Computer Vision: from Recognition to Geometry

Lecture 6: Recognition & Detection

http://media.ee.ntu.edu.tw/courses/cv/18F/

FB: NTUEE Computer Vision Fall 2018

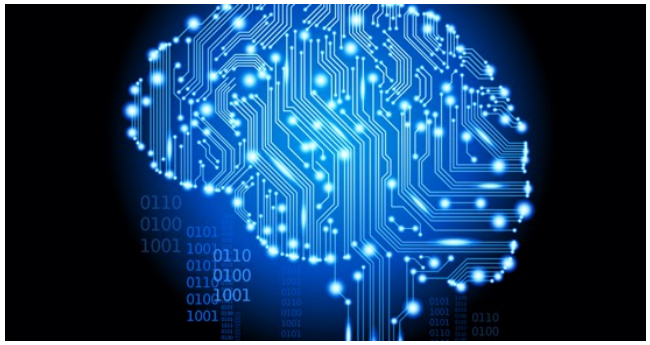
Yu-Chiang Frank Wang 王鈺強, Associate Professor

Dept. Electrical Engineering, National Taiwan University

2018/10/24

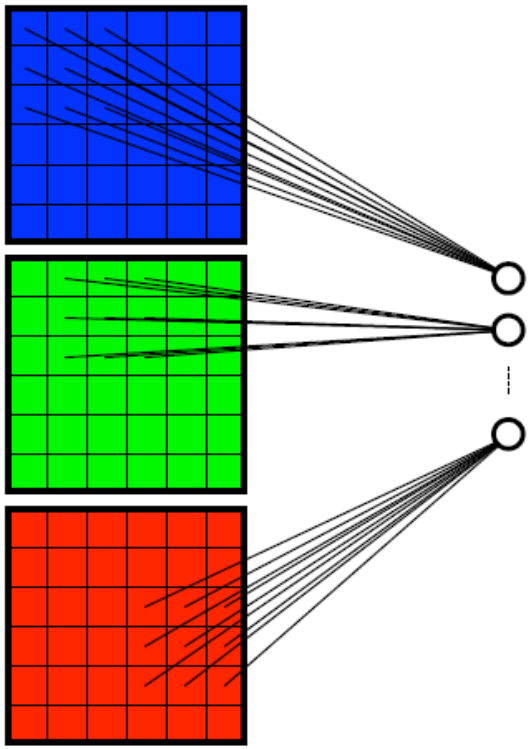
What's to Be Covered Today…

- Neural Networks & CNN
  - Convolutional Neural Networks
- Recognition & Detection
  - Recognition: From Interest Points to Bag-of-Words Models
  - Object Detection



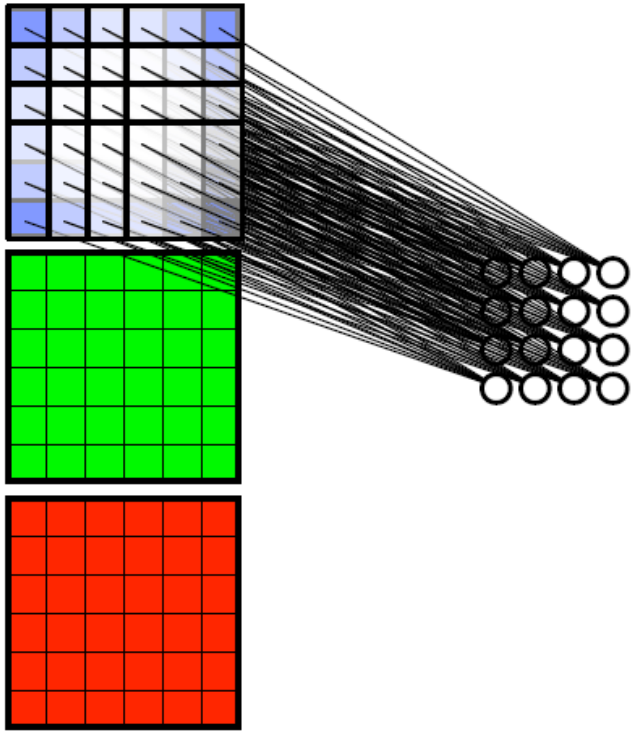
Convolutional Neural Networks

- Property I of CNN: Local Connectivity
  - Each neuron takes info only from a neighborhood of pixels.



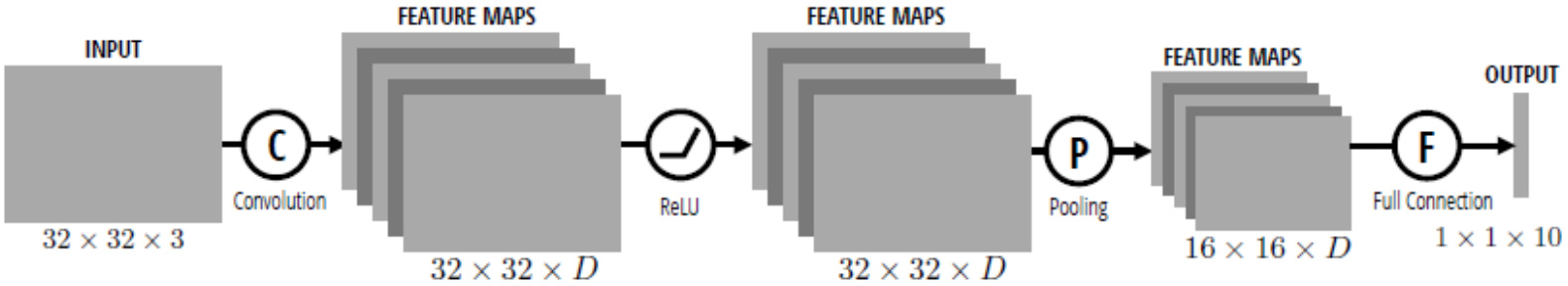
Convolutional Neural Networks

- Property II of CNN: Weight Sharing
  - Neurons connecting all neighborhoods have identical weights.

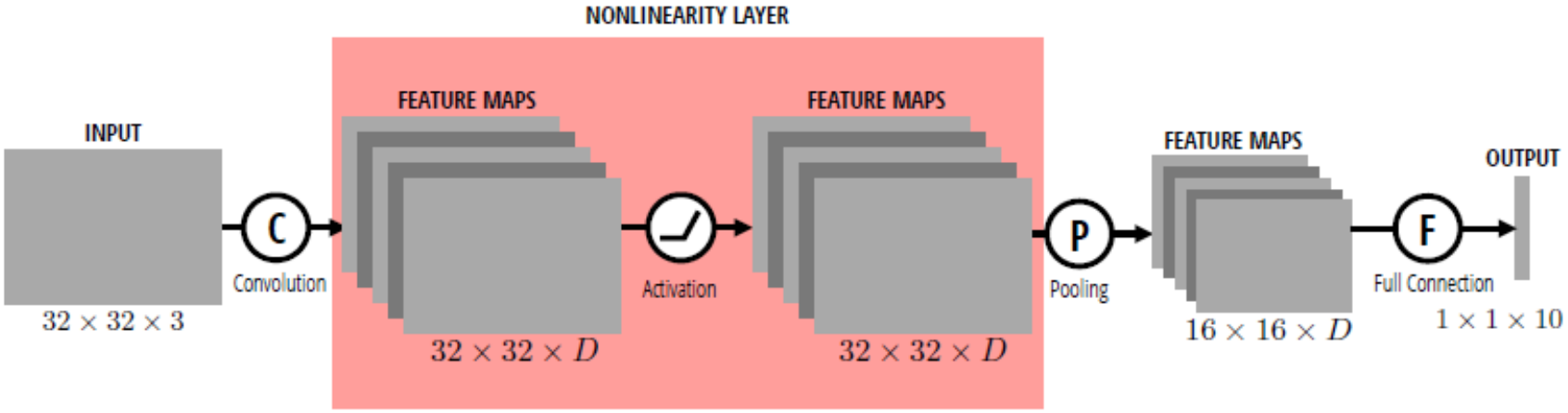


CNN

- Which layers are linear?
  - Convolution layer
  - Pooling layer
  - FC layer
  - Activation function
  - Softmax layer
- What if no nonlinear layer in CNN?



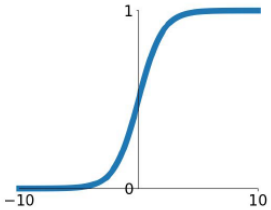
Nonlinearity Layer in CNN



Nonlinearity Layer (cont'd)

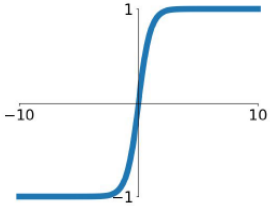
Sigmoid

$$\sigma(x) = \frac{1}{1+e^{-x}}$$



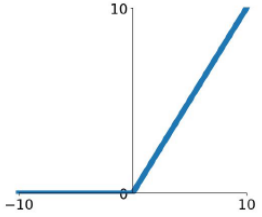
tanh

$$\tanh(x)$$



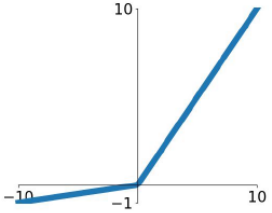
ReLU

$$\max(0, x)$$



Leaky ReLU

$$\max(0.1x, x)$$

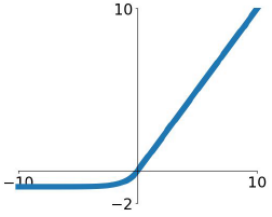


Maxout

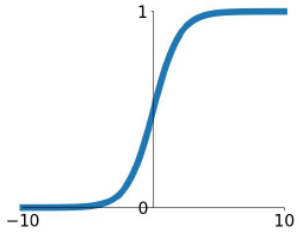
$$\max(w_1^T x + b_1, w_2^T x + b_2)$$

ELU

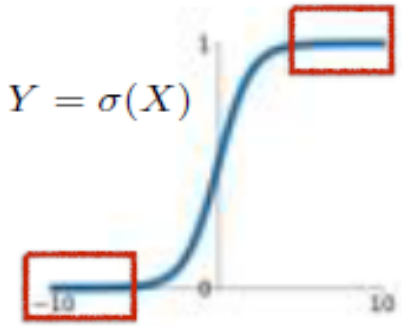
$$\begin{cases} x & x \geq 0 \\ \alpha(e^x - 1) & x < 0 \end{cases}$$



$$\sigma(x) = \frac{1}{1+e^{-x}}$$



- Sigmoid
  - Similar to step functions, but differentiable ☺
  - Easy to saturate ☹
    - Gradient vanishing problem



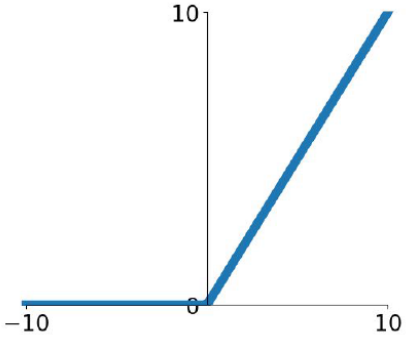
Chain rule

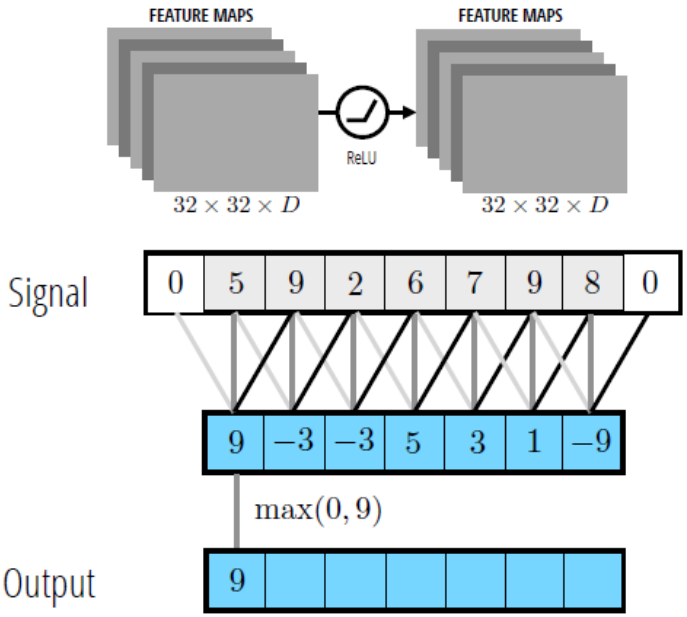
$$\frac{\partial Y}{\partial X} = \frac{\partial Y}{\partial \sigma}\frac{\partial \sigma}{\partial X}$$

is almost 0 for most of X

Thus, $\frac{\partial Y}{\partial X}$ also goes to 0.

- ReLU (Rectified Linear Unit)
  - Pixel by pixel computation of max(0, x)
  - Prevent gradient vanishing when x > 0 ☺
  - Computationally efficient
  - Biologically plausible
  - Still loose gradient when x < 0 ☹

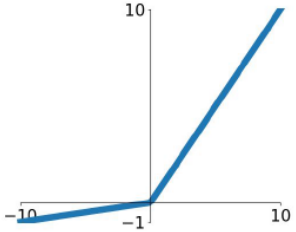




- Leaky ReLU & Exponential LU (ELU)
  - No gradient vanishing problem
  - Can be used instead of ReLU

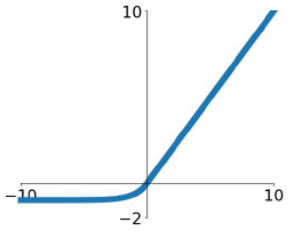
Leaky ReLU

$$\max(0.1x, x)$$



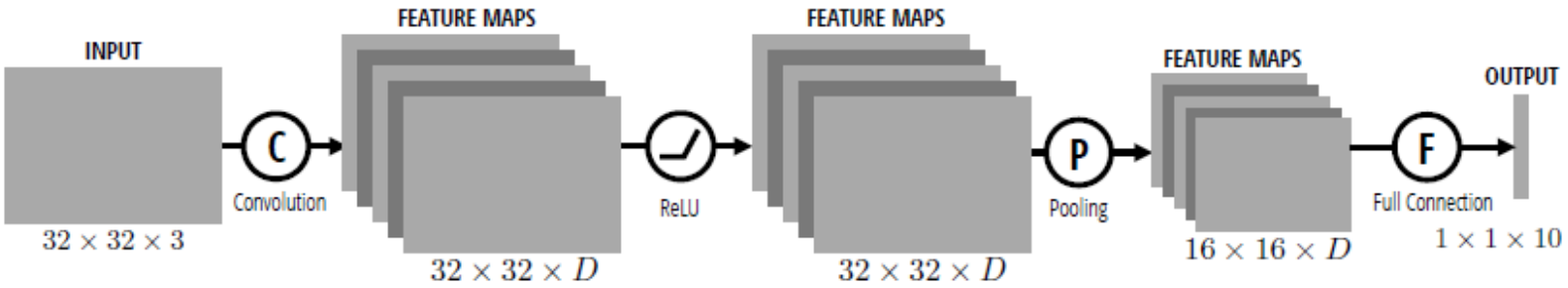
ELU

$$\begin{cases} x & x \geq 0 \\ \alpha(e^x - 1) & x < 0 \end{cases}$$

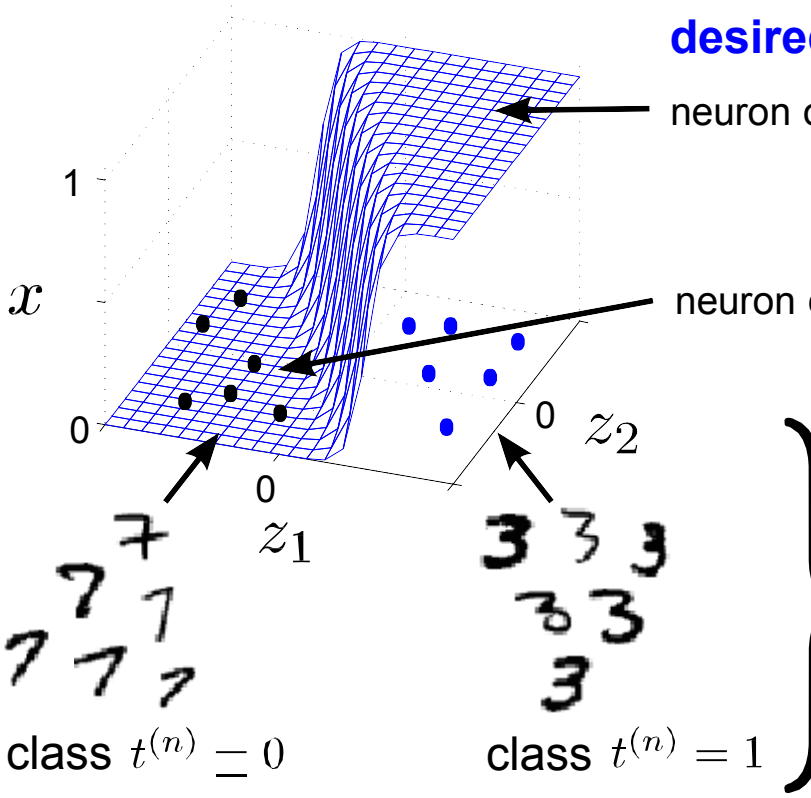


Output Layer in CNN

- Loss function



Recall that…Training a Single Neuron



desired result of training:

neuron outputs $\mathrm{x}(\boldsymbol{z}^{(n)}; \boldsymbol{w}) \approx 1$ for $t^{(n)} = 1$

neuron outputs $\mathrm{x}(\boldsymbol{z}^{(n)}; \boldsymbol{w}) \approx 0$ for $t^{(n)} = 0$

training data

$\{\boldsymbol{z}^{(n)}\}_{n=1}^{N}$ $\{t^{(n)}\}_{n=1}^{N}$

inputs class labels

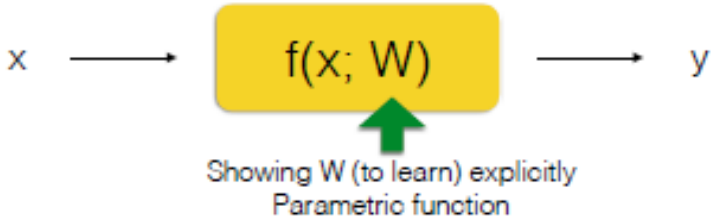
objective function:

$$G(\boldsymbol{w}) = -\sum_n \left[t^{(n)} \log \mathrm{x}(\boldsymbol{z}^{(n)}; \boldsymbol{w}) + (1 - t^n) \log \left(1 - \mathrm{x}(\boldsymbol{z}^{(n)}; \boldsymbol{w})\right)\right] \geq 0$$

surprise $-\log p(\text{outcome})$ when observing $t^{(n)}$

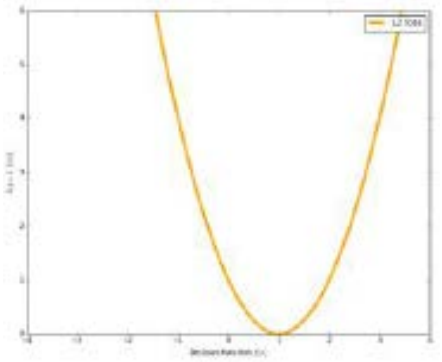
relative entropy between $\mathrm{x}(\boldsymbol{z}^{(n)}; \boldsymbol{w})$ and $t^{(n)}$

} encourages neuron output to match training data



- Loss function L
  - Recall that L measures how well learned W can map input X to output Y
  - E.g., L2 loss

$$Loss = \frac{1}{N}\sum_{i=1}^{N} \| y_i - f(x_i; W, b)\|_2^2$$



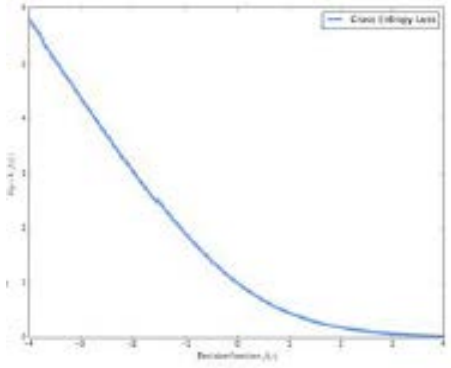
Perceptual loss: 算feature之間的loss

  - E.g., Cross-entropy loss

$$Loss = -\frac{1}{N}\sum_{i=1}^{N} \hat{y_i}\, log(P(y_i = j\,|x_i))$$

$$P(y_i = j\,|x_i) = \frac{e^{f(x_i;W,b)}}{\sum_{j=1}^{C} e^{f(x_j;W,b)}}$$

i.e., softmax



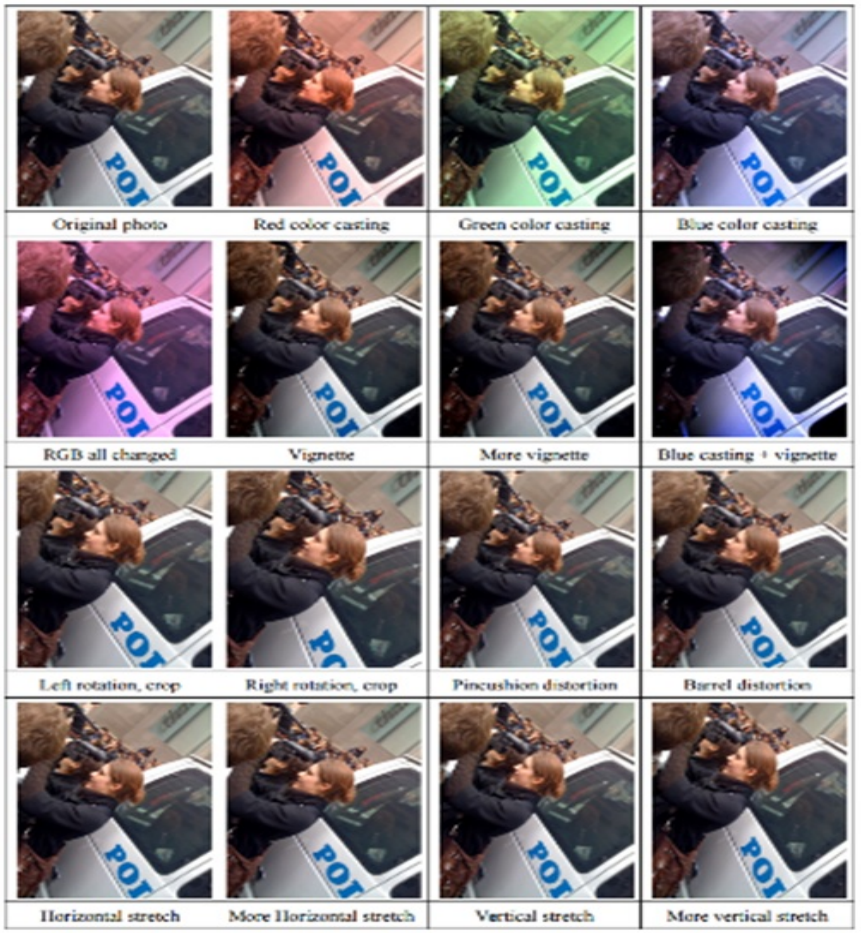
Training Convolutional Neural Networks

- Backpropagation +
  stochastic gradient descent with momentum
  - Neural Networks: Tricks of the Trade
- Data augmentation
- Dropout
- Batch normalization

擴增

Data Augmentation (Jittering)

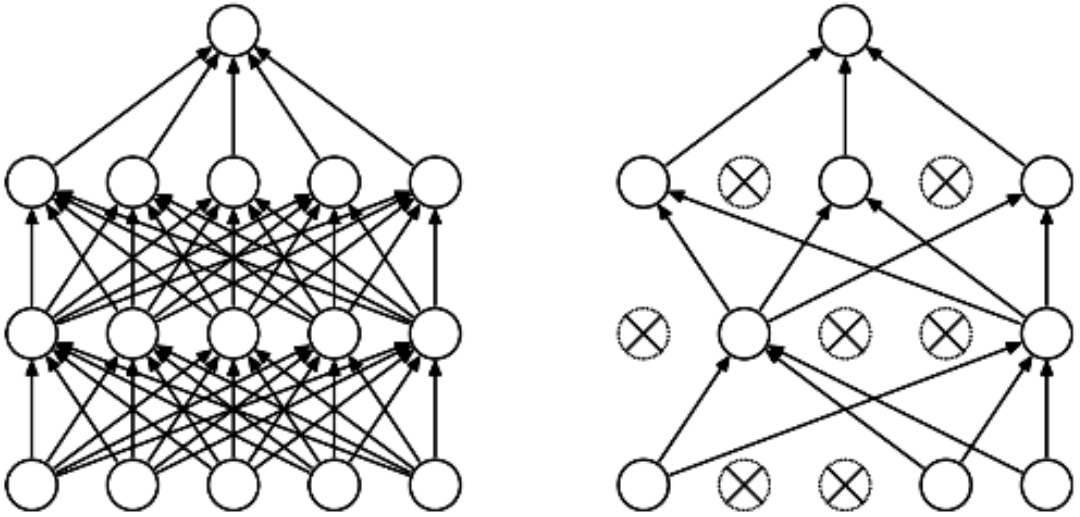
- Create *virtual* training samples
  - Horizontal flip
  - Random crop
  - Color casting
  - Geometric distortion



Deep Image [Wu et al. 2015]

Dropout

N個臭皮匠 勝過一個諸葛亮



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

Intuition: successful conspiracies

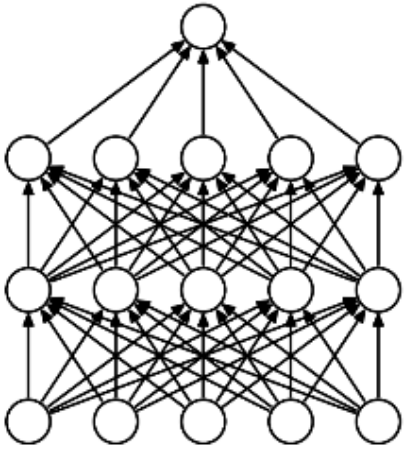
- 50 people planning a conspiracy

- Strategy A: plan a big conspiracy involving 50 people
  - Likely to fail. 50 people need to play their parts correctly.

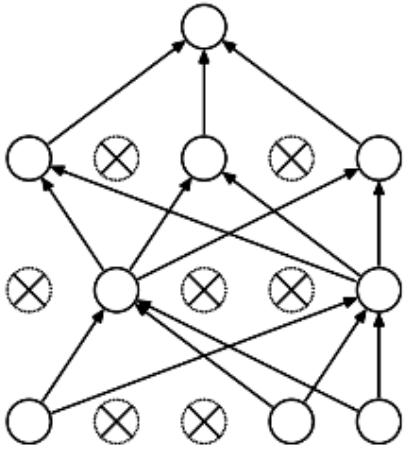
- Strategy B: plan 10 conspiracies each involving 5 people
  - Likely to succeed!

Dropout: A simple way to prevent neural networks from overfitting [Srivastava JMLR 2014]

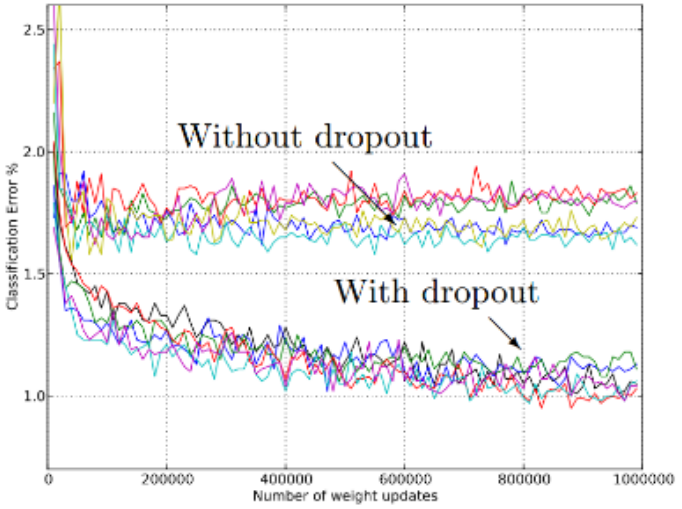
Dropout

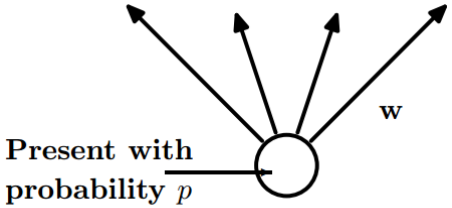


(a) Standard Neural Net

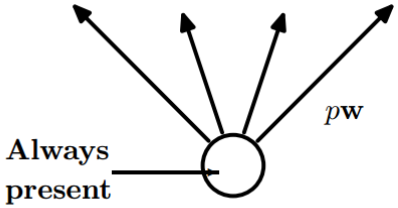


(b) After applying dropout.





(a) At training time



(b) At test time

Main Idea: approximately combining exponentially many different neural network architectures efficiently

Model	Top-1 (val)	Top-5 (val)	Top-5 (test)
SVM on Fisher Vectors of Dense SIFT and Color Statistics	-	-	27.3
Avg of classifiers over FVs of SIFT, LBP, GIST and CSIFT	-	-	26.2
Conv Net + dropout (Krizhevsky et al., 2012)	40.7	18.2	-
Avg of 5 Conv Nets + dropout (Krizhevsky et al., 2012)	38.1	16.4	16.4

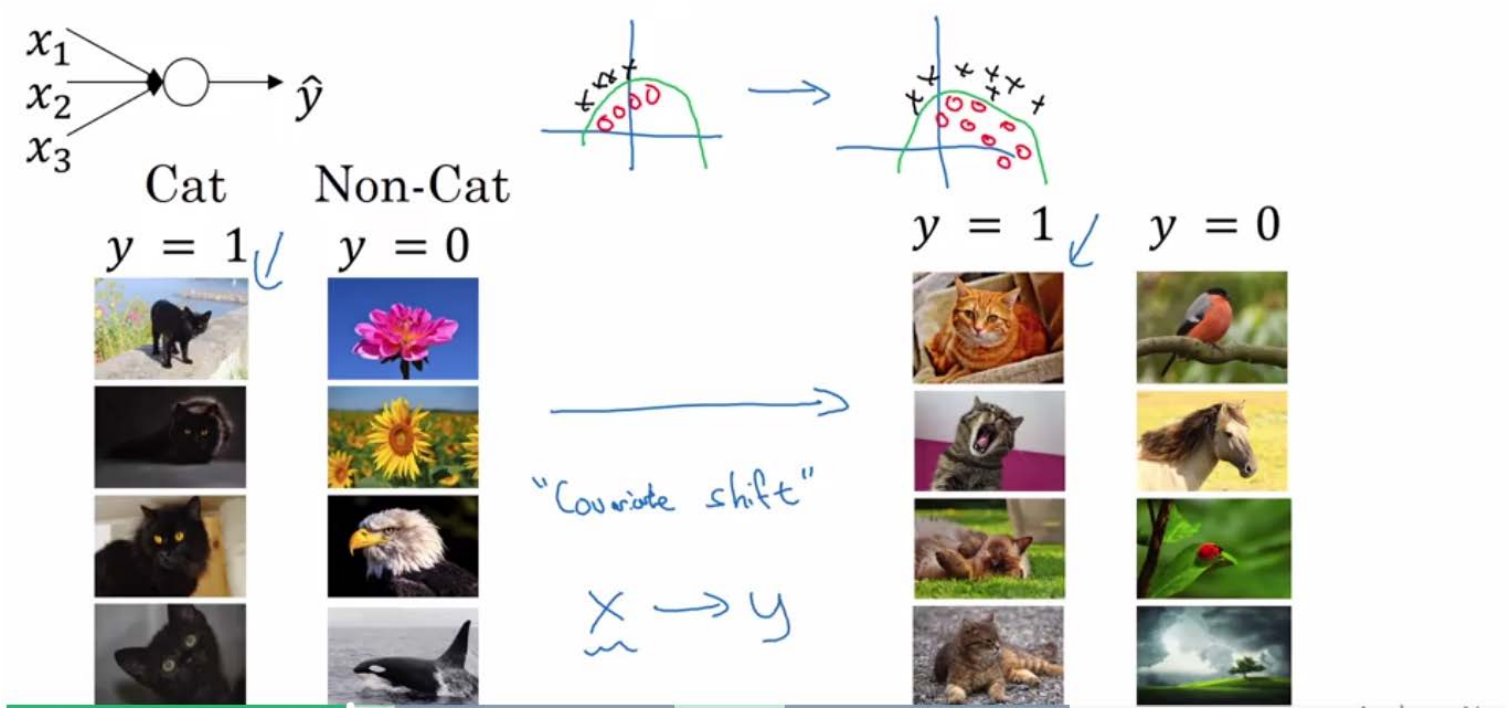
Table 6: Results on the ILSVRC-2012 validation/test set.

Dropout: A simple way to prevent neural networks from overfitting [Srivastava JMLR 2014]

Batch Normalization [Ioffe and Szegedy, 2015]

zero-mean normalization

"You need unit Gaussian activations? Just make them so."

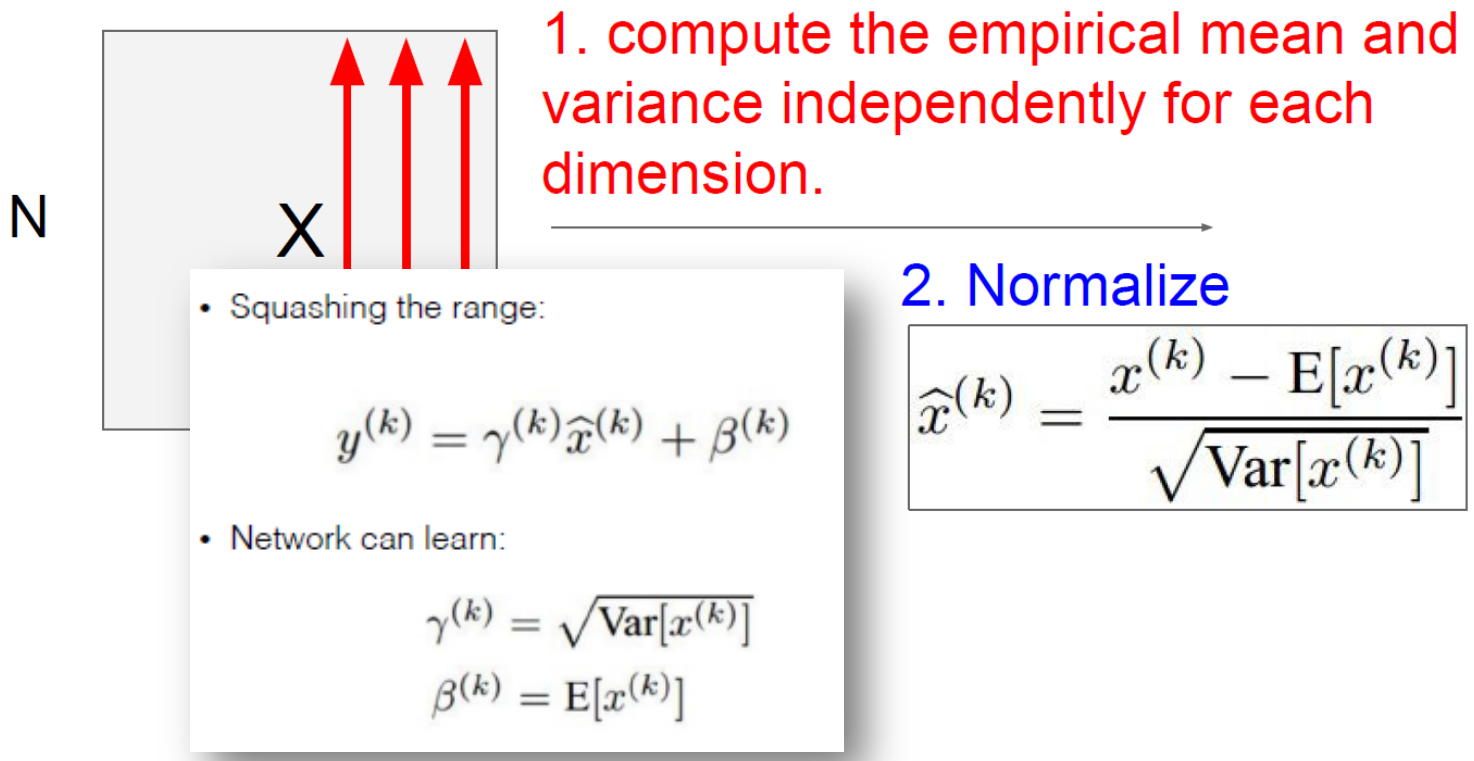


Credit: Andrew Ng

Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift [Ioffe and Szegedy 2015]

Batch Normalization [Ioffe and Szegedy, 2015]

"You need unit Gaussian activations? Just make them so."



Usually inserted after convolutional or FC layers, and before nonlinearity.

Credit: Fei-Fei Li

Batch Normalization (cont'd)

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

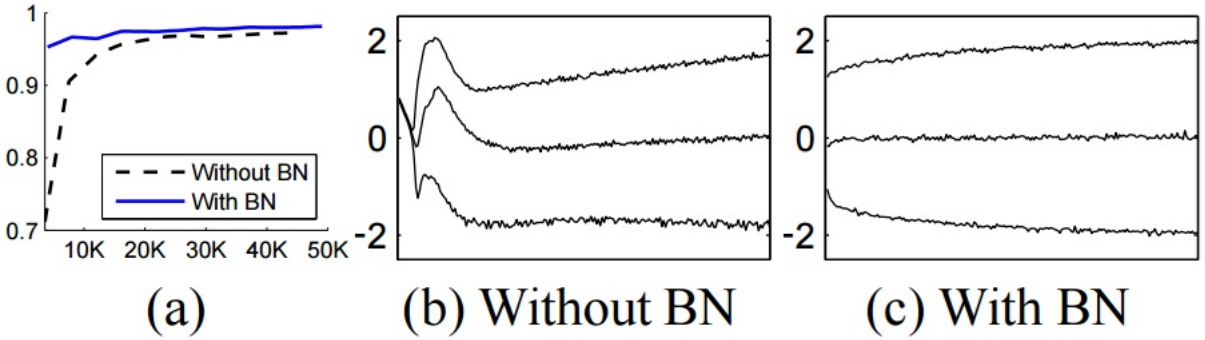
Output: $\{y_i = \mathrm{BN}_{\gamma,\beta}(x_i)\}$

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

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

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

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



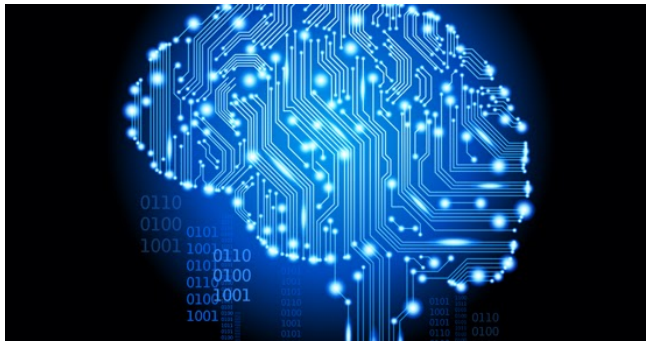
(a) (b) Without BN (c) With BN

Batch Normalization (cont'd)

- At test time
  - Batch Norm layer functions differently.
  - The mean/std are not computed based on the batch. Instead, a single fixed empirical mean of activations during training is used.
  - E.g., mean/std are estimated during training via running average
- Why Batch Normalization?
  - Improves gradient flow
  - Allows higher learning rate
  - Reduces strong dependence on initialization
  - Regularization

What's to Be Covered Today…

- Neural Networks & CNN
  - Convolutional Neural Networks
- Recognition & Detection
  - Recognition: From Interest Points to Bag-of-Words Models
  - Object Detection



Recall that: Interest Points?

- Registration & Correspondence
  - Identifying corresponding points/patches/regions across images
  - Apps: matching, alignment, stitching, etc.

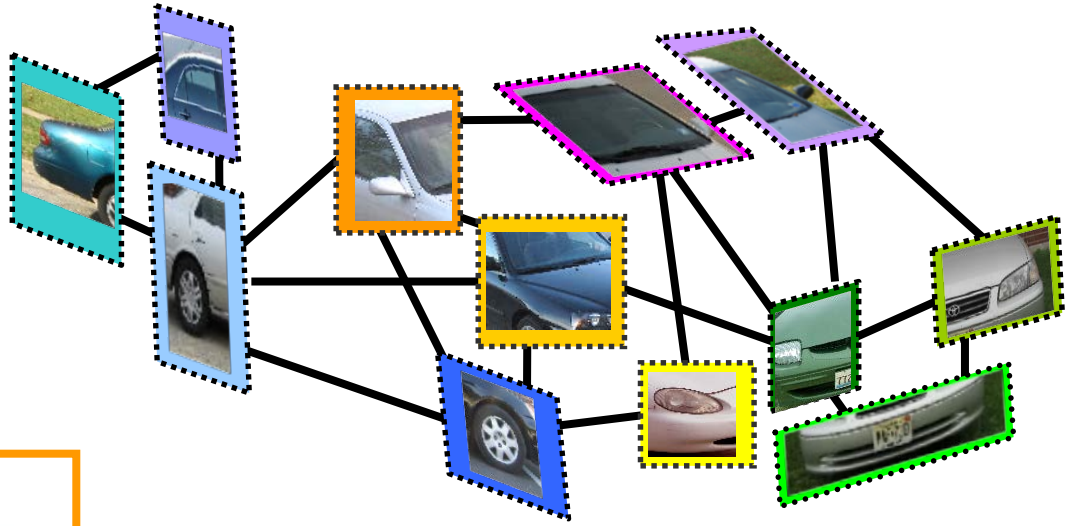
Why Interest Points? (cont'd)

- Example: panorama

Credit: Matt Brown

Why Interest Points? (cont'd)

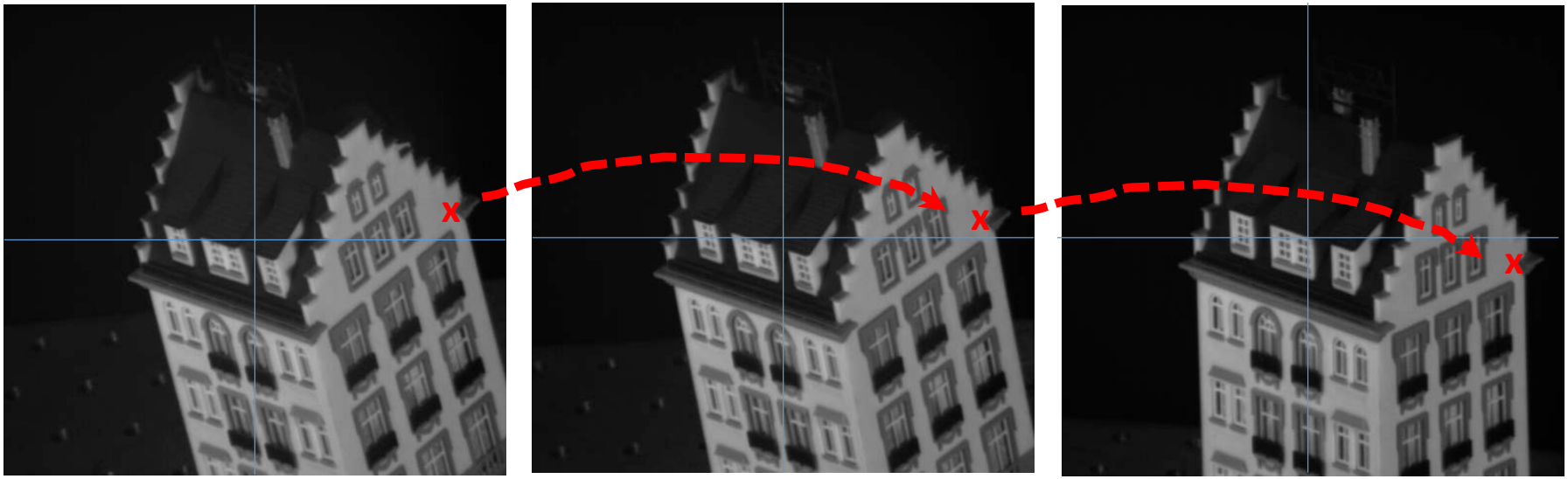
- Example: fitting 3D models



Credit: Silvio Savarese

Why Interest Points? (cont'd)

- Example: tracking



frame 0 frame 22 frame 49

Why Interest Points? (cont'd)

- Examples
  - Image alignment
  - 3D reconstruction
  - Motion tracking
  - Object recognition
  - Robot navigation
  - Indexing and database retrieval

About Interest Points

- Desirable properties for local features
  - **Locality**

    Features are local, and robust to noise
  - **Quantity**

    A large number would be expected.
  - **Distinctiveness**

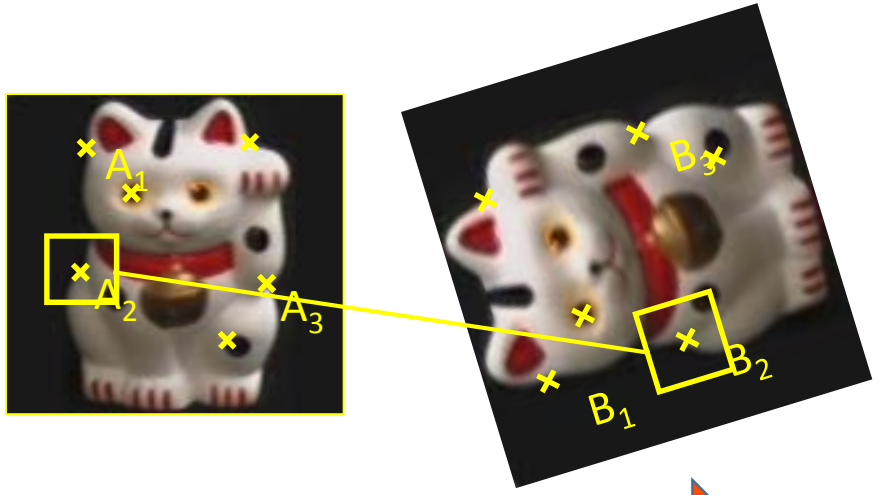
    Differentiate a variety of images of interest (e.g., objects, etc.)
  - **Repeatability**

    Able to detect the same interest points of interest
  - **Compact & Efficiency**

    Real-time performance would be desirable.

About Interest Points

- Key Trade-offs



Detection

Few Points

More distinctive representation
Robust detection
Precise localization

More Points

Robust to occlusion
Works with less texture

Description

More Distinctive

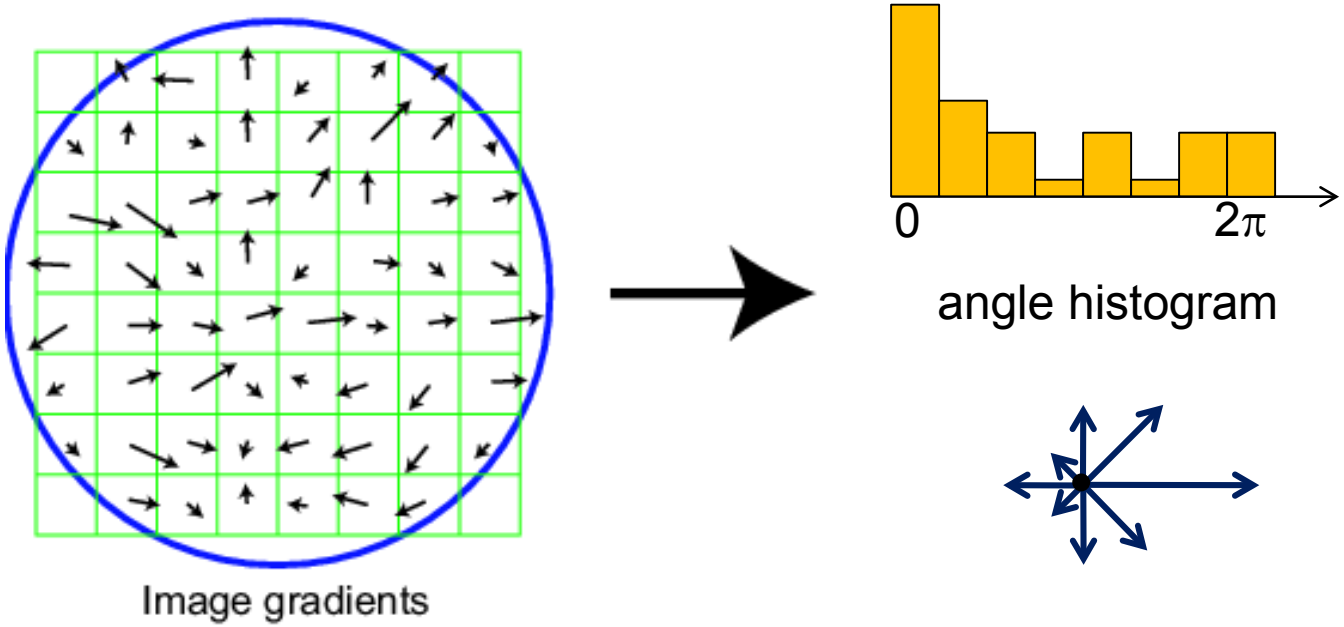
Minimize wrong matches

More Flexible

Robust to expected variations
Maximize correct matches

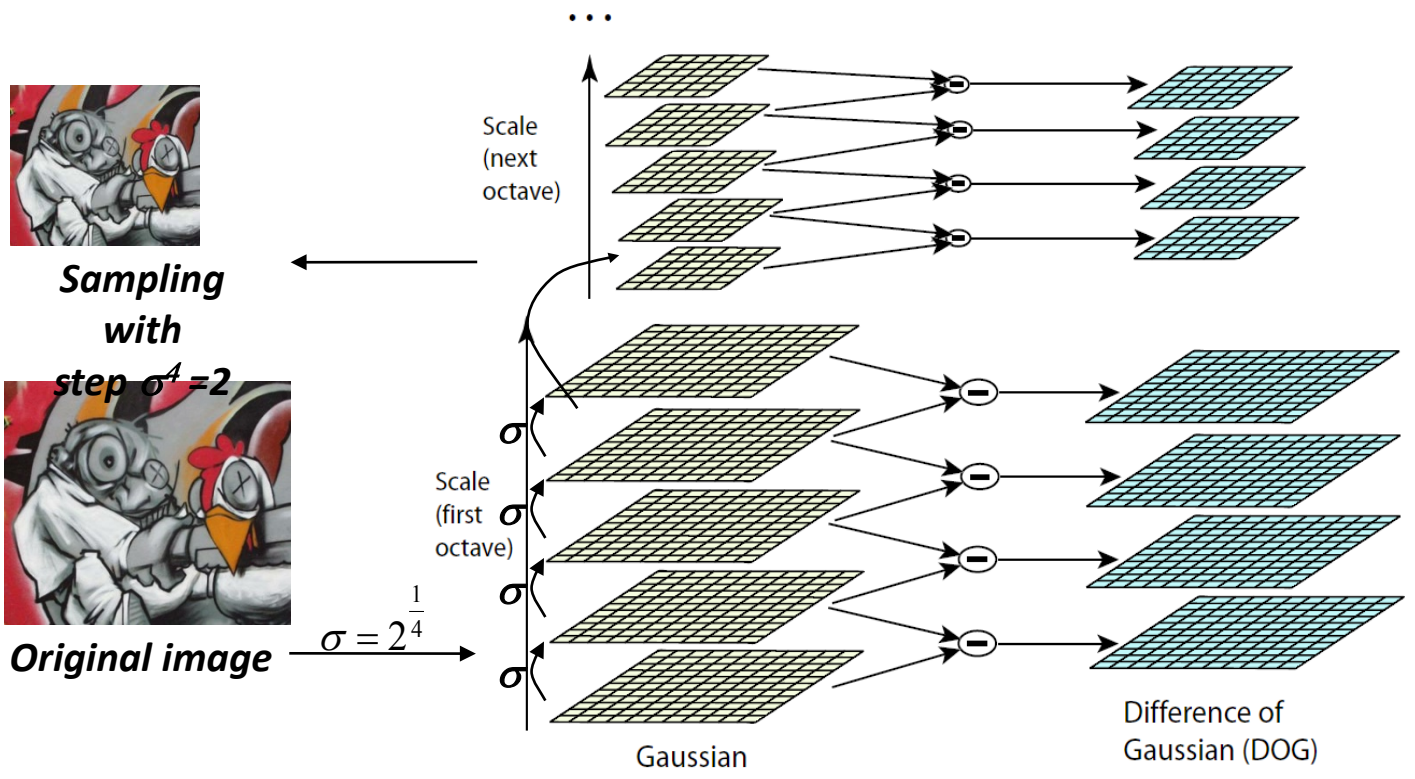
Scale Invariant Feature Transform (SIFT)

- Key Ideas
  - Take a 4 x 4 (= 16 grids) square window around each detected keypoint
  - Compute edge orientation (angle of the gradient - 90°) for each pixel in it
  - Throw out weak edges (threshold gradient magnitude)
  - Create histogram of surviving edge orientations

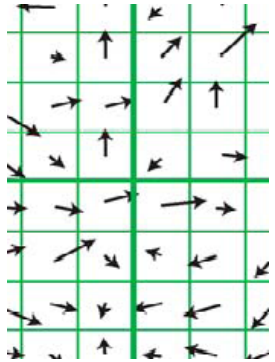


Scale Invariant Feature Transform (SIFT)

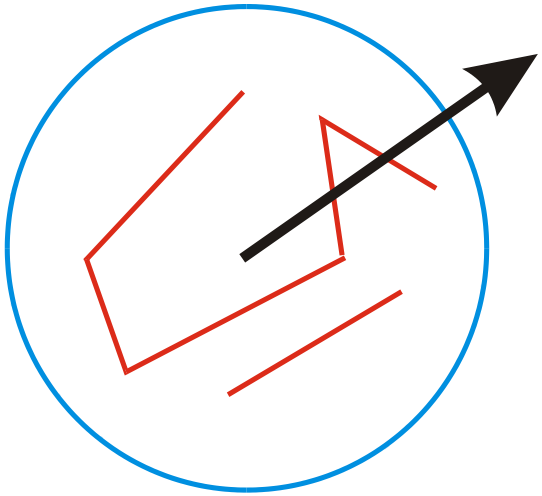
- Step 1. Keypoint Detection/Localization
  - Eliminate edge responses

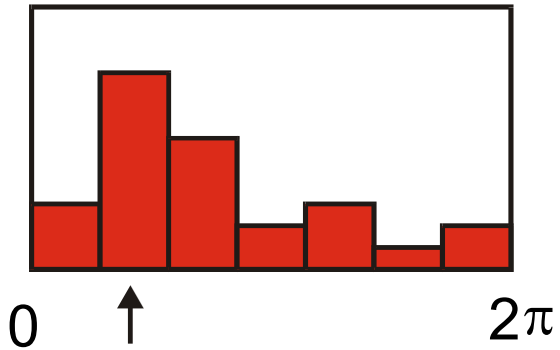


Scale Invariant Feature Transform (SIFT)



- Step 2. Orientation Normalization
  - Caluculate orientation and magnitude of gradients in each pixel
  - Histogram of orientations of sample points near keypoint.



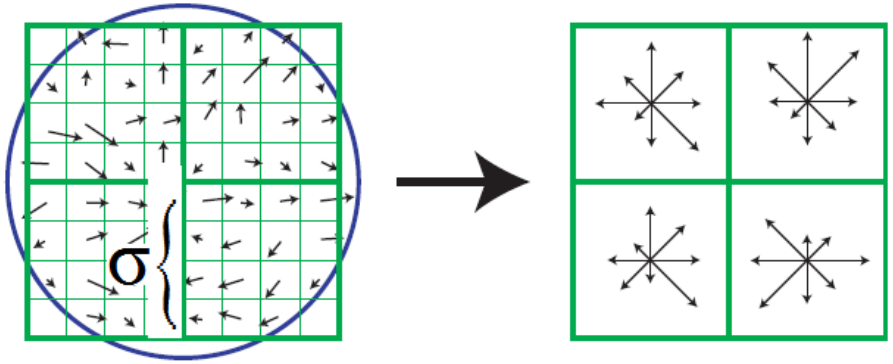


$$m(x, y) = \sqrt{(L(x+1, y) - L(x-1, y))^2 + (L(x, y+1) - L(x, y-1))^2}$$

$$\theta(x, y) = \tan^{-1}((L(x, y+1) - L(x, y-1))/(L(x+1, y) - L(x-1, y)))$$

Scale Invariant Feature Transform (SIFT)

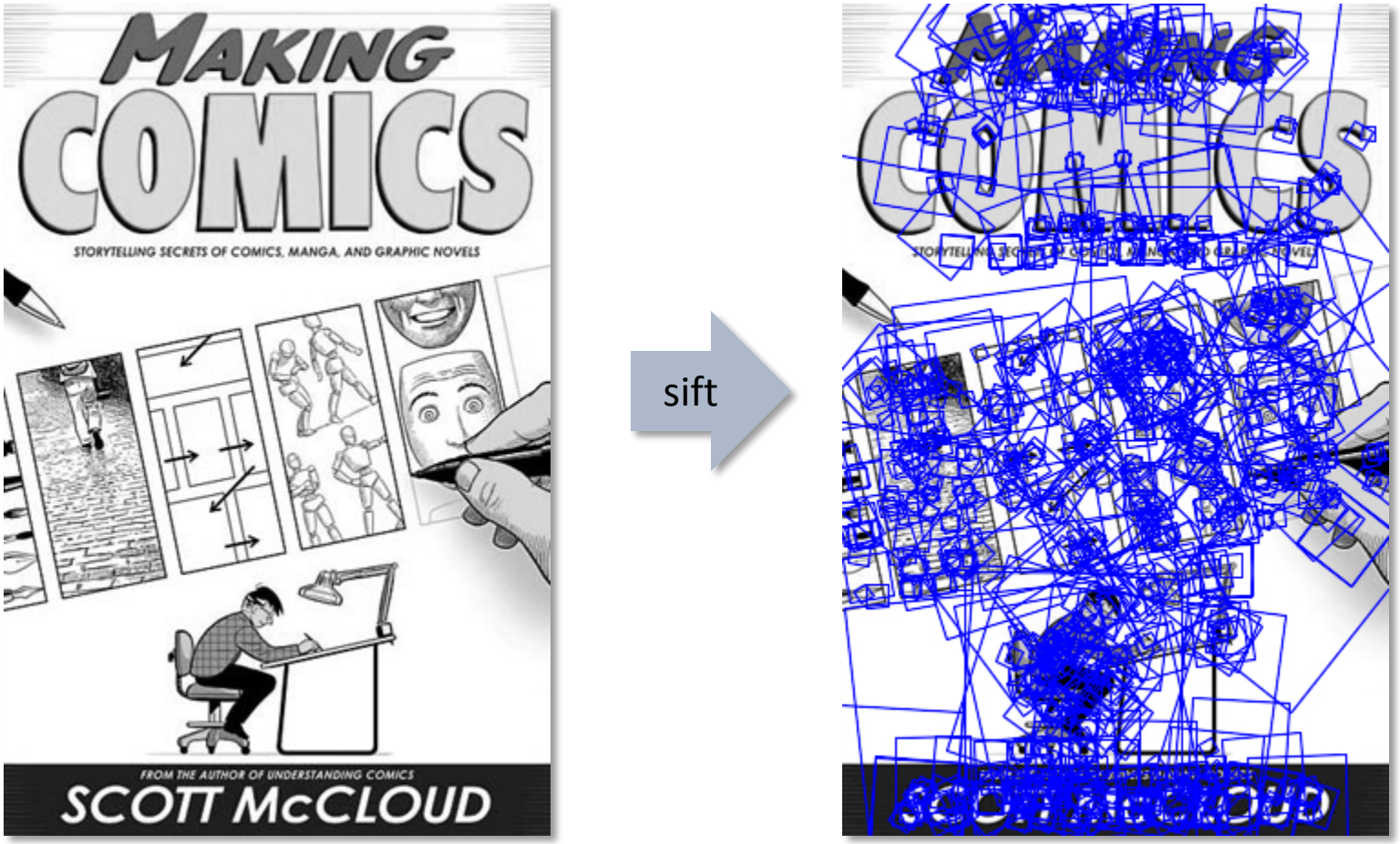
- Step 3. Keypoint Descriptor
  - Recall that orienation has been normalized.
  - **3-1**. Divide sample points around keypoint in 4 x 4 = 16 regions
    (4 regions shown in the bottom-right figure)
  - **3-2**. Calculate histogram of orientations with 8 bins for each region
    (followed by Trilinear interpolation + Vector normalization)
  - Excluding (x, y), total dimension # of SIFT descriptor:



  - Invariant to local scale & orientation. What about out-of-plane rotation?
  - Invariant for illumination and 3D viewpoint changes?

Scale Invariant Feature Transform (SIFT)

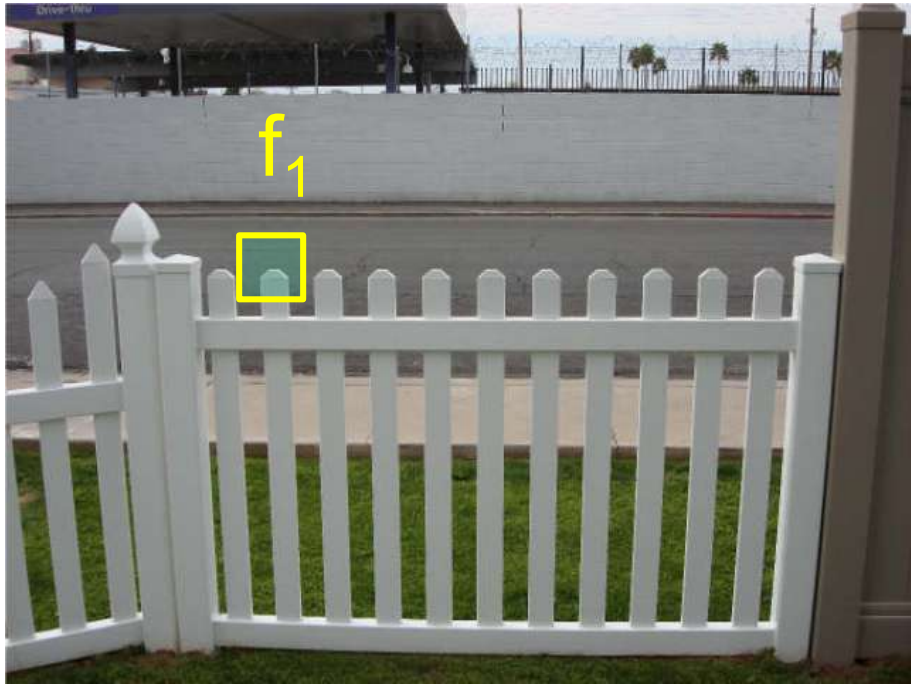
- Example



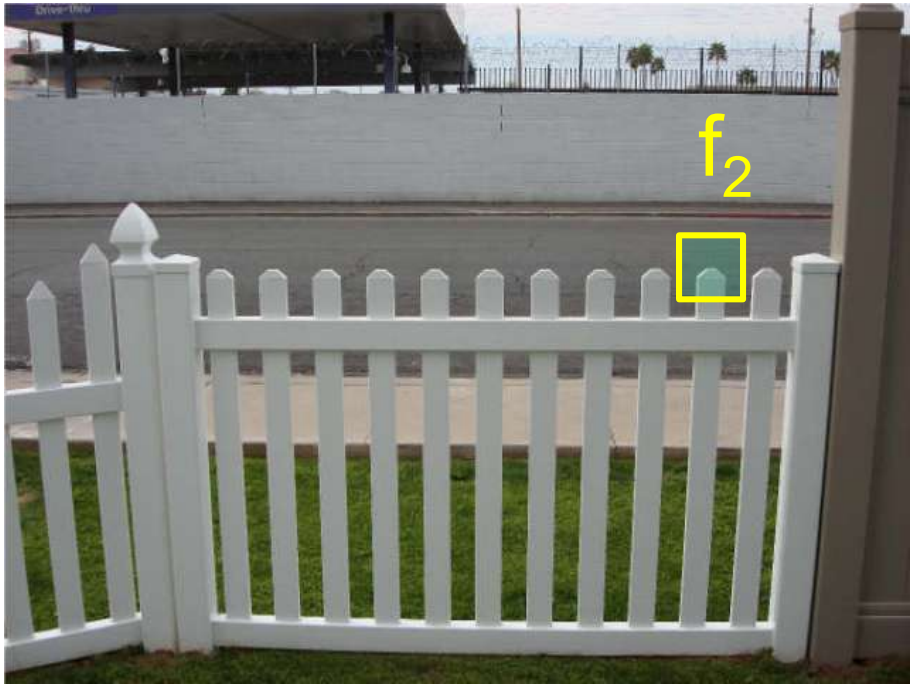
868 SIFT features

Interest Points for Image Matching

- Given a region of interest f_1 in I_1, how to find the best match f_2 in I_2?
  1. Define distance function that compares two descriptors f_1 & f_2
  2. Test all the features in I_2, find the one with min distance



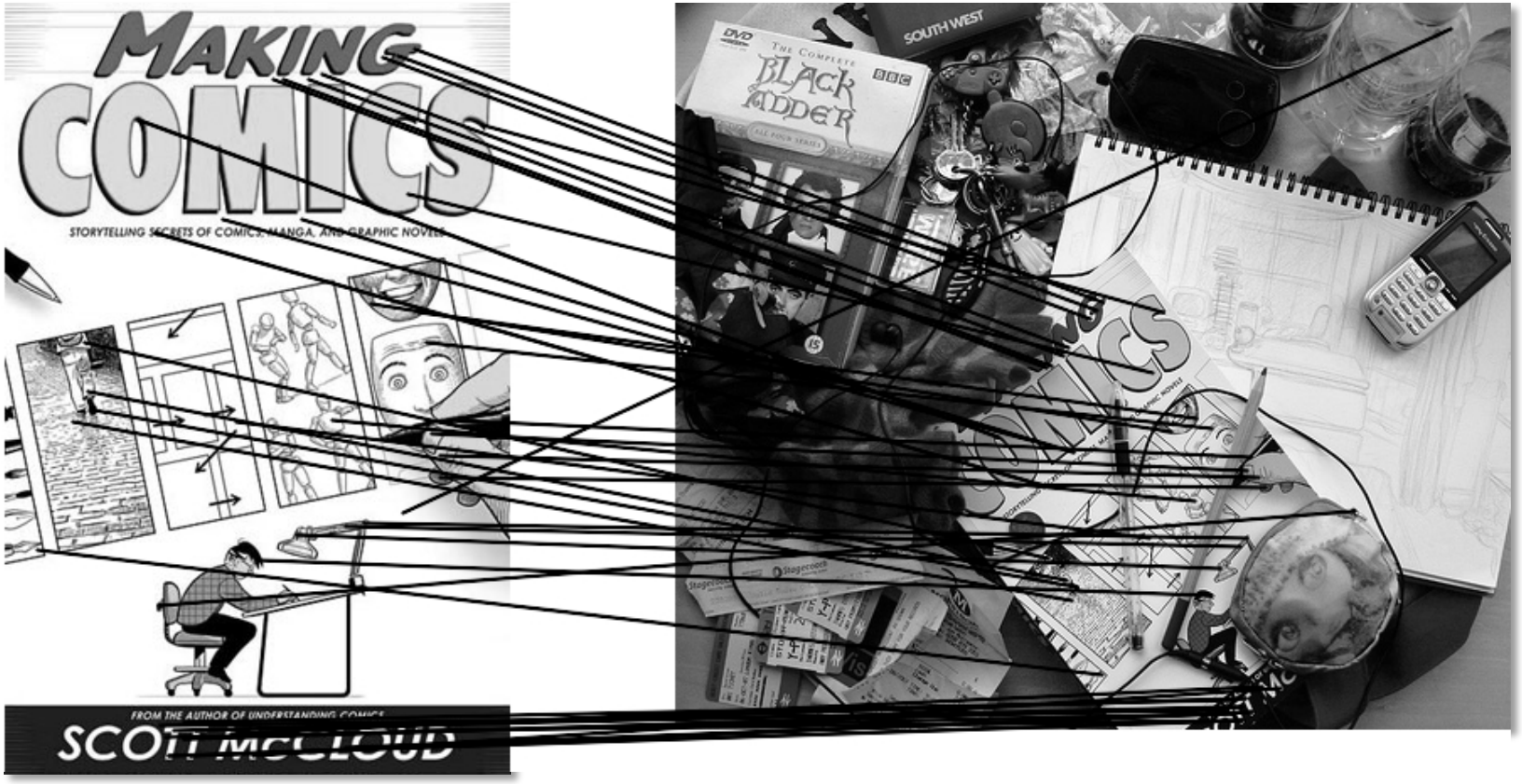
I_1



I_2

Interest Points for Image Matching

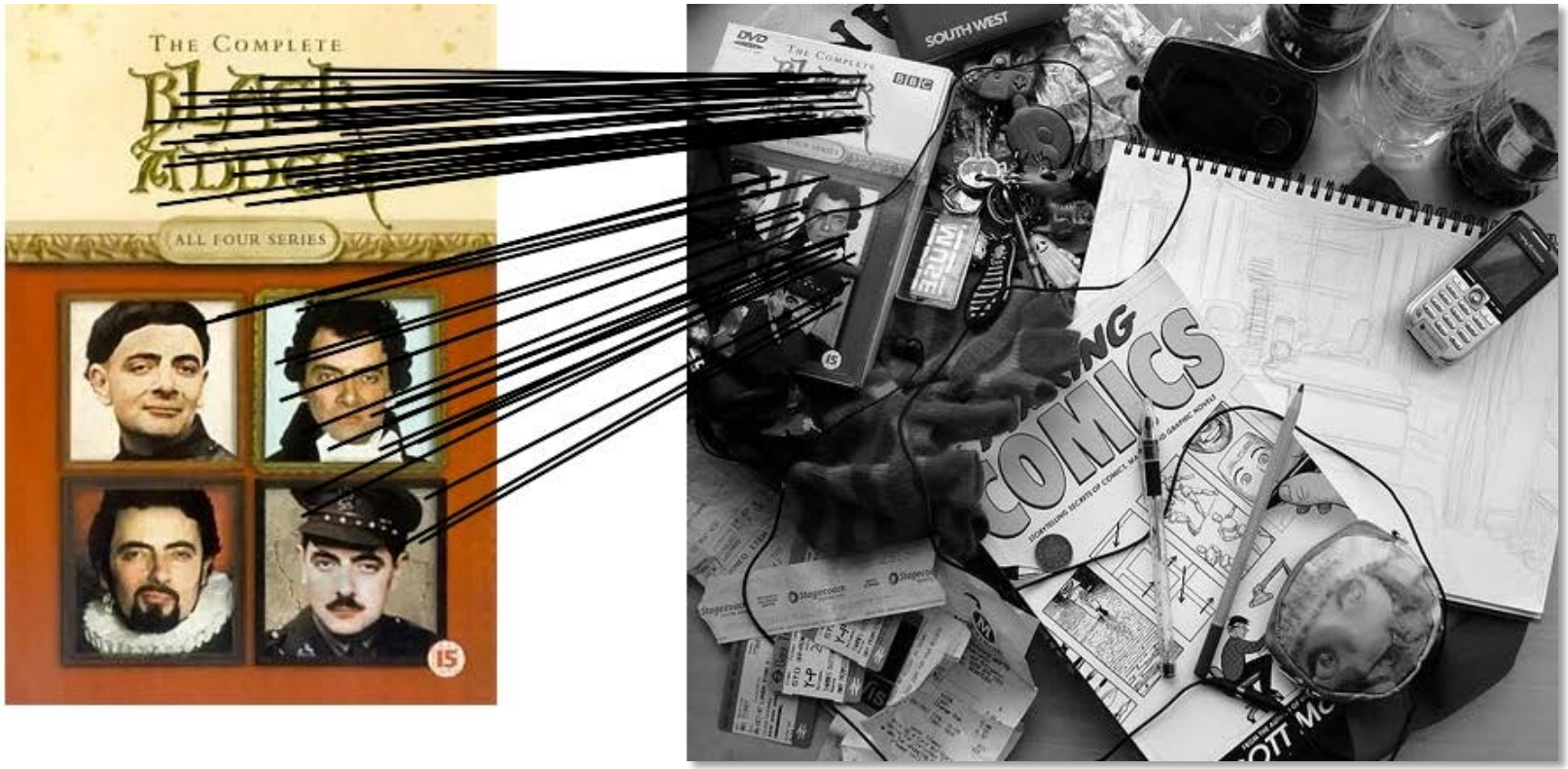
- Examples



51 matches

Interest Points for Image Matching

- Examples

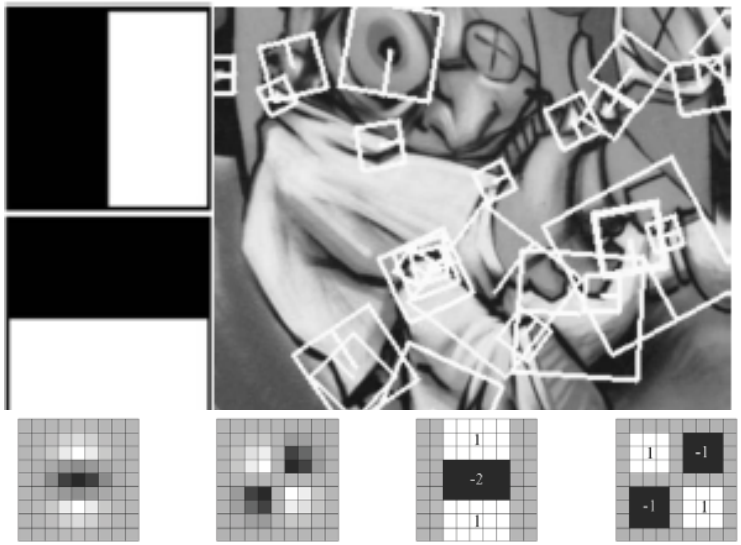


58 matches

✓ Disregard outlier pairs by RANdom SAmple Consensus (RANSAC) algorithm

Recent Advances in Interest Points

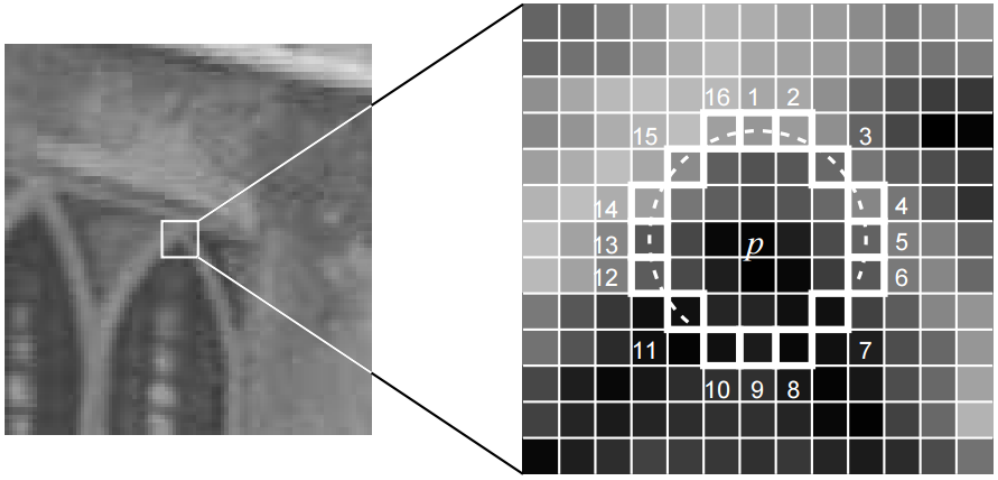
- Speeded Up Robust Features (SURF)
  - Fast approximation of SIFT
  - Efficient computation by 2D box filters & integral images
  - Equivalent quality for object identification
  - GPU implementation available
    - Feature extraction @ 200Hz
      (detector + descriptor, 640×480 img) http://www.vision.ee.ethz.ch/~surf



[Bay, ECCV'06], [Cornelis, CVGPU'08]

Recent Advances in Interest Points

- Binary Descriptors
  - BRIEF: Binary Robust Independent Elementary Features, ECCV 10
  - ORB (Oriented FAST and Rotated BRIEF), CVPR 11
  - BRISK: Binary robust invariant scalable keypoints, ICCV 11
  - Freak: Fast retina keypoint, CVPR 12
  - LIFT: Learned Invariant Feature Transform, ECCV 16



Features from Accelerated Segment Test, ECCV 06

What's to Be Covered Today…

- Neural Networks & CNN
  - Convolutional Neural Networks
- Recognition & Detection
  - Recognition: From Interest Points to Bag-of-Words Models
  - Object Detection

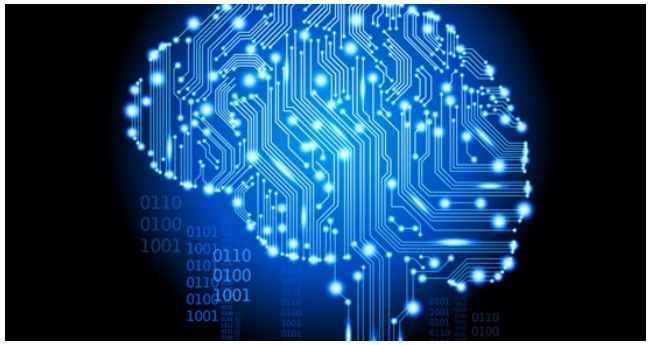
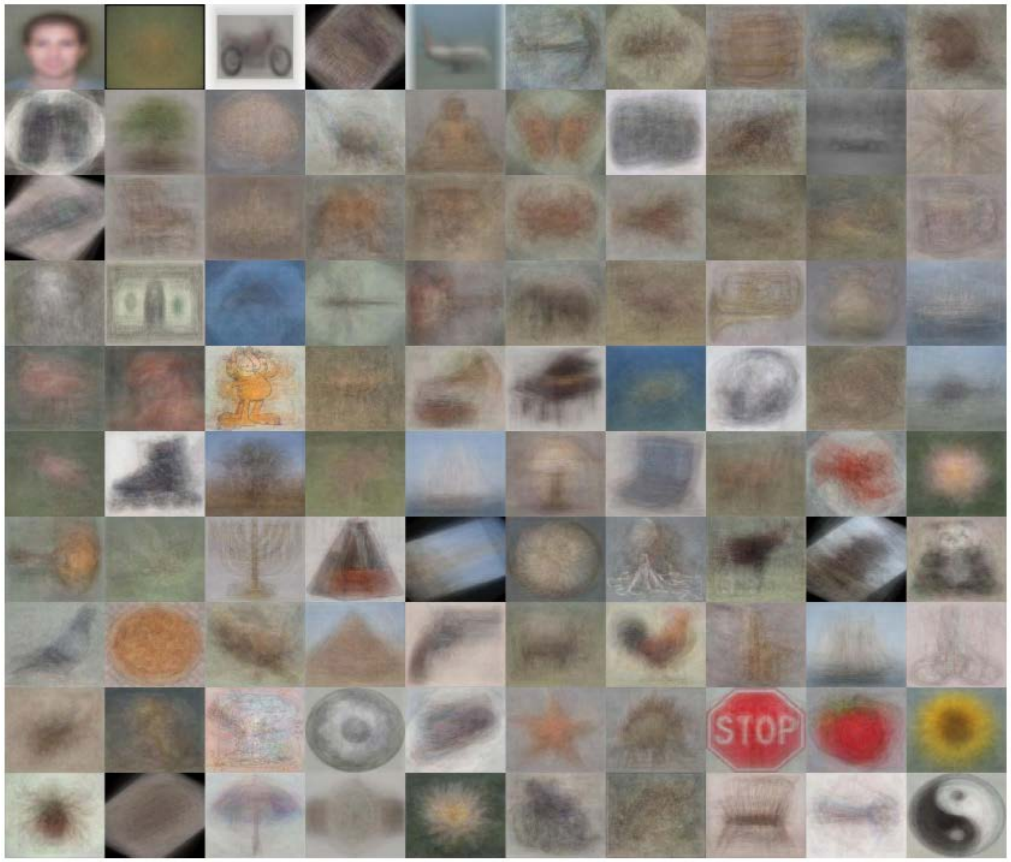


Image Categorization

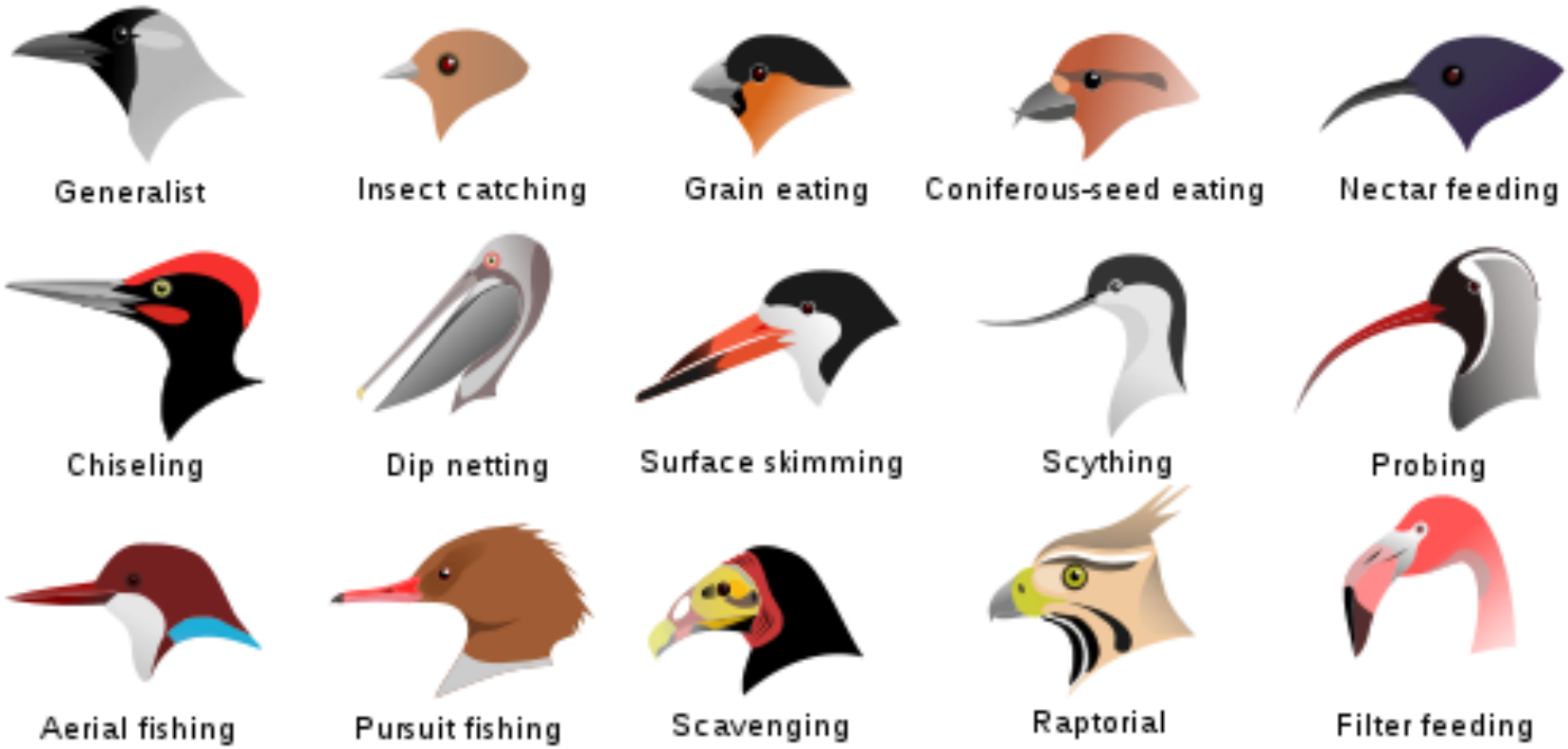
- Object Recognition



Average Object Images of Caltech 101

Image Categorization

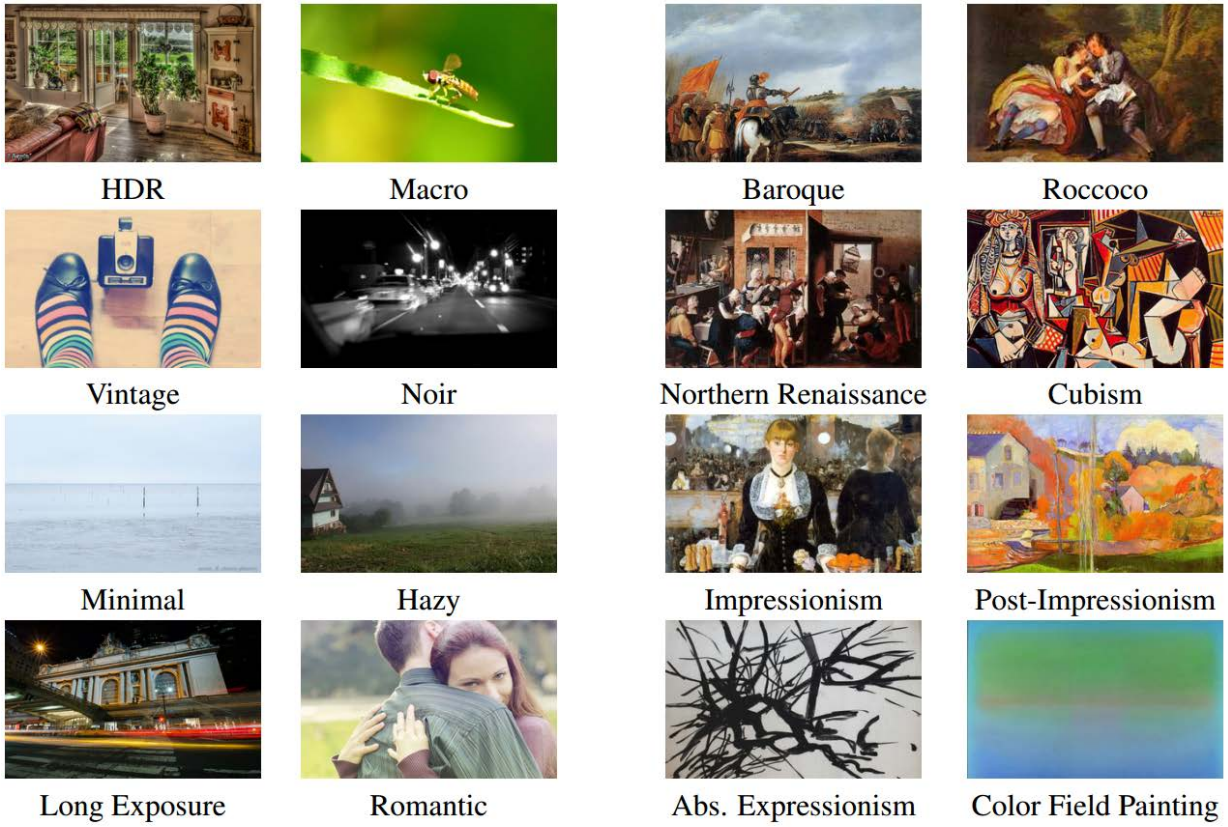
- Fine-Grained Recognition



Visipedia Project

Image Categorization

- Image style recognition



[Karayev et al. BMVC 2014]

Image Categorization

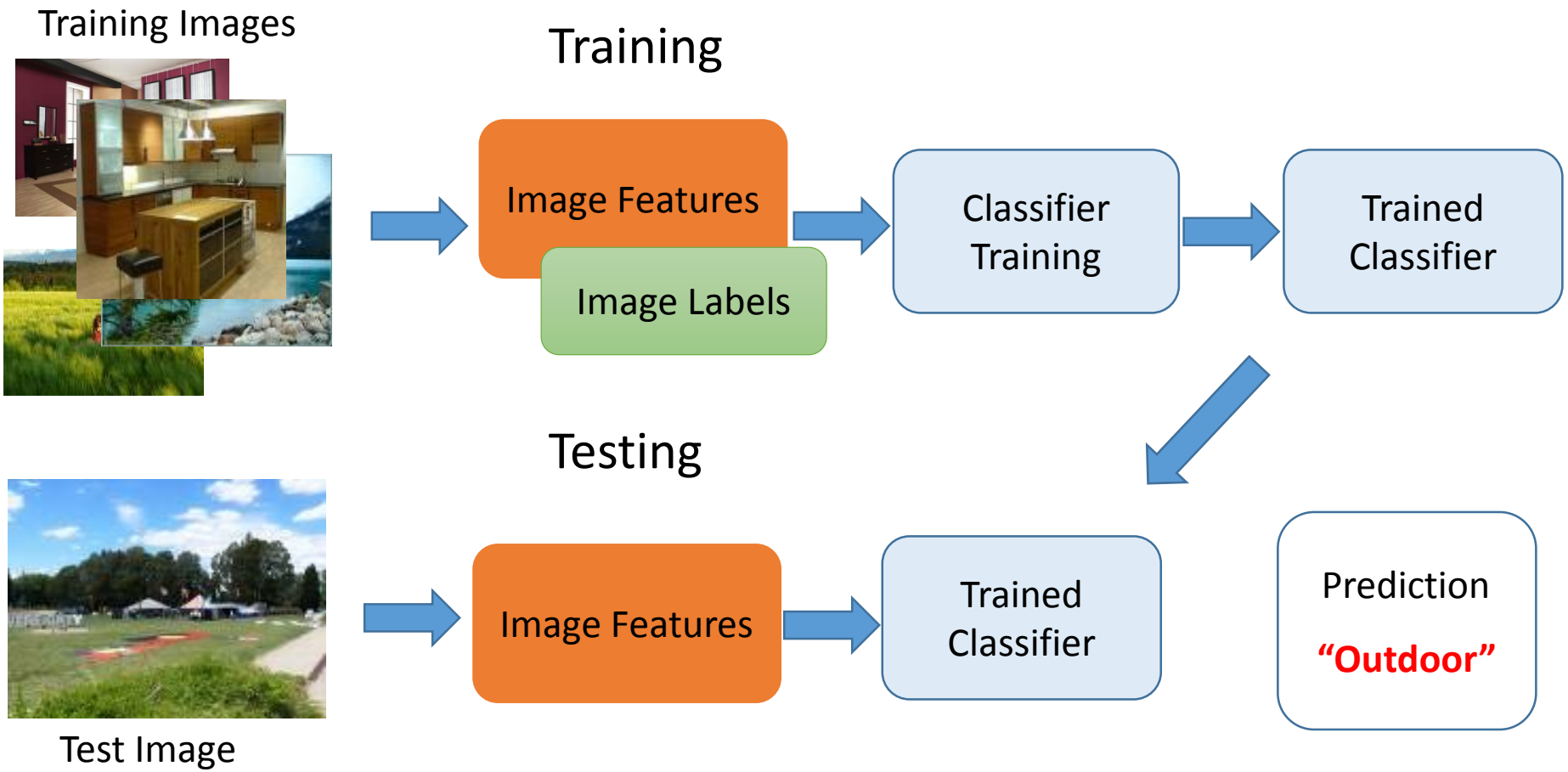
- Dating historical photos

1940 1953 1966 1977

[Palermo et al. ECCV 2012]

Supervised Learning for Visual Classification

- Training vs. Testing Phases

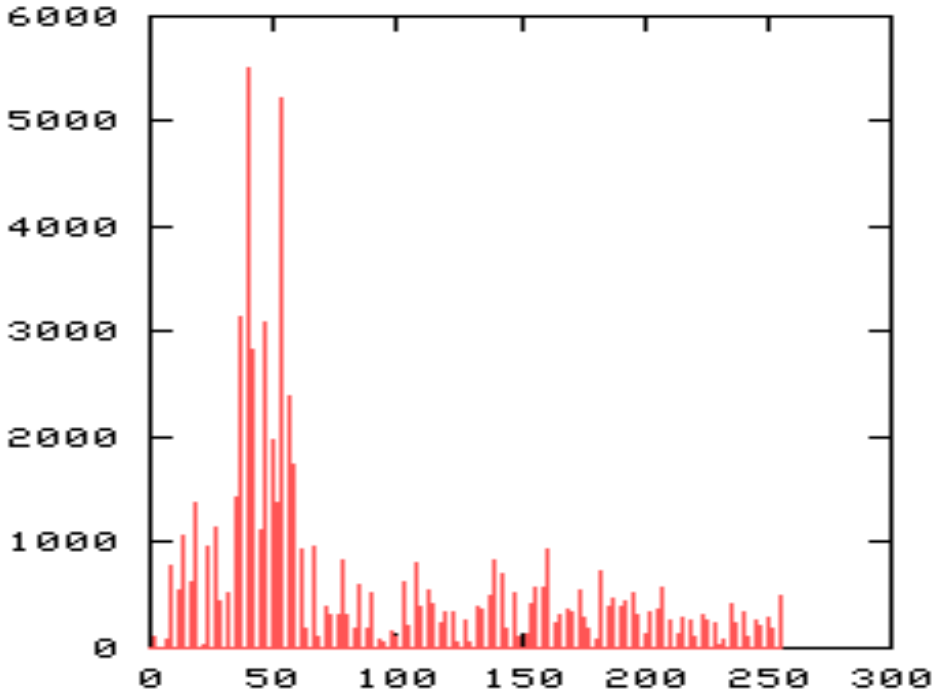


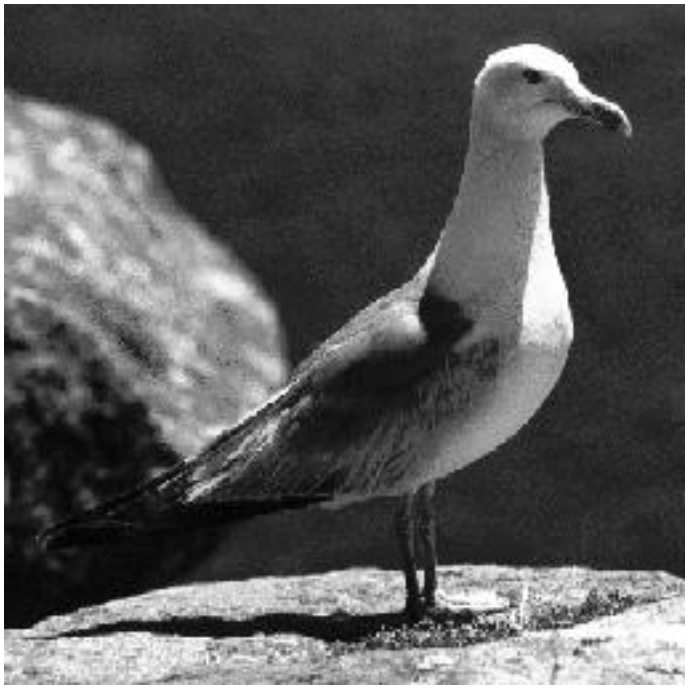
What Are the Right Features?
(When deep features are not applicable…)

- Depending on the task of interest!
- Possible choices
  - Object: shape
    - Local shape info, shading, shadows, texture
  - Scene : geometric layout
    - linear perspective, gradients, line segments
  - Material properties: albedo, feel, hardness
    - Color, texture
  - Action: motion
    - Optical flow, tracked points

Image Representation: Histograms

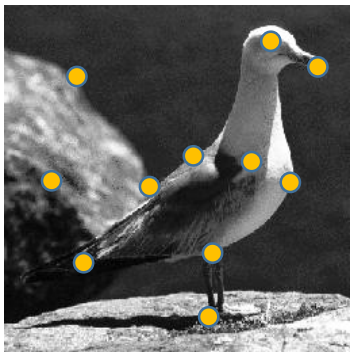
- Global histogram
  - Possible to describe color, texture, depth, or even interest points!





Images from Dave Kauchak

Image Representation: Histograms



- Take images with 2D features/descriptors as an example

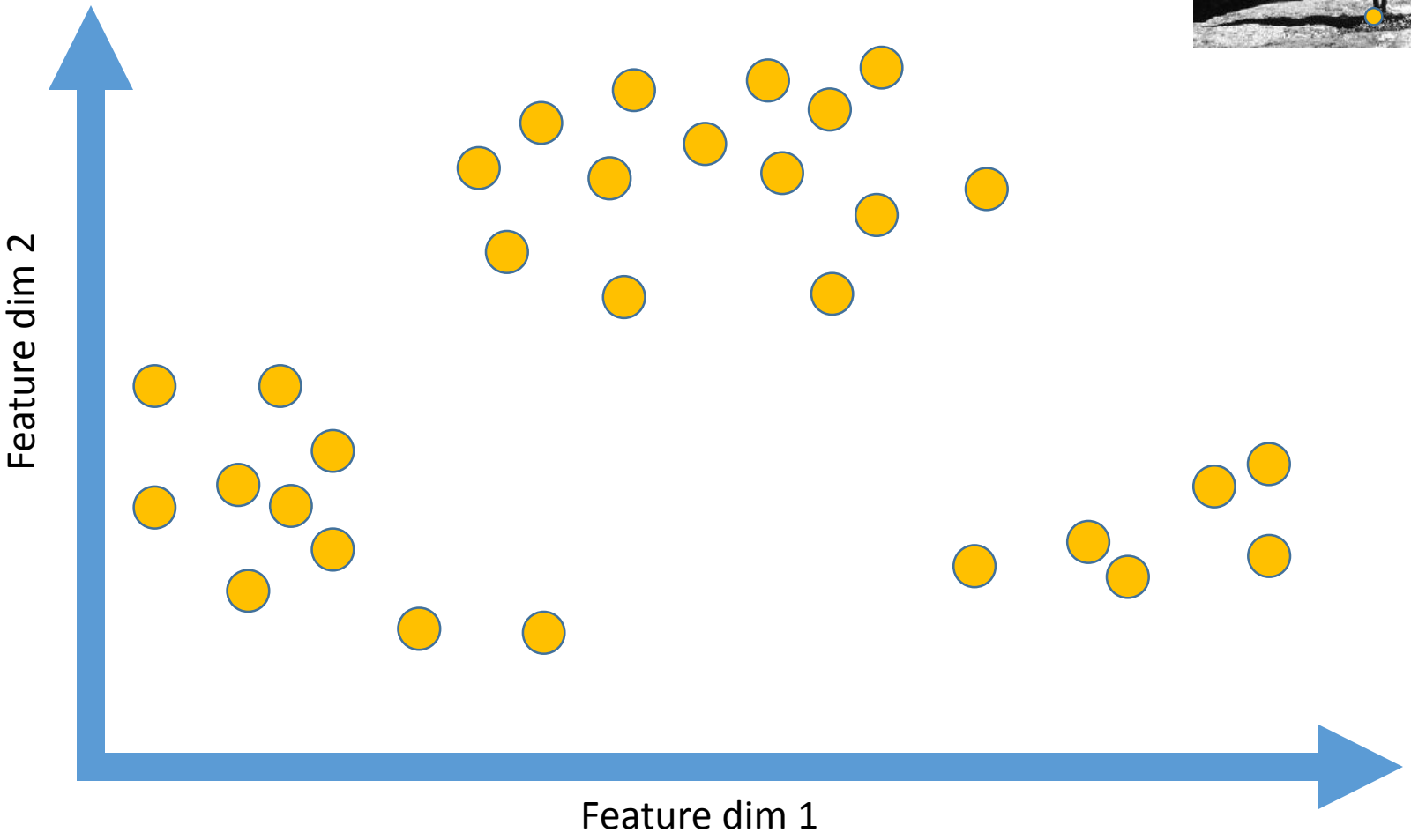


Image Representation: Histograms

- # of occurrence of data in each bin
- Marginal histogram of feature 1

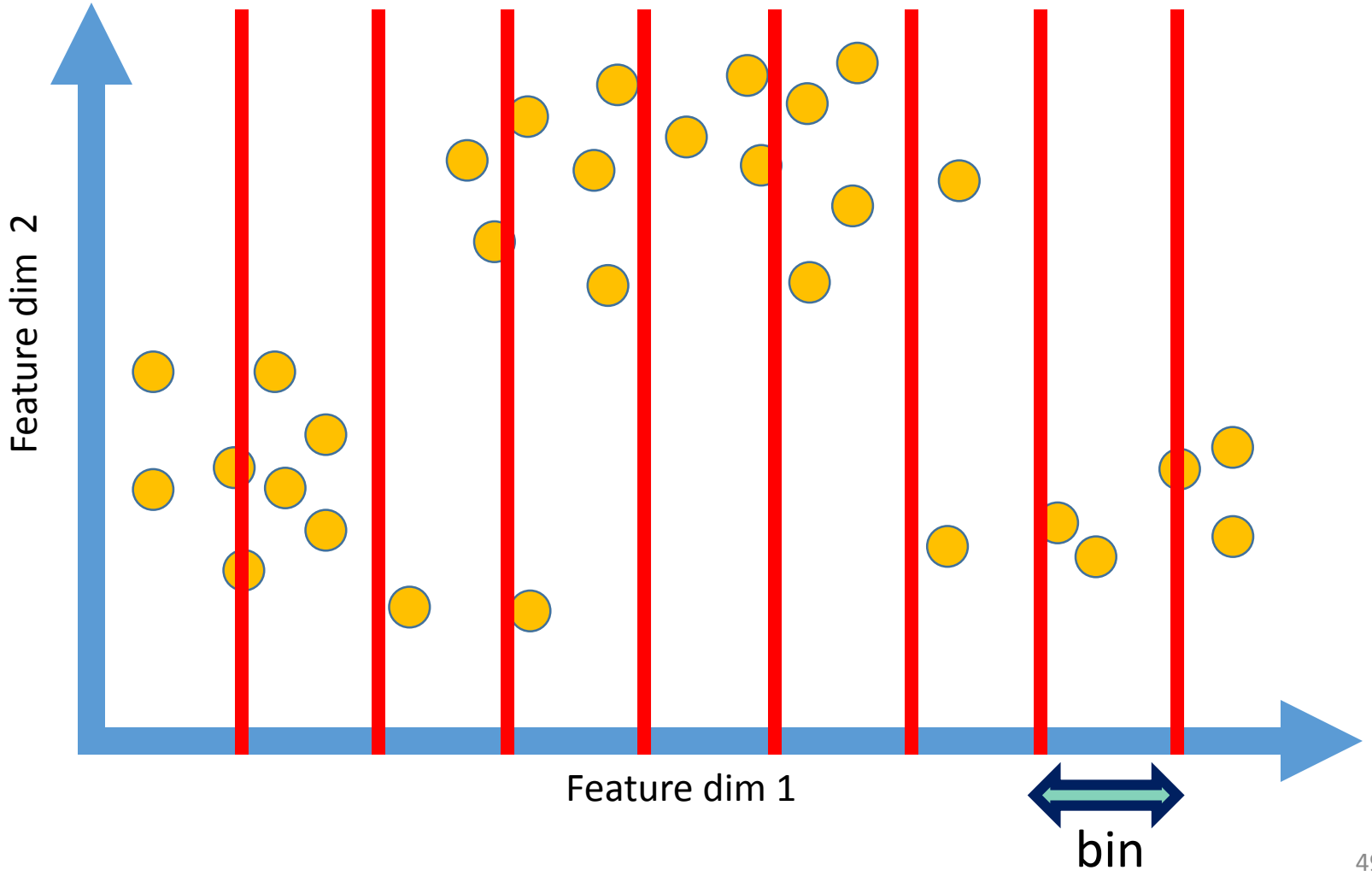


Image Representation: Histograms

- # of occurrence of data in each bin
- Marginal histogram of feature dim #2

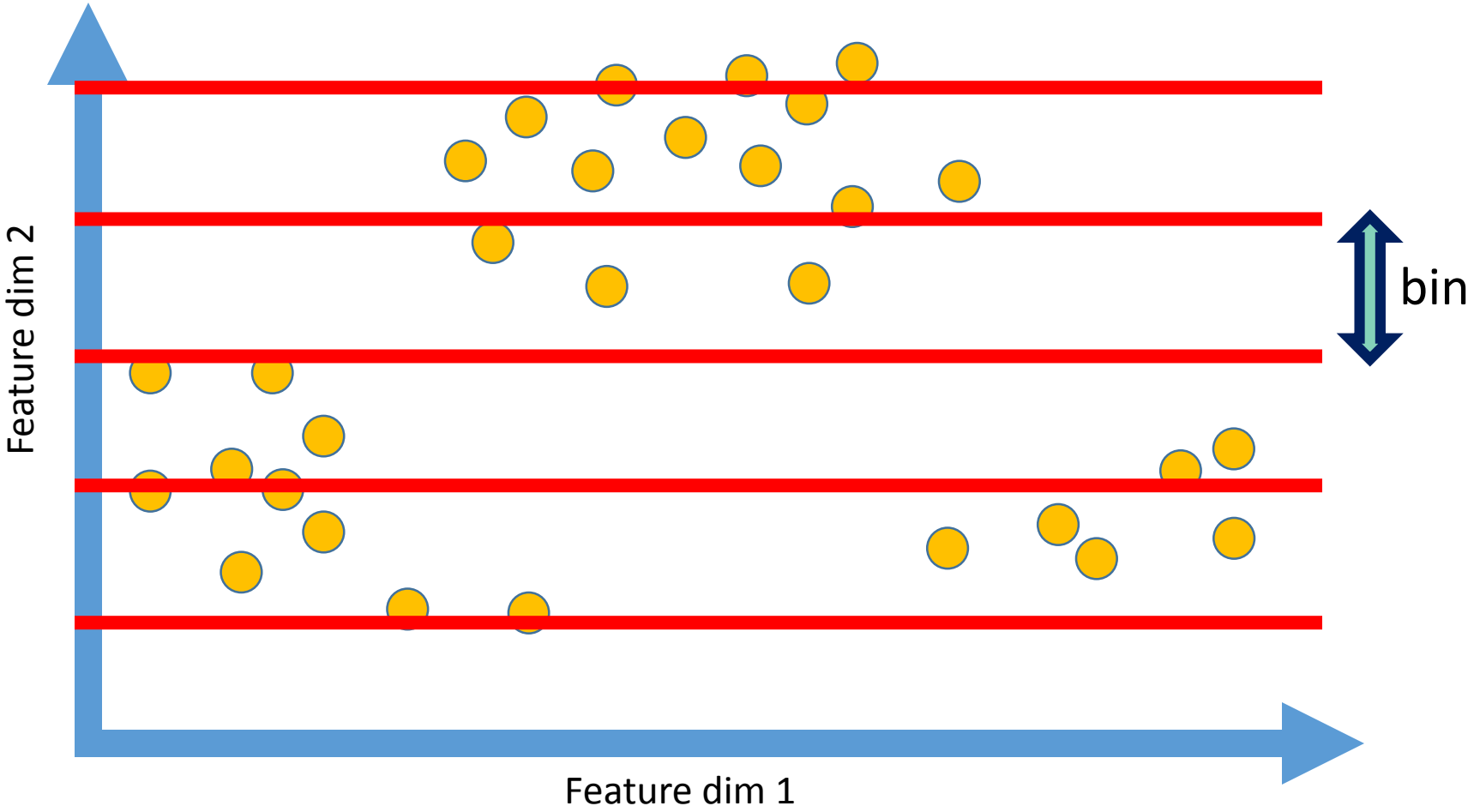


Image Representation: Histograms

- Better modeling (quantization) of multi-dimensional data
- Clustering

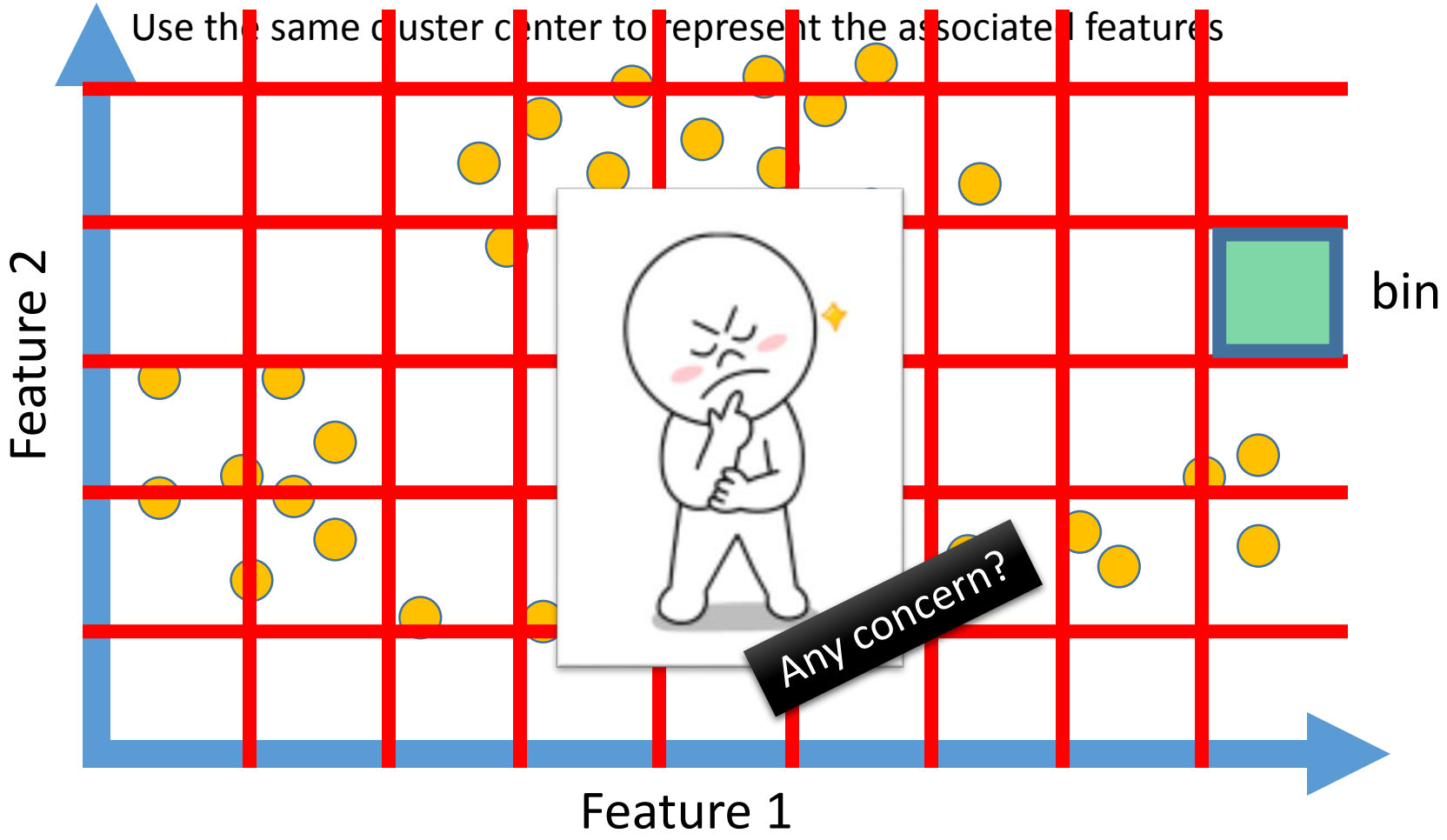
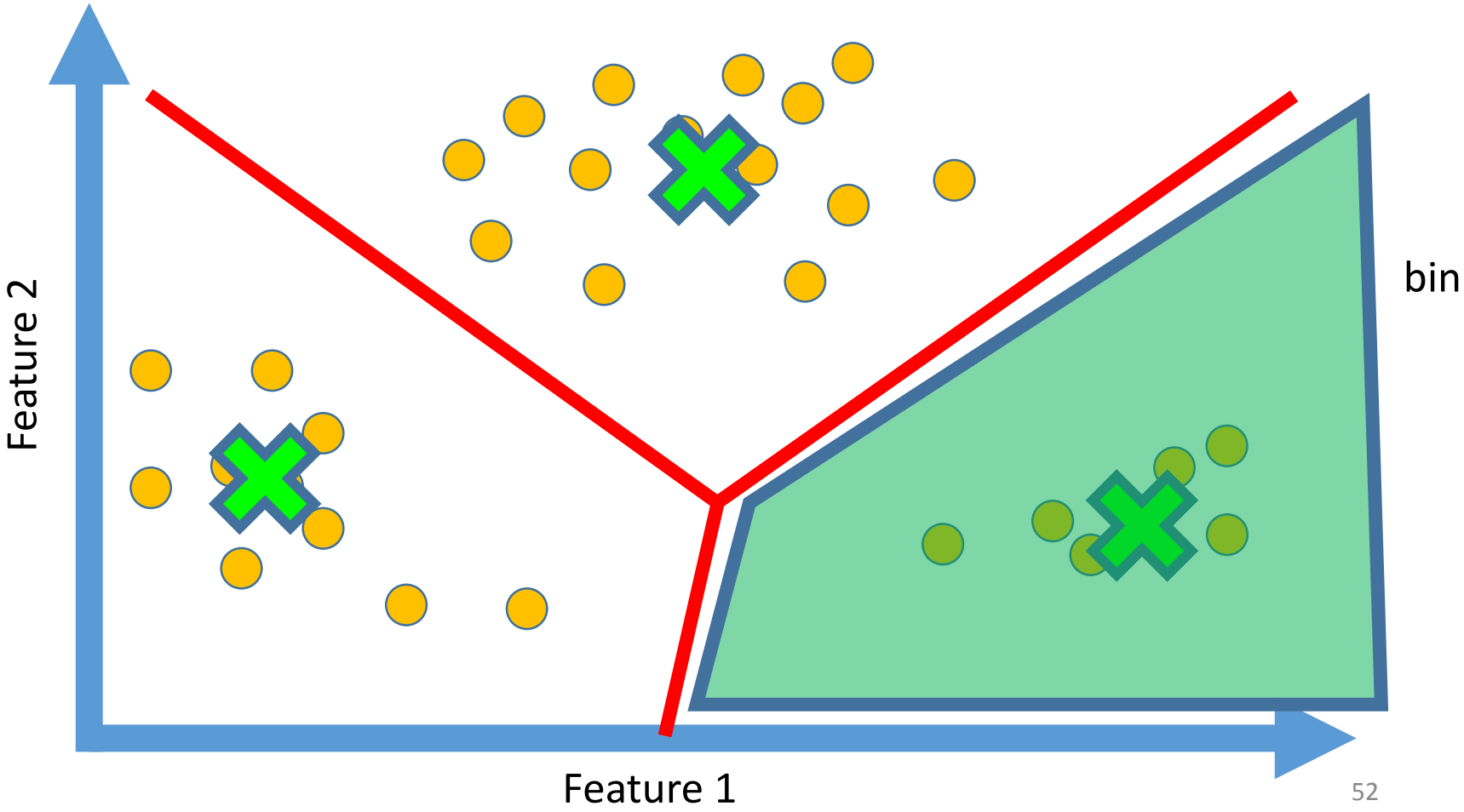


Image Representation: Histograms

- Better modeling (quantization) of multi-dimensional data
- Clustering
  - Use the same cluster center to represent the associated features



Remarks on Histogram-Based Image Representation

- Quantization
  - Grids vs. clusters

Fewer Bins
Need less data
Coarser representation

More Bins
Need more data
Finer representation

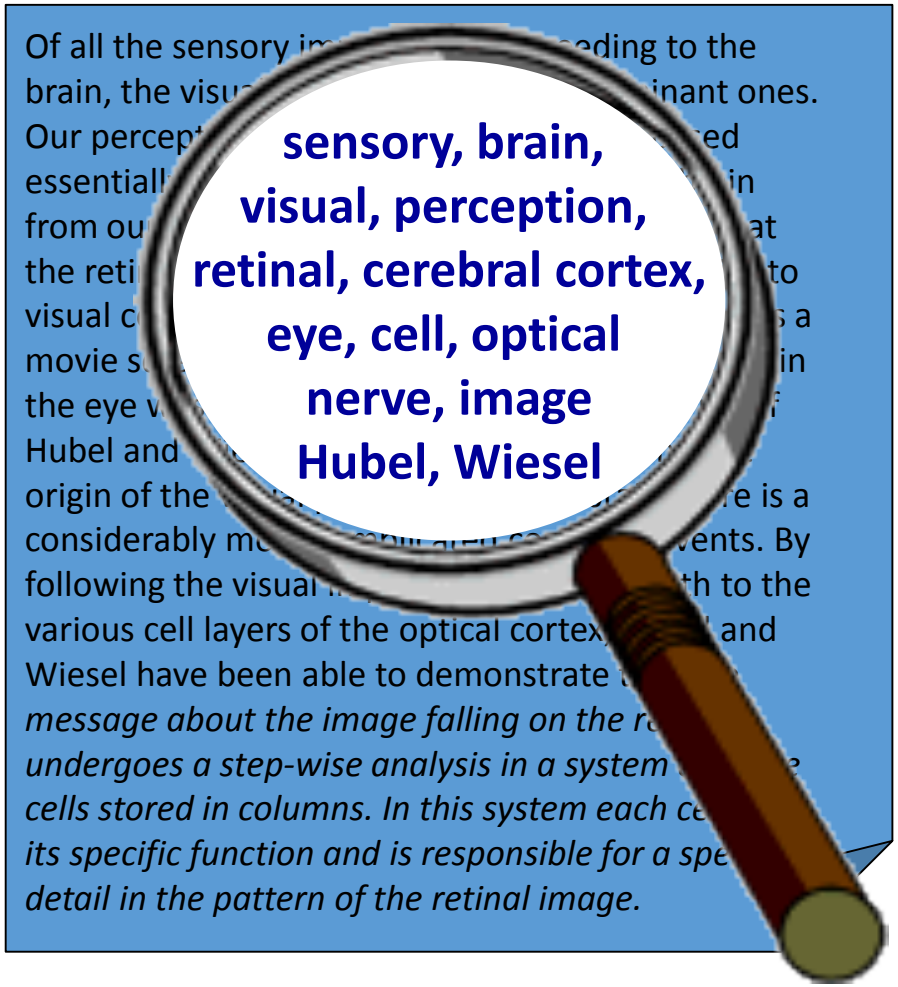
- Popular distance metrics
  - Euclidean distance
  - Histogram intersection kernel
  - Chi-squared distance
  - Earth mover's distance
    (min cost to transform one distribution to another)

$$\text{histint}(h_i, h_j) = 1 - \sum_{m=1}^{K} \min\left(h_i(m), h_j(m)\right)$$

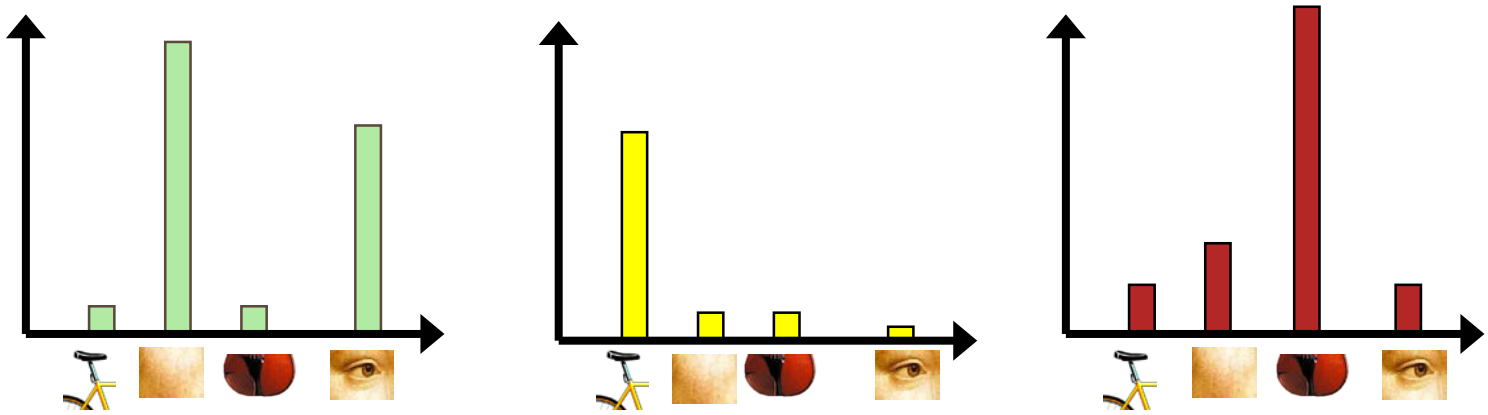
$$\chi^2(h_i, h_j) = \frac{1}{2} \sum_{m=1}^{K} \frac{[h_i(m) - h_j(m)]^2}{h_i(m) + h_j(m)}$$

Bag-of-Words Models for Image Classification

- Analogy to document categorization



Bag of Words (or Visual Words)

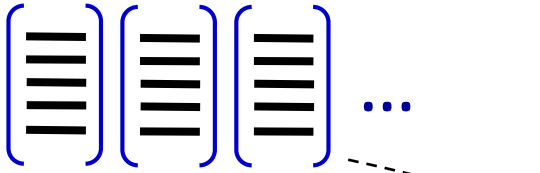


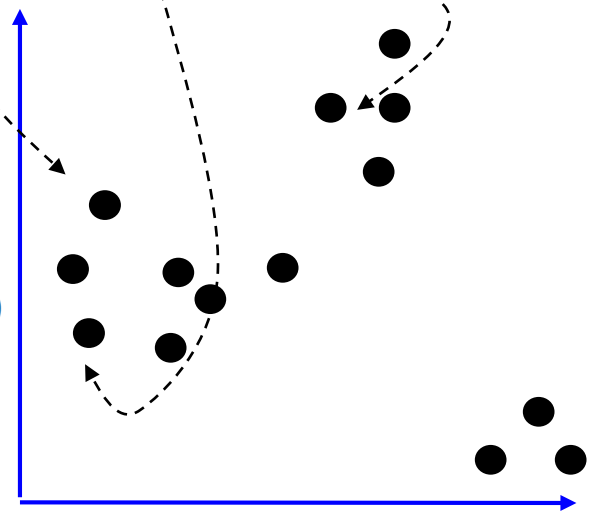
Bag-of-Words for Image Classification

- Training

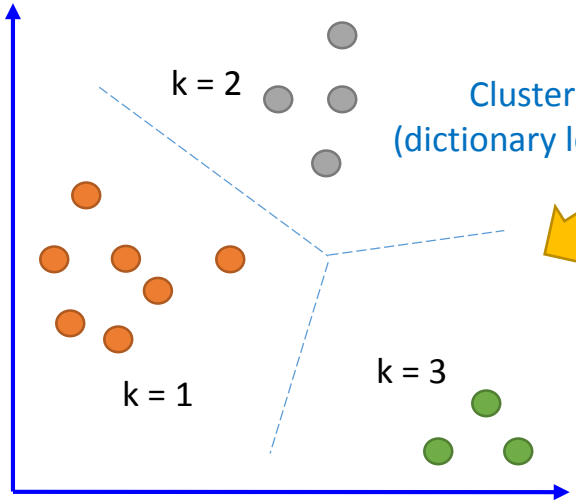
Training images

Interest point detection



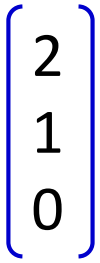


Clustering (dictionary learning)



Quantization (w/ normalization)

Feature Encoding

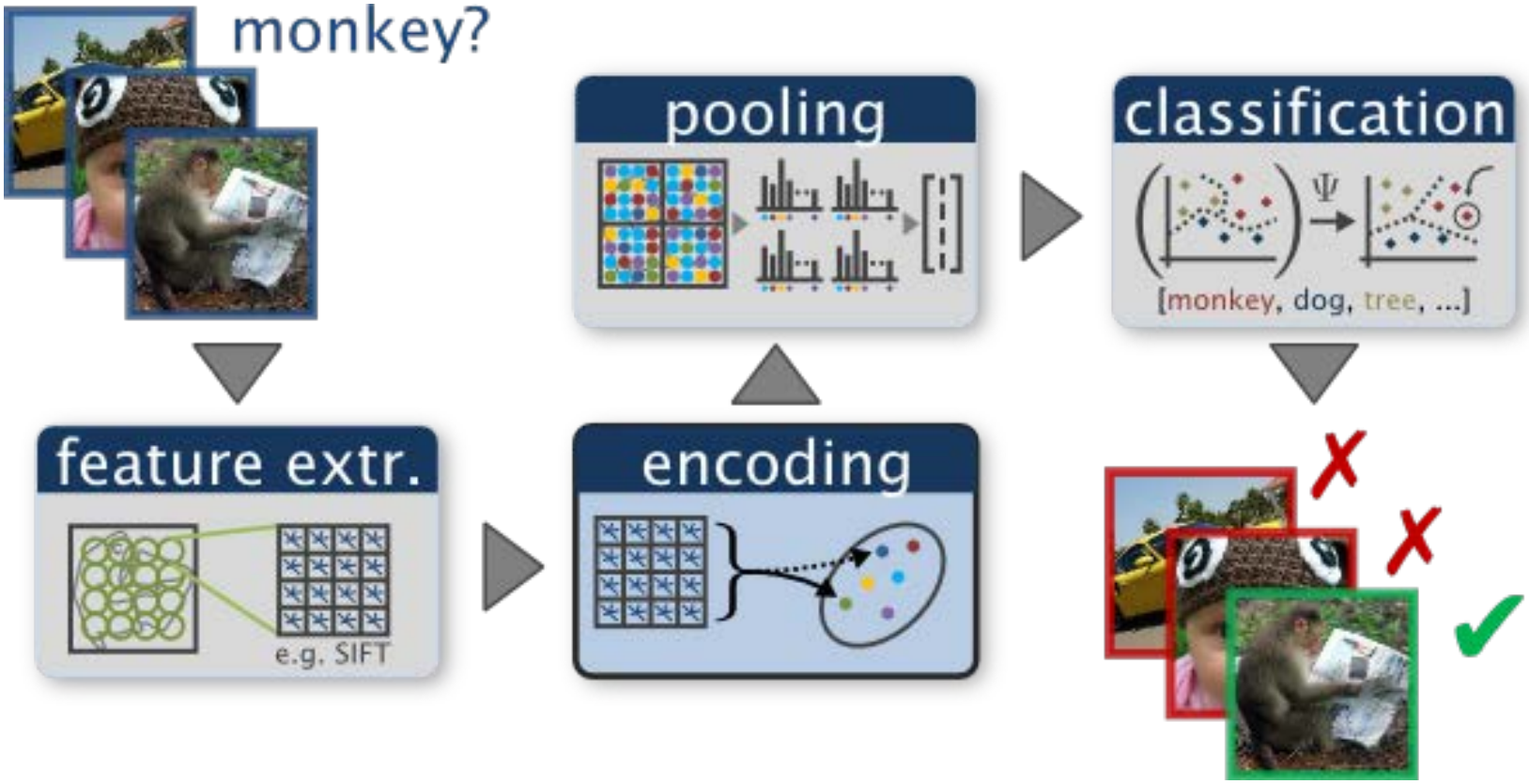


Bag-of-Words for Image Classification

- Testing

Bag-of-Words for Image Classification

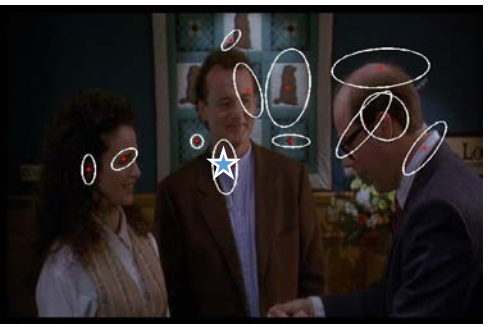
- Overview

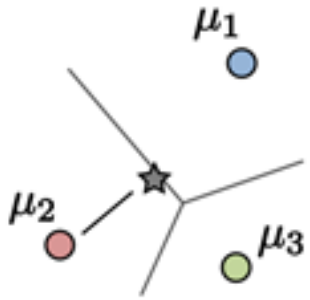


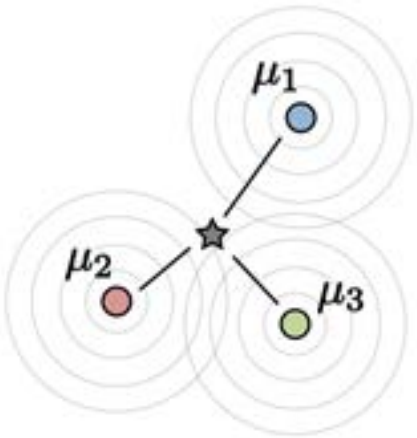
[Chatfieldet al. BMVC 2011]

About Feature Encoding for Bag-of-Words

- Hard vs. soft assignments to clusters

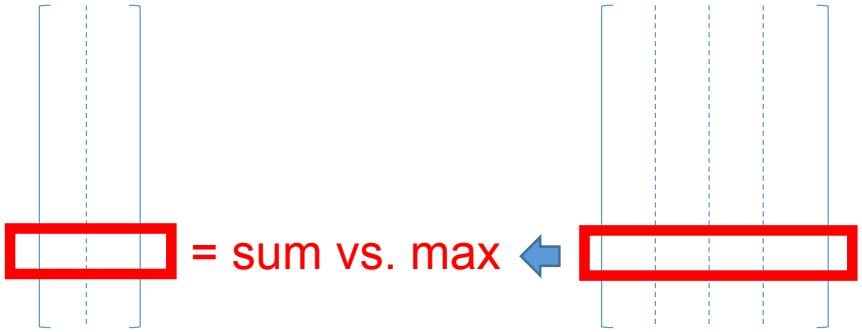


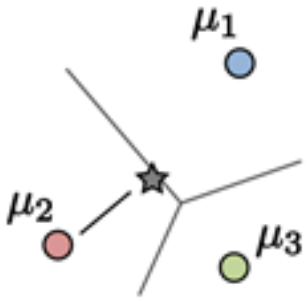


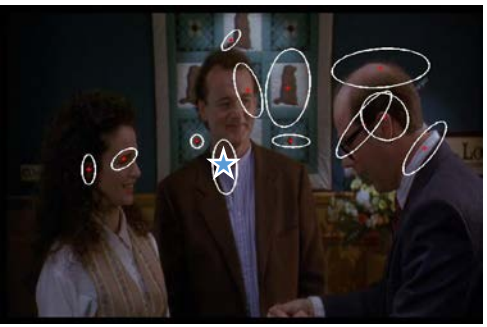


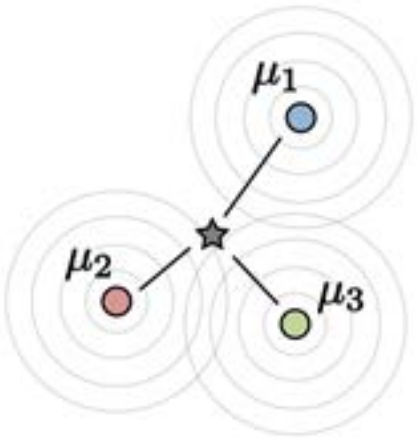
About Feature Encoding for Bag-of-Words

- Sum vs. max pooling



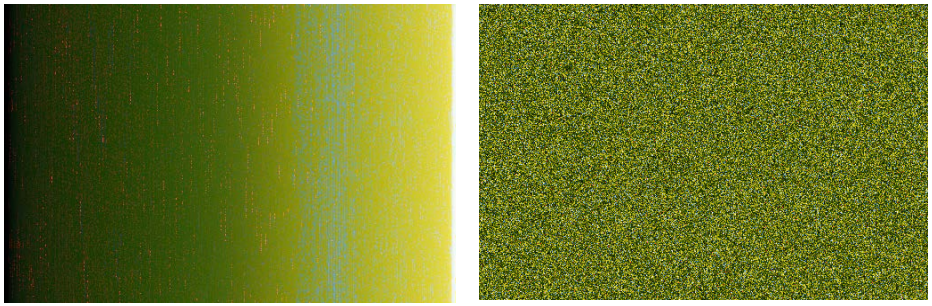


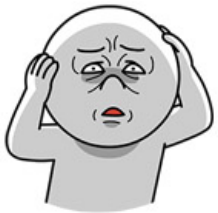


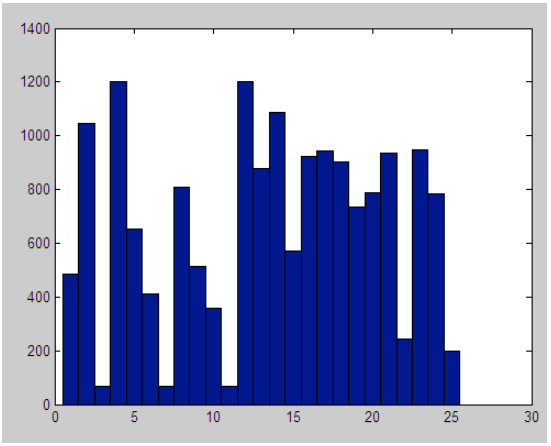


Final Remarks on BoW

- What's the limitation?
  - Loss of…
  - What's the possible solution?

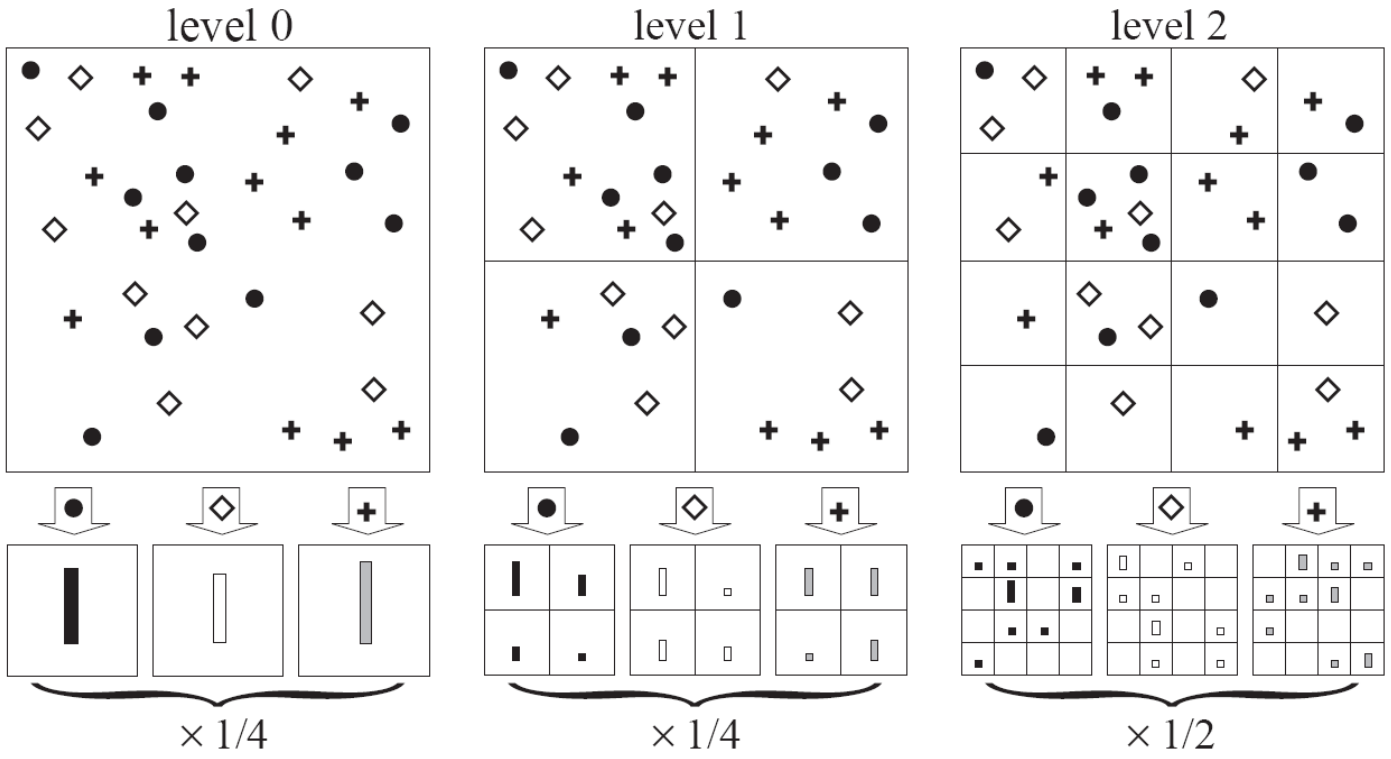






Final Remarks on BoW

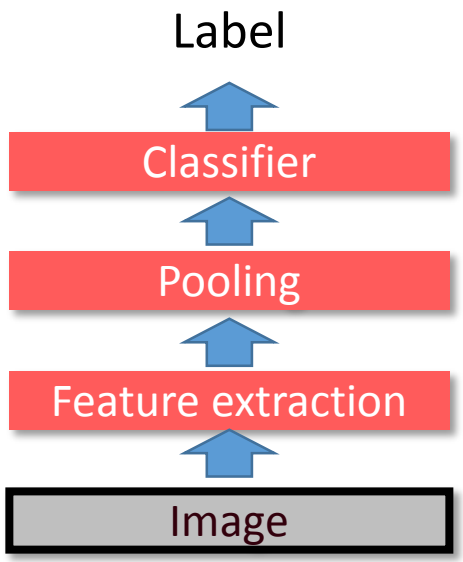
- Spatial pyramid
  - Compute BoW in each spatial grid + concatenation

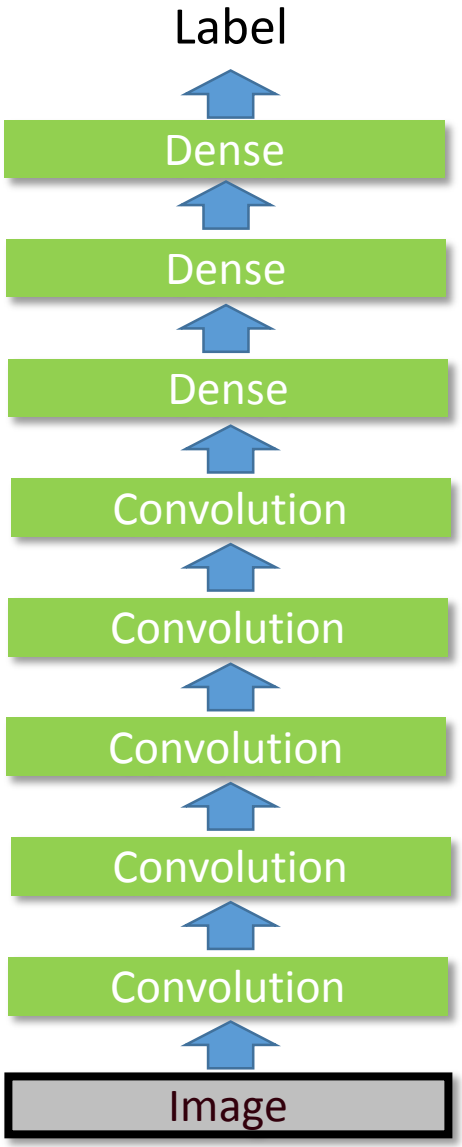


[Lazebnik et al. CVPR 2006]

Shallow vs. Deep Learning for Image Classification

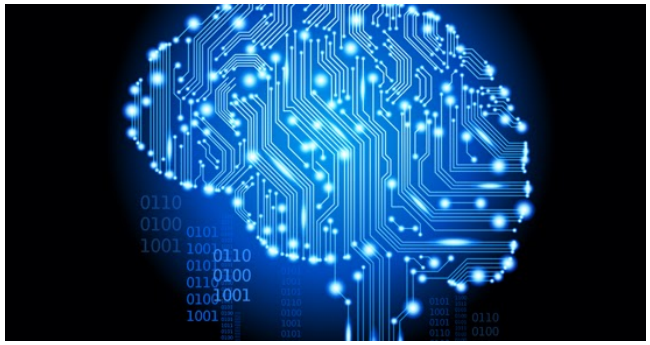
- Engineered vs. deeply learned features
  - A sufficient amount of training data
  - GPUs (optional ☺)



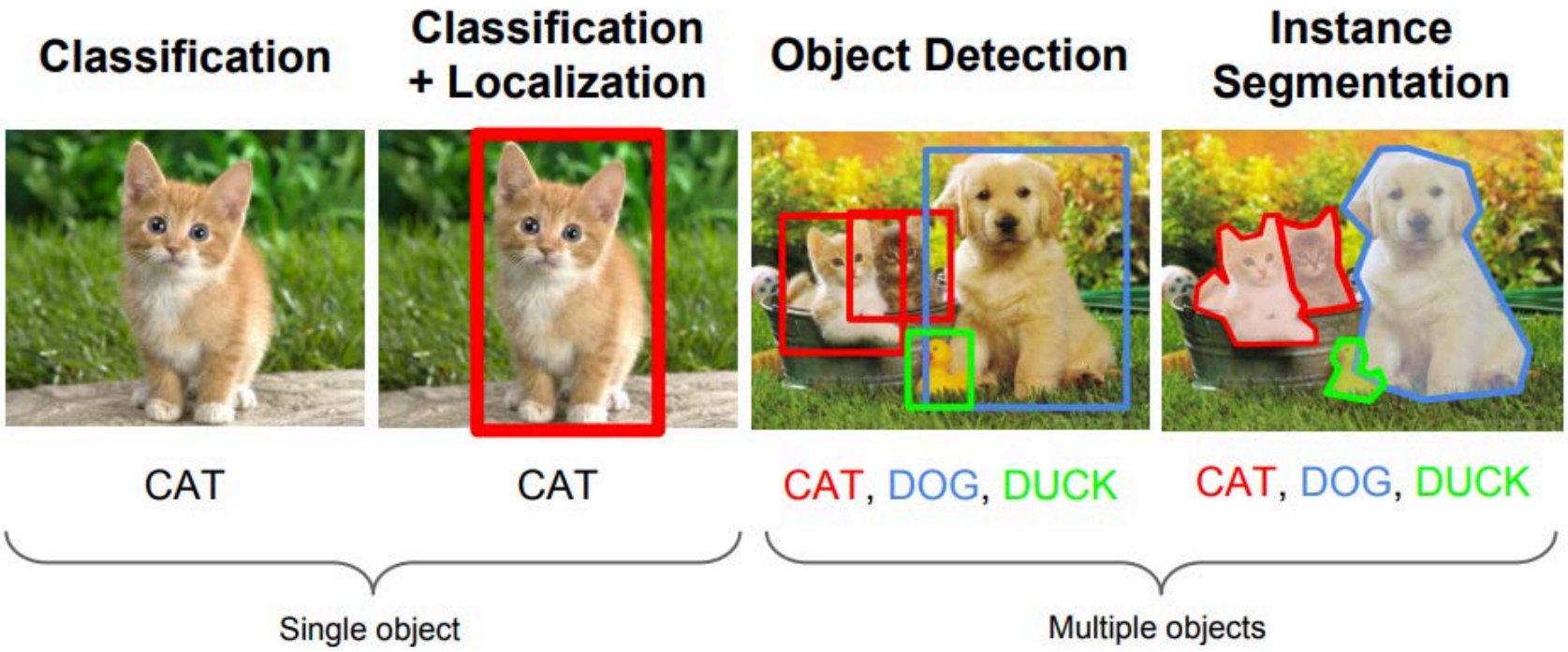


What's to Be Covered Today…

- Neural Networks & CNN
  - Convolutional Neural Networks
- Recognition & Detection
  - Recognition: From Interest Points to Bag-of-Words Models
  - Object Detection



Roadmap

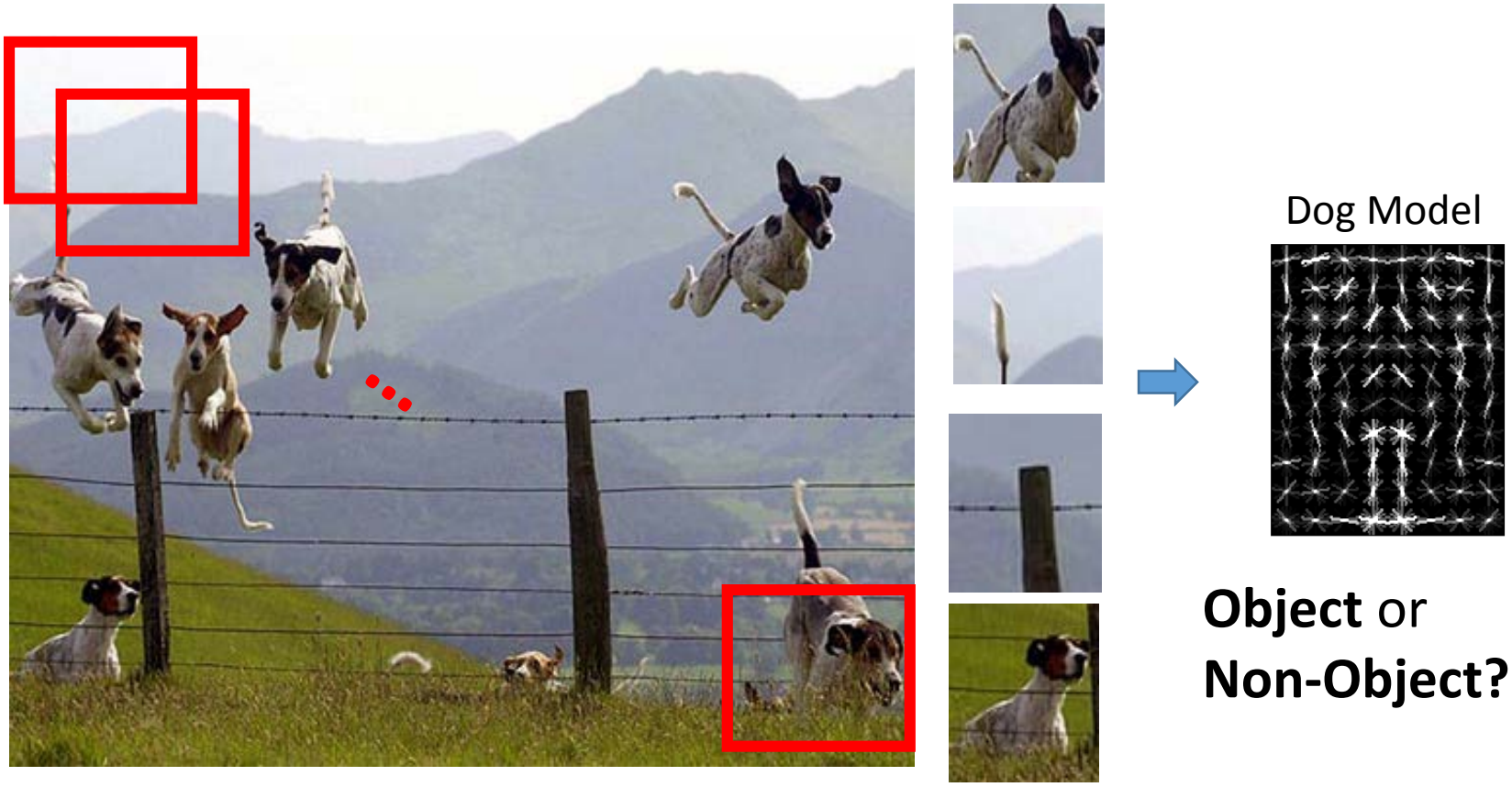


Slide from A. Karpathy

Demo

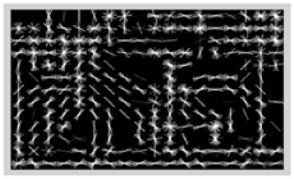
Object Category Detection

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

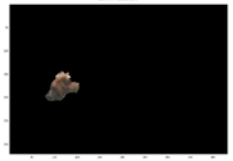


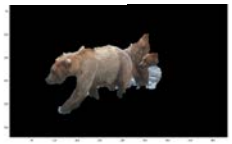
General Process of Object Recognition

Specify Object Model



Generate Hypotheses





Score Hypotheses

Gradient based or CNN features, usually based on summary representation with classification/voting results

Resolve Detections

Rescore each proposed object based on the entire candidate set

Challenges in Modeling the Object Classes

Illumination

Object pose

Clutter

Occlusion

Intra-class appearance

Viewpoint

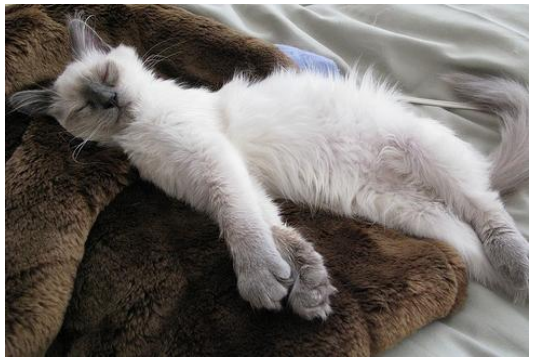
Slide from K. Grauman, B. Leibe

Challenges in Modeling the Non-object Classes

True Detection

Bad Localization

Confused with Similar Object



Misc. Background

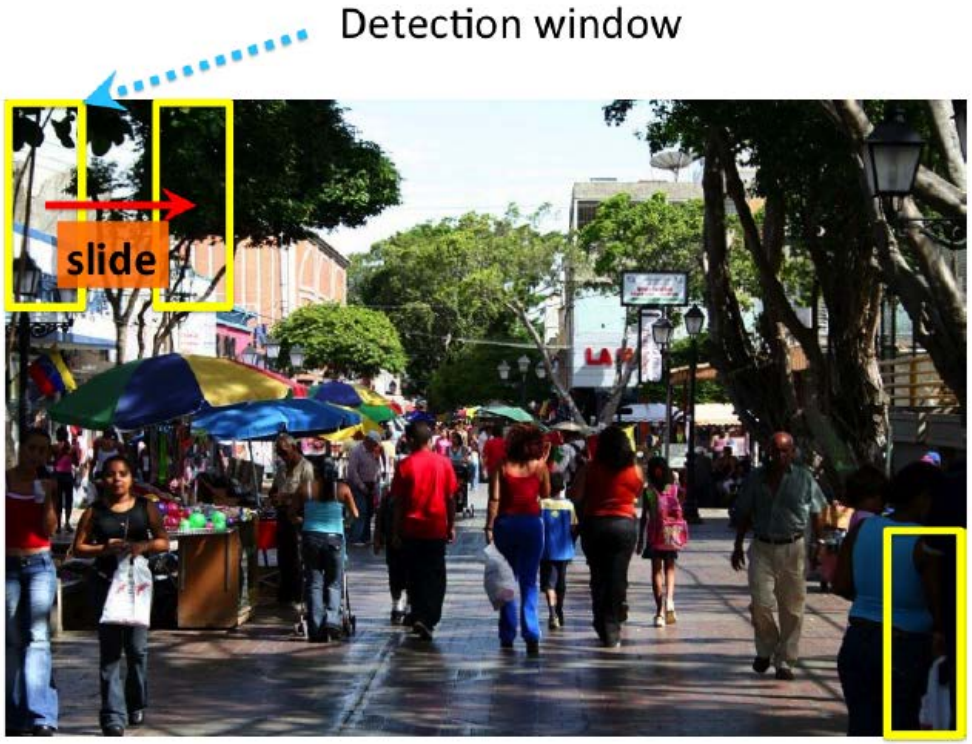
Confused with Dissimilar Objects

Type of Approaches

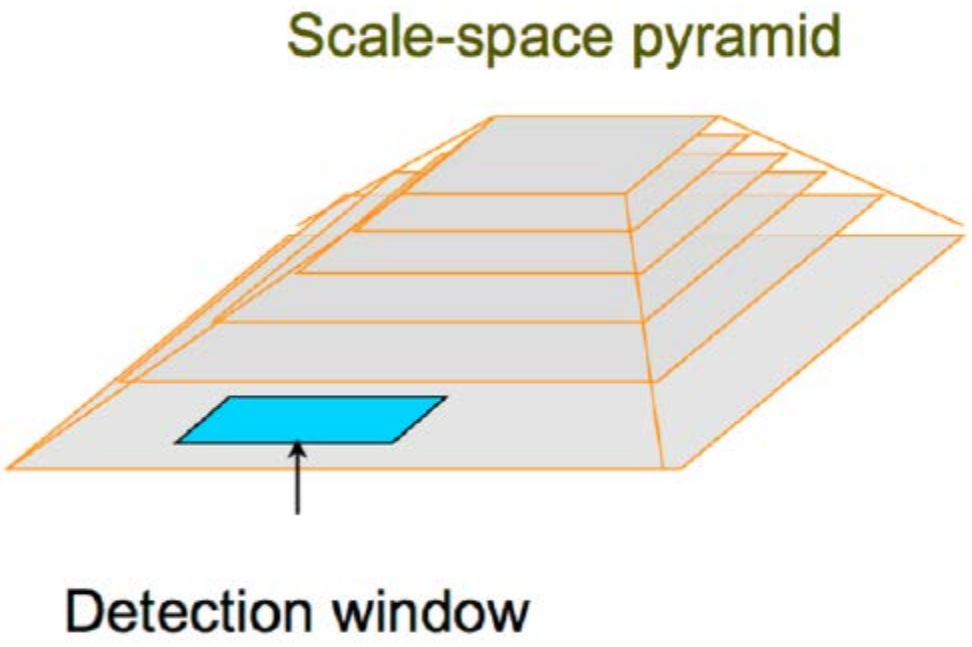
- Sliding Windows
  - "Slide" a box around image
  - Classify each cropped image inside the box and determine it's an object or not
  - E.g., HOG (person) detector by Dalal and Triggs (2005)
    Deformable part-based model by Felzenswalb et al. (2010)
    Real-time (face) detector by Viola and Jones (2001)
- Region (Object) Proposals
  - Generate region (object) proposals
  - Classify each image region and determine it's an object or not

The HOG Detector

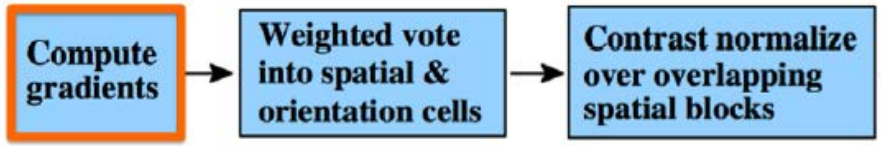
- Sliding window detector find objects in 4 steps:
  - Inspect every window
  - Extract features in window
  - Classify & accept window if score > threshold
  - Clean-up (post-processing) stage



- Step 1: Inspect every window
  - Objects can vary in sizes, what to do?
  - Sliding window + image pyramid!



- Step 2: Extract Features in Window
  - Histogram of Gradients (HOG) features
  - Similar to SIFT in some ways…



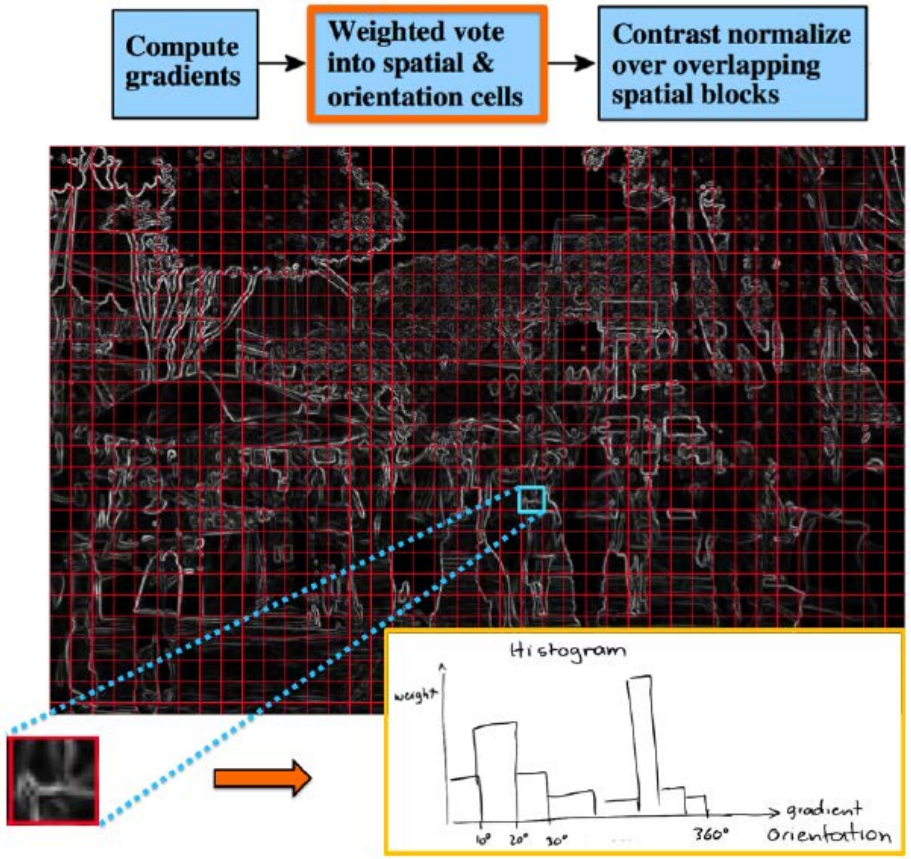
- Step 2: Extract Features in Window
  - Histogram of Gradients (HOG) features
  - Ways to compute image gradients…

Mask Type	1D centered	1D uncentered	1D cubic-corrected	2x2 diagonal	3x3 Sobel
Operator	[-1, 0, 1]	[-1, 1]	[1, -8, 0, 8, -1]	$\begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix}$ $\begin{bmatrix} -1 & 0 \\ 0 & 1 \end{bmatrix}$	$\begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}$ $\begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix}$
Miss rate at 10^{-4} FPPW	11%	12.5%	12%	12.5%	14%

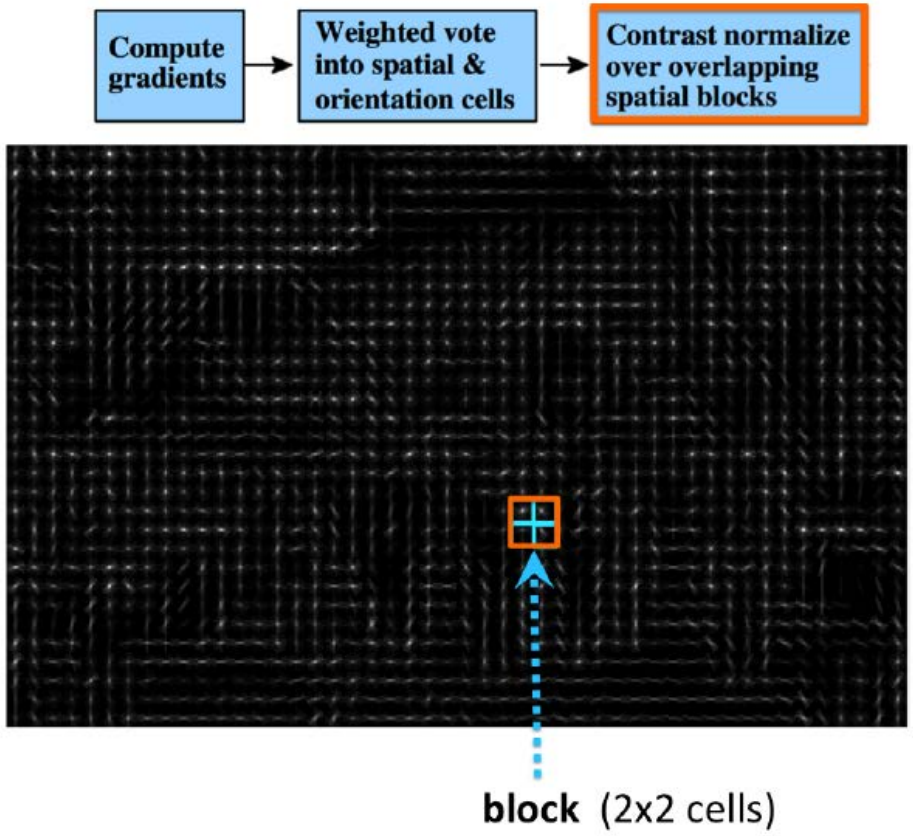
(Miss rate: smaller is better)

This gradient filter gives the best performance

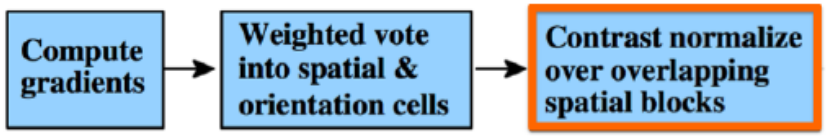
- Step 2: Extract Features in Window
  - Histogram of Gradients (HOG) features
  - Divide the image into non-overlapping cells (grids) of 8 x 8 pixels
  - Compute a histogram of orientations in each cell (similar to SIFT), resulting in a 9-dimensional feature vector.

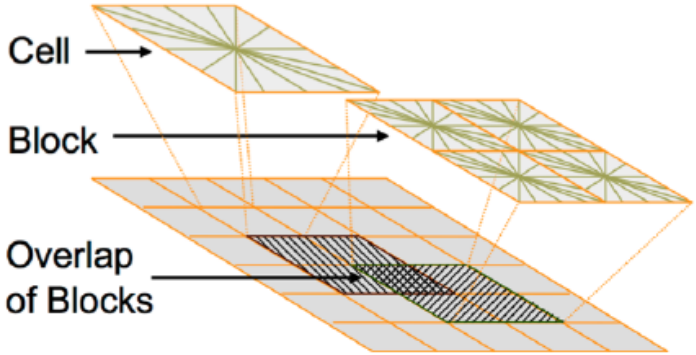


- Step 2: Extract Features in Window
  - Histogram of Gradients (HOG) features
  - Divide the image into non-overlapping cells (grids) of 8 x 8 pixels
  - Compute a histogram of orientations in each cell (similar to SIFT), resulting in a 9-dimensional feature vector.
  - We now take blocks, where each has 2 x 2 cells.



- Step 2: Extract Features in Window
  - Compute a histogram of orientations in each cell (similar to SIFT), resulting in a 9-dimensional feature vector.
  - We now take blocks, where each has 2 x 2 cells.
  - Normalize each feature vector, such that each block has unit norm. This does not change the dim of the feature, just the magnitude.

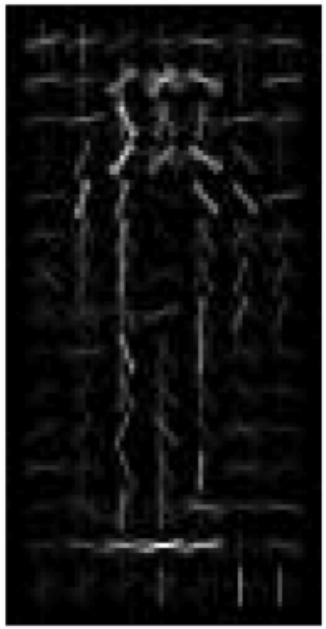




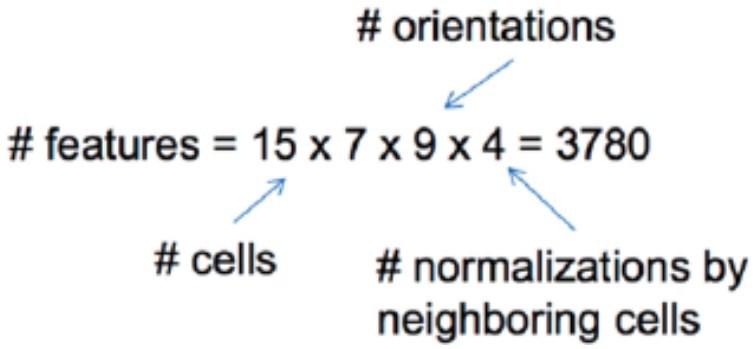
Feature vector f = [..., ..., ...]

L2 normalization in each block: $$\mathbf{f} = \frac{\mathbf{f}}{\sqrt{||\mathbf{f}||_2^2 + \epsilon^2}}$$

- Step 2: Extract Features in Window
  - Normalize each feature vector, such that each block has unit norm. This does not change the dim of the feature, just the magnitude.
  - Since each cell is in 4 blocks, we have 4 different normalizations, and we make each one into separate features.
  - For each person class, window is 15 x 7 HOG cells.
  - We vectorize each the feature matrix in each window.



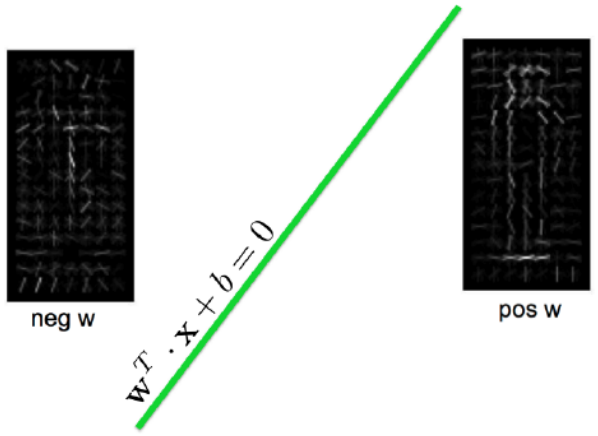
Final descriptor for window
(**person class in this case**)



- Step 3: Detection (Classify & accept window if score > threshold)
  - Train a window classifier
  - Use the trained classifier to predict presence of object class in each window

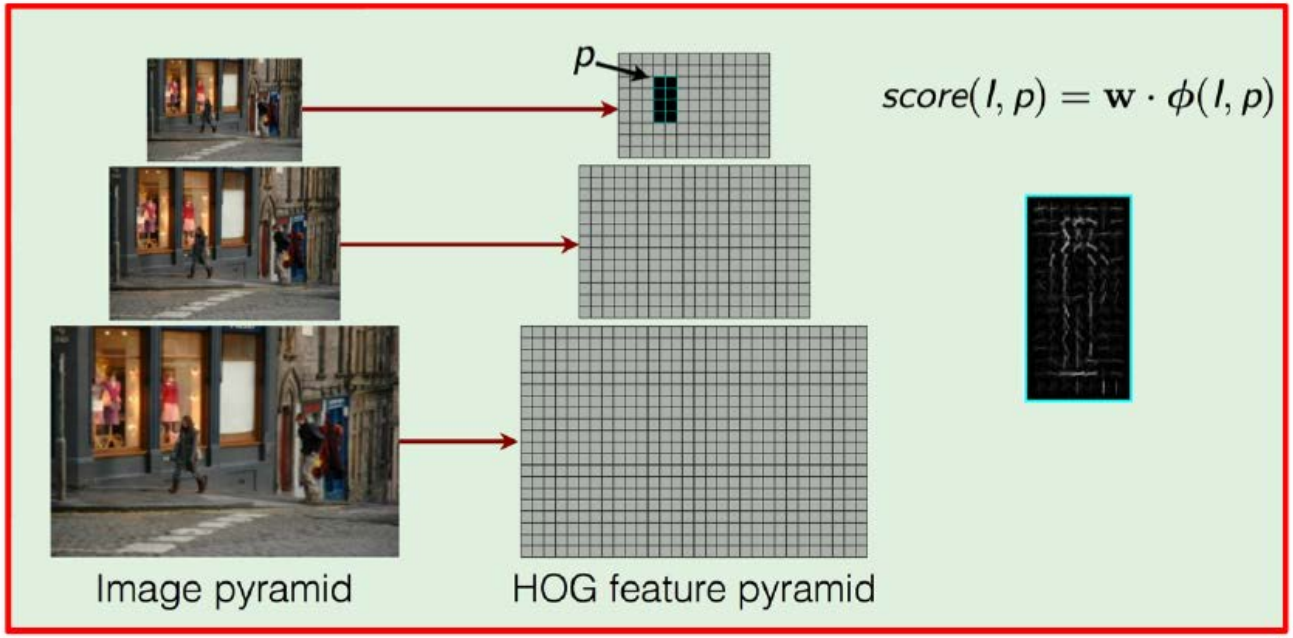
positive training examples

negative training examples

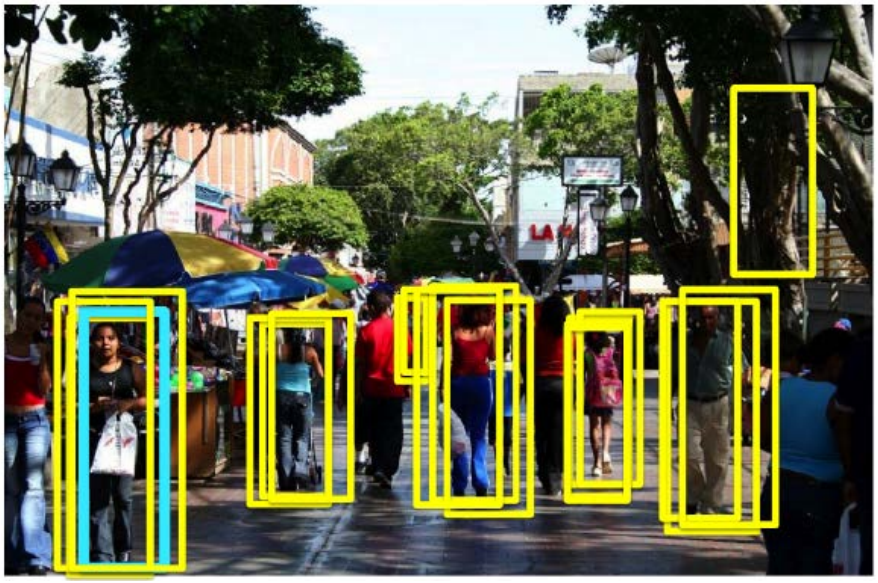


Train classifier. SVM (Support Vector Machines) is typically used.

- Step 3: Detection (Classify & accept window if score > threshold)
  - Train a window classifier
  - Use the trained classifier to predict presence of object class in each window
  - During testing, compute the score $\mathbf{w}^T\mathbf{x}+\mathbf{b}$ in each location, which can be viewed as performing cross-correlation (or convolution) with template $\mathbf{w}$ (and add bias $\mathbf{b}$).



- Step 4: Cleaning-Up
  - Perform a greedy algorithm of non-maxima suppression (NMS) to pick the bounding box with highest score



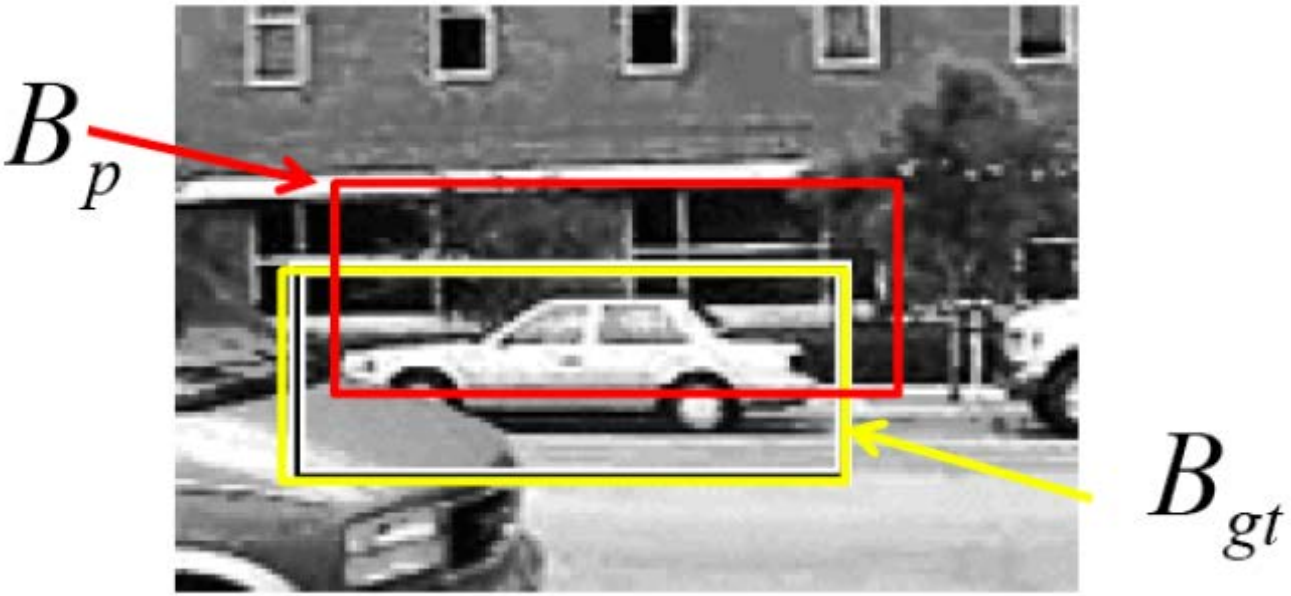
Non-maxima suppression (NMS)

$$\text{overlap} = \frac{\text{area}(box_1 \cup box_2)}{\text{area}(box_1 \cap box_2)} > 0.5$$ ⇒ remove box_2

- Remove all boxes that overlap more than XX (typically 50%) with the chosen box

- Evaluation
  - IOU (intersection over union)
  - E.g, detection is correct if IOU between bounding box and ground truth > 50%

$$a_0 = \frac{\text{area}(B_p \cap B_{gt})}{\text{area}(B_p \cup B_{gt})}$$



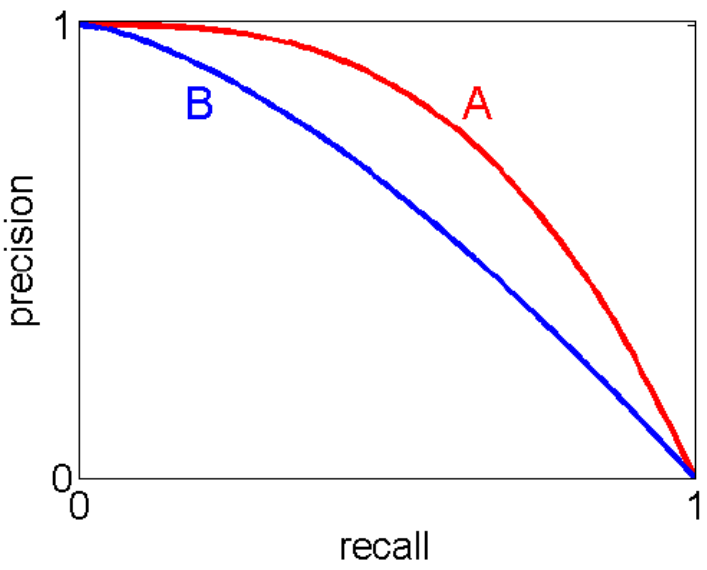
- Evaluation
  - IOU (intersection over union)
  - Precision and Recall
    - Sort all the predicted boxes according to scores, in a descending order
    - For each location in the sorted list,
      we compute precision and recall obtained when using top k boxes in the list.

$$\text{recall} = \frac{\#\text{correct boxes}}{\#\text{ground-truth boxes}}$$

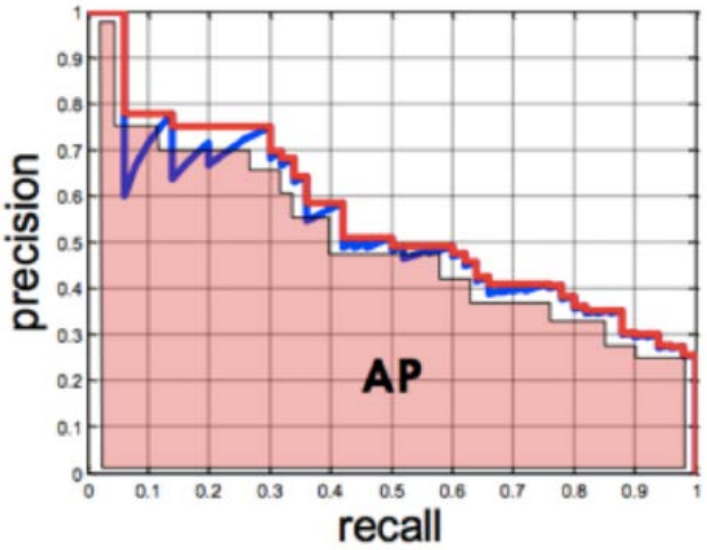
$$\text{precision} = \frac{\#\text{correct boxes}}{\#\text{all predicted boxes}}$$

$$\text{Precision} = \frac{True\ Positive}{True\ Positive + False\ Positive}$$

$$\text{Recall} = \frac{True\ Positive}{True\ Positive + False\ Negative}$$



- Evaluation
  - IOU (intersection over union)
  - Precision and Recall
  - Average Precision (AP):
    - Compute the area under P-R curve
    - Standard measure for detection evaluation
    - mean Average Precision (mAP): average of AP across classes



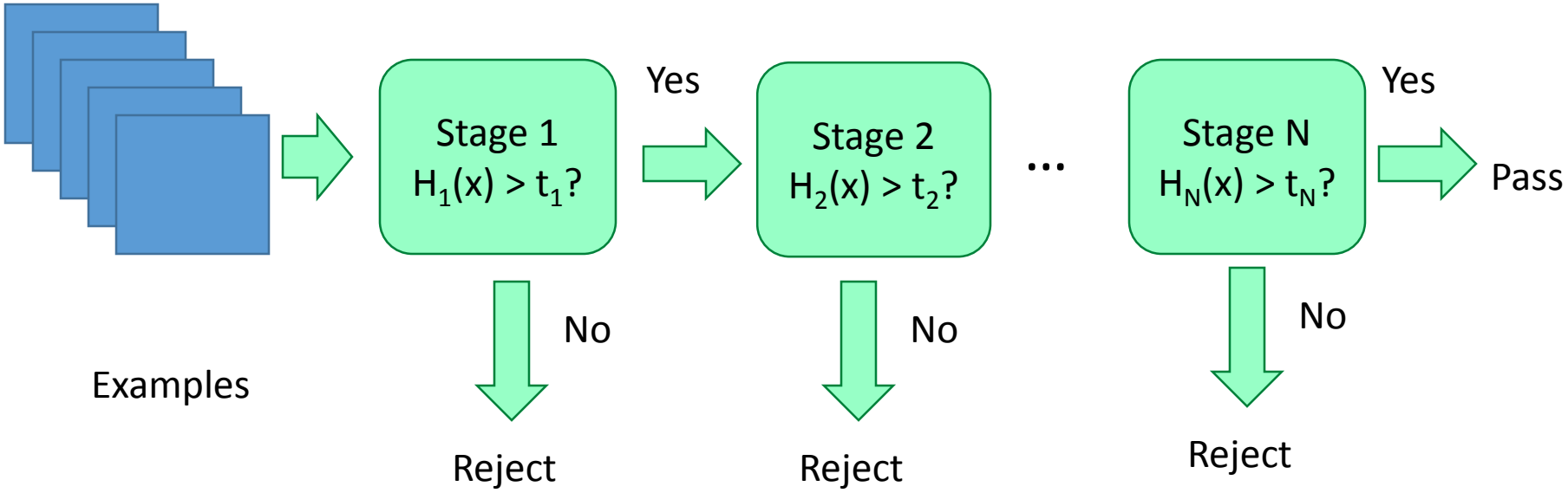
Viola-Jones Sliding Window Detector

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).

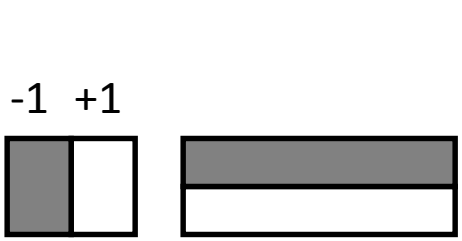
Cascade for Fast Detection



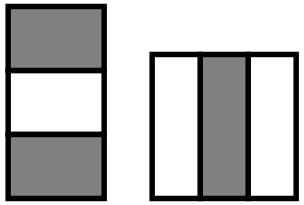
- Choose threshold for low false negative rate
- Fast classifiers early in cascade
- Slow classifiers later, but most examples don't get there

Features that are Fast to Compute

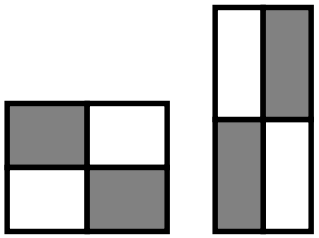
- “Haar-like features”
  - Differences of sums of intensity
  - Thousands, computed at various positions and scales within detection window



Two-rectangle features



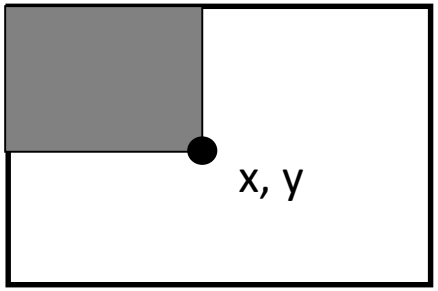
Three-rectangle features



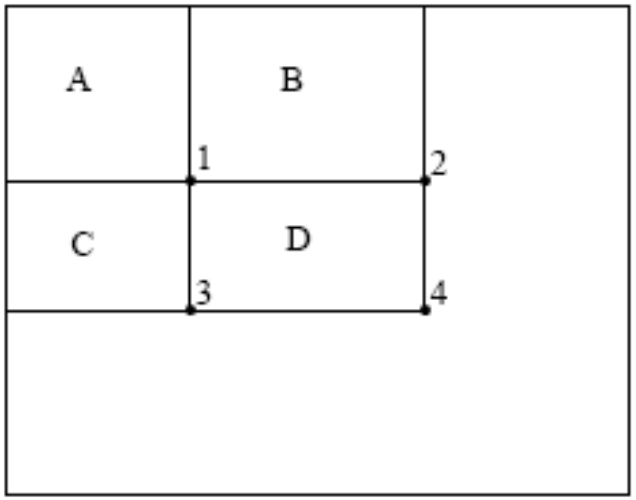
Etc.

Integral Images

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



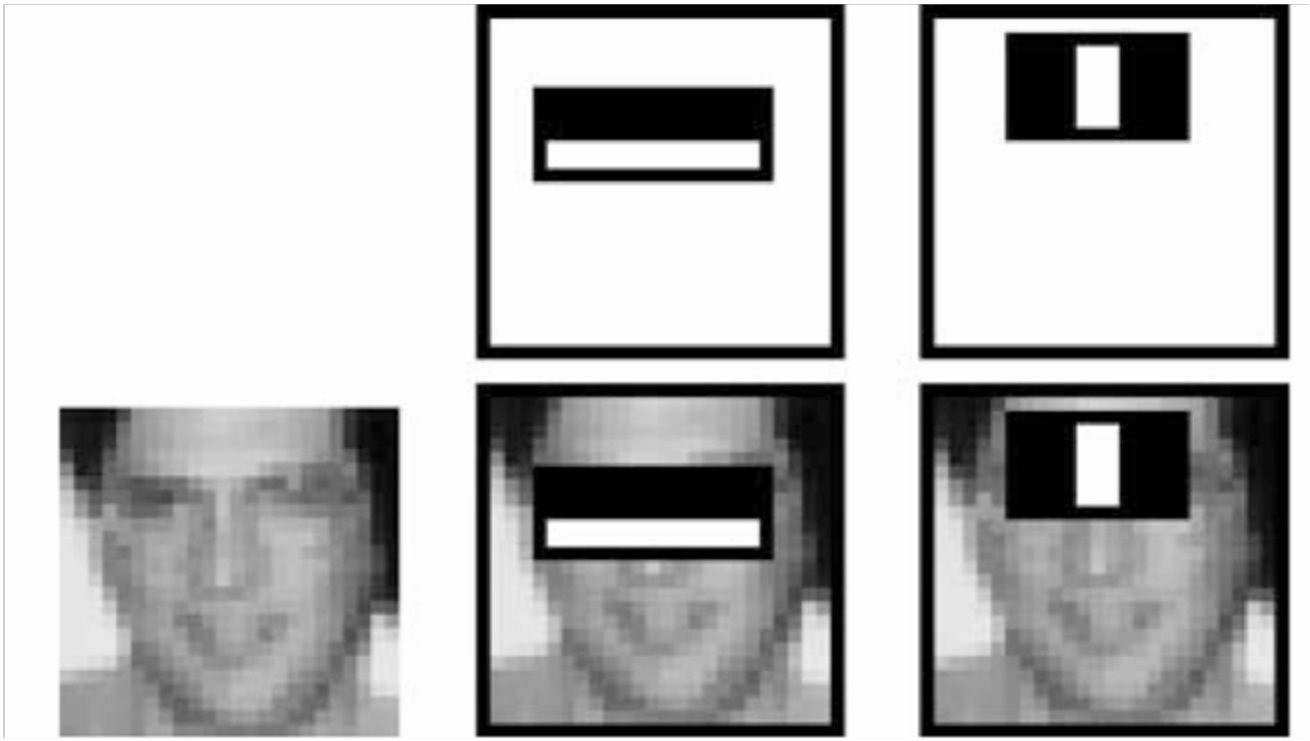
ii(x,y) = Sum of the values in the grey region



How to compute B-A?

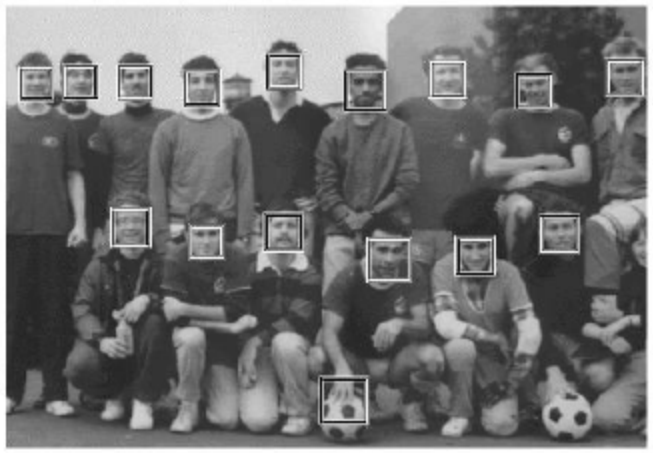
How to compute A+D-B-C?

Top 2 Selected Features for Face Detection



Viola Jones Results

Speed = 15 FPS (in 2001)


False detections / Detector	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

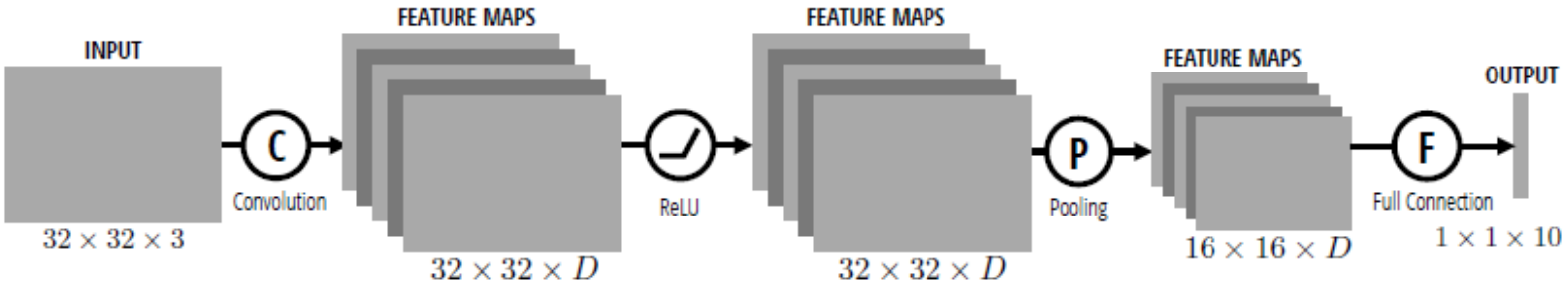
Something to Think About…

- 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?

Recall that

- Convolutional Neural Networks

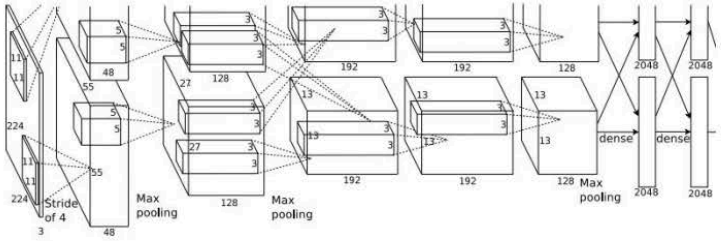


CNN as Feature Extractor

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

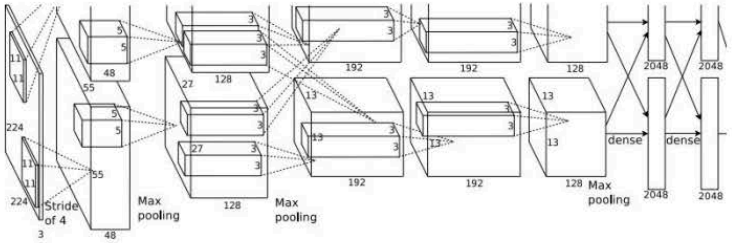


Dog? NO
Cat? NO
Background? YES

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

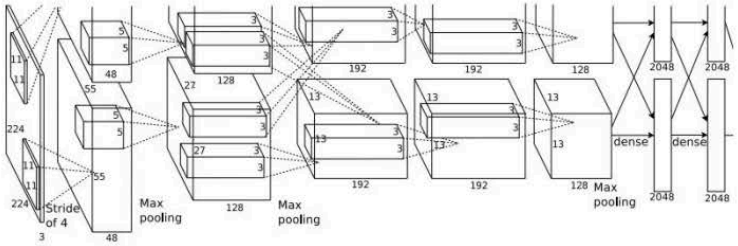


Dog? YES
Cat? NO
Background? NO

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

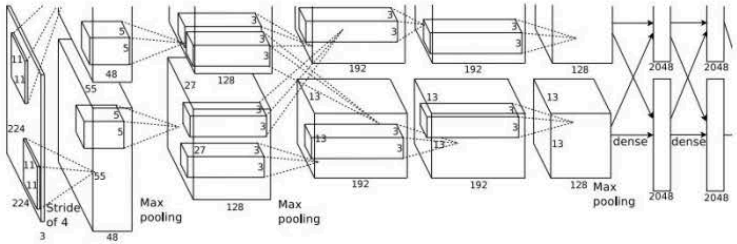


Dog? YES
Cat? NO
Background? NO

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? NO
Cat? YES
Background? NO

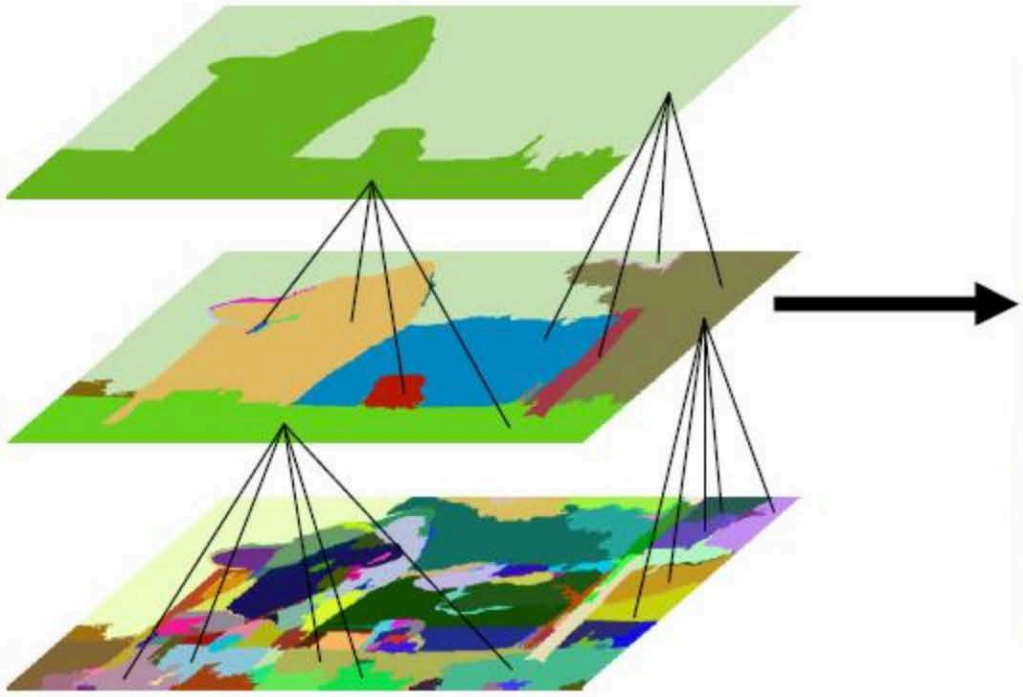
Slides by Justin Johnson

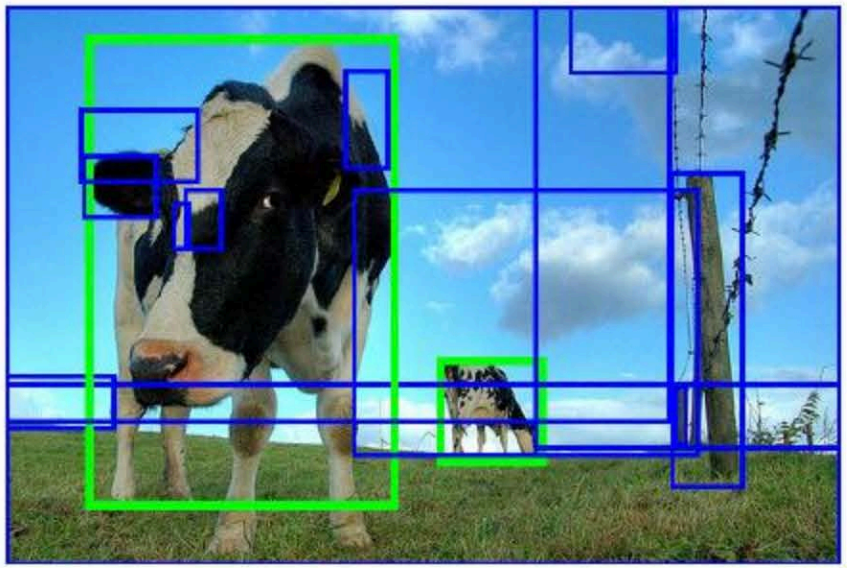
CNN as Feature Extractor

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

Region Proposal

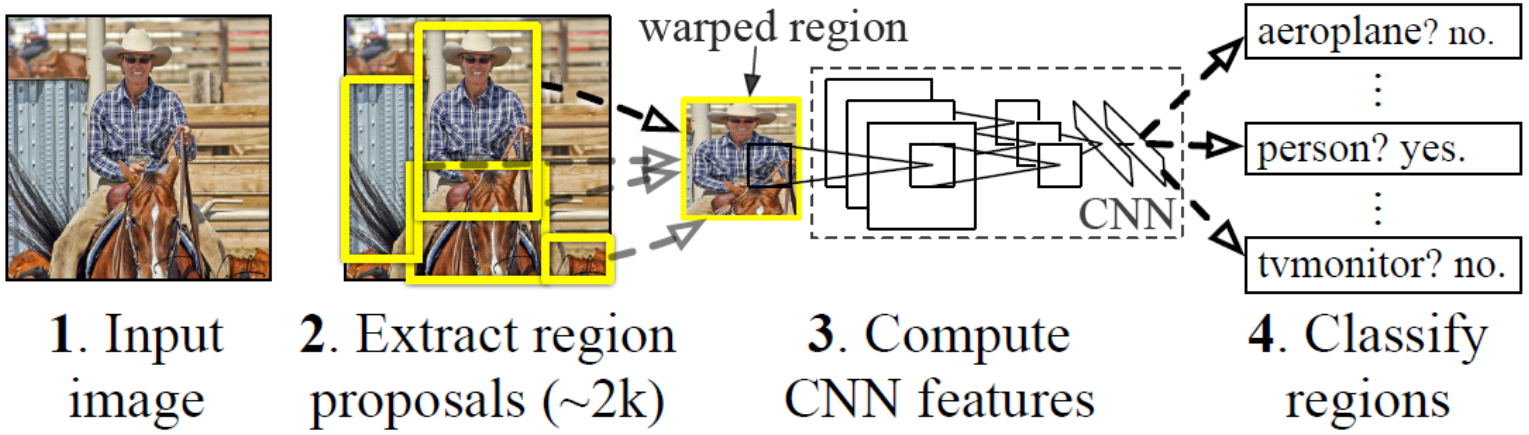
- Solution
  - Use some fast algorithms to filter out some regions first, and only feed the potential regions (i.e., region proposals) into CNN
  - E.g., selective search





Uijilings et al. IJCV 2013

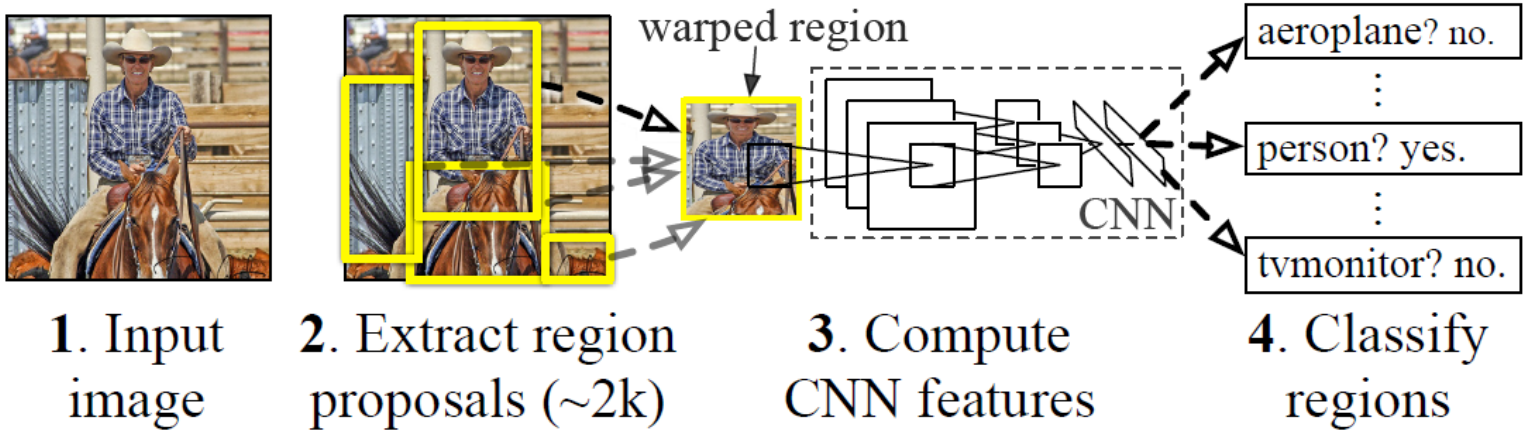
R-CNN (Girshick et al. CVPR 2014)



- Replace sliding windows with "selective search" region proposals (Uijilings et al. IJCV 2013)
- Extract rectangles around regions and resize to 227x227 pixels
- 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

R-CNN (Girshick et al. CVPR 2014)



- Ad hoc training objectives
  - Fine-tune network with softmax classifier (log loss)
  - Train post-hoc linear SVMs for each class (hinge loss)
  - Train post-hoc bounding-box regressors (least squares loss)
- Training is extremely slow with lots of disk space.

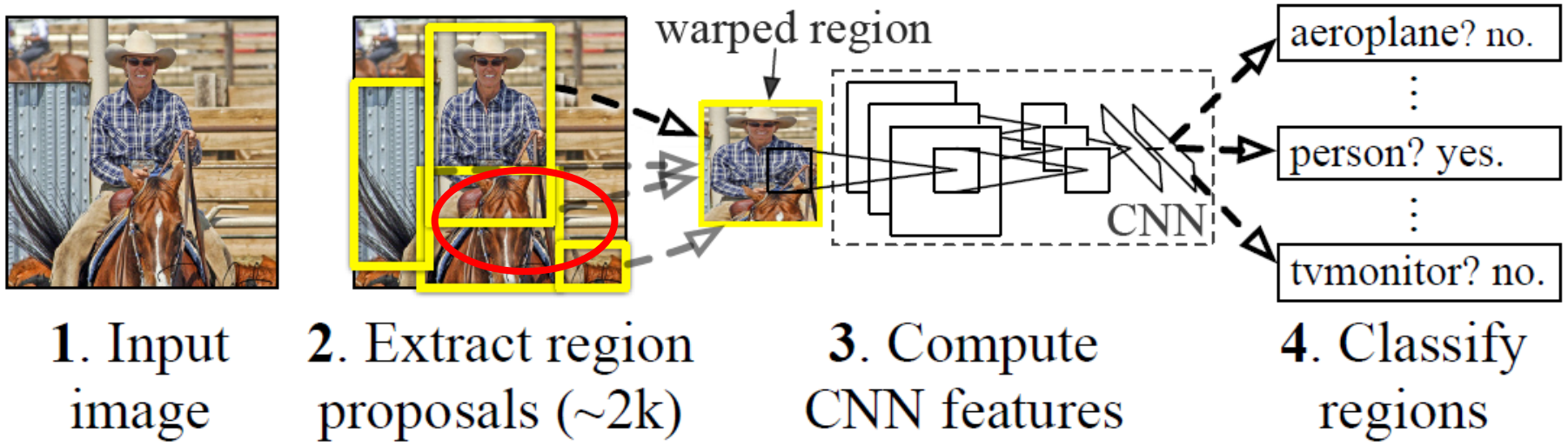
http://arxiv.org/pdf/1311.2524.pdf

Bounding Box Regression

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

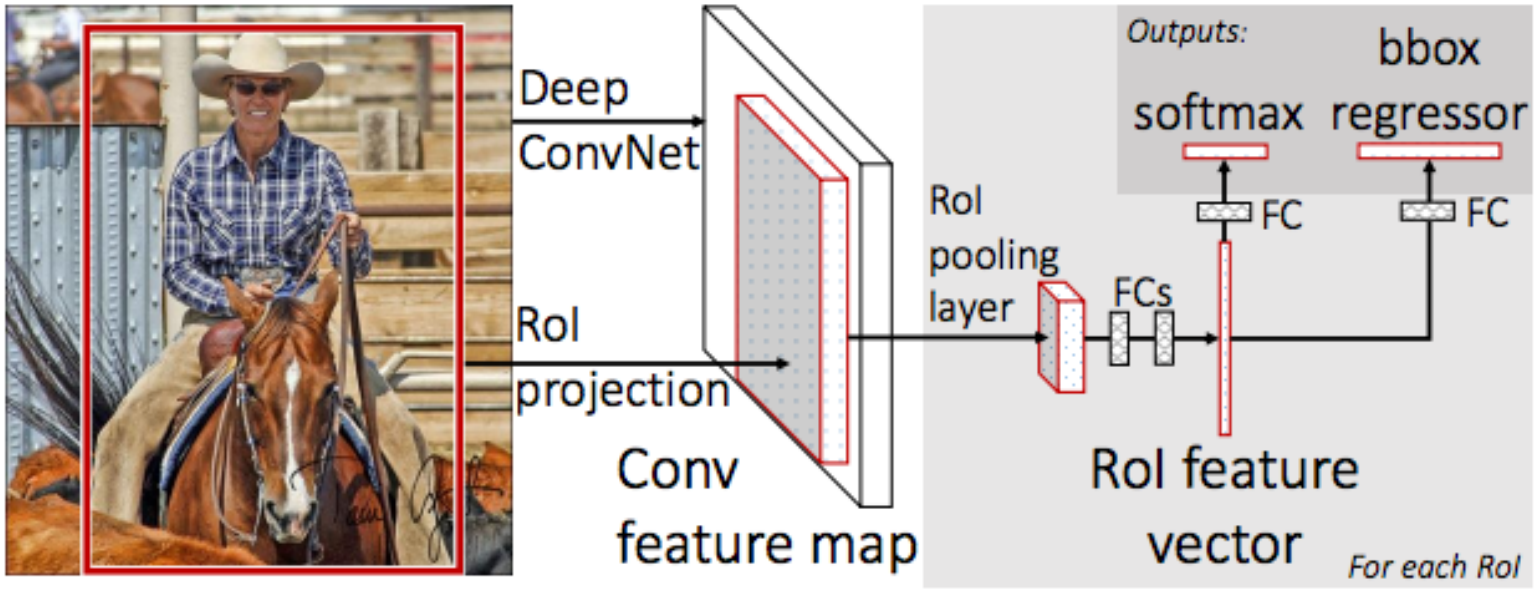
R-CNN (Girshick et al. CVPR 2014)

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



Fast R-CNN (Girshick ICCV 2015)

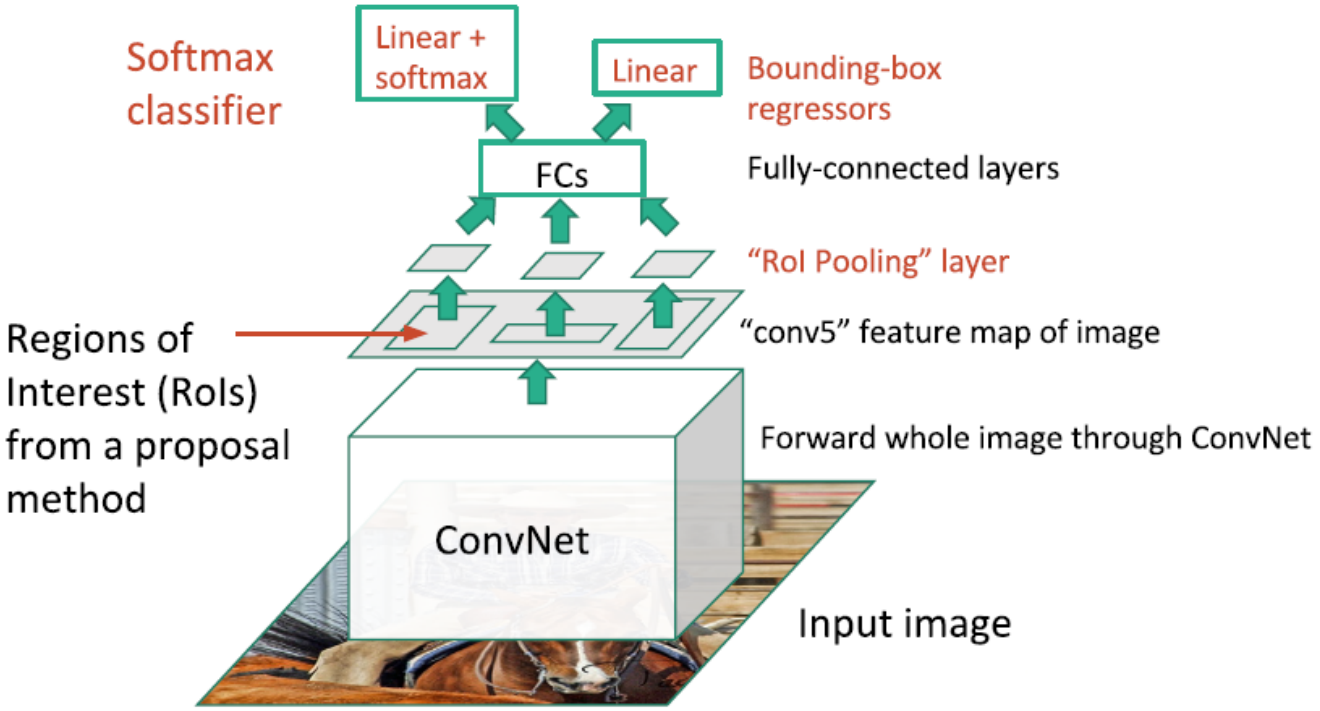
- Solution
  - Why not feed the whole image into CNN only once! Then crop features instead of image itself



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

Fast R-CNN (Girshick ICCV 2015)

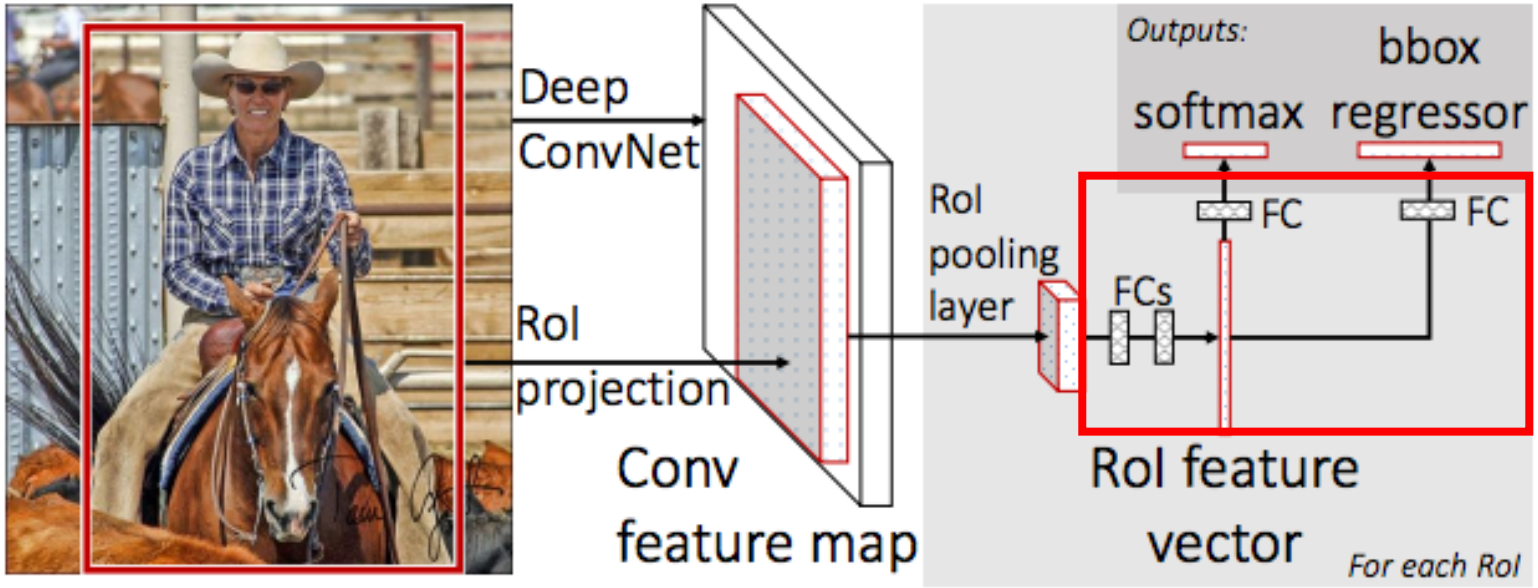
- Solution
  - Why not feed the whole image into CNN only once! Then crop features instead of image itself



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

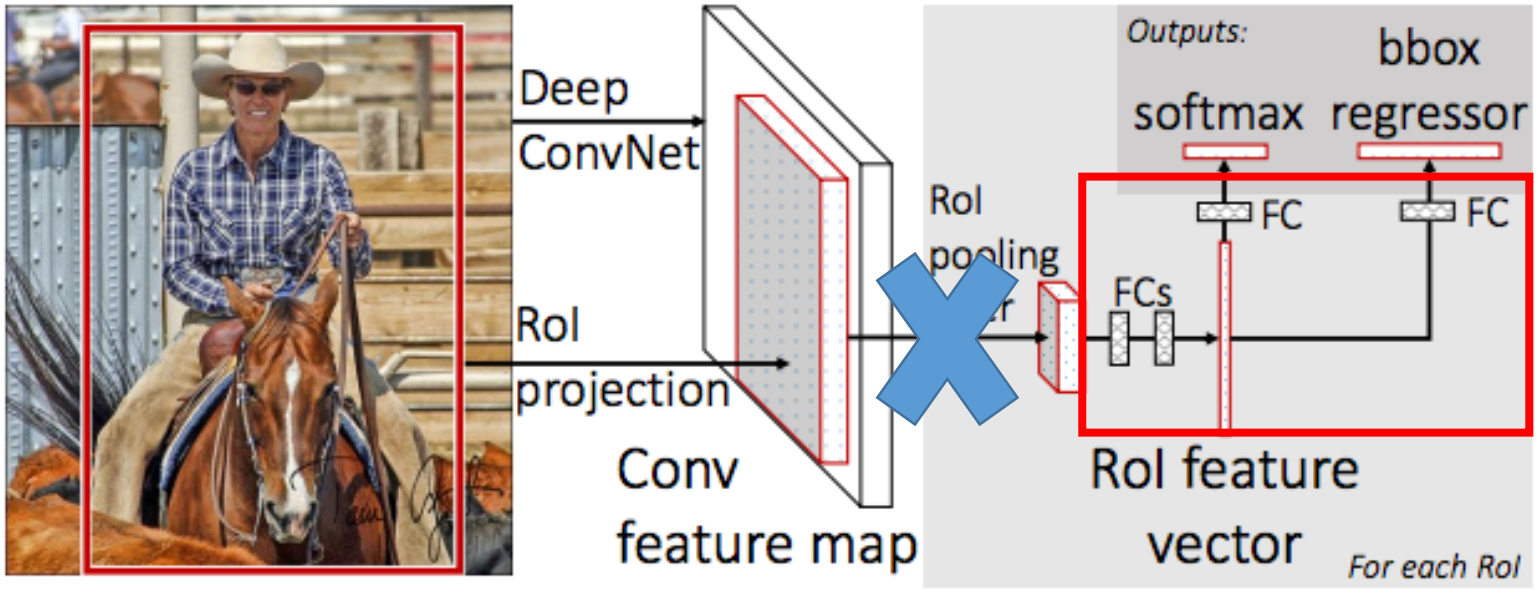
Fast R-CNN (Girshick ICCV 2015)

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



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
  - 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…

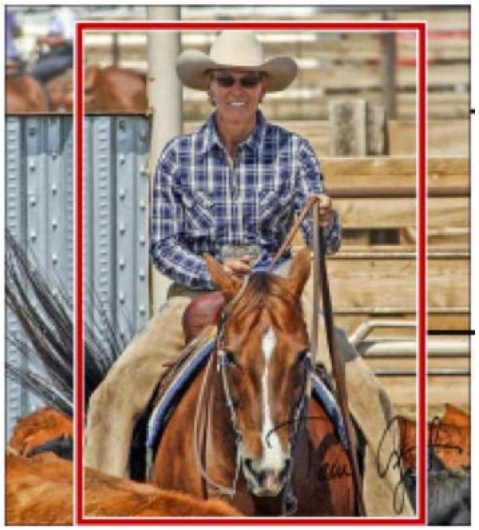


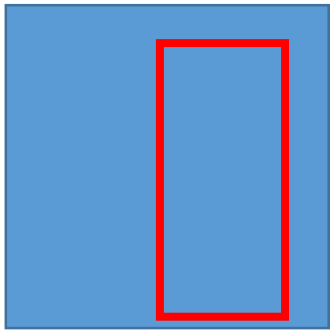
Fast R-CNN (Girshick ICCV 2015)

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

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

RoI Pooling

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

0	1	2	3
4	5	6	7
8	9	10	11
12	13	14	15

Feature map
4 x 4 x 1

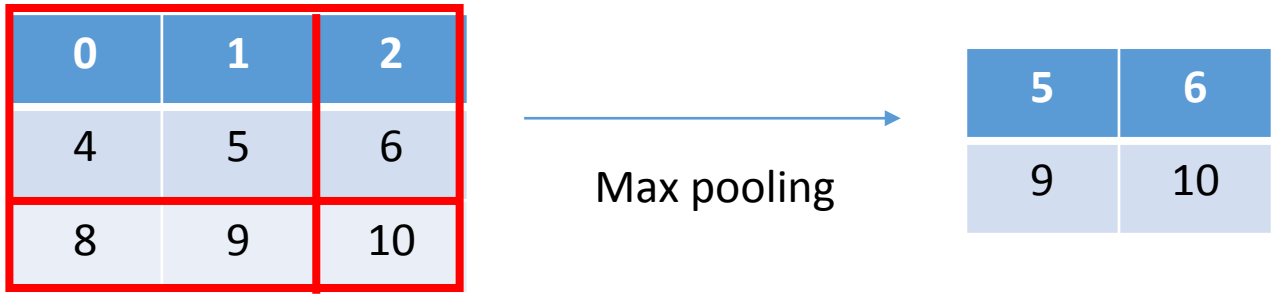
Make it as
2x2 grids

0	1	2
4	5	6
8	9	10

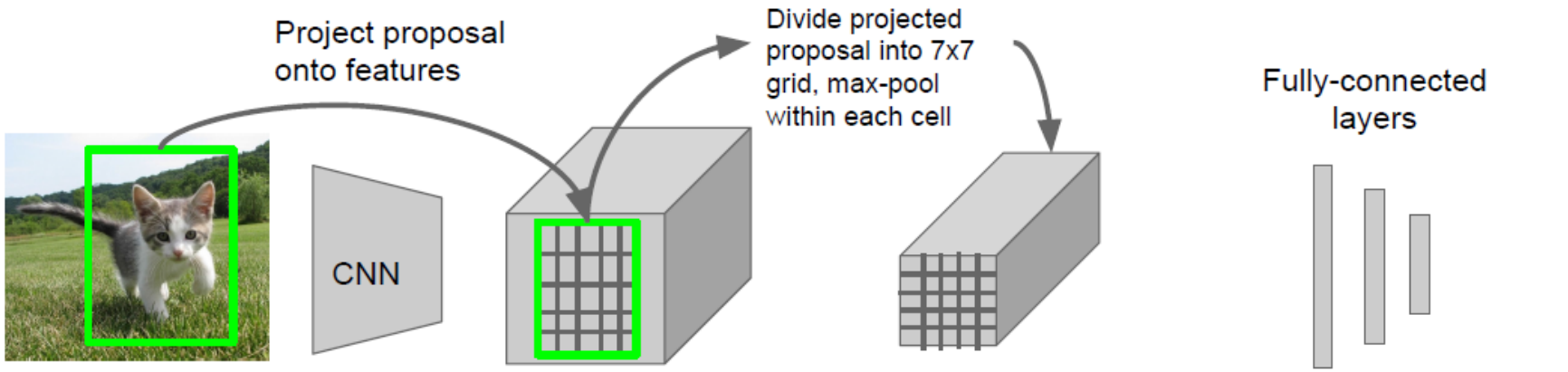
1	2	3
5	6	7
9	10	11
13	14	15

RoI Pooling

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



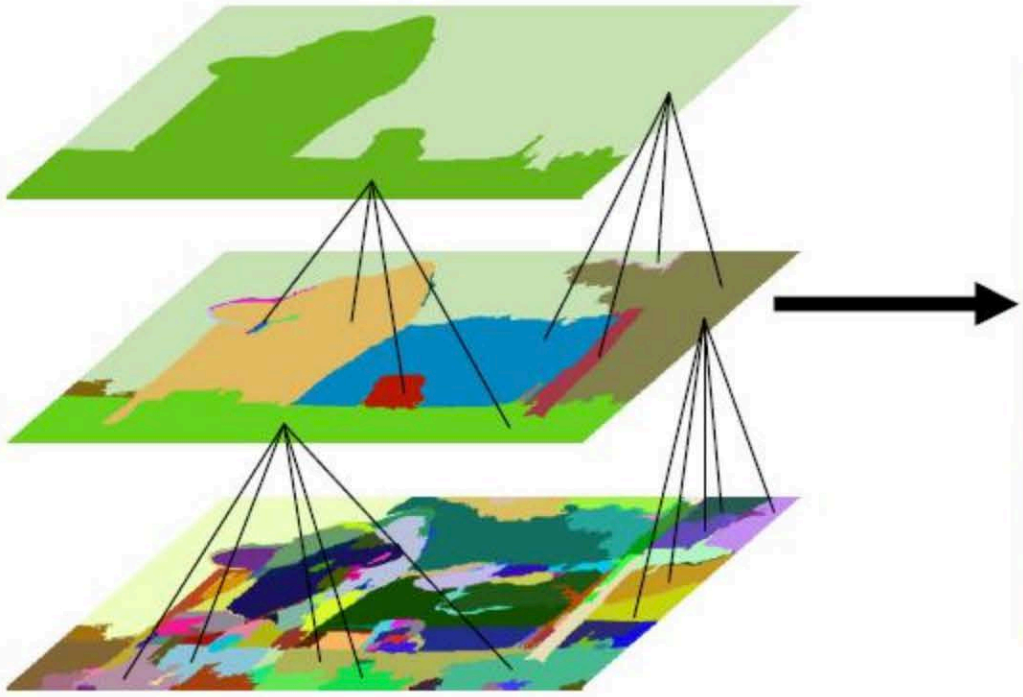
RoI Pooling

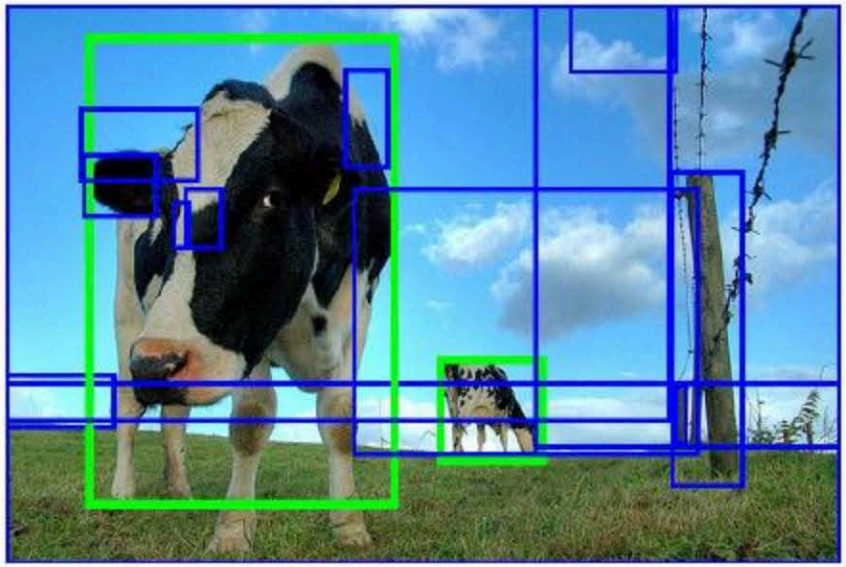


Girshick, "Fast R-CNN", ICCV 2015.

Fast R-CNN (Girshick ICCV 2015)

- What could be the problems?
  - Why we need the region proposal pre-processing step? That's still not "deep learning"…

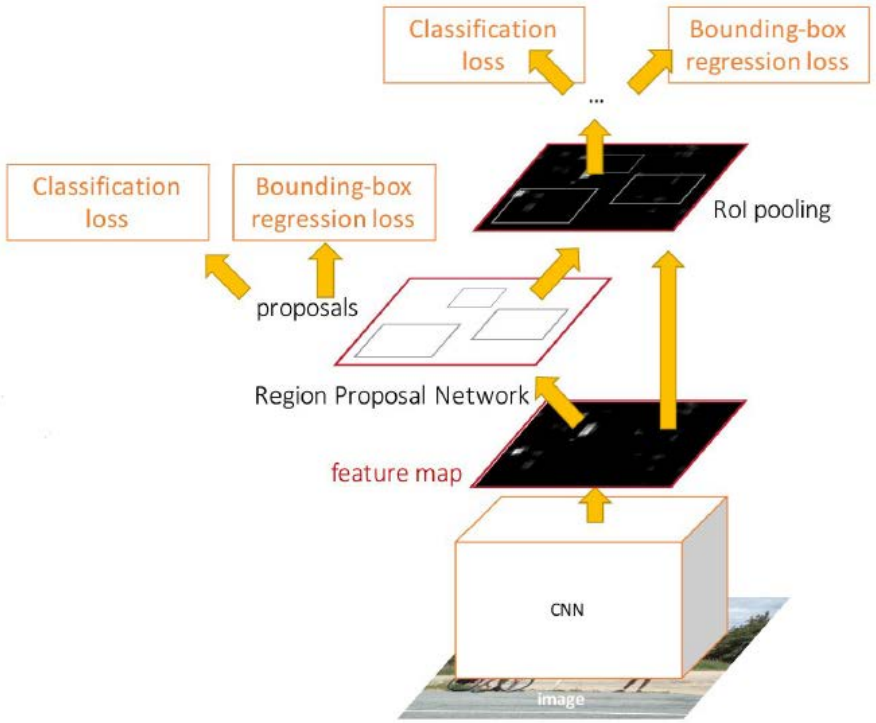




Uijilings et al. IJCV 2013

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

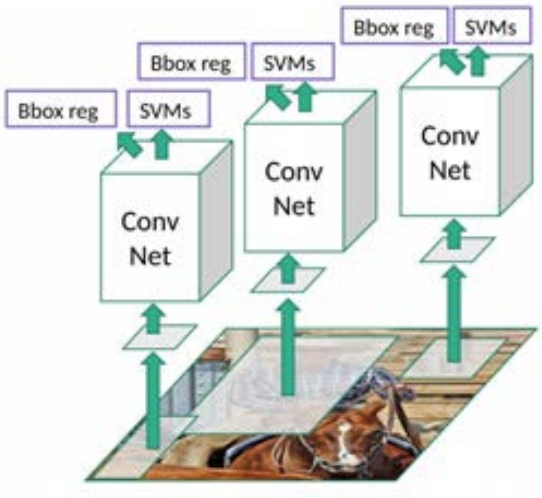
- Solution
  - Why not generate region proposals using CNN?
    -> Insert Region Proposal Network (RPN) to predict proposals from features
  - Jointly train with 4 losses:
    - RPN classification loss
    - RPN regress box coordinates
    - Final classification loss
    - Final box coordinates

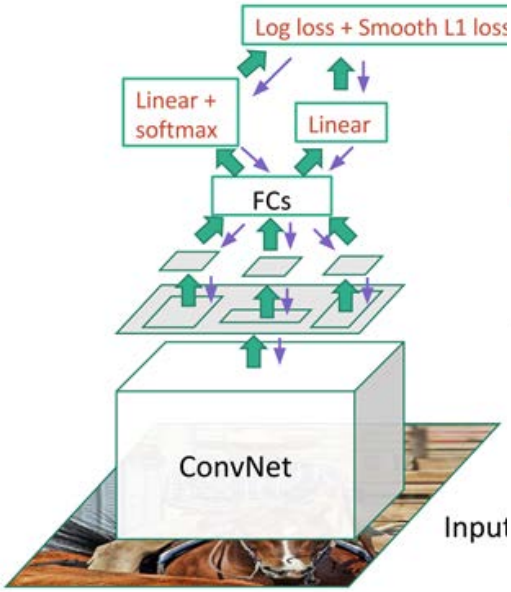


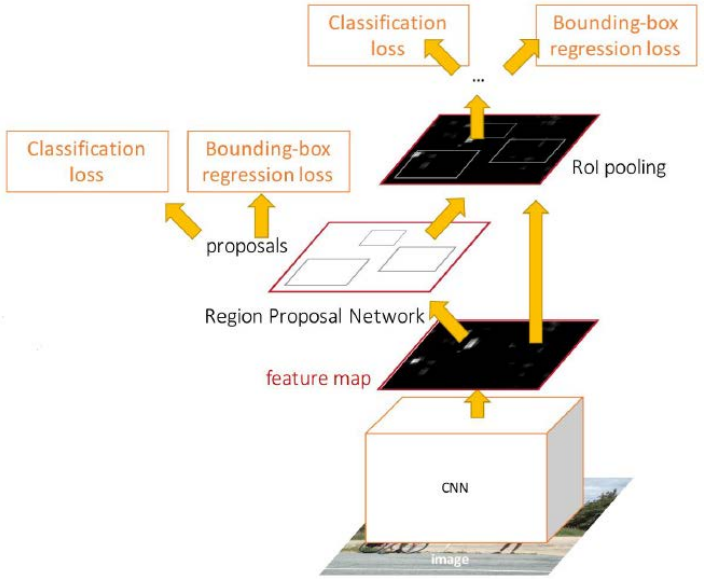
https://arxiv.org/pdf/1506.01497.pdf

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

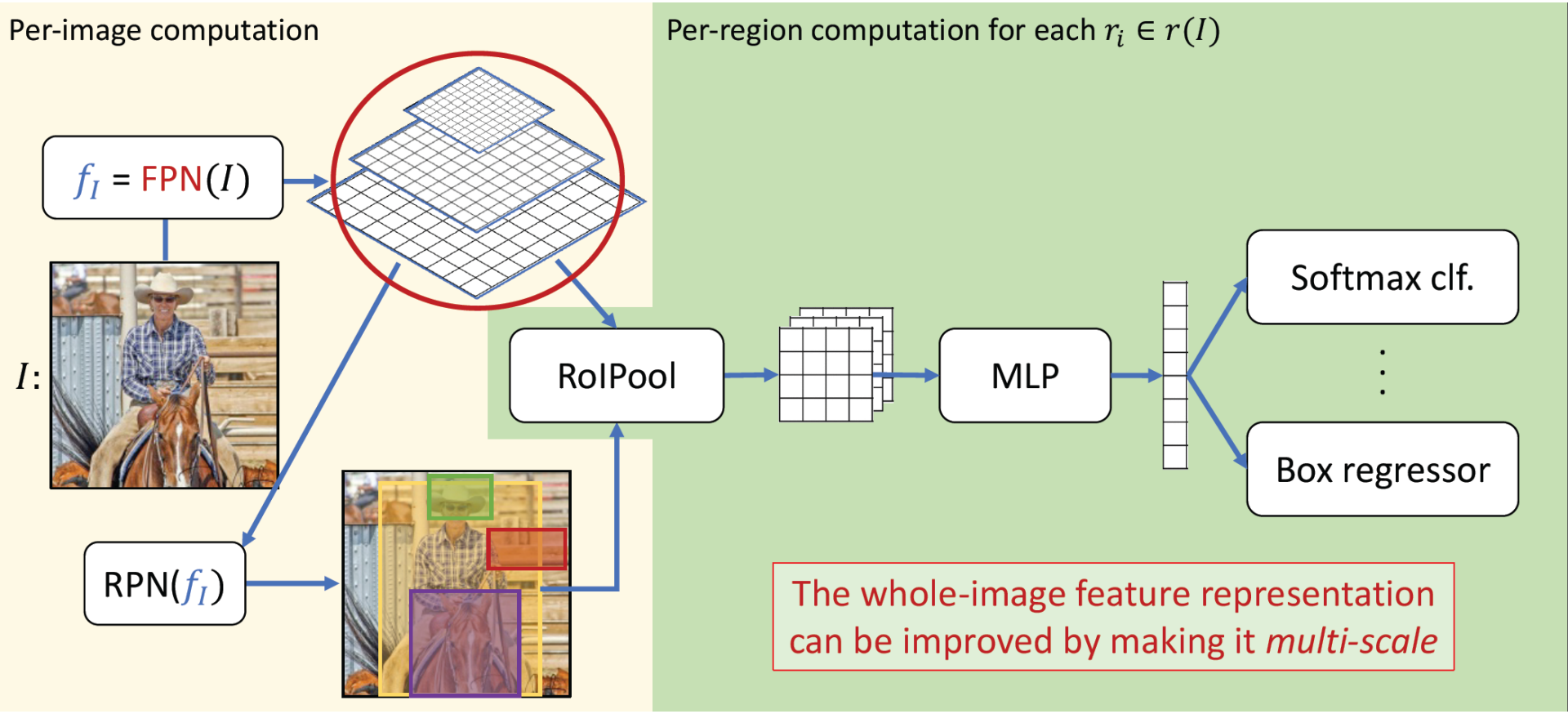
R-CNN, Fast R-CNN, & Faster R-CNN







Faster R-CNN with Feature Pyramid Network



Slide credit: Ross Girshick

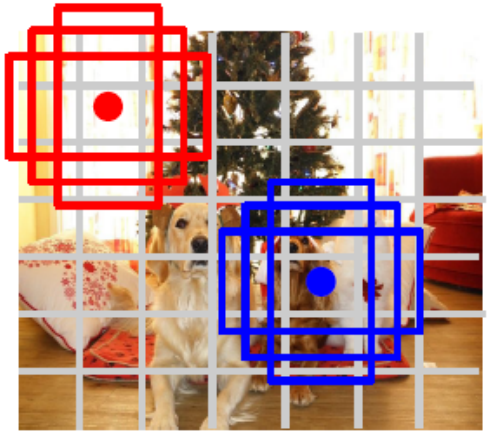
Faste*r* R-CNN (Ren et al. NIPS 2015)

- What could be the problems
  - Two-stage detection pipeline is still too slow for real-time detection in videos

Detection without Proposals: YOLO/SSD

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)

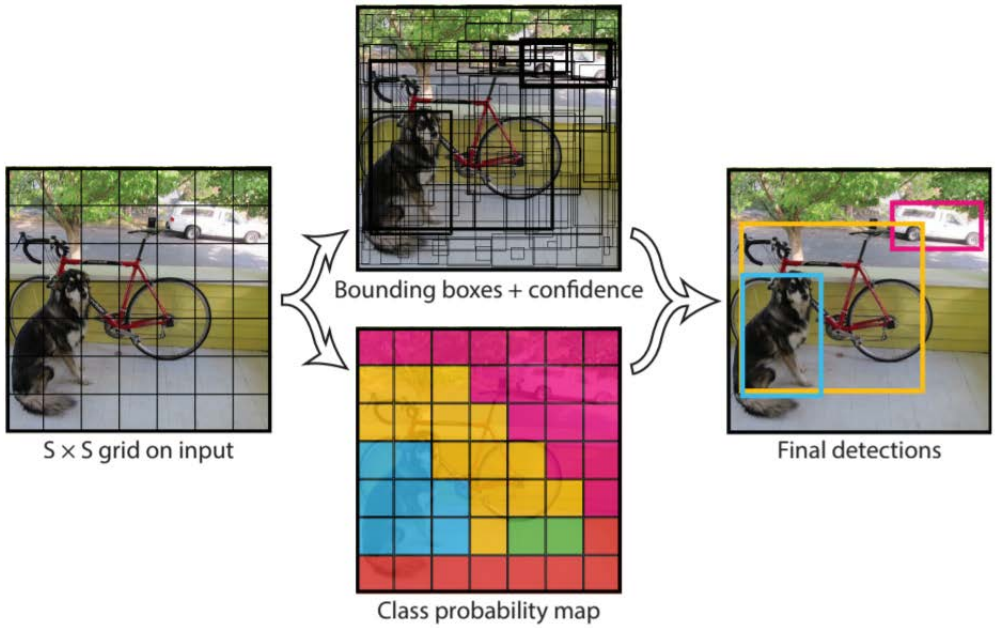
Output:
7 x 7 x (5 * B + C)

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

You Only Look Once (YOLO)

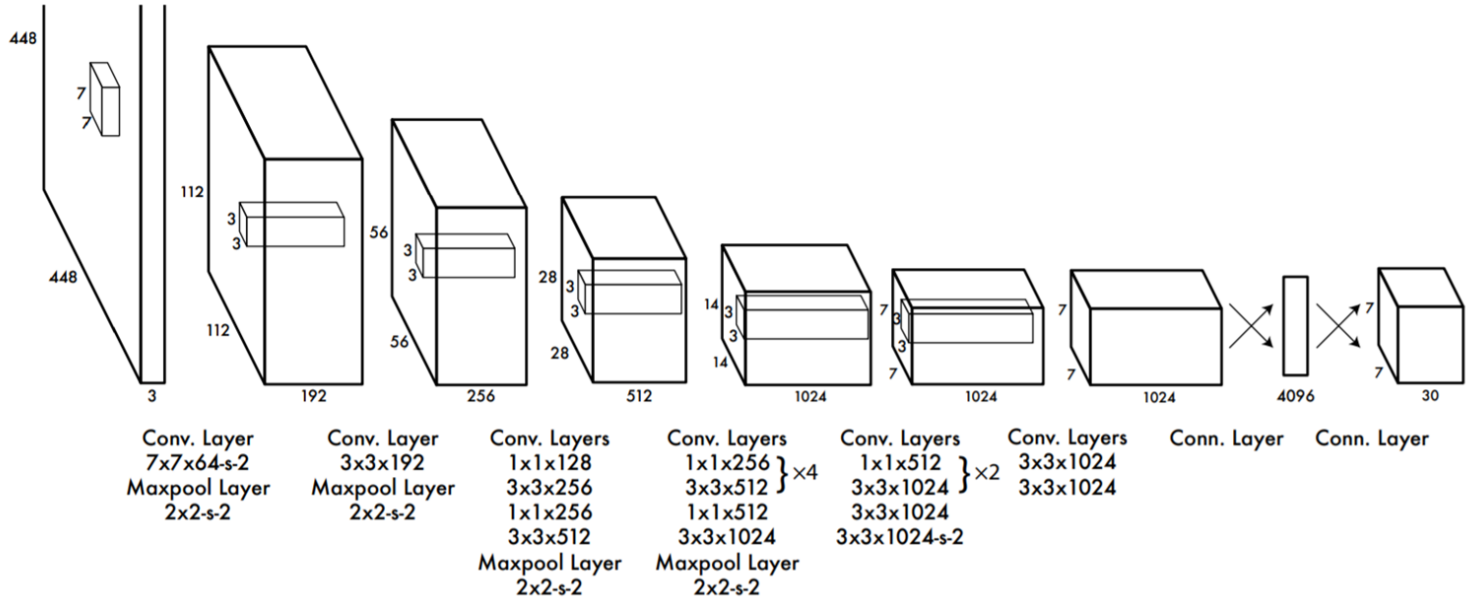
Divide the image into an $S \times S$ grid and for each grid cell predicts B bounding boxes, confidence for those boxes, and C class probabilities.

These predictions are encoded as an $S \times S \times (B * 5 + C)$ tensor.



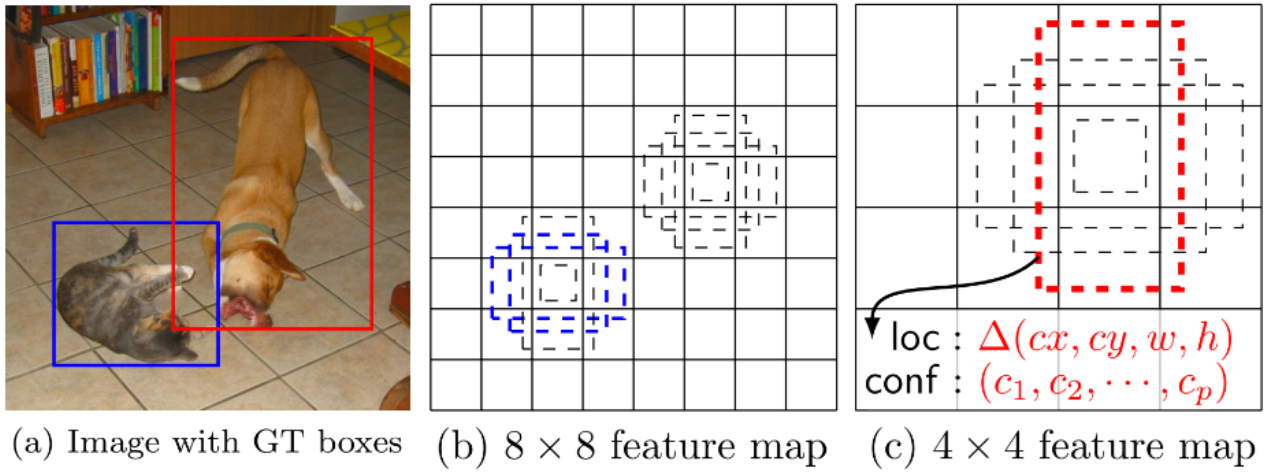
You Only Look Once (YOLO)

No region proposal needed!



Single Shot MultiBox Detector (SSD)

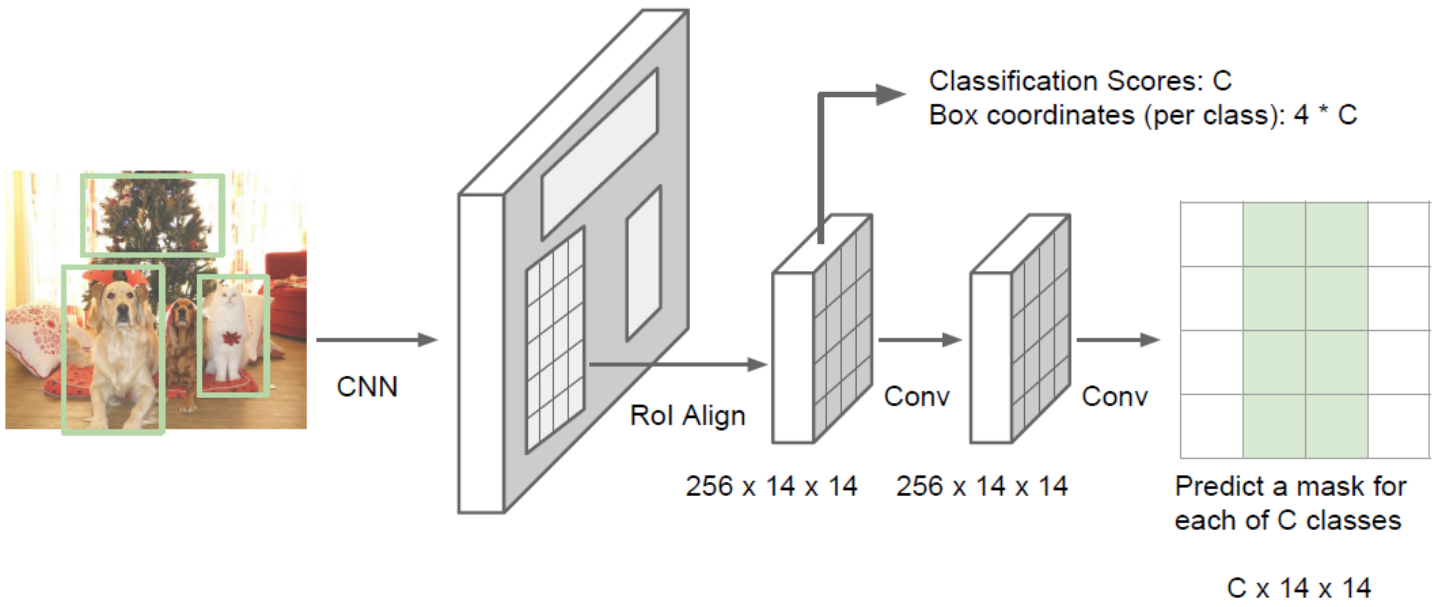
Propose multiple default boxes per grid at different scales



(a) Image with GT boxes (b) 8×8 feature map (c) 4×4 feature map

Mask R-CNN

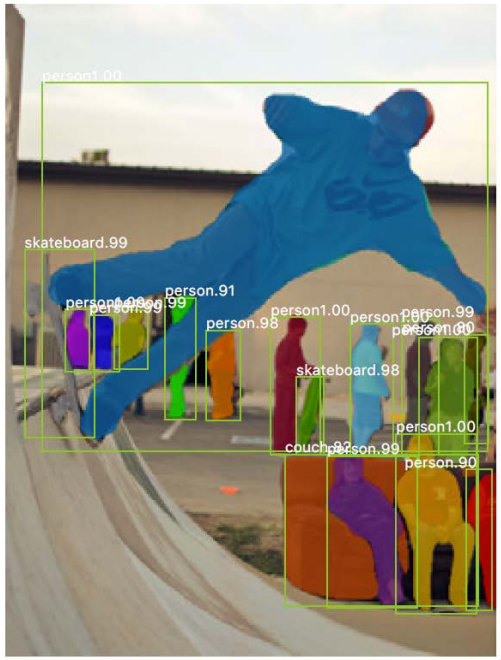
- Same network as Faster R-CNN, except
  - Bilinearly interpolate when extracting 7x7 cells of ROI features for better alignment of features to image
  - Instance segmentation: produce a mask for each object category
  - Keypoint prediction: produce a 56x56 mask for each keypoint (to label single pixel as correct keypoint)

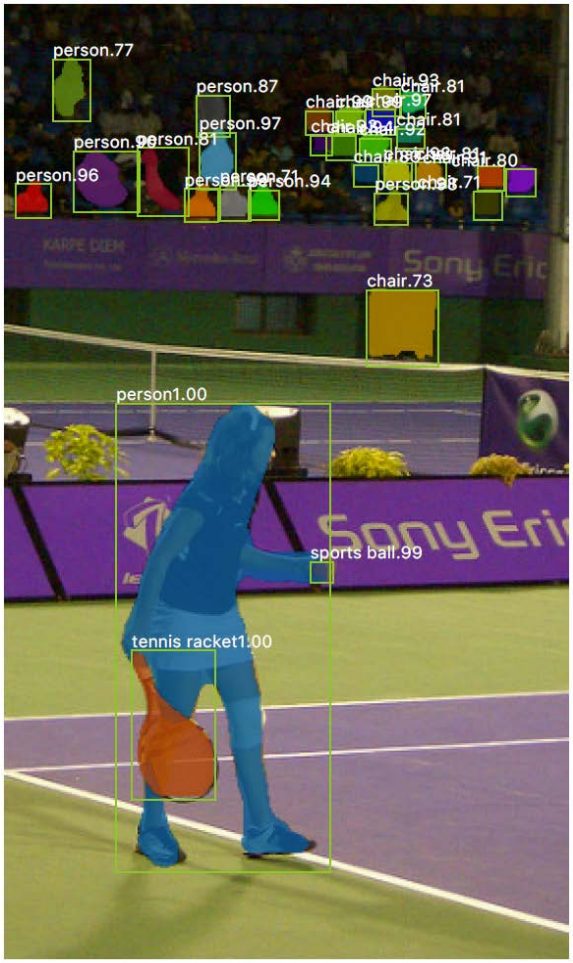


He et al, "Mask R-CNN", arXiv 2017

Mask R-CNN

- Very good results!





What We Have Learned So Far…

- Neural Networks & CNN
  - Convolutional Neural Networks
- Recognition & Detection
  - Recognition: From Interest Points to Bag-of-Words Models
  - Object Detection

- HW 2 is out and due 11/13 11pm!