

Convolutional Feature Maps

Elements of efficient (and accurate)
CNN-based object detection

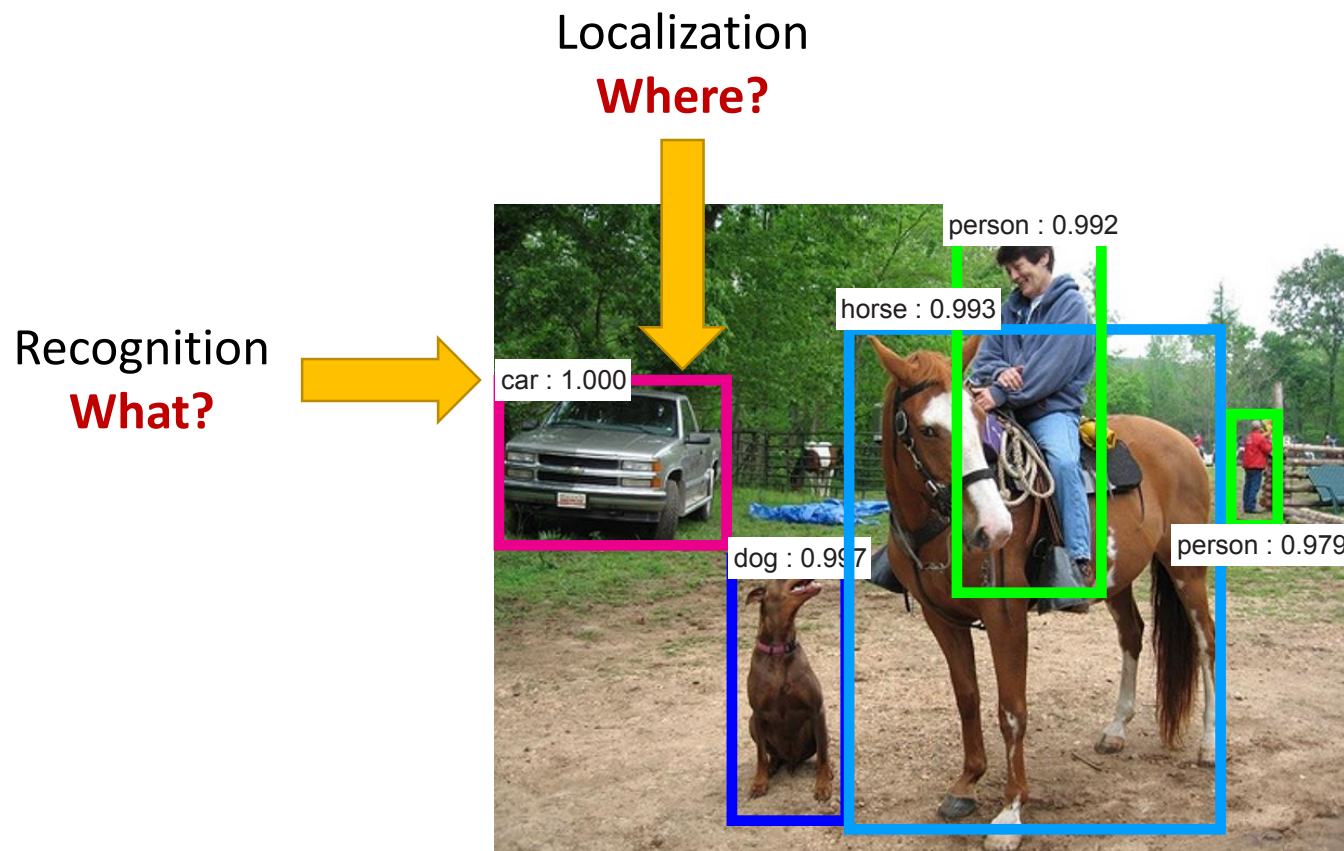
Kaiming He

Microsoft Research Asia (MSRA)

Overview of this section

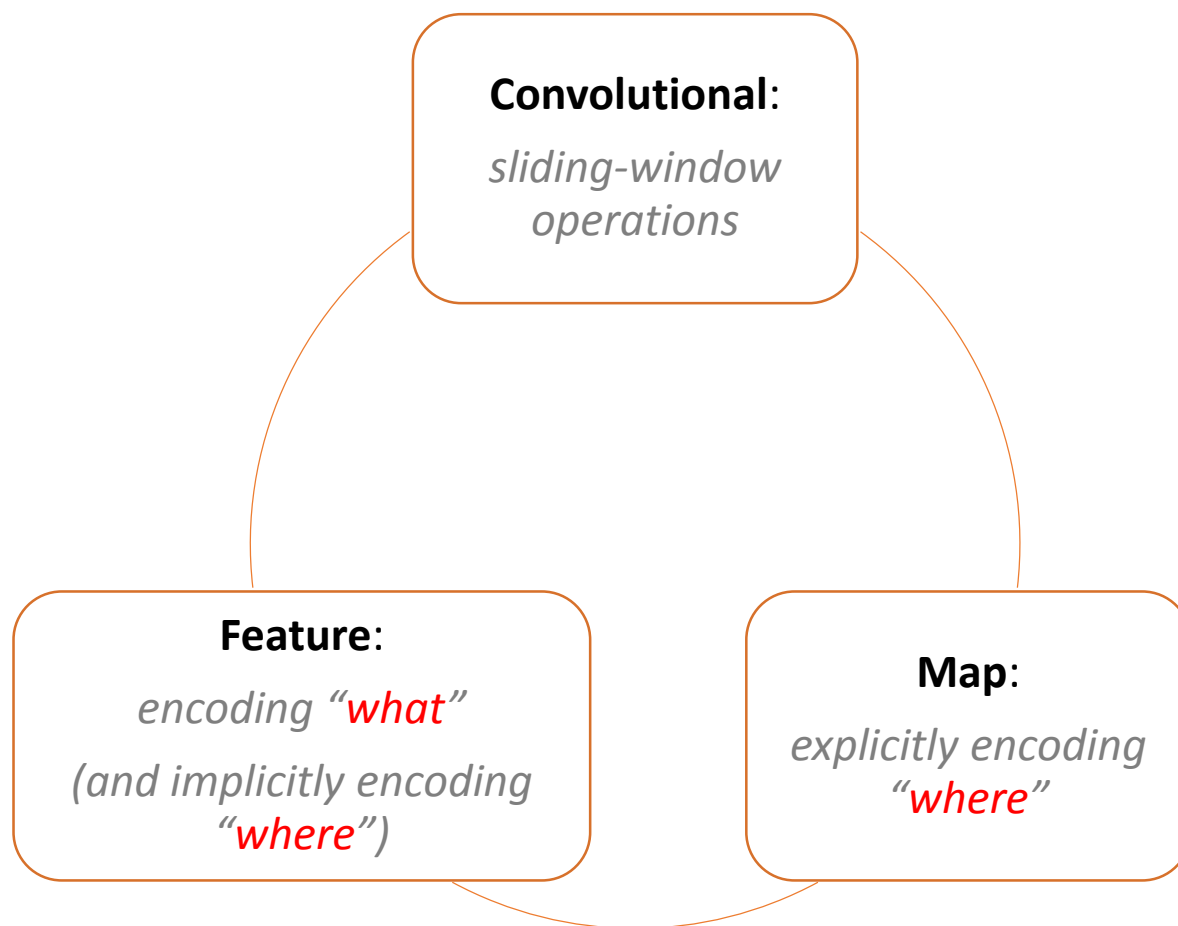
- Quick introduction to convolutional feature maps
 - Intuitions: into the “black boxes”
 - How object detection networks & region proposal networks are designed
 - Bridging the gap between “hand-engineered” and deep learning systems
- Focusing on forward propagation (inference)
 - Backward propagation (training) covered by Ross’s section

Object Detection = What, and Where



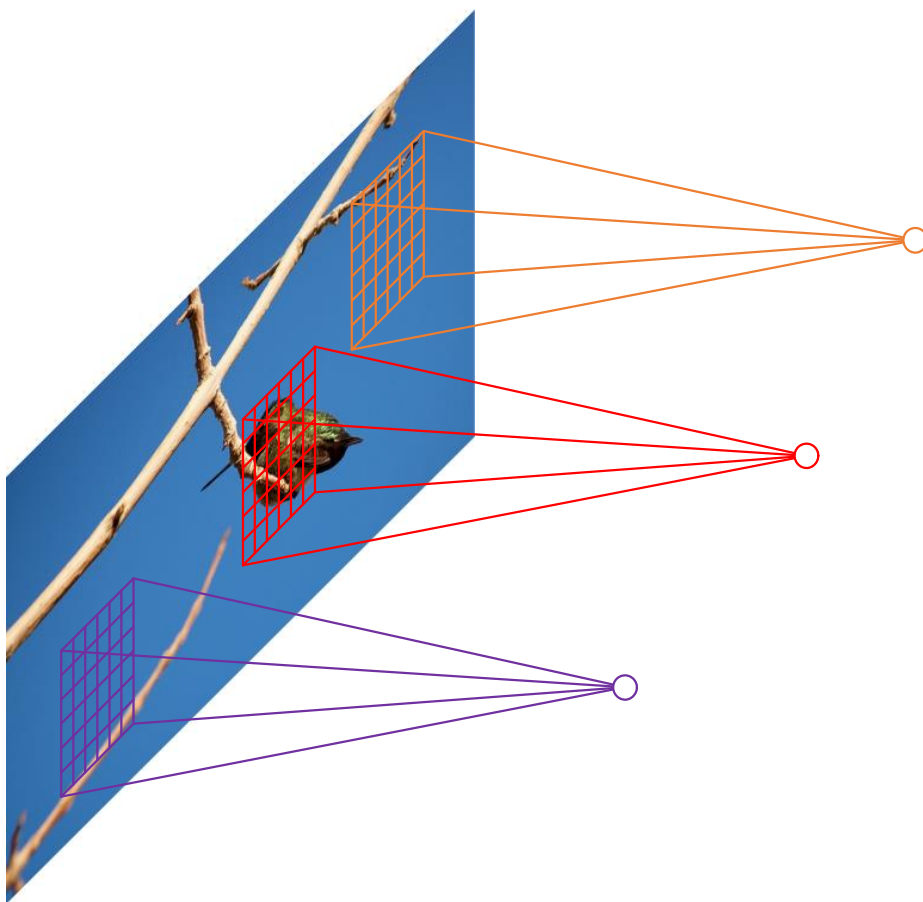
- We need a building block that tells us “what and where”...

Object Detection = What, and Where



Convolutional Layers

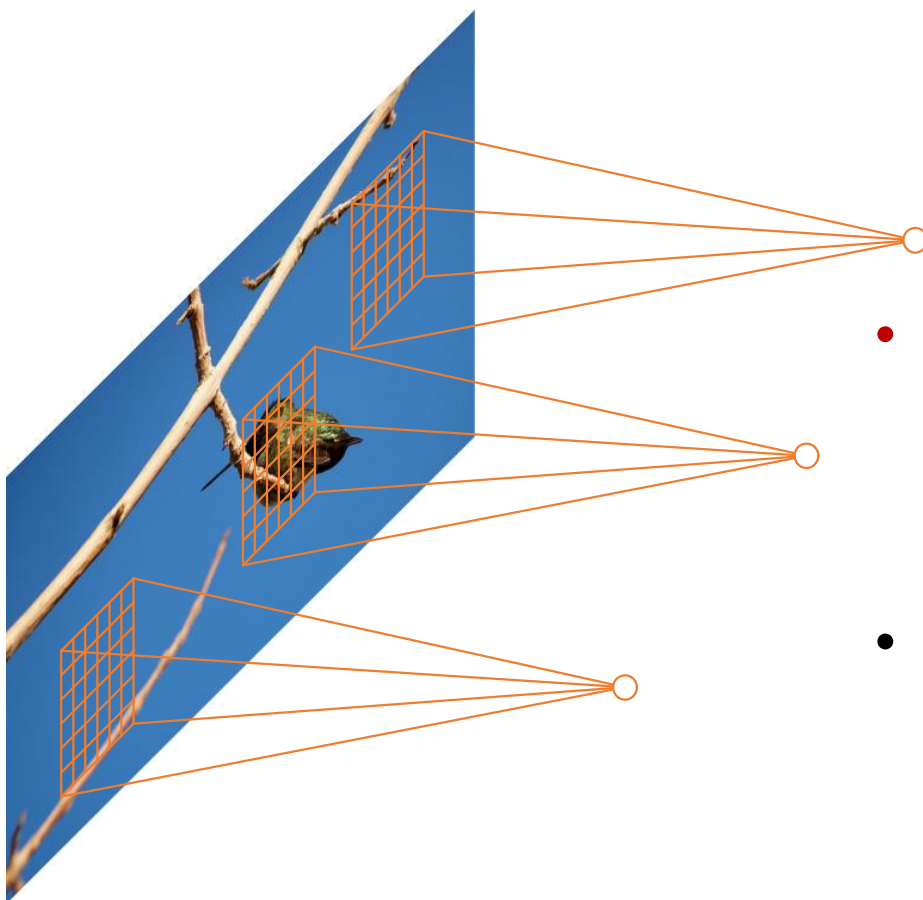
- Convolutional layers are **locally connected**



- a filter/kernel/window slides on the image or the previous map
- the **position** of the filter explicitly provides information for localizing
- local spatial information w.r.t. the window is encoded in the channels

Convolutional Layers

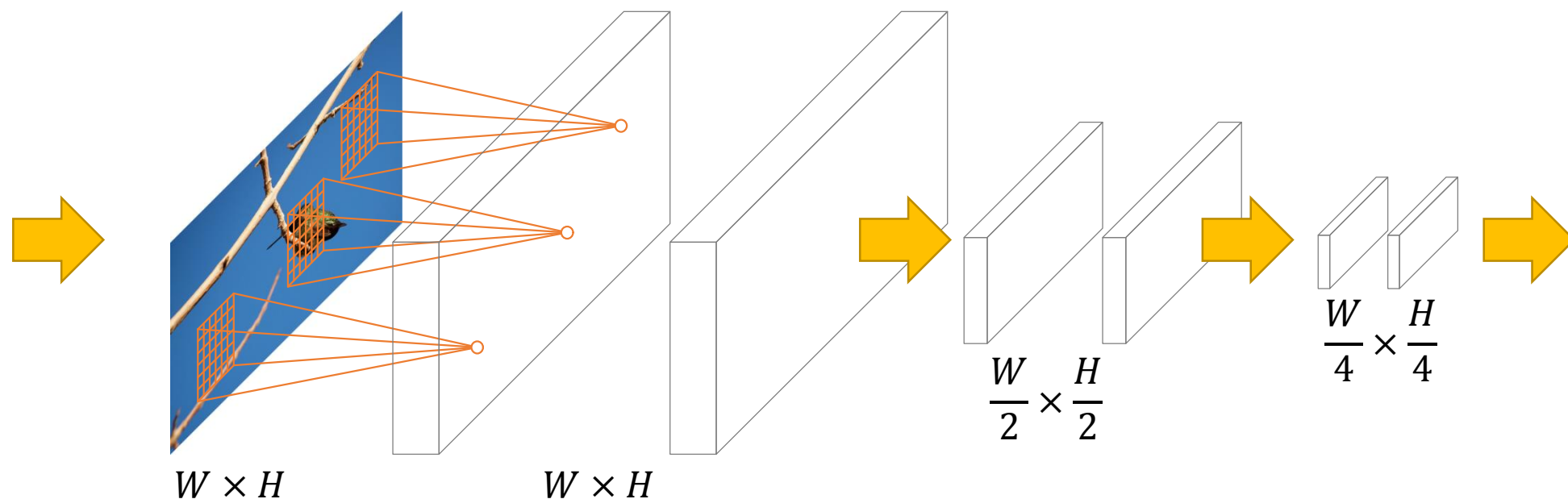
- Convolutional layers share weights spatially: **translation-invariant**



- **Translation-invariant**: a translated region will produce the same response at the correspondingly translated position
- A local pattern's convolutional response can be **re-used** by different candidate regions

Convolutional Layers

- Convolutional layers can be applied to **images of any sizes**, yielding **proportionally-sized** outputs



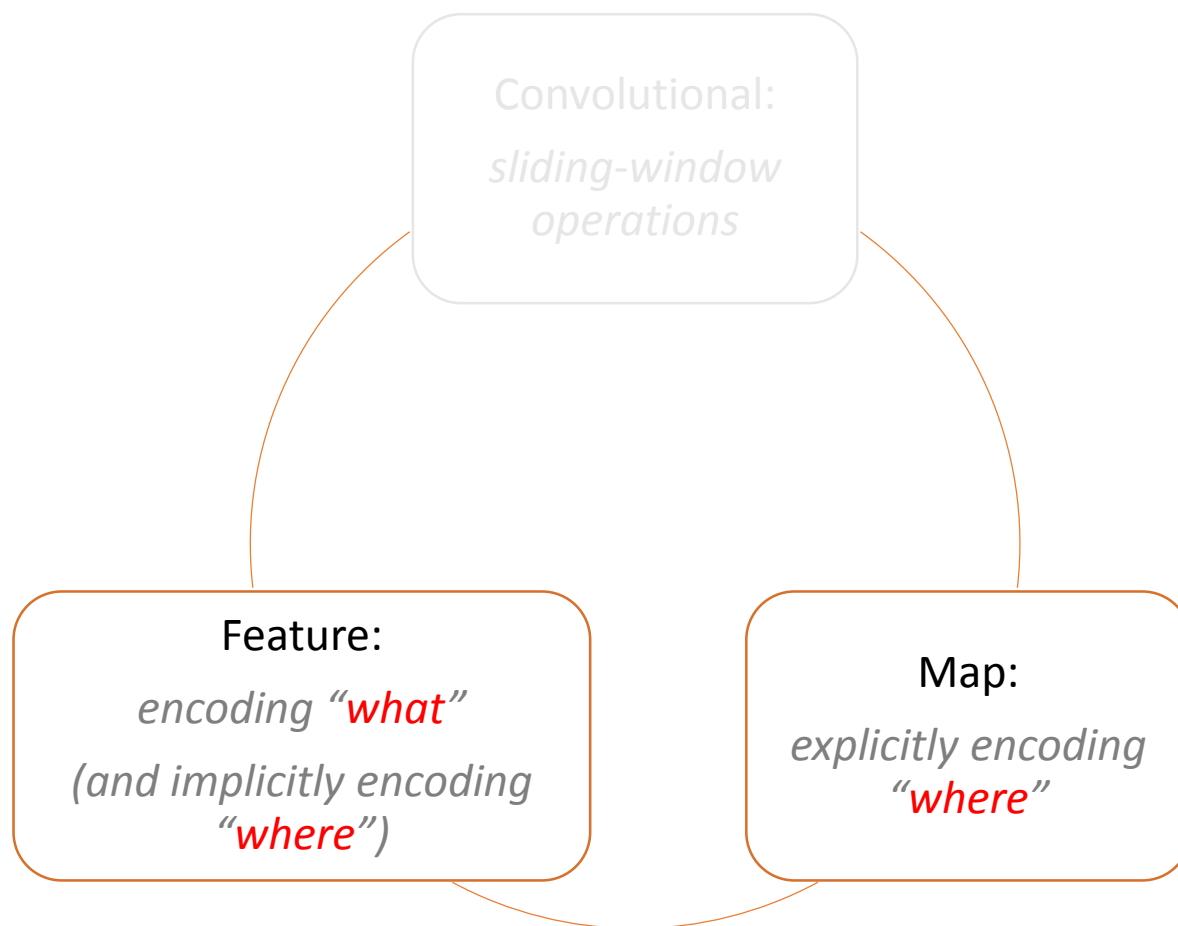
HOG by Convolutional Layers

- Steps of computing HOG:
 - Computing image gradients
 - Binning gradients into 18 directions
 - Computing cell histograms
 - Normalizing cell histograms
- Convolutional perspectives:
 - Horizontal/vertical edge filters
 - Directional filters + gating (non-linearity)
 - Sum/average pooling
 - Local response normalization (LRN)

see [Mahendran & Vedaldi, CVPR 2015]

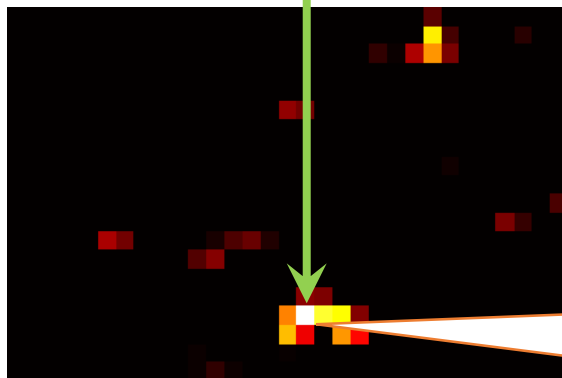
HOG, dense SIFT, and many other “hand-engineered” features are convolutional feature maps.

Feature Maps = features and their locations



Feature Maps = features and their locations

ImageNet images with **strongest** responses of this channel



one feature map of conv₅
(#55 in 256 channels of a model
trained on ImageNet)



Intuition of *this* response:

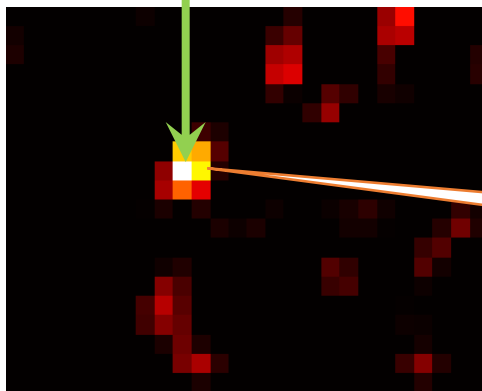
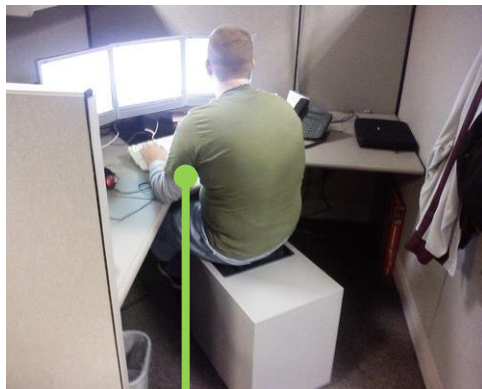
There is a “**circle-shaped**” object (likely a tire) **at this position**.

What

Where

Feature Maps = features and their locations

ImageNet images with **strongest** responses of this channel



one feature map of conv₅
(#66 in 256 channels of a model
trained on ImageNet)



Intuition of *this* response:
There is a “**λ-shaped**” object (likely an underarm) **at this position**.

What

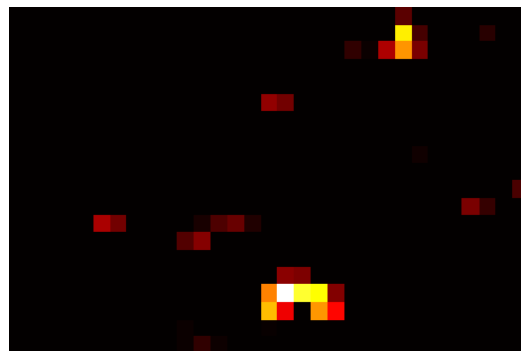
Where

Feature Maps = features and their locations

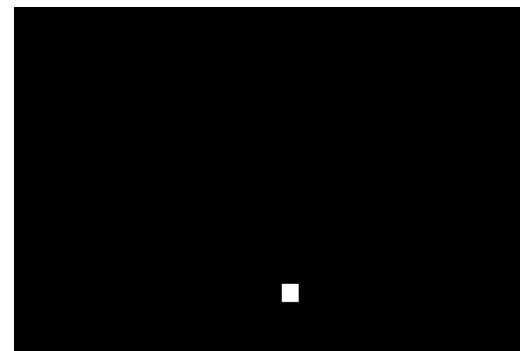
- Visualizing **one response** (by Zeiler and Fergus)



image



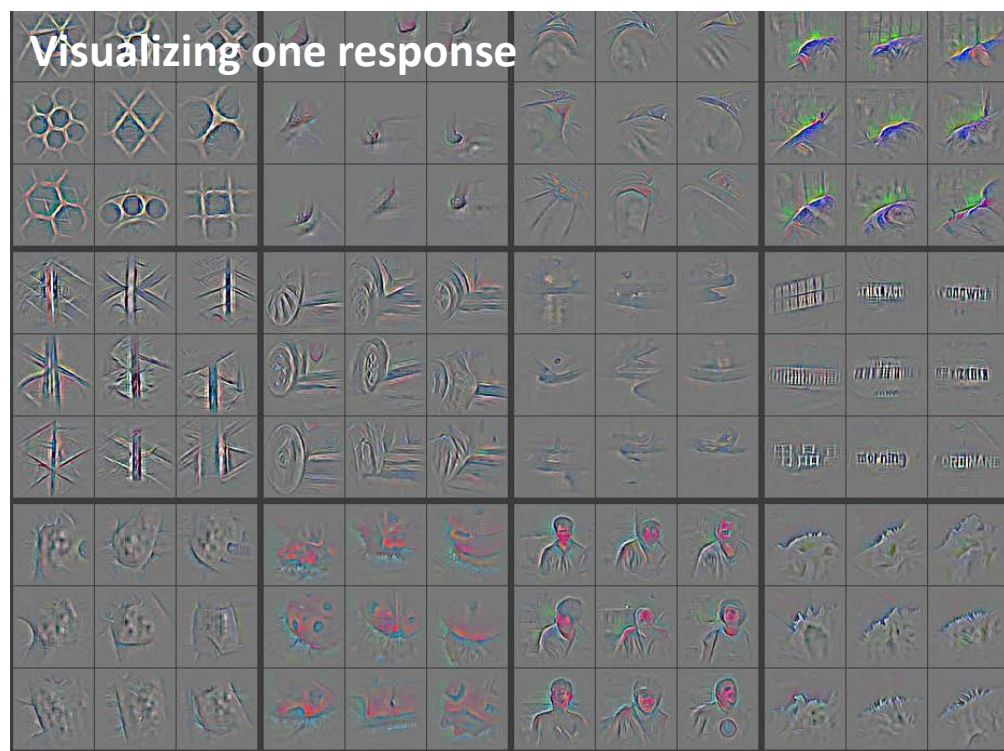
a feature map



keep one response
(e.g., the strongest)



Feature Maps = features and their locations



conv3

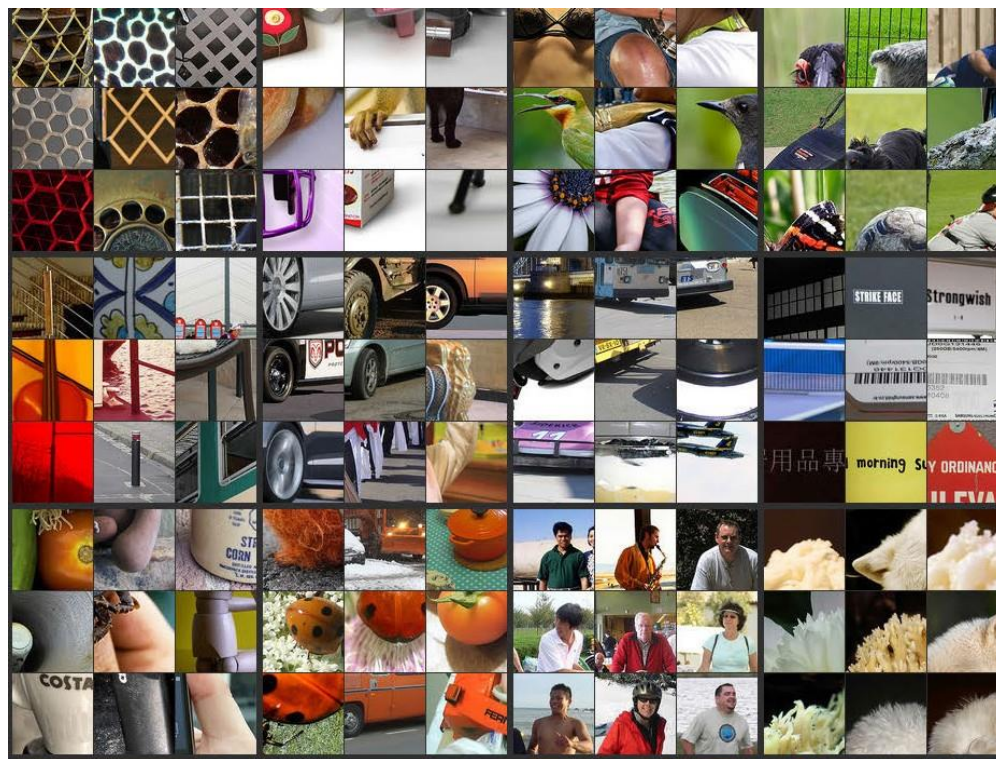
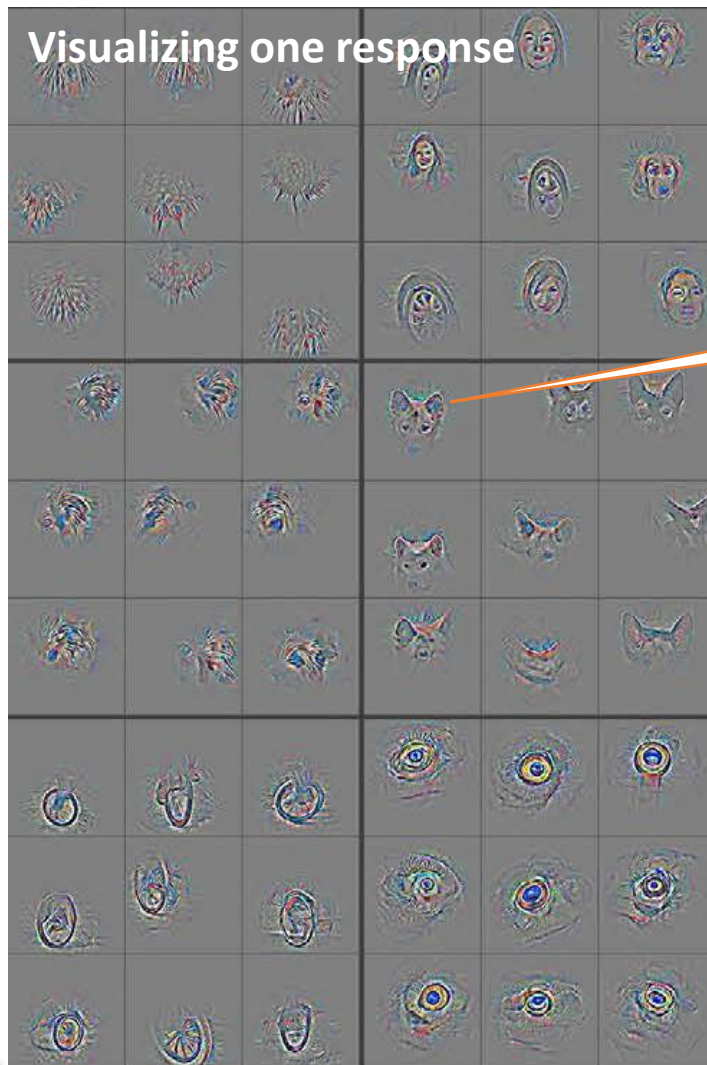


image credit: Zeiler & Fergus

Feature Maps = features and their locations

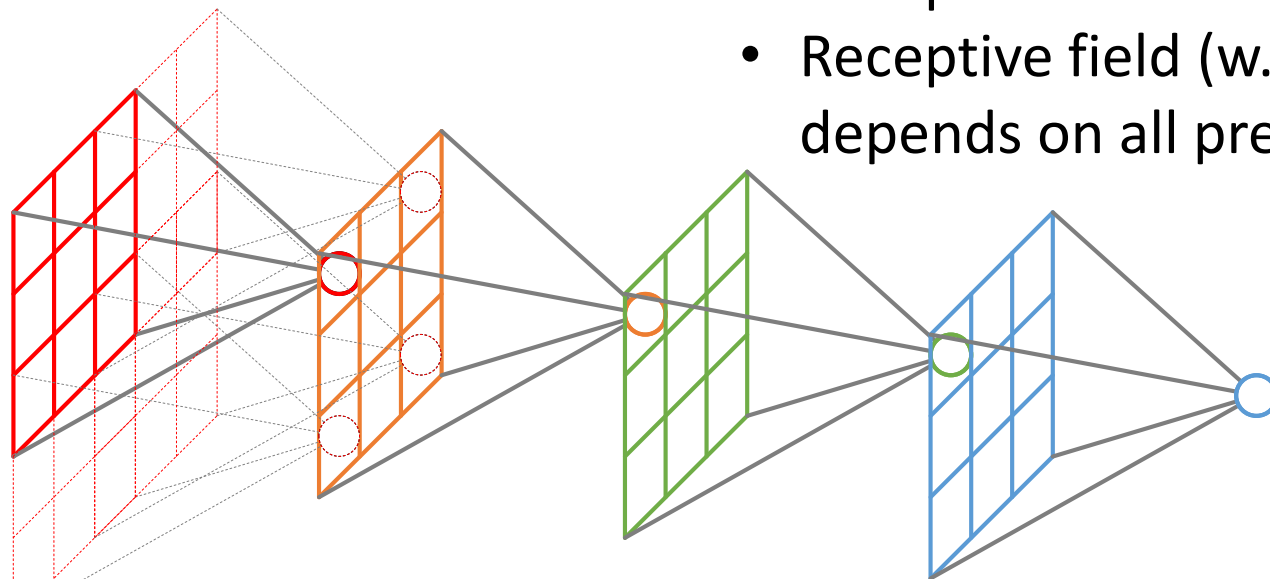


Intuition of *this* visualization:
There is a “dog-head” shape at this position.

- **Location** of a feature:
explicitly represents *where* it is.
- **Responses** of a feature:
encode *what* it is, and implicitly
encode finer position information –

*finer position information is
encoded in the channel dimensions
(e.g., bbox regression from
responses at one pixel as in RPN)*

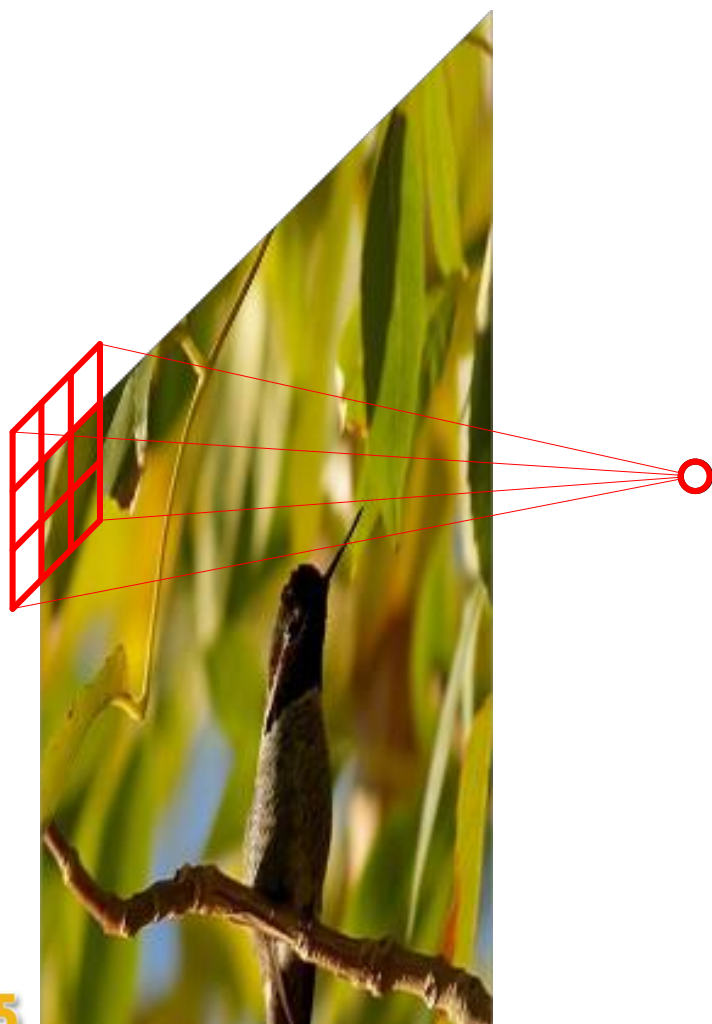
Receptive Field



- Receptive field of the first layer is the filter size
- Receptive field (w.r.t. input image) of a deeper layer depends on all previous layers' filter size and strides

- **Correspondence** between a feature map pixel and an image pixel is not unique
- Map a feature map pixel to **the center of the receptive field** on the image in the SPP-net paper

Receptive Field



How to compute the center of the receptive field

- A simple solution
 - For each layer, pad $\lfloor F/2 \rfloor$ pixels for a filter size F (e.g., pad 1 pixel for a filter size of 3)
 - On each feature map, the response at $(0, 0)$ has a receptive field centered at $(0, 0)$ on the image
 - On each feature map, the response at (x, y) has a receptive field centered at (Sx, Sy) on the image (stride S)

- A general solution

$$i_0 = g_L(i_L) = \alpha_L(i_L - 1) + \beta_L,$$

$$\alpha_L = \prod_{p=1}^L S_p,$$

$$\beta_L = 1 + \sum_{p=1}^L \left(\prod_{q=1}^{p-1} S_q \right) \left(\frac{F_p - 1}{2} - P_p \right)$$

See [Karel Lenc & Andrea Vedaldi]
"R-CNN minus R". BMVC 2015.

Region-based CNN Features

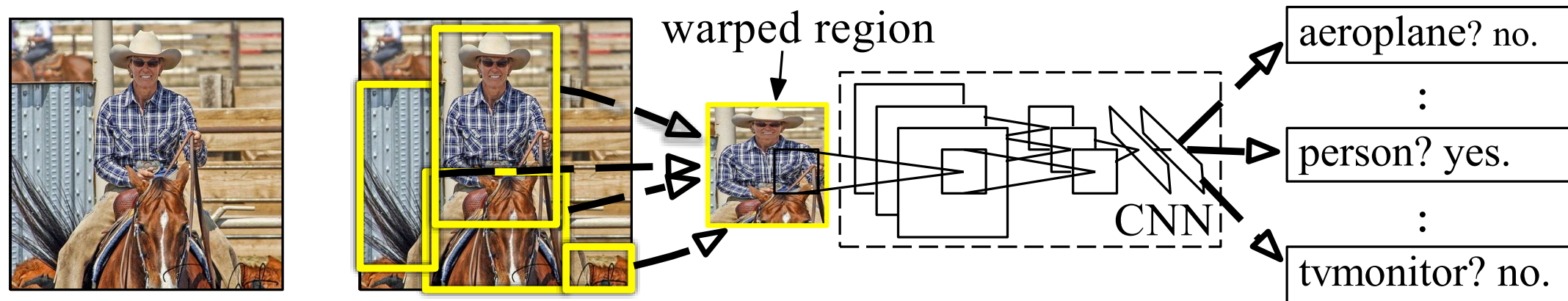


figure credit: R. Girshick et al.

input image

region proposals
~2,000

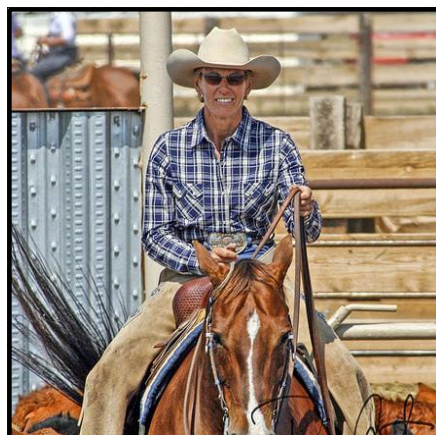
1 CNN for each region

classify regions

R-CNN pipeline

Region-based CNN Features

- Given proposal regions, what we need is **a feature for each region**
- R-CNN: **cropping an image region** + CNN on region, requires 2000 CNN computations
- What about **cropping feature map regions**?



Regions on Feature Maps



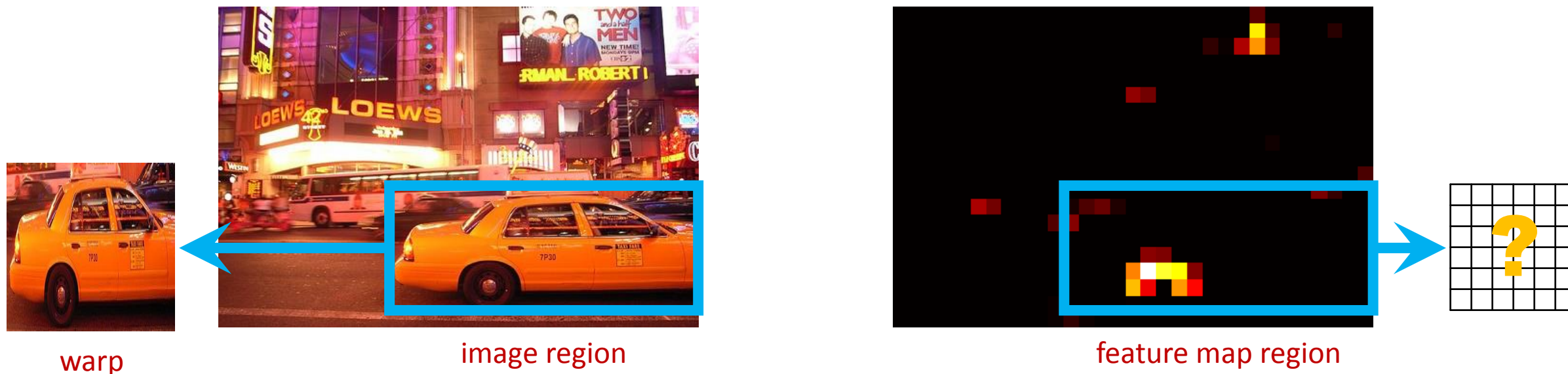
image region



feature map region

- Compute convolutional feature maps on the entire image **only once**.
- Project an image region to a **feature map region** (using correspondence of the receptive field center)
- Extract a region-based feature from the feature map region...

Regions on Feature Maps



- **Fixed-length** features are required by fully-connected layers or SVM
- But how to produce a fixed-length feature from a feature map region?
- Solutions in traditional computer vision: Bag-of-words, SPM...

Bag-of-words & Spatial Pyramid Matching

SIFT/HOG-based
feature maps

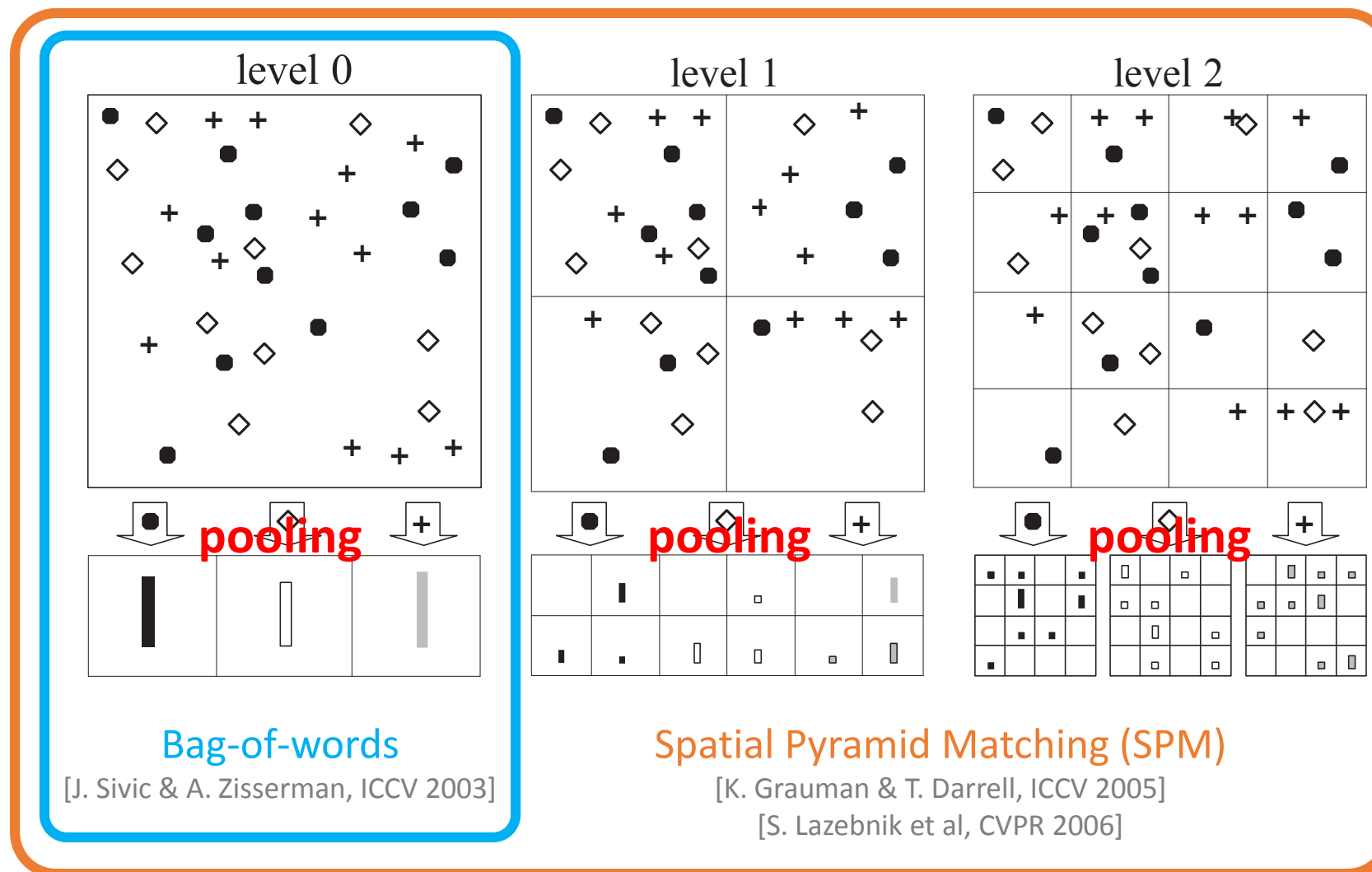
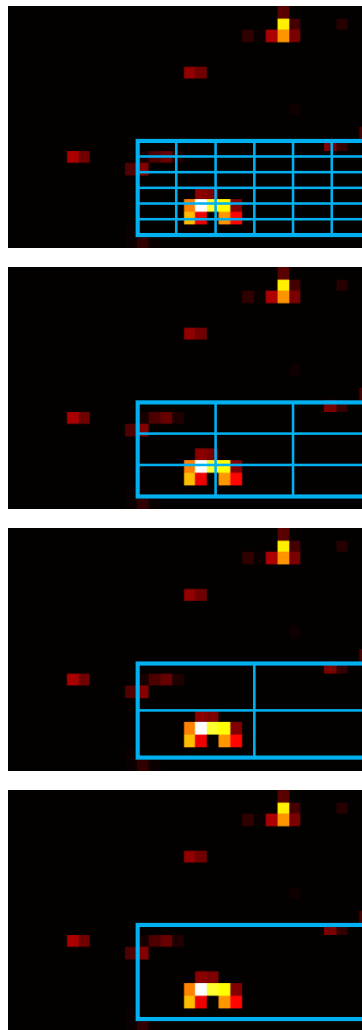
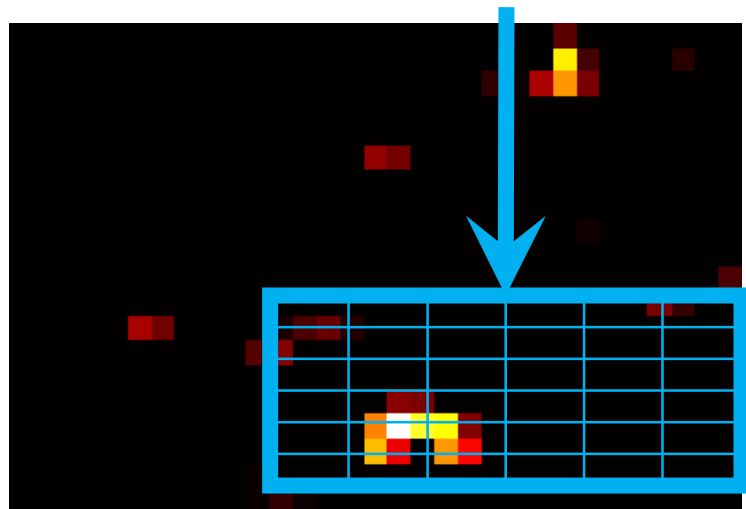


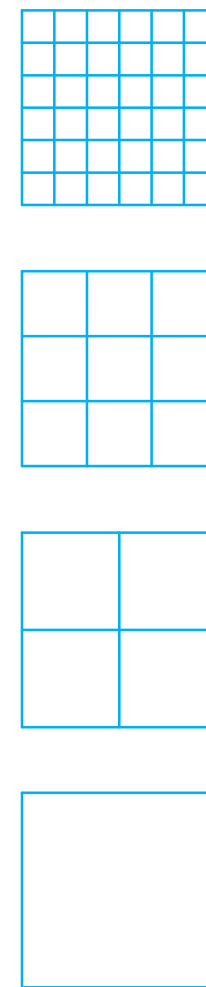
figure credit: S. Lazebnik et al.

Spatial Pyramid Pooling (SPP) Layer

- fix the number of bins (instead of filter sizes)
- **adaptively-sized** bins



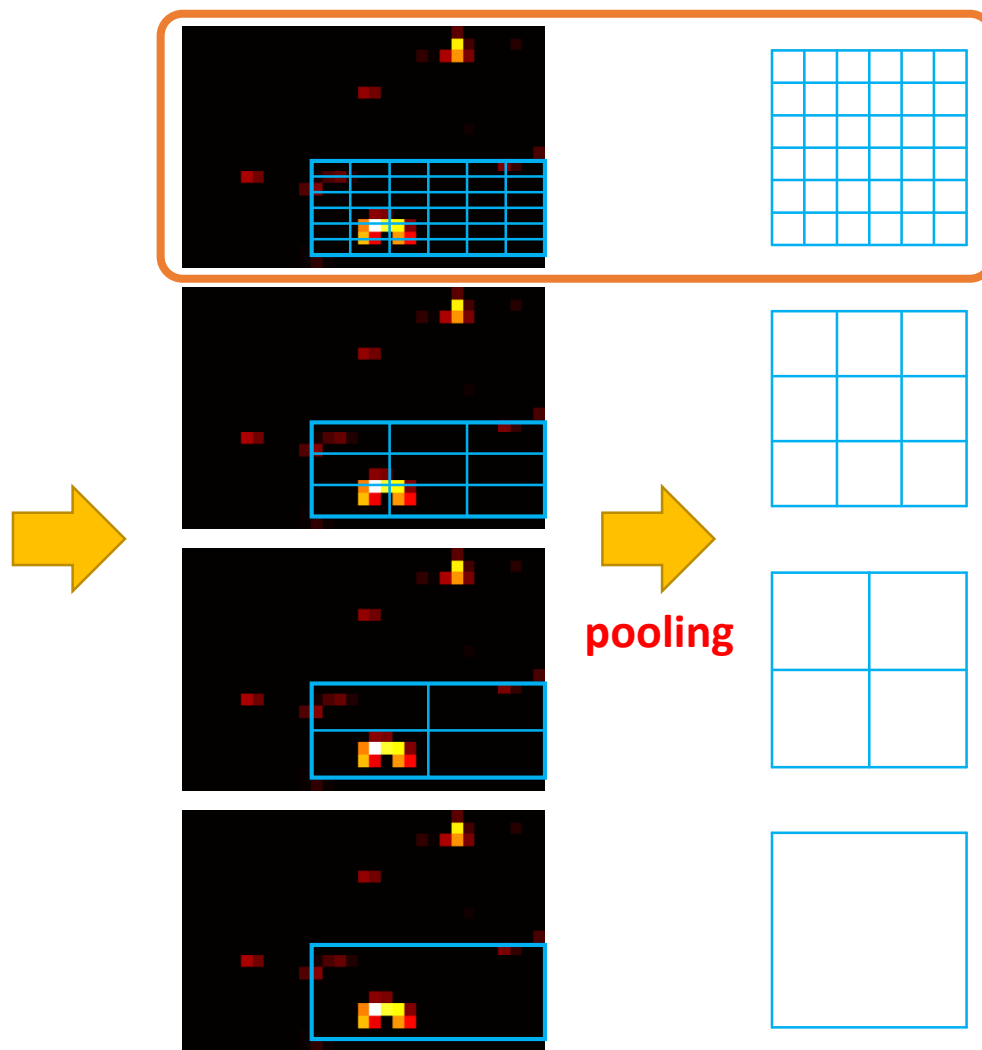
pooling



a finer level maintains explicit spatial information

concatenate,
fc layers...

a coarser level removes explicit spatial information
(bag-of-features)



- Pre-trained nets often have a **single-resolution** pooling layer (7x7 for VGG nets)
 - To adapt to a pre-trained net, a **“single-level”** pyramid is useable
 - Region-of-Interest (RoI) pooling
- [R. Girshick, ICCV 2015]

concatenate,
fc layers...

Single-scale and Multi-scale Feature Maps

- Feature Pyramid
 - Resize the input image to multiple scales
 - Compute feature maps for each scale
 - Used for HOG/SIFT features and convolutional features (OverFeat [Sermanet et al. 2013])



Single-scale and Multi-scale Feature Maps

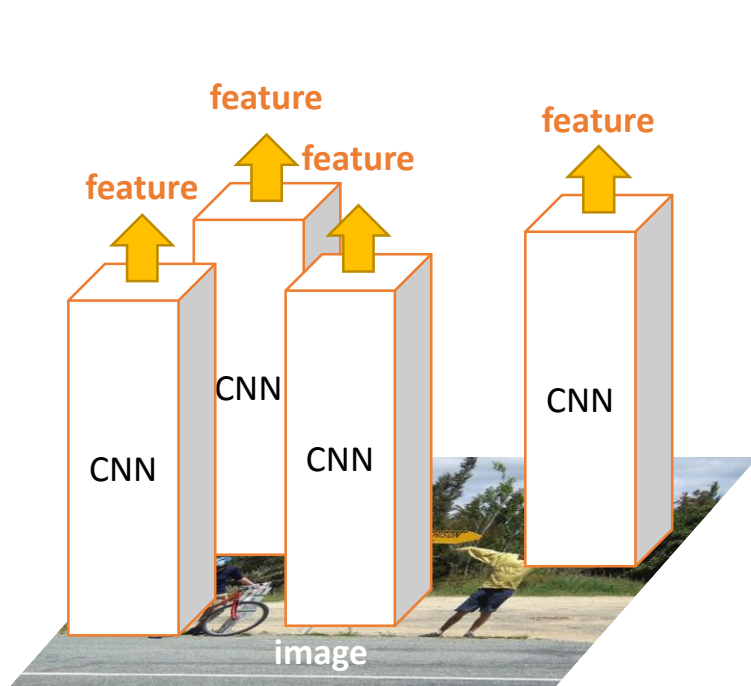
- But deep convolutional feature maps perform well **at a single scale**

| | SPP-net 1-scale | SPP-net 5-scale |
|-------------------------------------|---------------------------|--------------------|
| pool ₅ | 43.0 | 44.9 |
| fc ₆ | 42.5 | 44.8 |
| fine-tuned fc ₆ | 52.3 | 53.7 |
| fine-tuned fc ₇ | 54.5 | 55.2 |
| fine-tuned fc ₇ bbox reg | 58.0 | 59.2 |
| conv time | 0.053s | 0.293s |
| fc time | 0.089s | 0.089s |
| total time | 0.142s | 0.382s |

- Also observed in Fast R-CNN and VGG nets
- Good speed-vs-accuracy tradeoff
- Learn to be scale-invariant from pre-training data (ImageNet)
- (note: but if good accuracy is desired, feature pyramids are still needed)

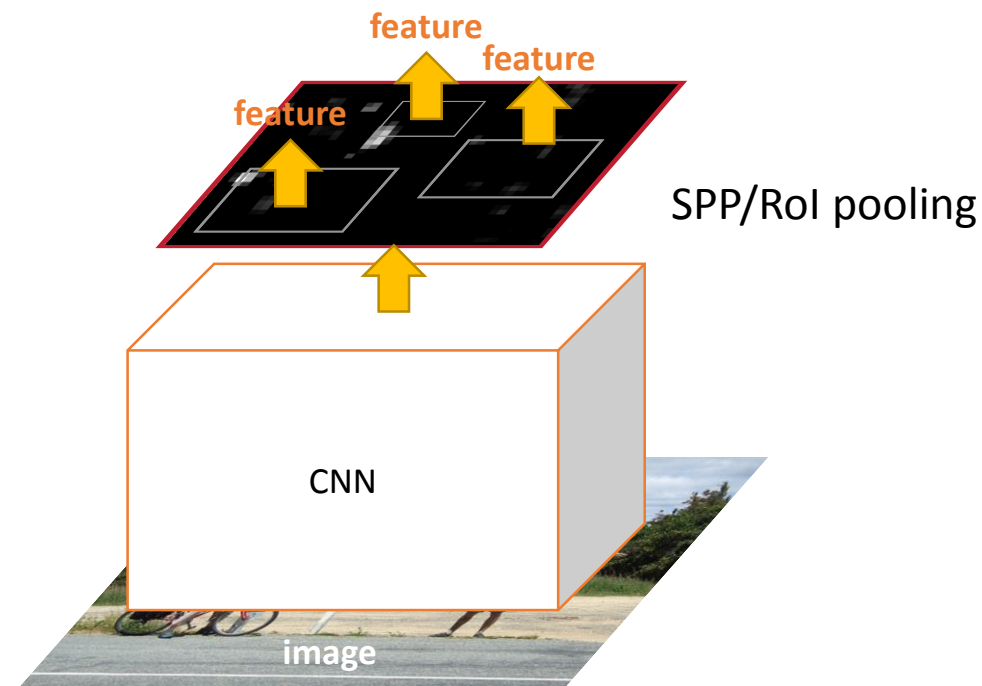
detection mAP on PASCAL VOC 2007, with ZF-net pre-trained on ImageNet
this table is from [K. He, et al. 2014]

R-CNN vs. Fast R-CNN (forward pipeline)



R-CNN

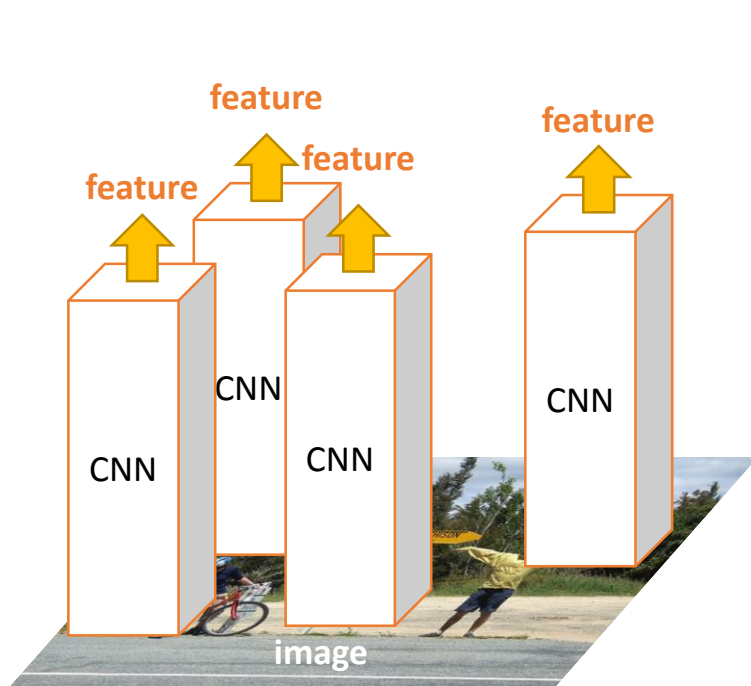
- Extract image regions
- 1 CNN per region (2000 CNNs)
- Classify region-based features



SPP-net & Fast R-CNN (the same forward pipeline)

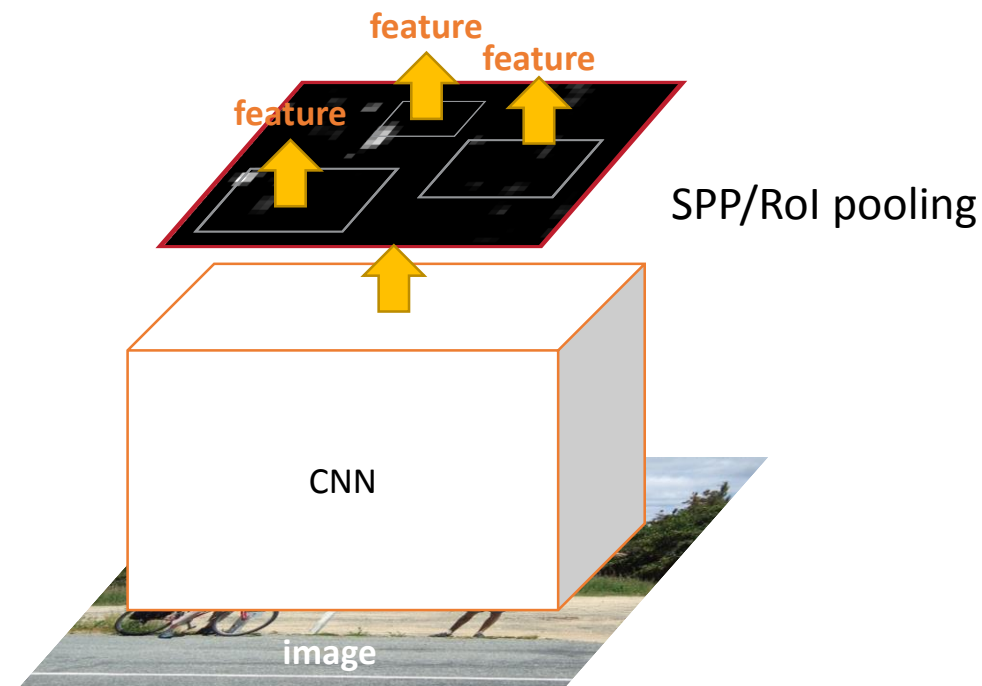
- 1 CNN on the entire image
- Extract features from feature map regions
- Classify region-based features

R-CNN vs. Fast R-CNN (forward pipeline)



R-CNN

- Complexity: $\sim 224 \times 224 \times 2000$



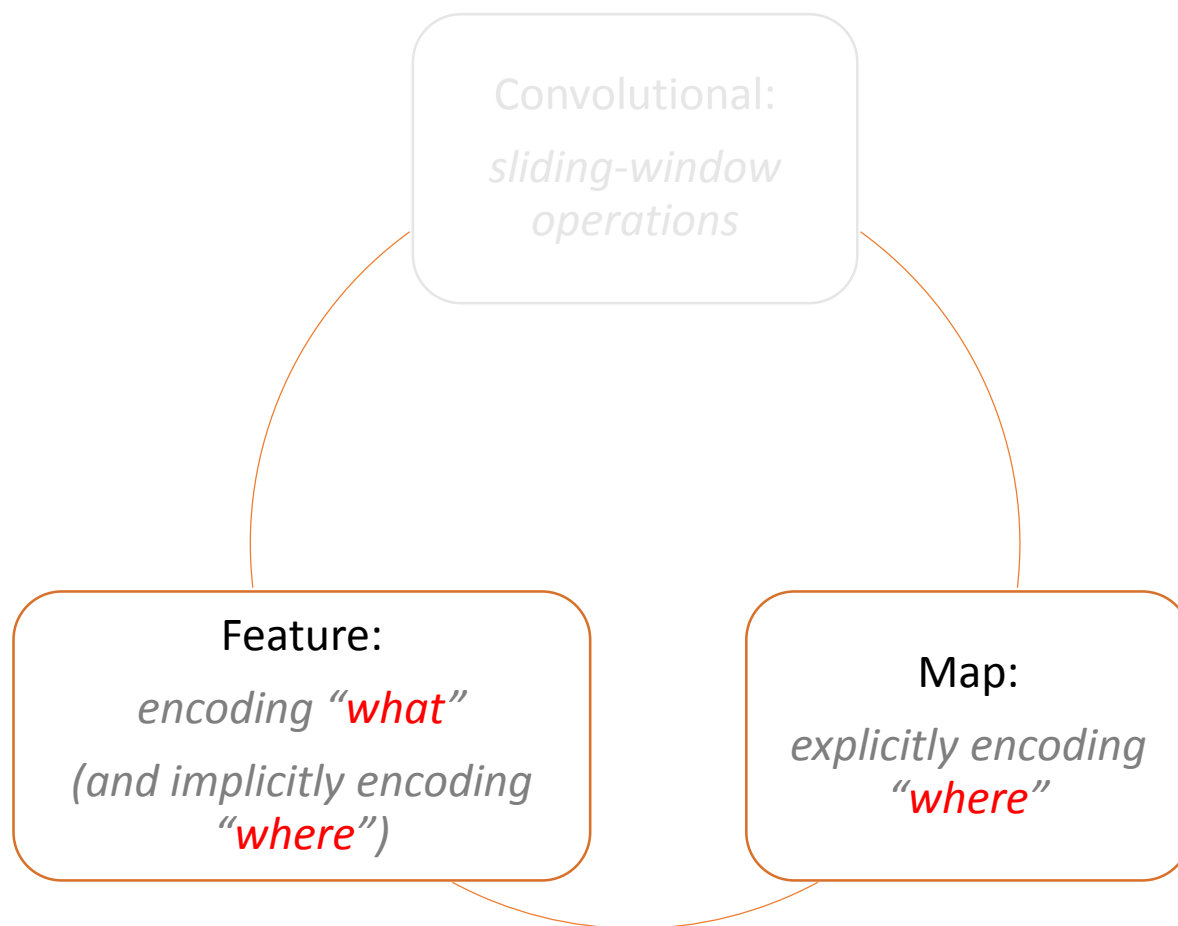
SPP-net & Fast R-CNN (the same forward pipeline)

- Complexity: $\sim 600 \times 1000 \times \mathbf{1}$
- $\sim \mathbf{160x}$ faster than R-CNN

Region Proposal from Feature Maps

- Object detection networks are fast (0.2s)...
- but what about **region proposal**?
 - Selective Search [Uijlings et al. ICCV 2011]: 2s per image
 - EdgeBoxes [Zitnick & Dollar. ECCV 2014]: 0.2s per image
- Can we do region proposal **on the same set of feature maps**?

Feature Maps = features and their locations

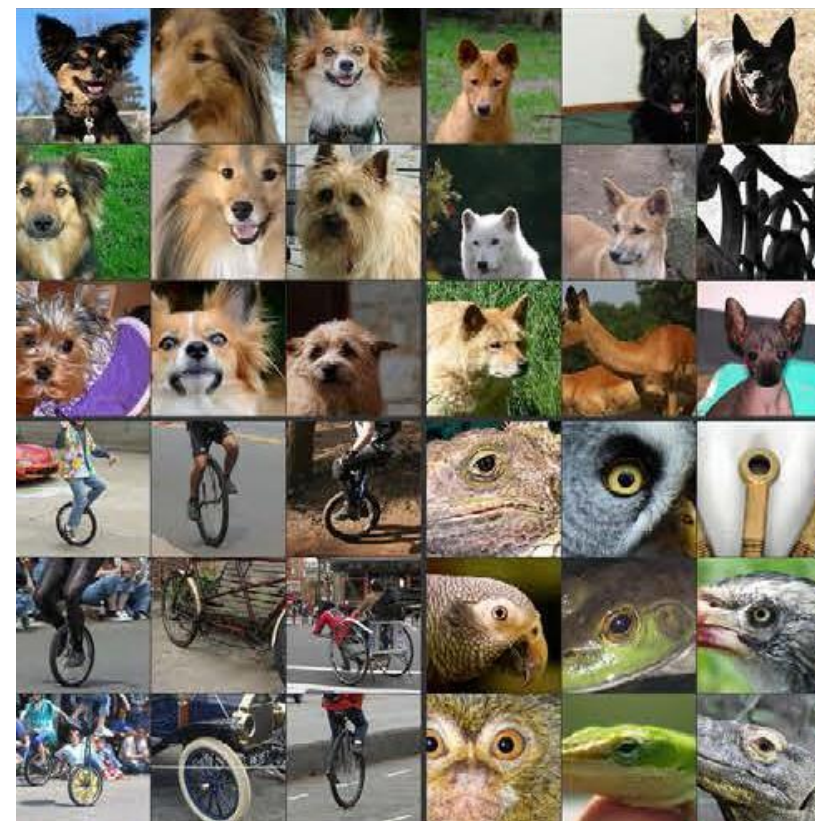
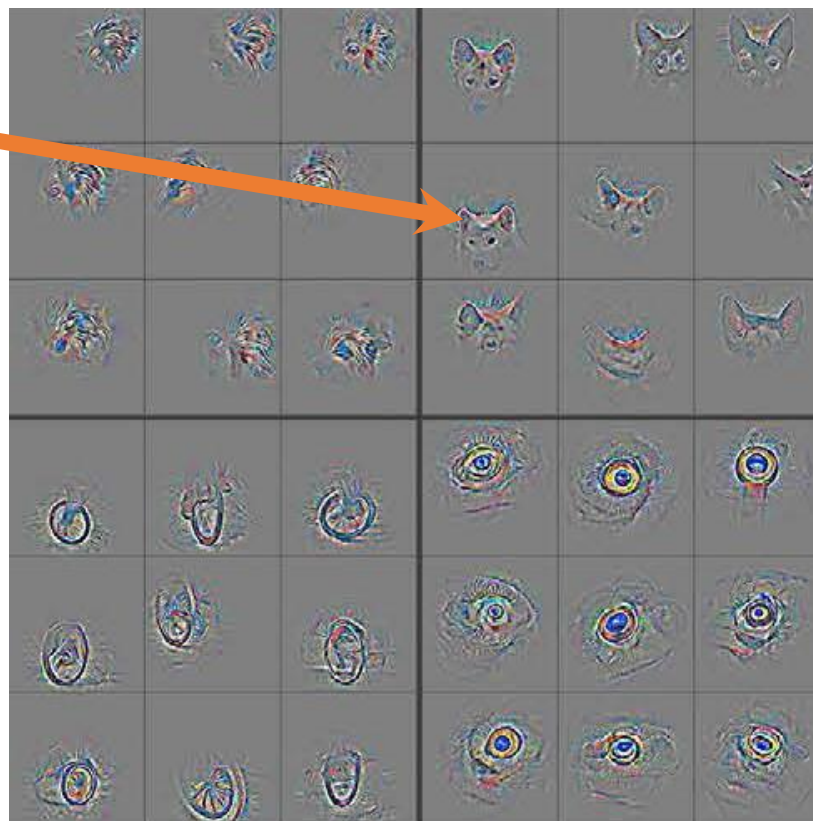


Region Proposal from Feature Maps

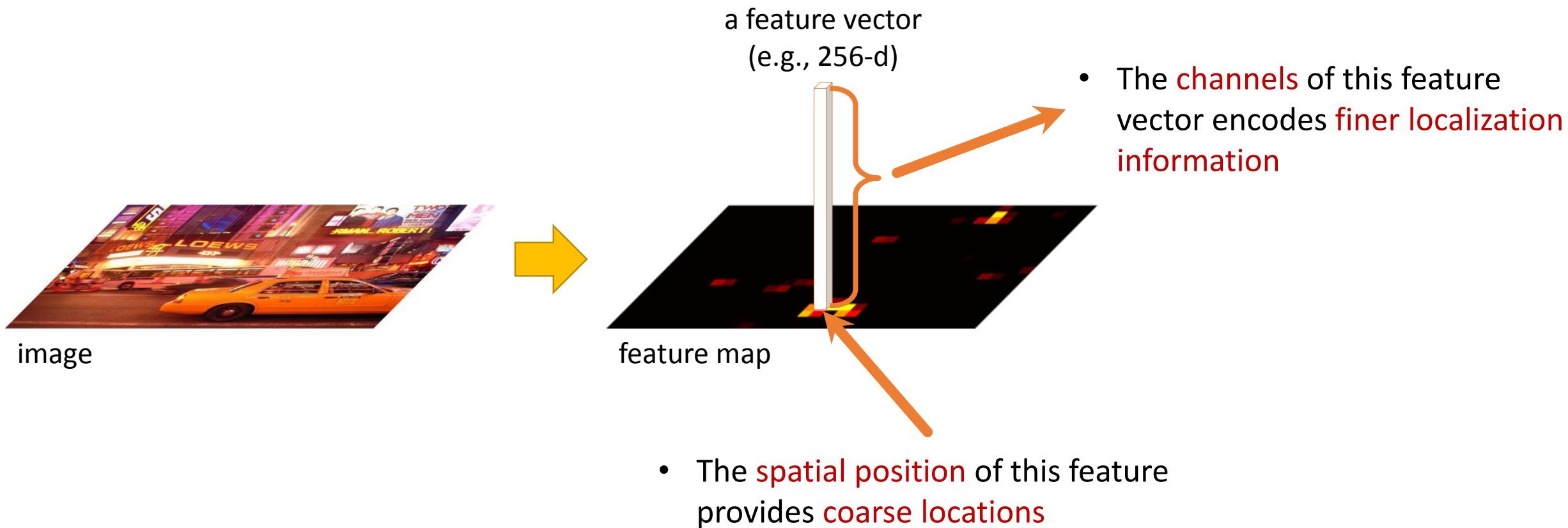
- By decoding **one response** at a single pixel, we can still roughly see the object outline*
- Finer localization information** has been encoded in the channels of a convolutional feature response
- Extract this information for better localization...

* Zeiler & Fergus's method traces unpooling information so the visualization involves more than a single response. But other visualization methods reveal similar patterns.

Revisiting visualizations from Zeiler & Fergus

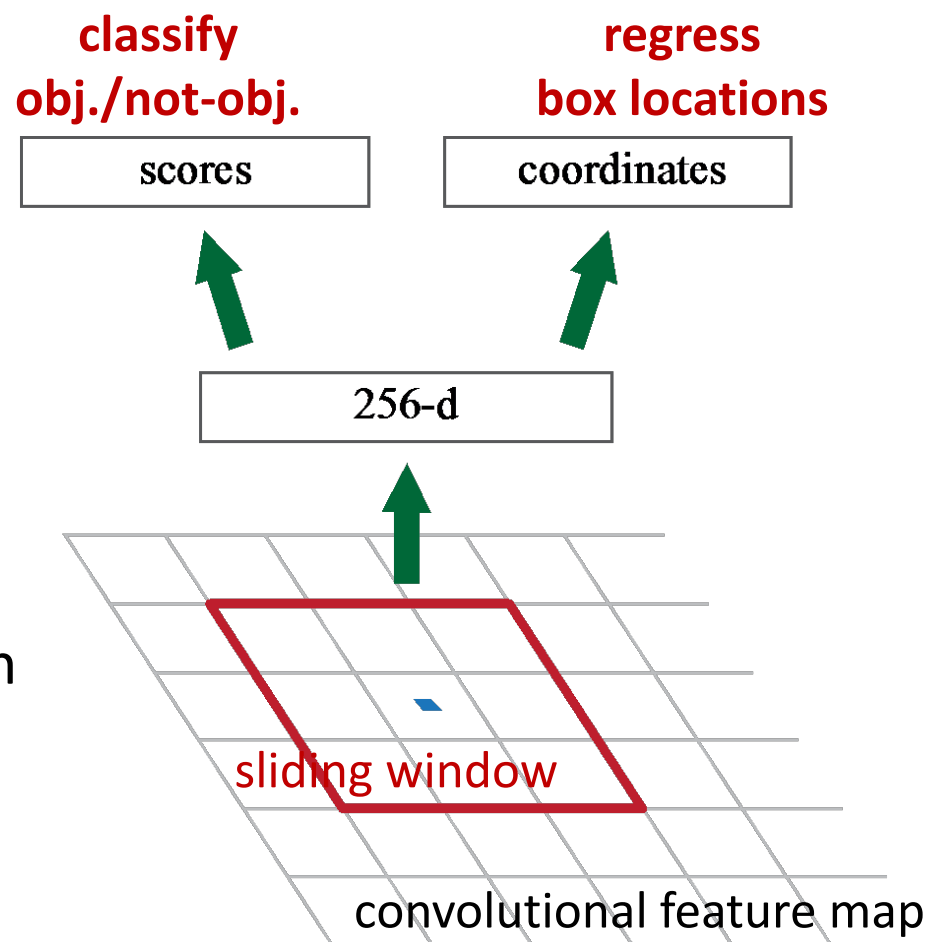


Region Proposal from Feature Maps



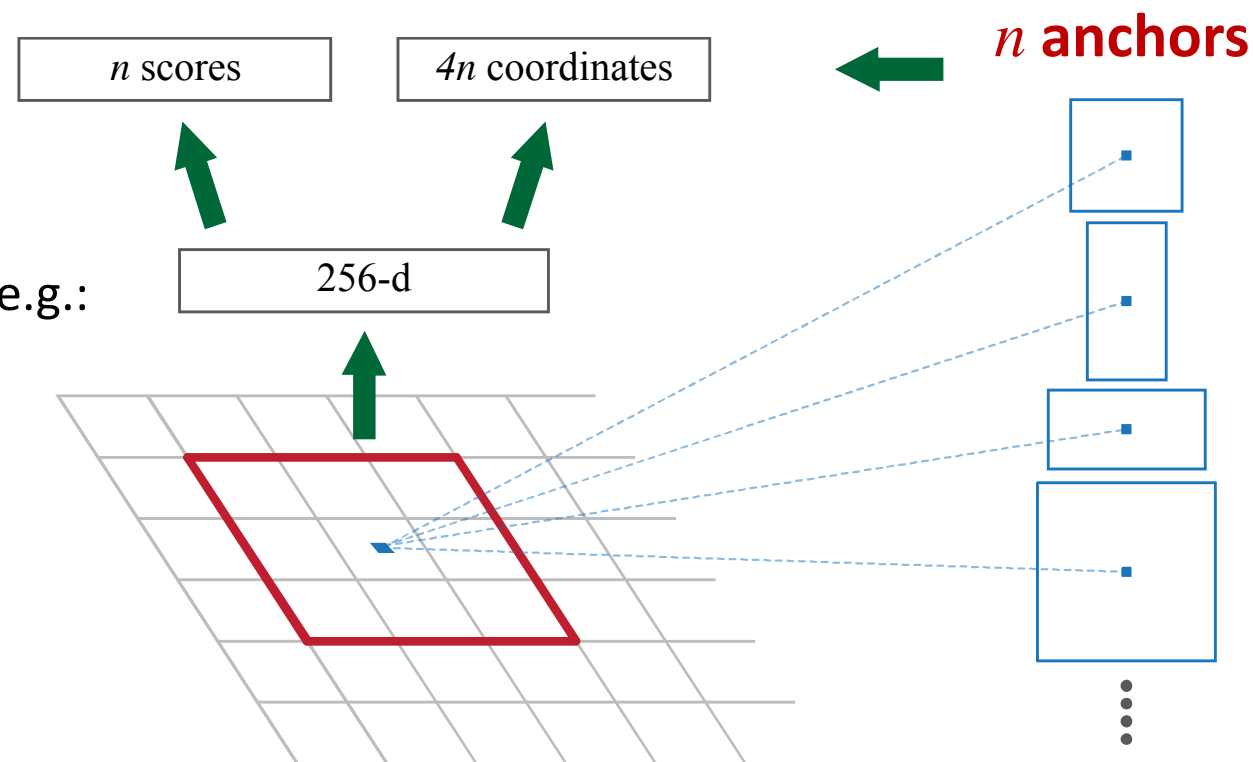
Region Proposal Network

- Slide a small window on the feature map
- Build a small network for:
 - classifying object or not-object, and
 - regressing bbox locations
- Position of the sliding window provides localization information **with reference to the image**
- Box regression provides finer localization information **with reference to this sliding window**



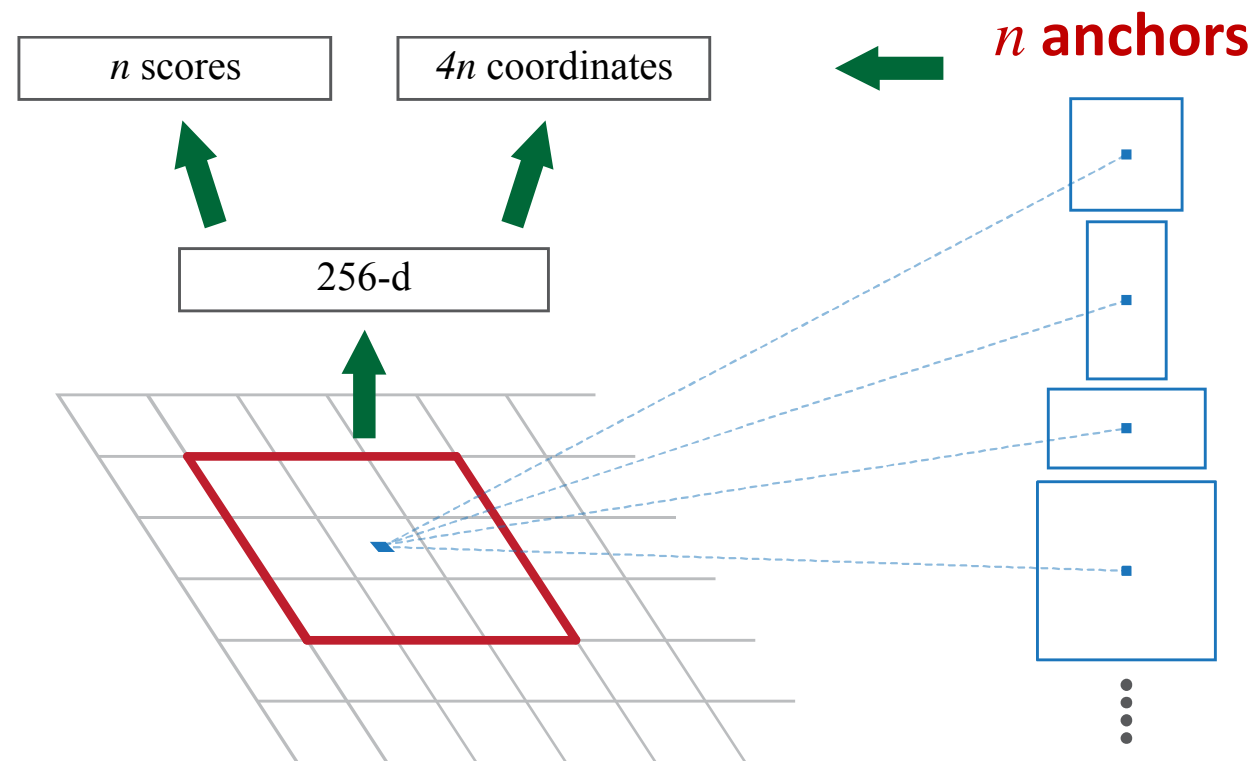
Anchors as references

- **Anchors**: pre-defined reference boxes
 - Box regression is with reference to anchors: regressing an anchor box to a ground-truth box
- Object probability is with reference to anchors, e.g.:
 - anchors as positive samples: if $\text{IoU} > 0.7$ or IoU is max
 - anchors as negative samples: if $\text{IoU} < 0.3$



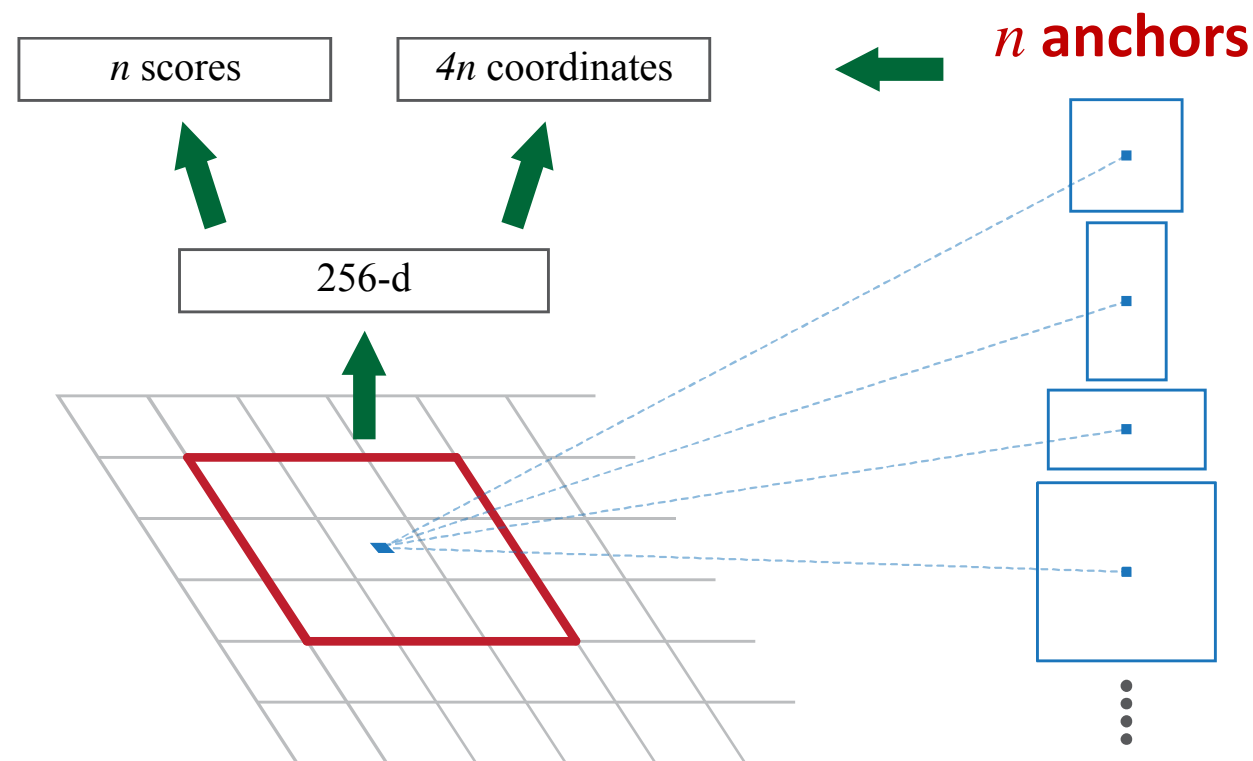
Anchors as references

- **Anchors**: pre-defined reference boxes
- **Translation-invariant** anchors:
 - the same set of anchors are used at each sliding position
 - the same prediction functions (with reference to the sliding window) are used
 - a translated object will have a translated prediction



Anchors as references

- **Anchors**: pre-defined reference boxes
- **Multi-scale/size** anchors:
 - multiple anchors are used at each position:
e.g., 3 scales (128^2 , 256^2 , 512^2) and 3 aspect ratios (2:1, 1:1, 1:2) yield 9 anchors
 - each anchor has its own prediction function
 - **single-scale** features, multi-scale predictions



Anchors as references

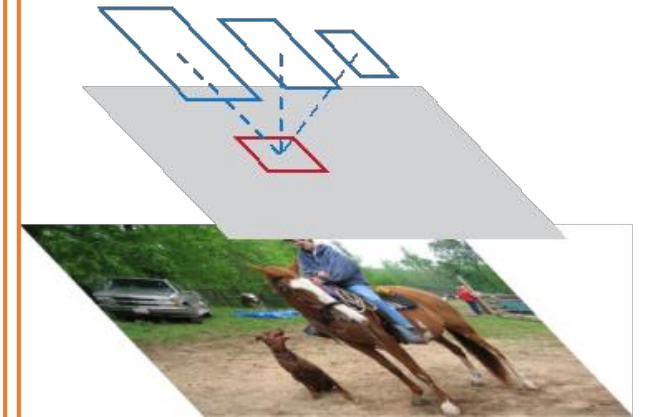
- Comparisons of **multi-scale** strategies



Image/Feature Pyramid



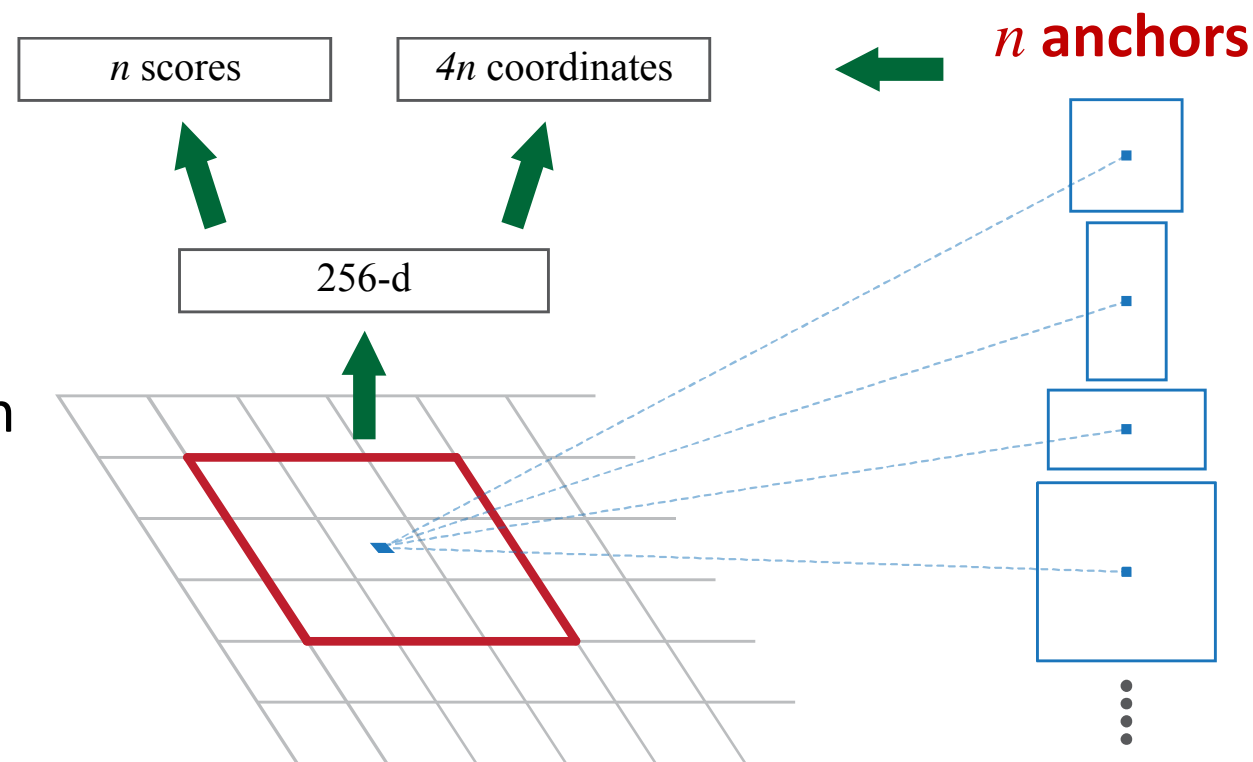
Filter Pyramid



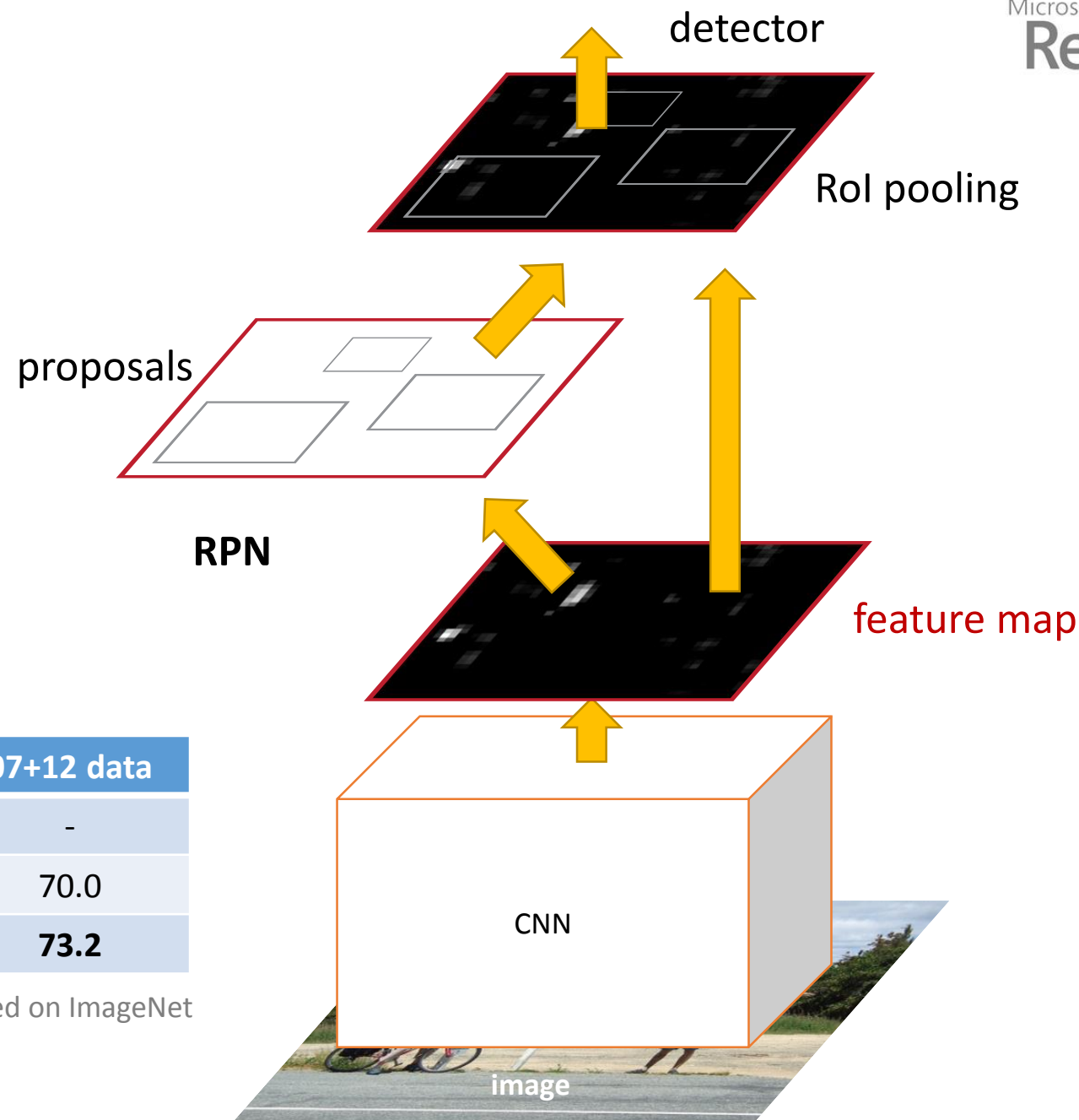
Anchor Pyramid

Region Proposal Network

- RPN is **fully convolutional** [Long et al. 2015]
- RPN is trained end-to-end
- RPN **shares** convolutional feature maps with the detection network (covered in Ross's section)



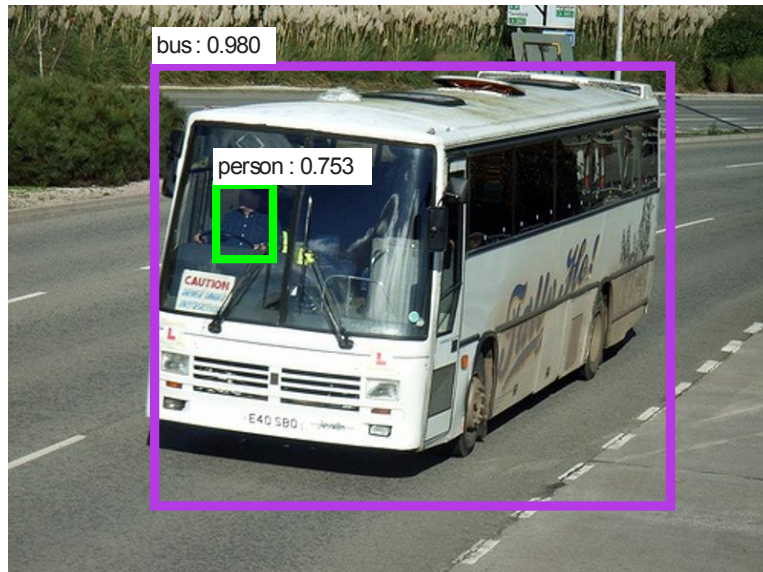
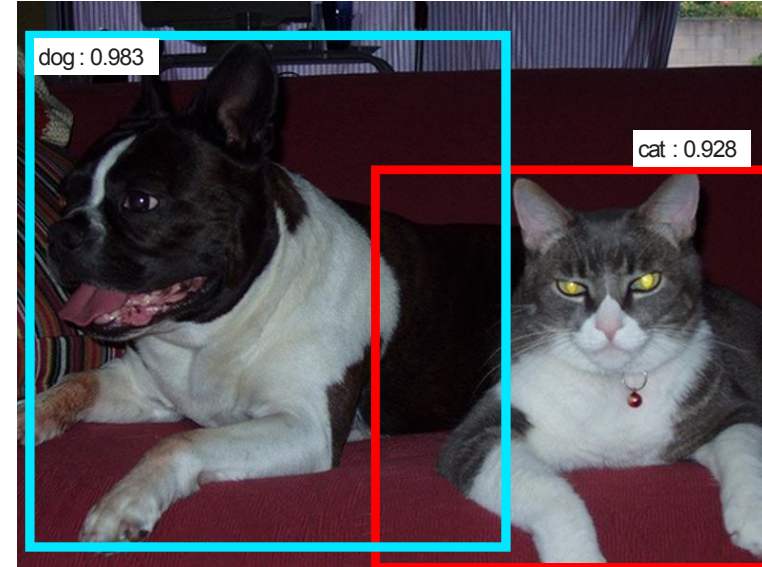
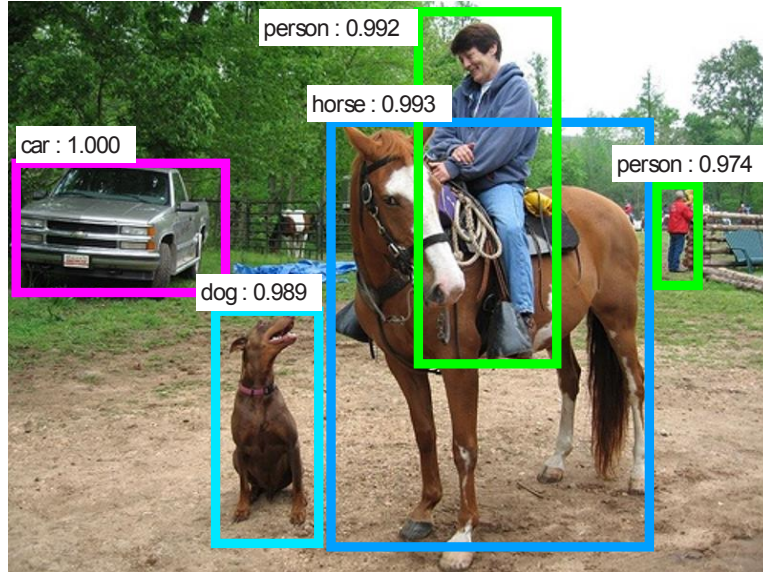
Faster R-CNN



| system | time | 07 data | 07+12 data |
|--------------|--------------|-------------|-------------|
| R-CNN | ~50s | 66.0 | - |
| Fast R-CNN | ~2s | 66.9 | 70.0 |
| Faster R-CNN | 198ms | 69.9 | 73.2 |

detection mAP on PASCAL VOC 2007, with VGG-16 pre-trained on ImageNet

Example detection results of Faster R-CNN



Keys to efficient CNN-based object detection

- Feature **sharing**
 - R-CNN => SPP-net & Fast R-CNN: sharing features **among proposal regions**
 - Fast R-CNN => Faster R-CNN: sharing features **between proposal and detection**
 - All are done by shared **convolutional feature maps**
- Efficient multi-scale solutions
 - **Single-scale** convolutional feature maps are good trade-offs
 - **Multi-scale anchors** are fast and flexible

Conclusion of this section

- Quick introduction to convolutional feature maps
 - Intuitions: into the “black boxes”
 - How object detection networks & region proposal networks are designed
 - Bridging the gap between “hand-engineered” and deep learning systems
- Focusing on forward propagation (inference)
 - Backward propagation (training) covered by Ross’s section