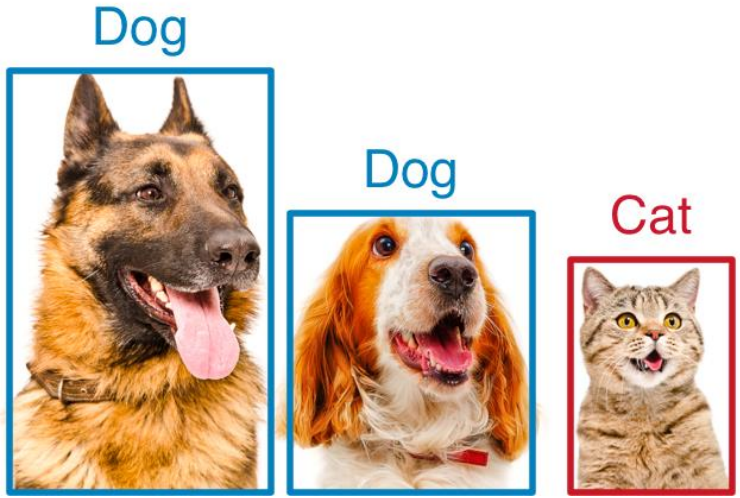
Advanced Deep Learning Architectures
COMP 5214 & ELEC 5680

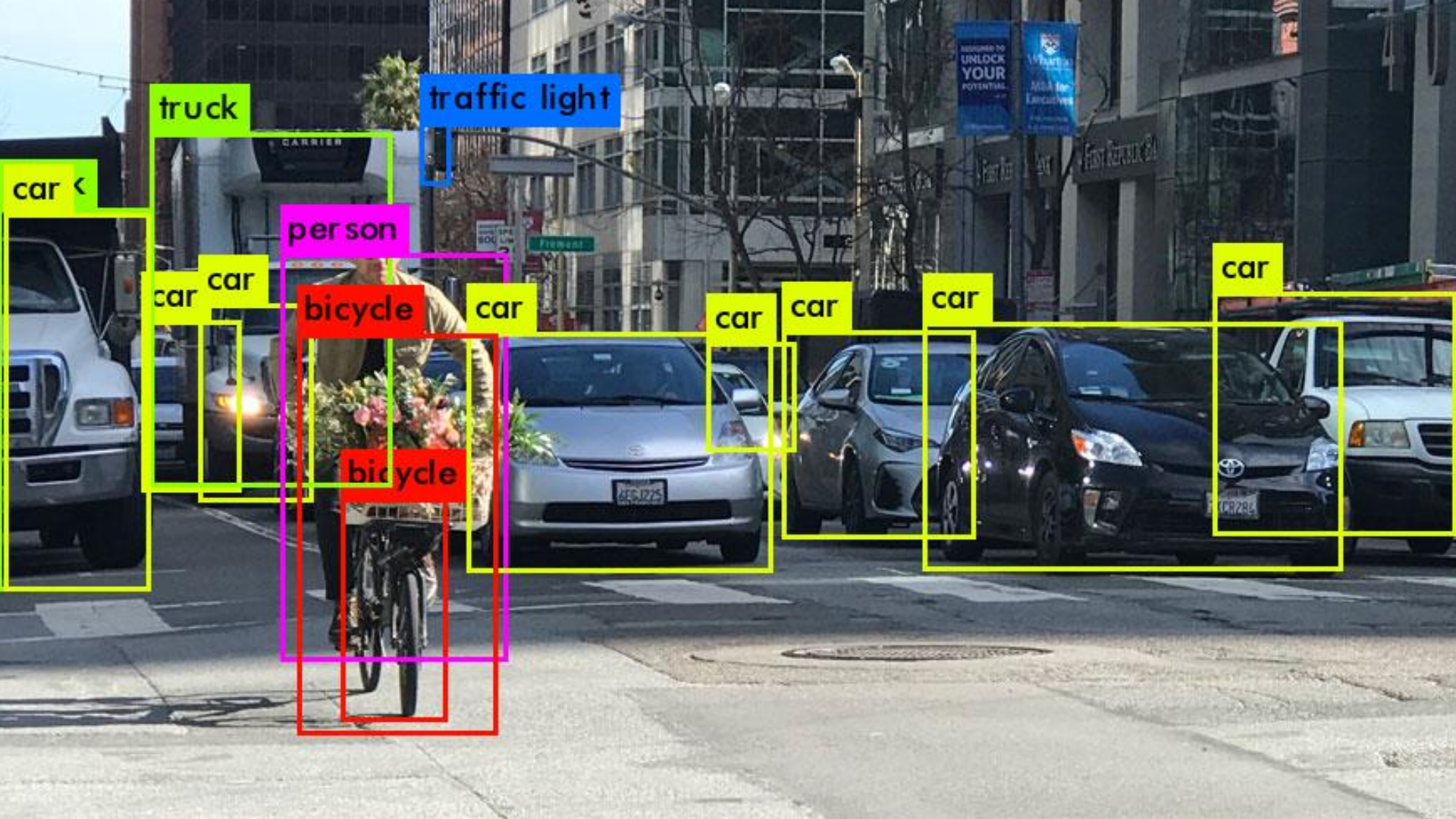
- Instructor: Dr. Qifeng Chen
- https://cqf.io

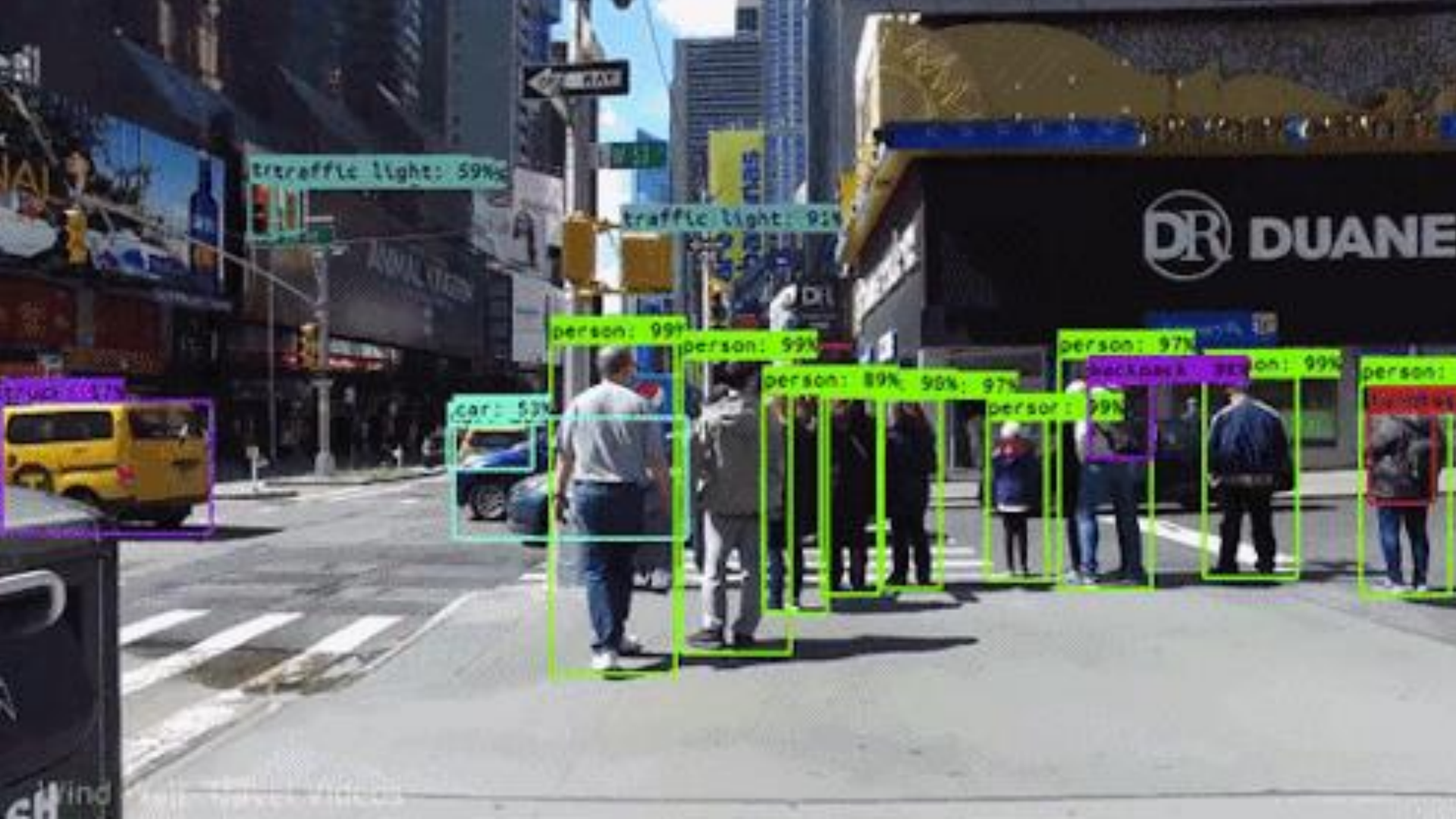
Credit: https://d2l.ai/

Image classification

Object Detection

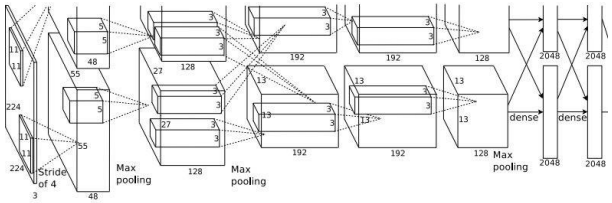



truck
traffic light
car
person
car
car
bicycle
car
car
car
car
car
bicycle


DR DUANE
traffic light: 59%
traffic light: 91%
person: 99%
person: 99%
person: 89%
person: 98%
person: 97%
person: 97%
person: 99%
car: 53%

Classification + Localization

This image is CC0 public domain



Fully Connected: 4096 to 1000

Class Scores
Cat: 0.9
Dog: 0.05
Car: 0.01
...

Vector: 4096

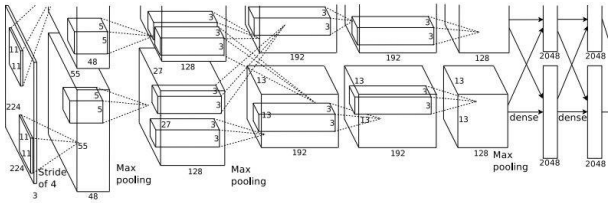
Fully Connected: 4096 to 4

Box Coordinates
(x, y, w, h)

Treat localization as a regression problem!

Classification + Localization

This image is CC0 public domain



Fully Connected: 4096 to 1000

Class Scores
Cat: 0.9
Dog: 0.05
Car: 0.01
...

Softmax Loss

Correct label: Cat

Vector: 4096

Fully Connected: 4096 to 4

Box Coordinates (x, y, w, h)

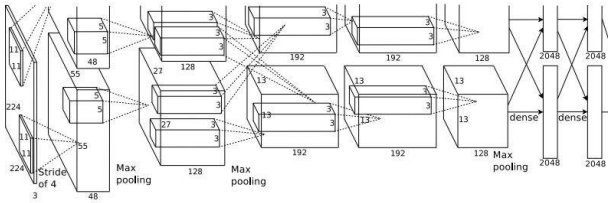
L2 Loss

Correct box: (x', y', w', h')

Treat localization as a regression problem!

Classification + Localization

This image is CC0 public domain



Fully Connected: 4096 to 1000

Class Scores
Cat: 0.9
Dog: 0.05
Car: 0.01
...

Softmax Loss

Correct label: Cat

Vector: 4096

Fully Connected: 4096 to 4

Box Coordinates (x, y, w, h)

L2 Loss

Correct box: (x', y', w', h')

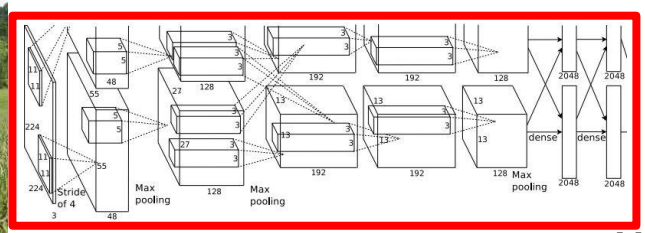
Multitask Loss

+ → **Loss**

Treat localization as a regression problem!

Classification + Localization

This image is CC0 public domain



Often pretrained on ImageNet (Transfer learning)

Vector: 4096

Fully Connected: 4096 to 1000

Class Scores
Cat: 0.9
Dog: 0.05
Car: 0.01
...

Softmax Loss

Correct label: Cat

Fully Connected: 4096 to 4

Box Coordinates (x, y, w, h)

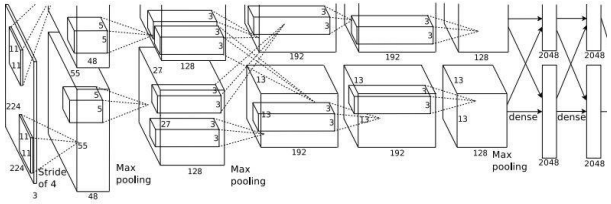
L2 Loss

Correct box: (x', y', w', h')

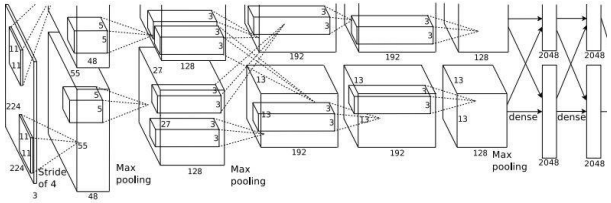
+ → **Loss**

Treat localization as a regression problem!

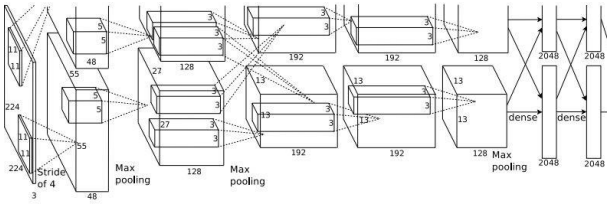
Object Detection as Regression?



CAT: (x, y, w, h)



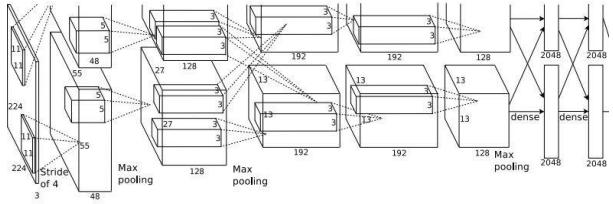
DOG: (x, y, w, h)
DOG: (x, y, w, h)
CAT: (x, y, w, h)



DUCK: (x, y, w, h)
DUCK: (x, y, w, h)
....

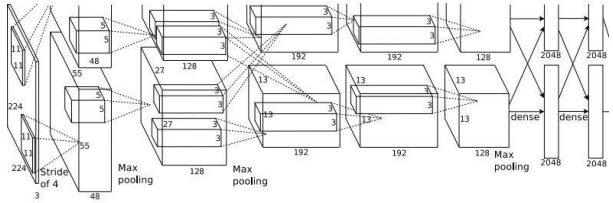
Object Detection as Regression?

Each image needs a different number of outputs!



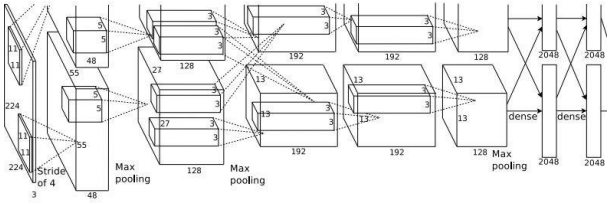
CAT: (x, y, w, h)

4 numbers



DOG: (x, y, w, h)
DOG: (x, y, w, h)
CAT: (x, y, w, h)

12 numbers

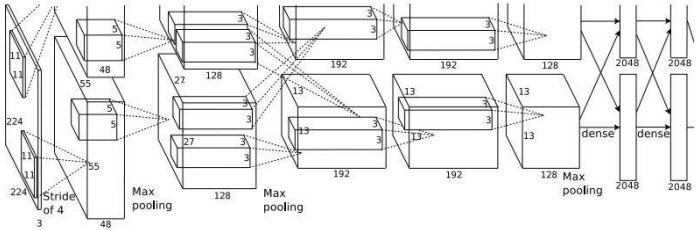


DUCK: (x, y, w, h)
DUCK: (x, y, w, h)
….

Many numbers!

Object Detection as Classification: Sliding Window

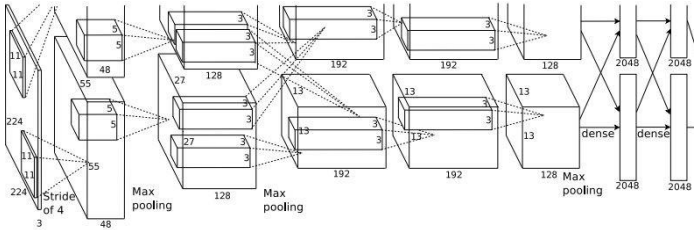
Apply a CNN to many different crops of the image, CNN classifies each crop as object or background



Dog? NO
Cat? NO
Background? YES

Object Detection as Classification: Sliding Window

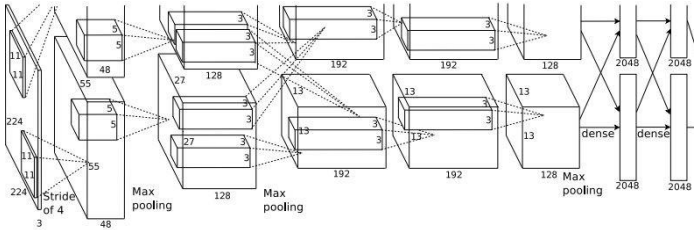
Apply a CNN to many different crops of the image, CNN classifies each crop as object or background



Dog? YES
Cat? NO
Background? NO

Object Detection as Classification: Sliding Window

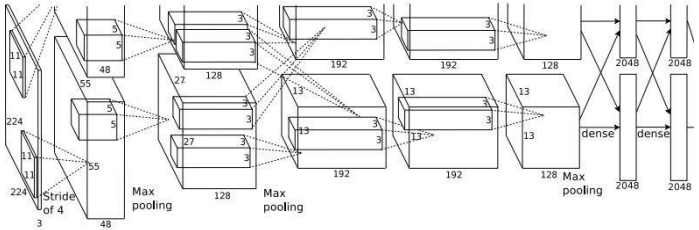
Apply a CNN to many different crops of the image, CNN classifies each crop as object or background



Dog? YES
Cat? NO
Background? NO

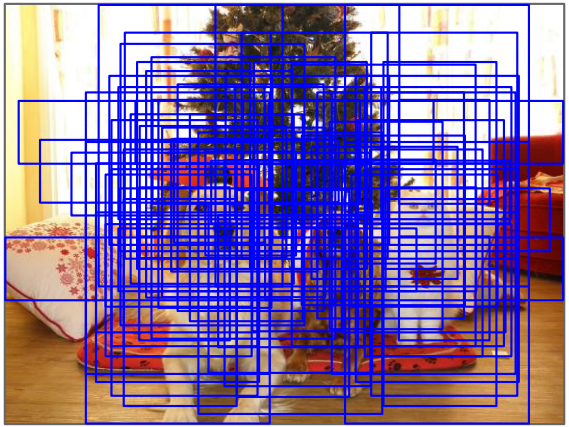
Object Detection as Classification: Sliding Window

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

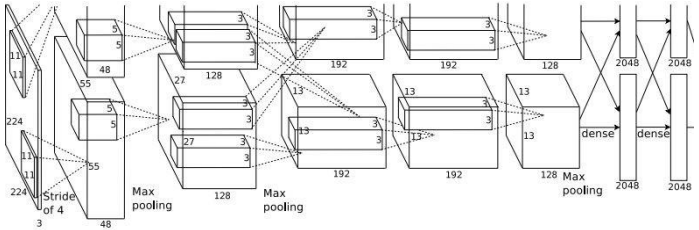


Dog? NO
Cat? YES
Background? NO

Object Detection as Classification: Sliding Window



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

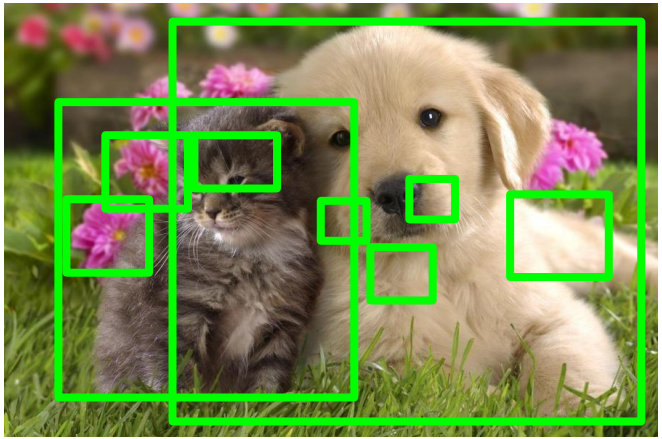


- Dog?
  NO
  Cat?
  YES
- Background
  ? NO

- Problem: Need to apply CNN to huge number of locations, scales, and aspect ratios, very computationally expensive!

Region Proposals / Selective Search

- Find “blobby” image regions that are likely to contain objects
- Relatively fast to run; e.g. Selective Search gives 2000 region proposals in a few seconds on CPU

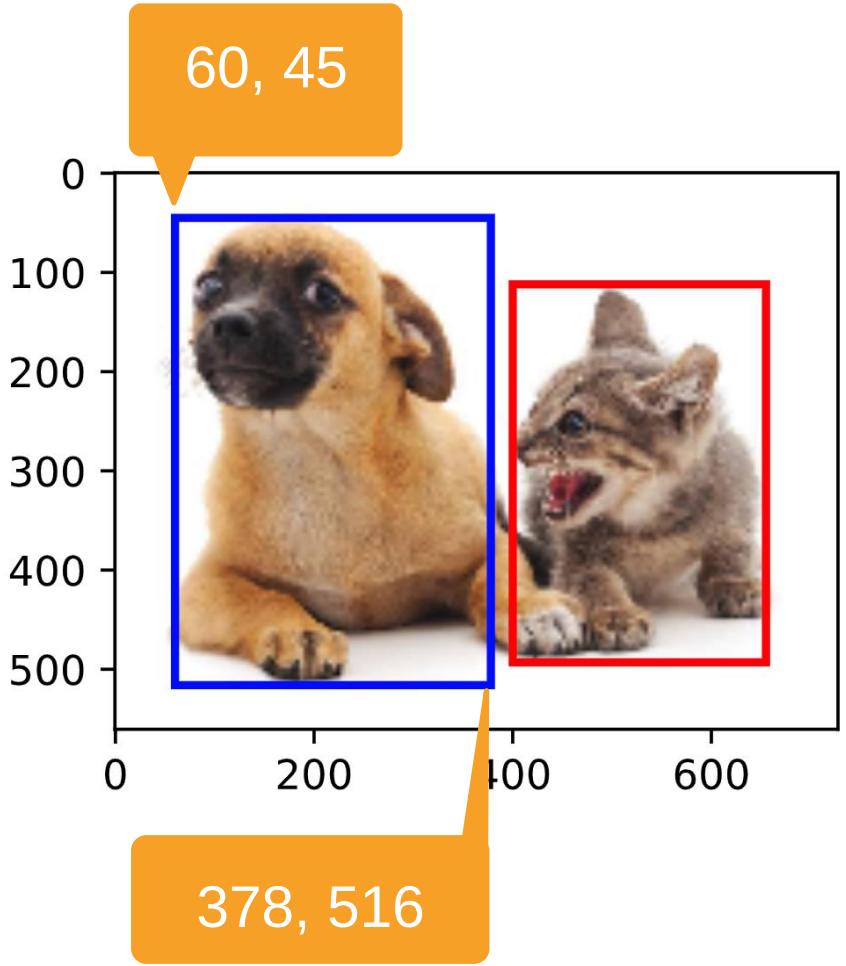


Alexe et al, “Measuring the objectness of image windows”, TPAMI 2012
Uijlings et al, “Selective Search for Object Recognition”, IJCV 2013
Cheng et al, “BING: Binarized normed gradients for objectness estimation at 300fps”, CVPR 2014
Zitnick and Dollar, “Edge boxes: Locating object proposals from edges”, ECCV 2014

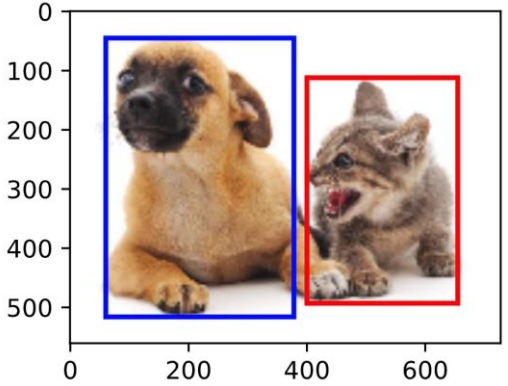
Bounding and Anchor Boxes

Bounding Box

- A bounding box can be defined by 4 numbers,
  - (top-left x,
    top-left y,
    bottom-right x,
    bottom-right y)
  - (top-left x,
    top-right y,
    width, height)



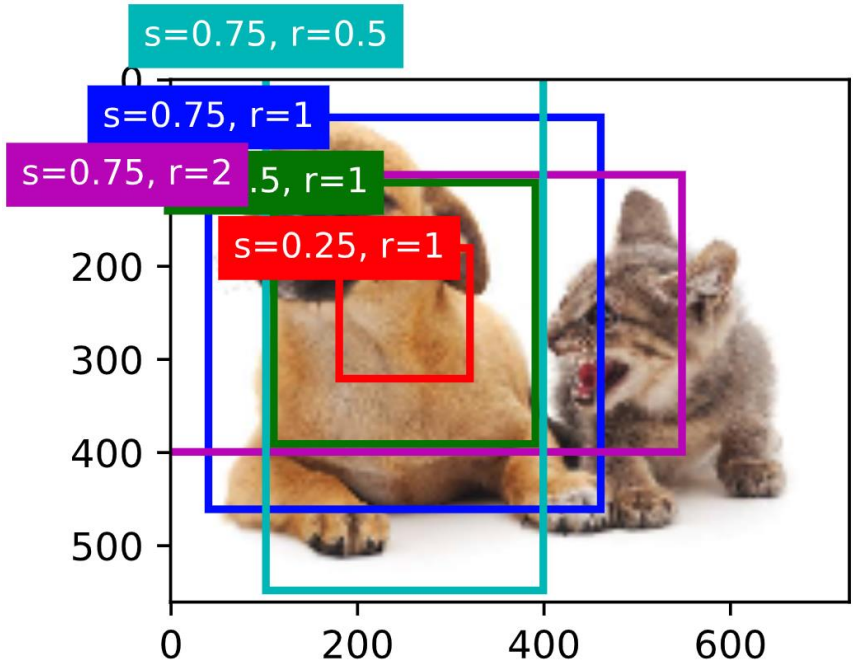
Object Detection Dataset



- Each row present an object
  - Image_filename, object_category, bounding box
- COCO (cocodataset.org)
  - 80 objects
  - 330K images
  - 1.5M objects

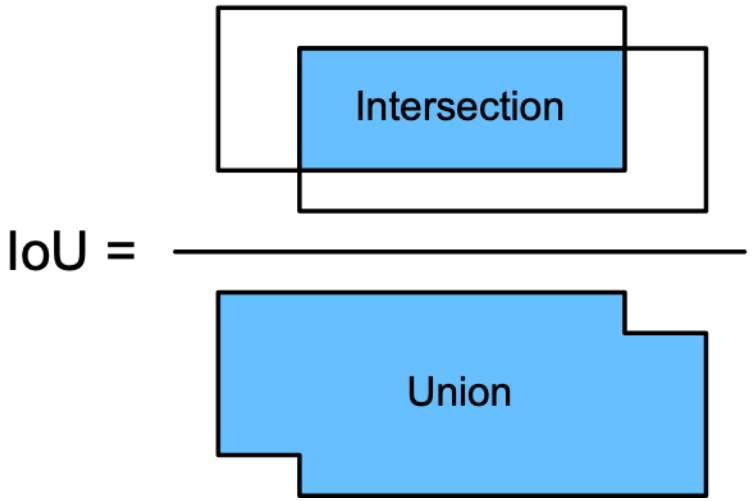
Anchor Boxes

- A detection algorithm often
  - Proposes multiple regions, called anchor boxes
  - Predict if an anchor box contains an object
  - If yes, predict the offset from the anchor box to the ground truth bounding box



IoU - Intersection over Union

- IoU measures the similarity between two boxes
  - 0 means no-overlapping
  - 1 means identical
- It's an especial case of Jacquard index
  - Given sets *A* and *B*

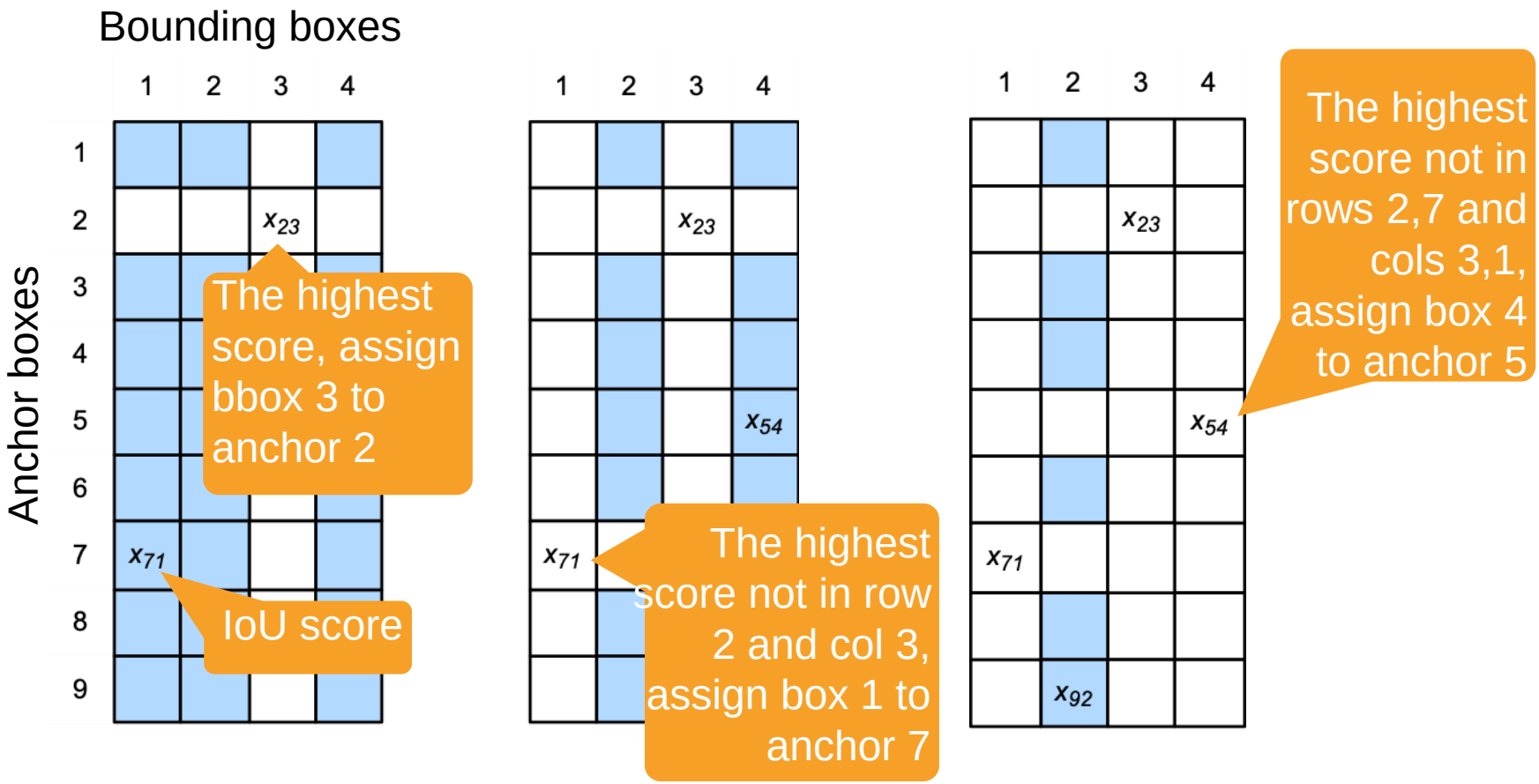


$$J(A,B) = \frac{|A \cap B|}{|A \cup B|}$$

Assign Labels to Anchor Boxes

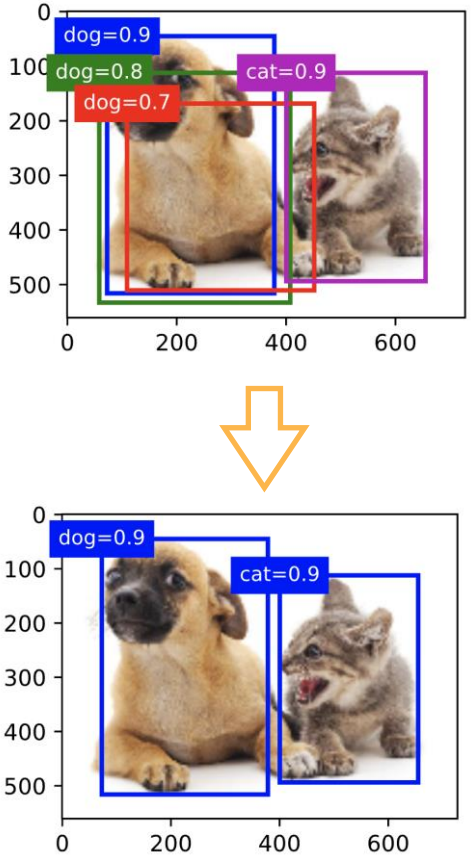
- Each anchor box is a training example
- Label each anchor box with
  - Background
  - Associate with a bounding box
- We may generate a large amount of anchor boxes
  - Leads to a large portion of negative examples

Assign Labels to Anchor Boxes

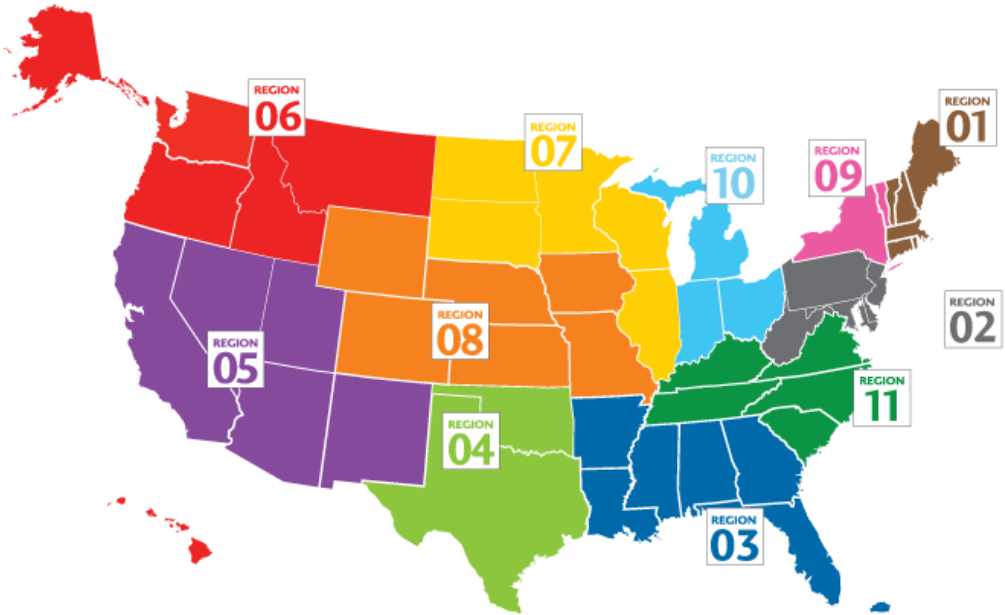


Output with non-maximum suppression (NMS)

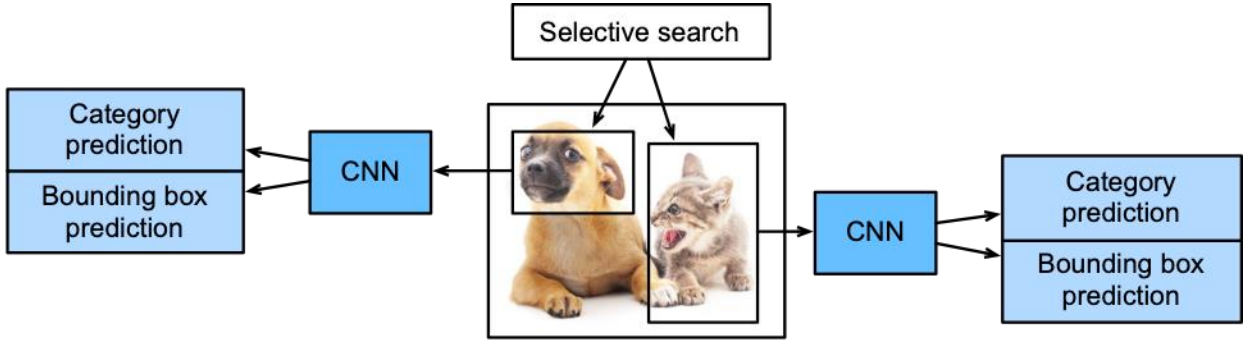
- Each anchor box generates one bounding box prediction
- Select the one with the highest score (not background)
- Remove all other predictions with IoU > θ compared to the selected one
- Repeat until all are selected or removed



Region-based CNNs



R-CNN



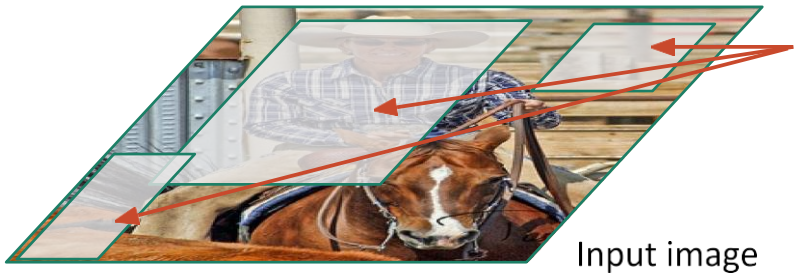
- Select anchor boxes with a heuristic algorithm
- Use a pre-trained networks to extract features for each anchor box
- Train a SVM to classify category
- Train a linear regression to predict bounding boxes

R-CNN

Input image

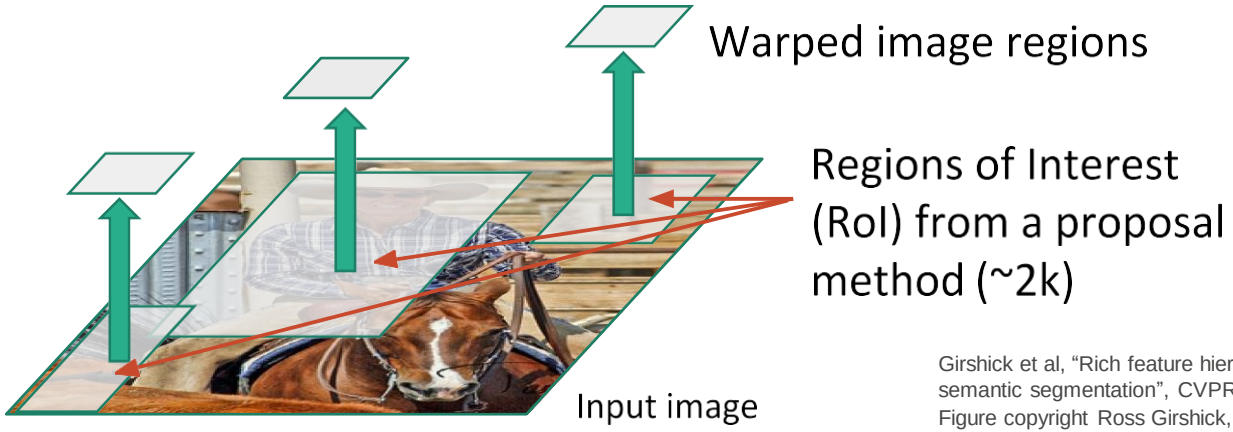
Girshick et al, “Rich feature hierarchies for accurate object detection and semantic segmentation”, CVPR 2014.

R-CNN



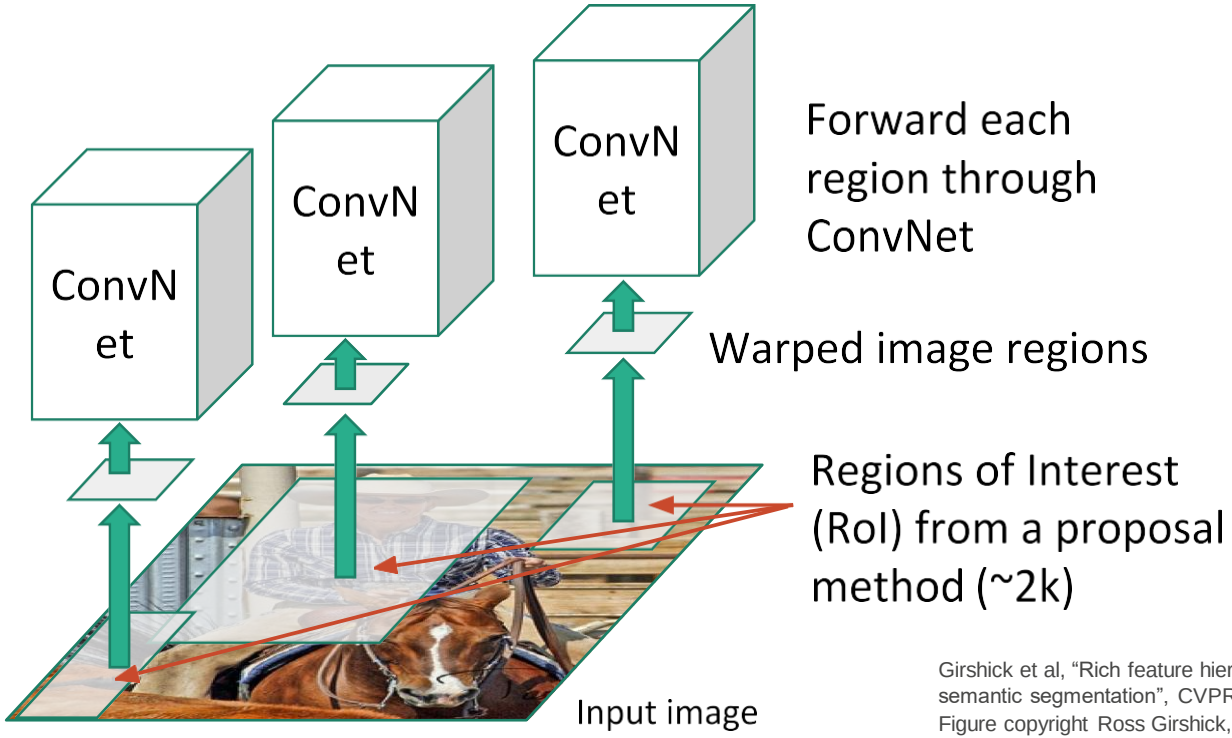
Girshick et al, “Rich feature hierarchies for accurate object detection and semantic segmentation”, CVPR 2014.
Figure copyright Ross Girshick, 2015; source. Reproduced with permission.

R-CNN



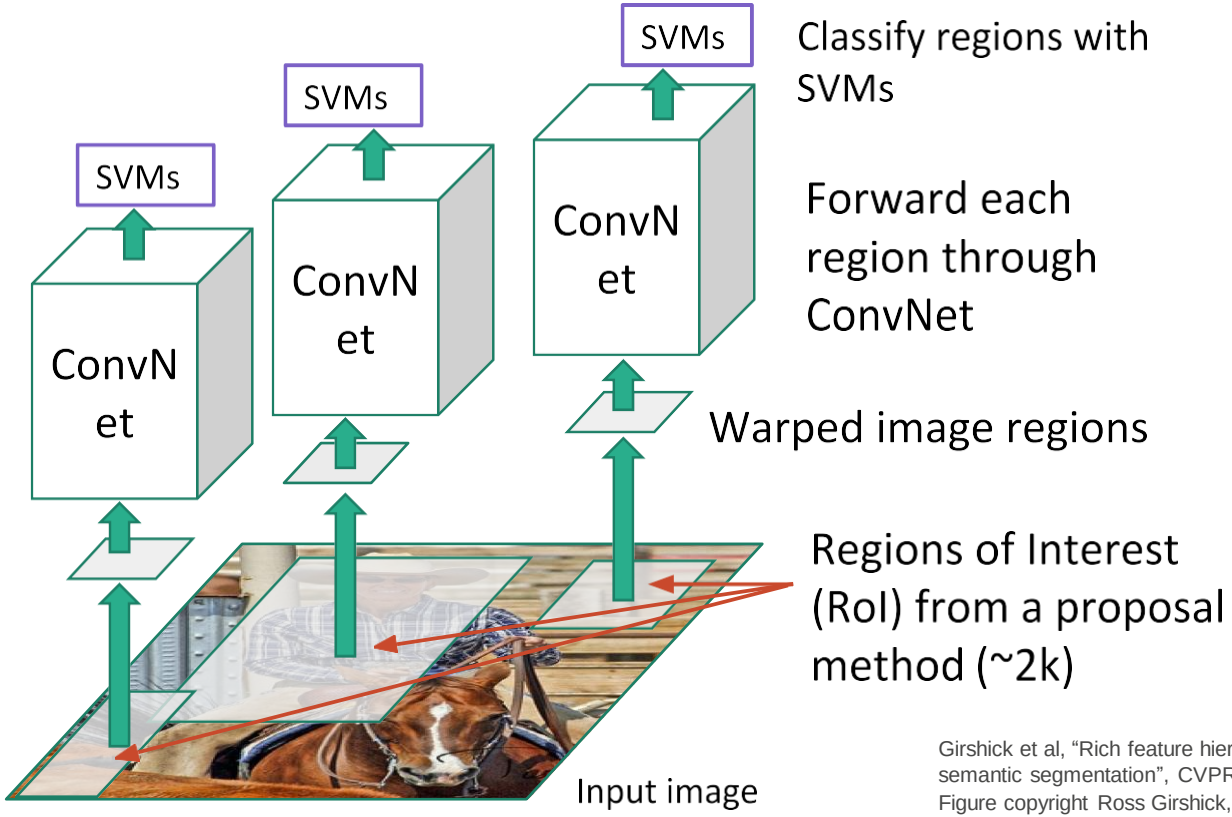
Girshick et al, “Rich feature hierarchies for accurate object detection and semantic segmentation”, CVPR 2014.
Figure copyright Ross Girshick, 2015; source. Reproduced with permission.

R-CNN



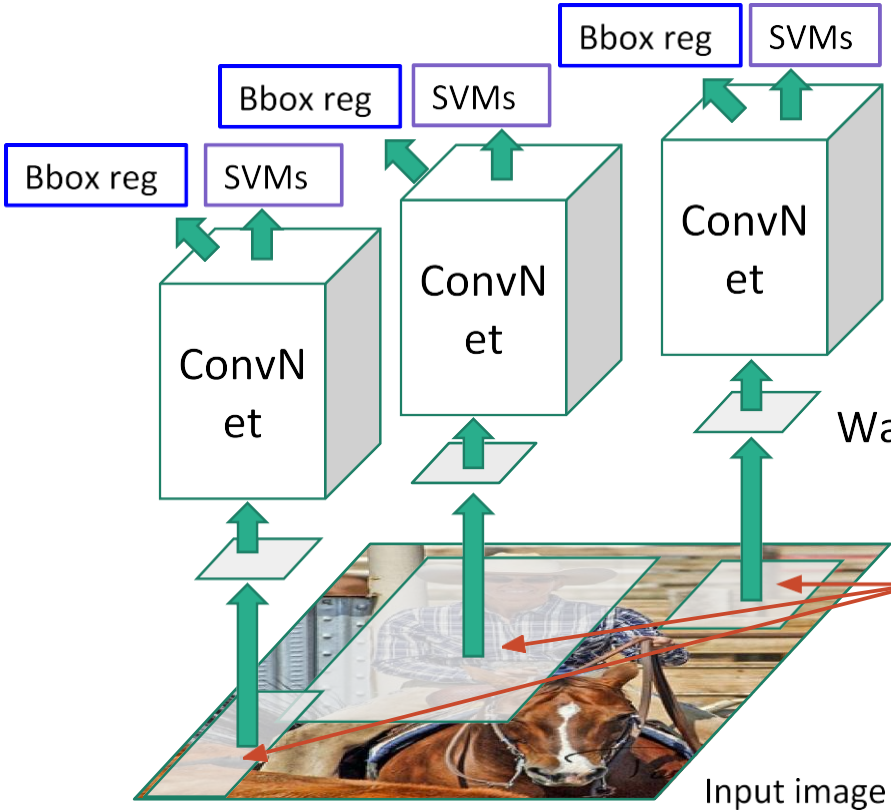
Girshick et al, “Rich feature hierarchies for accurate object detection and semantic segmentation”, CVPR 2014.
Figure copyright Ross Girshick, 2015; source. Reproduced with permission.

R-CNN



Girshick et al, “Rich feature hierarchies for accurate object detection and semantic segmentation”, CVPR 2014.
Figure copyright Ross Girshick, 2015; source. Reproduced with permission.

R-CNN



Linear Regression for bounding box offsets

Classify regions with SVMs

Forward each region through ConvNet

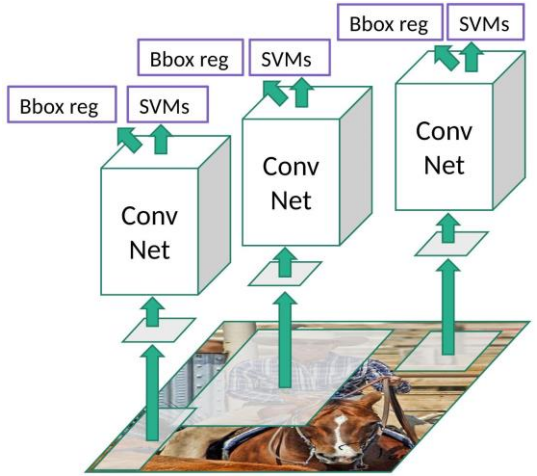
Warped image regions

Regions of Interest (RoI) from a proposal method (~2k)

Girshick et al, “Rich feature hierarchies for accurate object detection and semantic segmentation”, CVPR 2014.
Figure copyright Ross Girshick, 2015; source. Reproduced with permission.

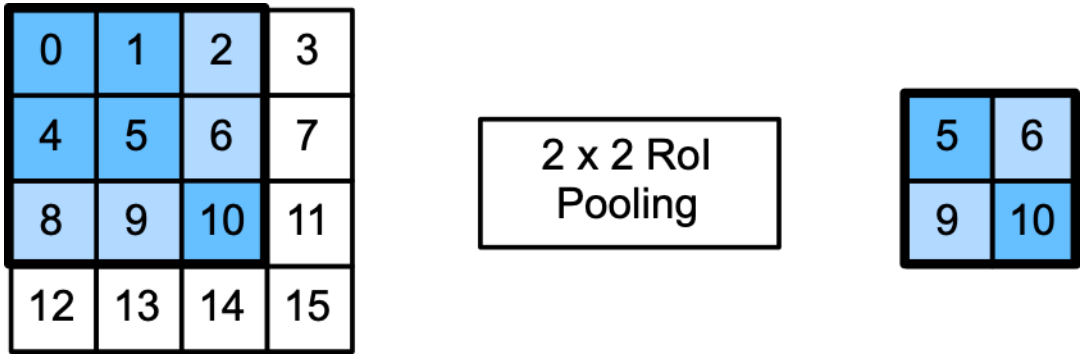
R-CNN: Problems

- Ad hoc training objectives
  - Fine-tune network with softmax classifier (log loss)
  - Train post-hoc linear SVMs (hinge loss)
  - Train post-hoc bounding-box regressions (least squares)
- Training is slow (84h), takes a lot of disk space
- Inference (detection) is slow
  - 47s / image with VGG16 [Simonyan & Zisserman. ICLR15]
  - Fixed by SPP-net [He et al. ECCV14]



Girshick et al, “Rich feature hierarchies for accurate object detection and semantic segmentation”, CVPR 2014.
Slide copyright Ross Girshick, 2015; source. Reproduced with permission.

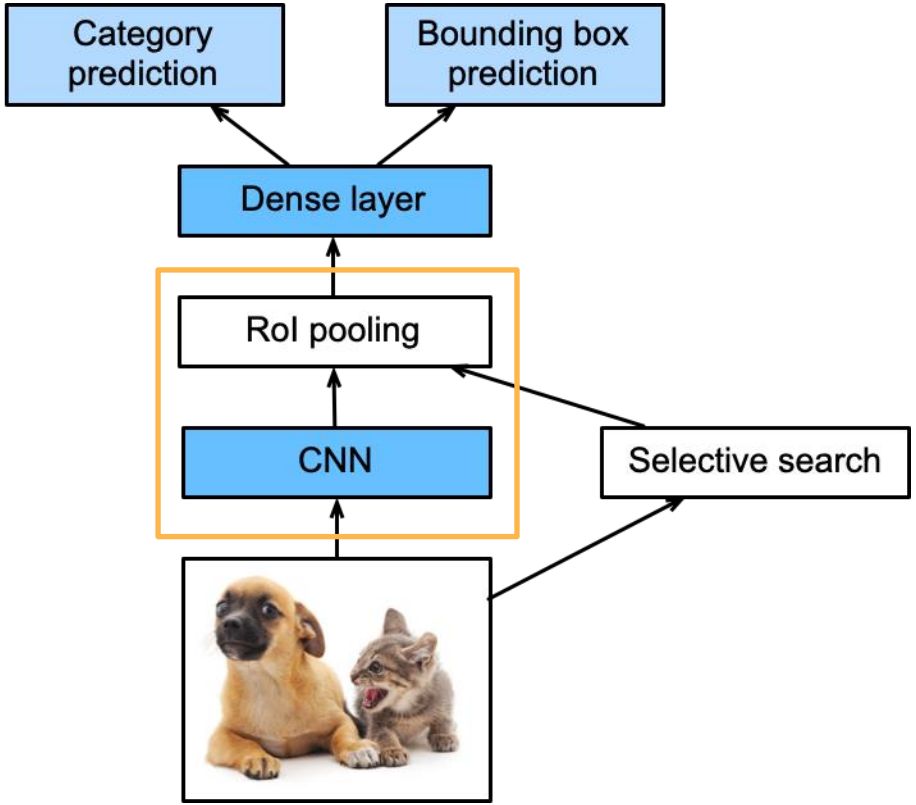
Region of Interest (RoI) Pooling



- Given an anchor box, uniformly cuts it into *n x m* blocks, output the maximal value in each block
- Returns *nm* values for each anchor box

Fast RCNN

- A CNN to extract features (fast)
- RoI pooling returns fixed length feature for each anchor box

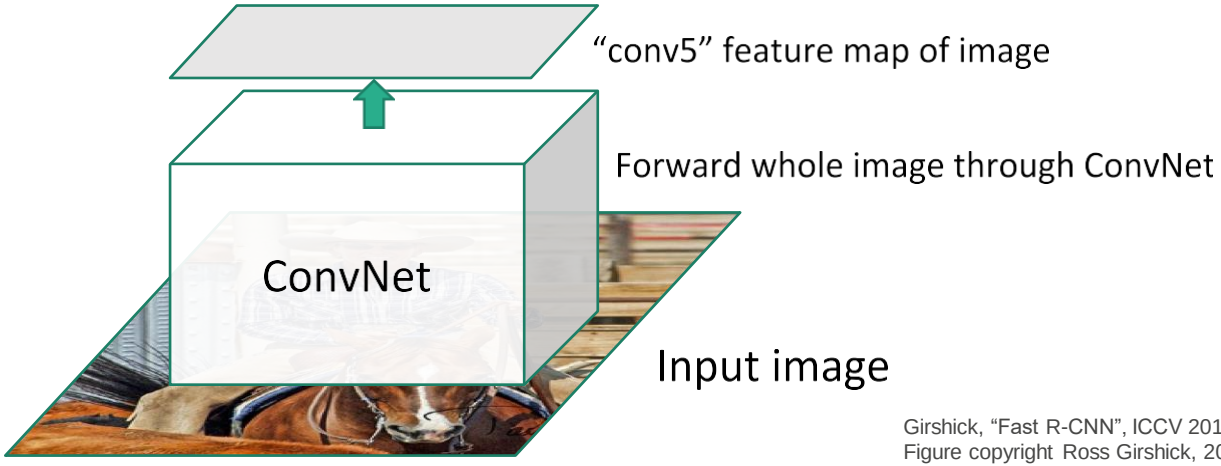


Fast R-CNN

Input image

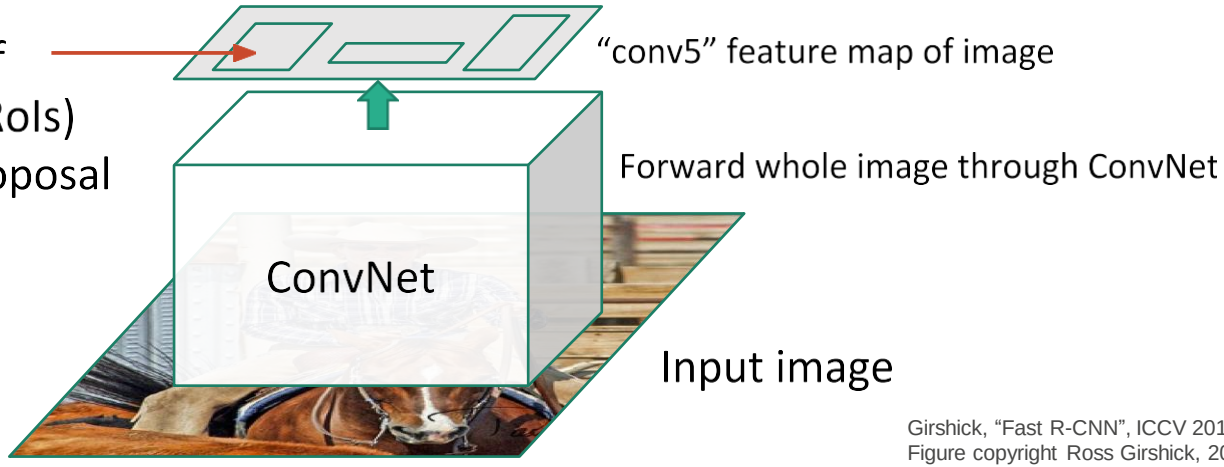
Girshick, “Fast R-CNN”, ICCV 2015.

Fast R-CNN



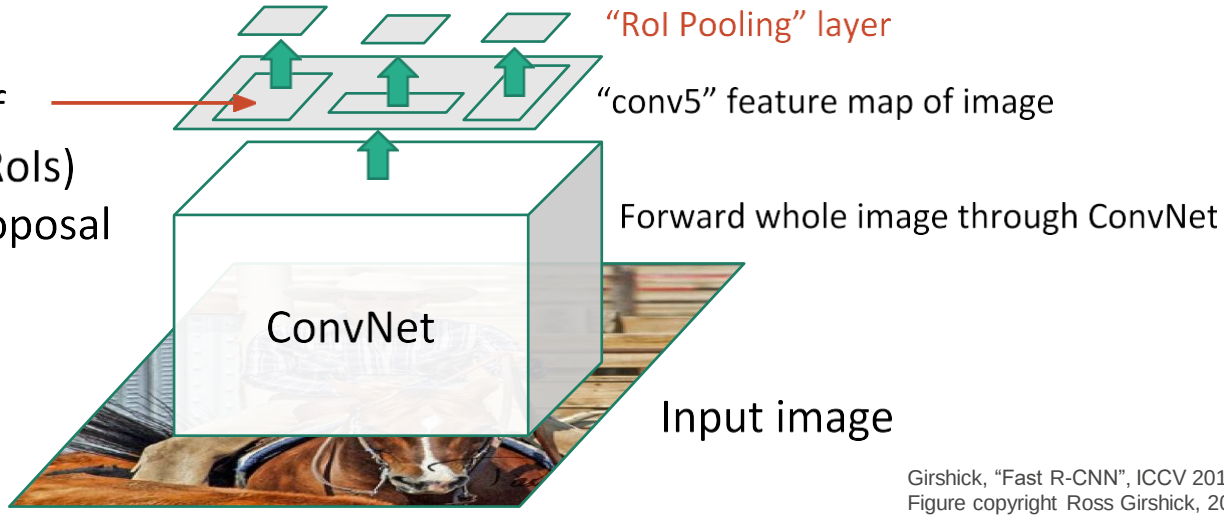
Girshick, “Fast R-CNN”, ICCV 2015.

Fast R-CNN



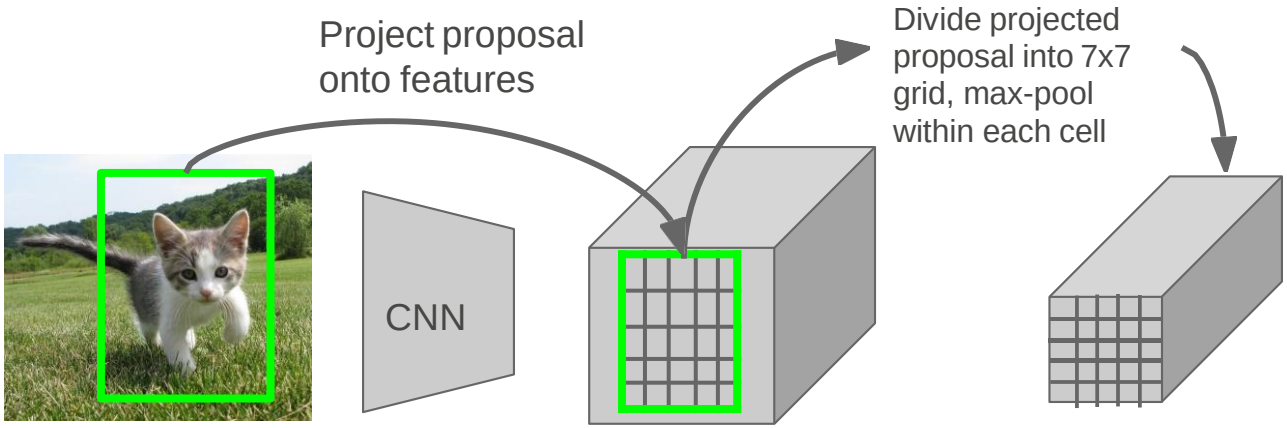
Girshick, "Fast R-CNN", ICCV 2015.

Fast R-CNN



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

Fast R-CNN: RoI Pooling



Hi-res input image:
3 x 640 x 480
with region proposal

Hi-res conv features:
512 x 20 x 15;

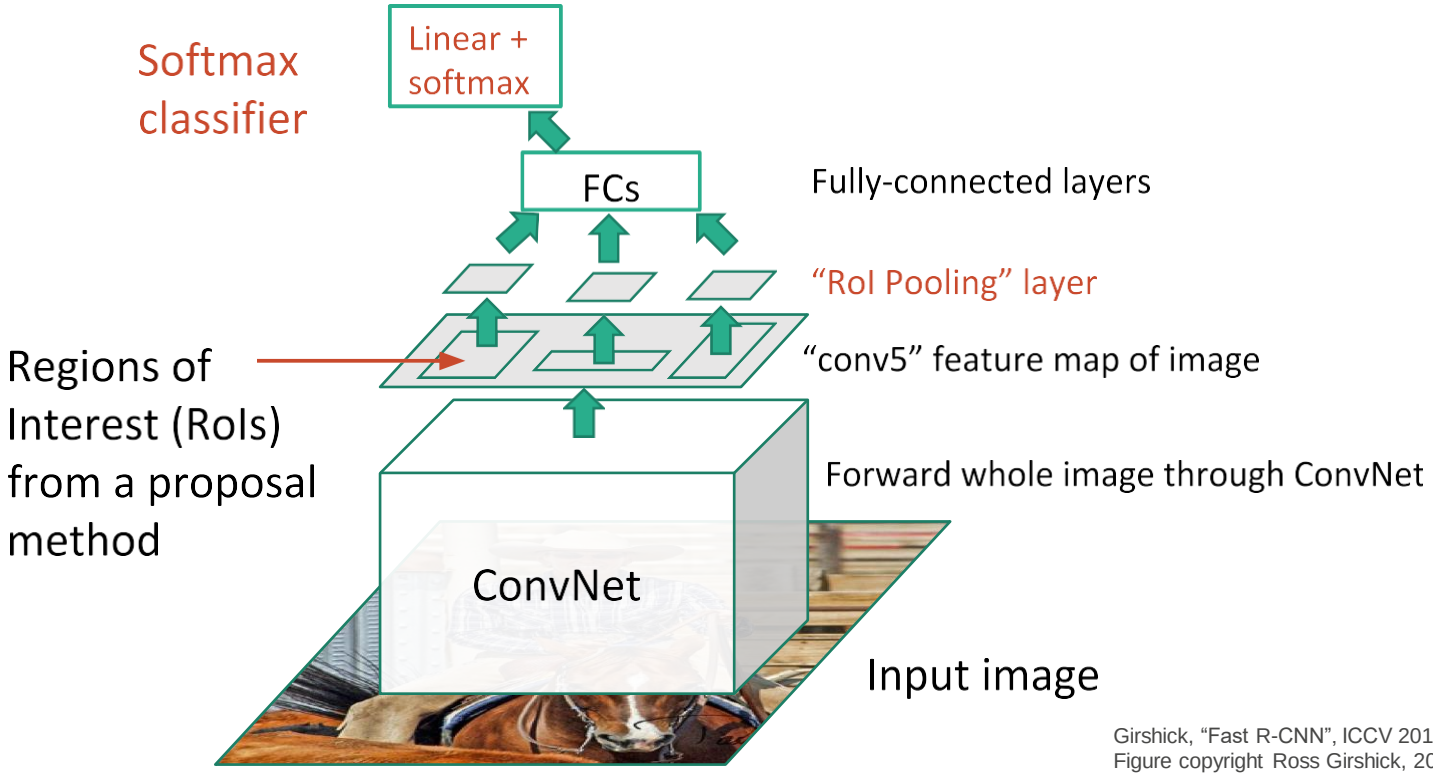
Projected region proposal is e.g.
512 x 18 x 8
(varies per proposal)

RoI conv features:
512 x 7 x 7
for region proposal

Fully-connected layers expect low-res conv features:
512 x 7 x 7

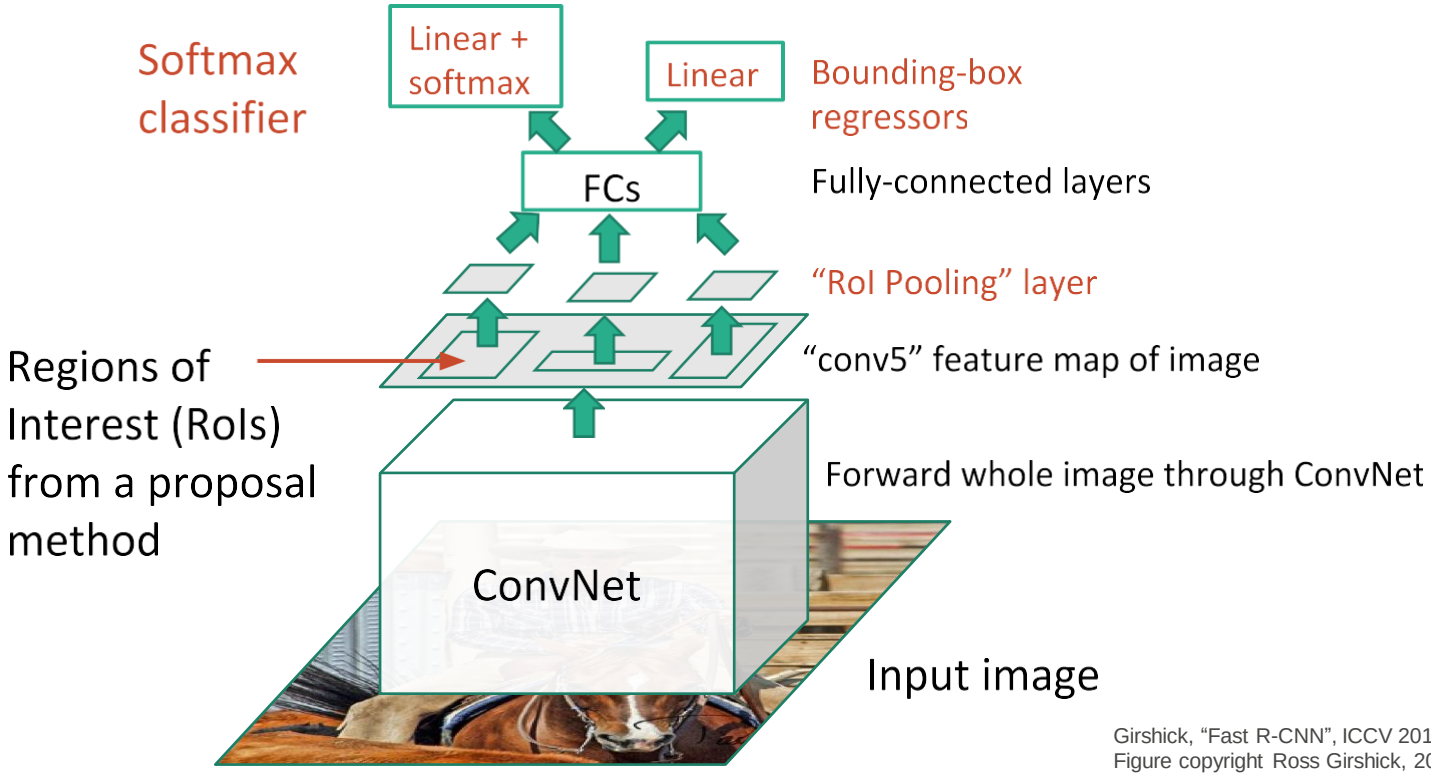
Girshick, "Fast R-CNN", ICCV 2015.

Fast R-CNN



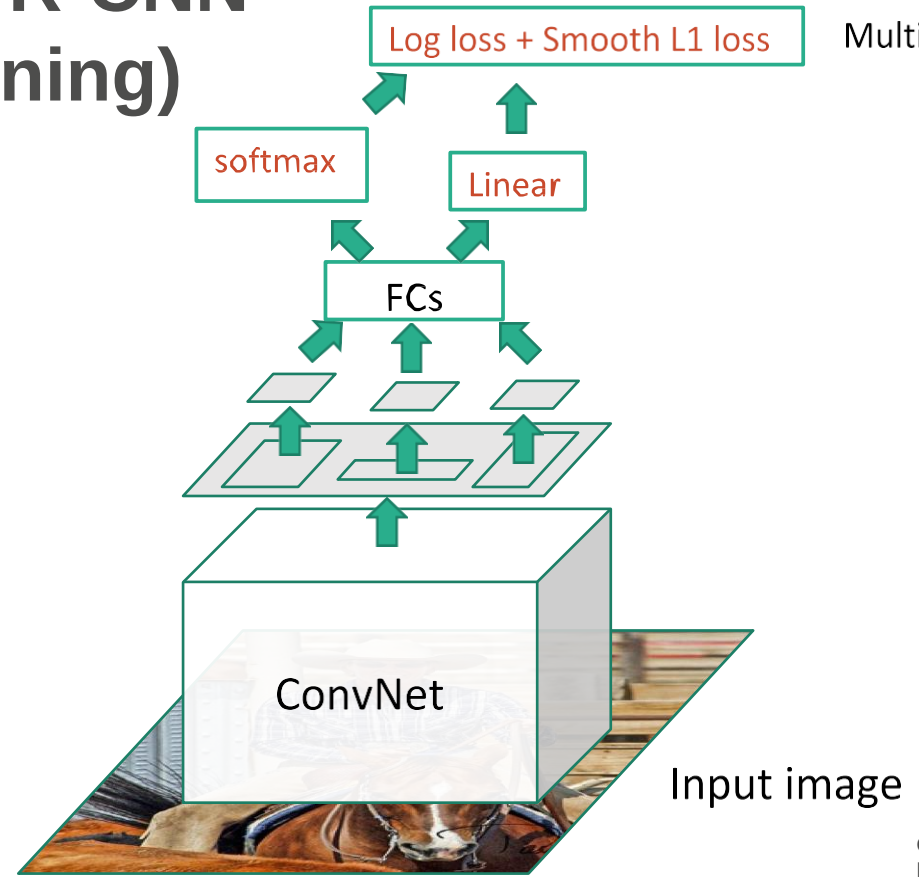
Girshick, "Fast R-CNN", ICCV 2015.
Figure copyright Ross Girshick, 2015; source. Reproduced with permission.

Fast R-CNN



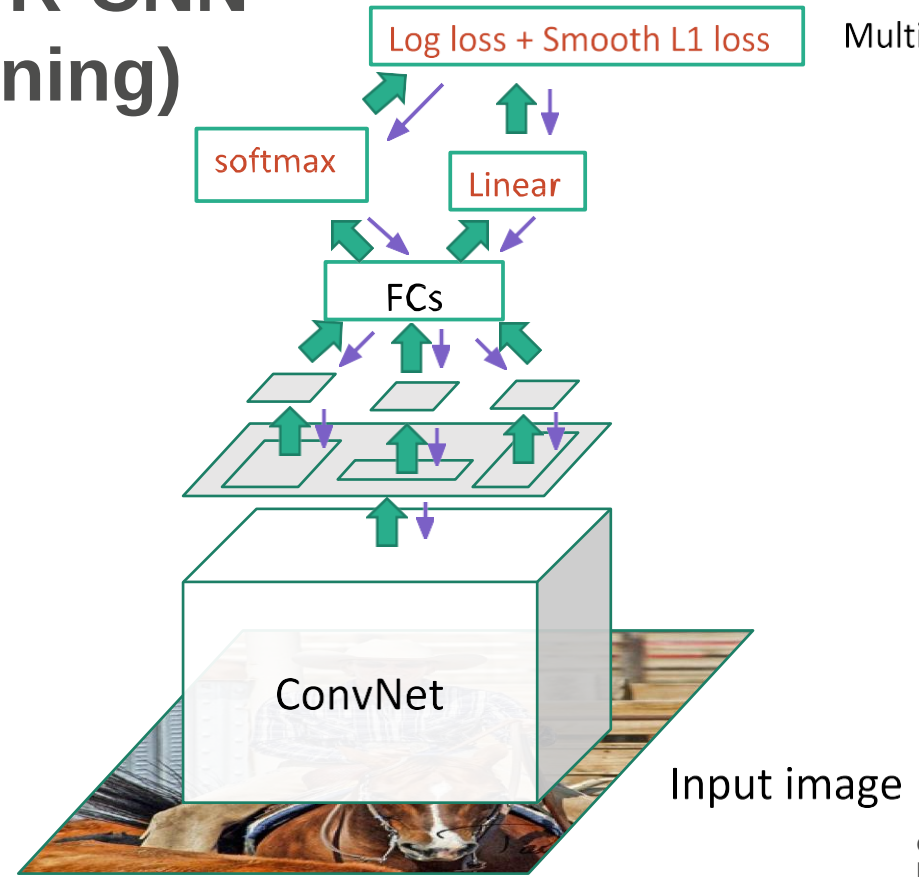
Girshick, "Fast R-CNN", ICCV 2015.

Fast R-CNN (Training)



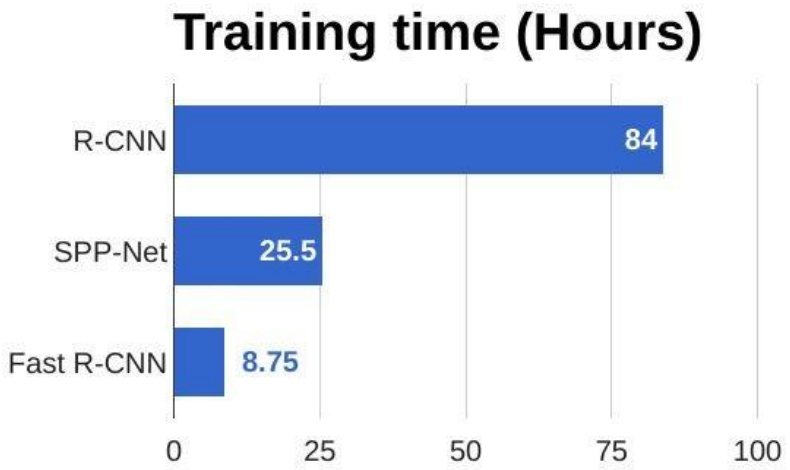
Girshick, “Fast R-CNN”, ICCV 2015.

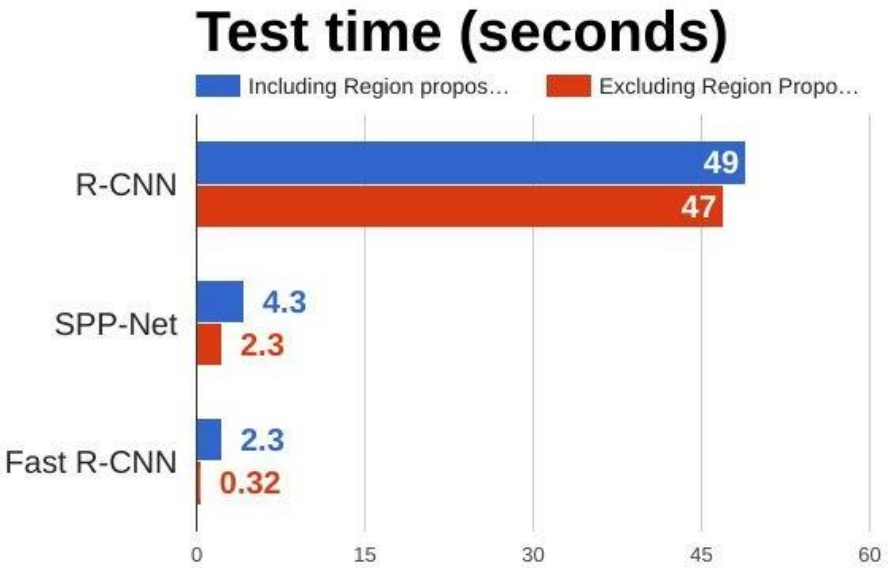
Fast R-CNN (Training)



Girshick, “Fast R-CNN”, ICCV 2015.

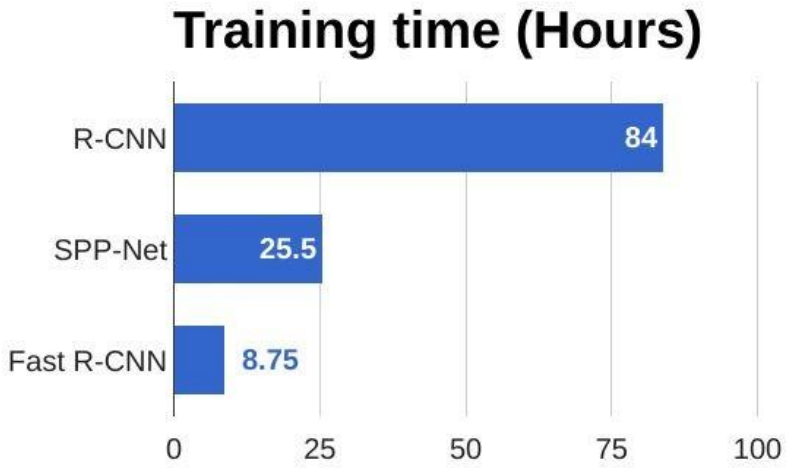
R-CNN vs SPP vs Fast R-CNN

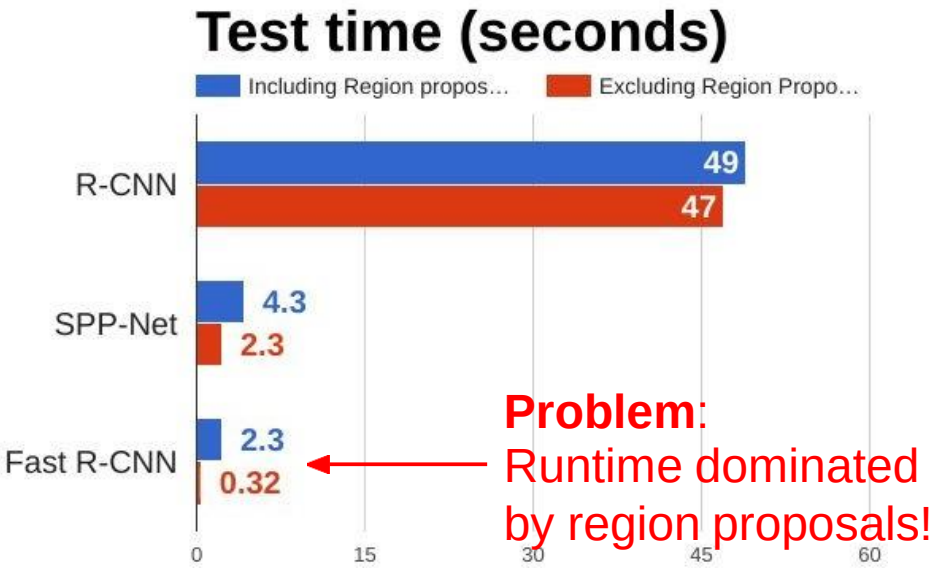




Girshick et al, “Rich feature hierarchies for accurate object detection and semantic segmentation”, CVPR 2014.
He et al, “Spatial pyramid pooling in deep convolutional networks for visual recognition”, ECCV 2014
Girshick, “Fast R-CNN”, ICCV 2015

R-CNN vs SPP vs Fast R-CNN





Problem:
Runtime dominated by region proposals!

Girshick et al, “Rich feature hierarchies for accurate object detection and semantic segmentation”, CVPR 2014.
He et al, “Spatial pyramid pooling in deep convolutional networks for visual recognition”, ECCV 2014
Girshick, “Fast R-CNN”, ICCV 2015

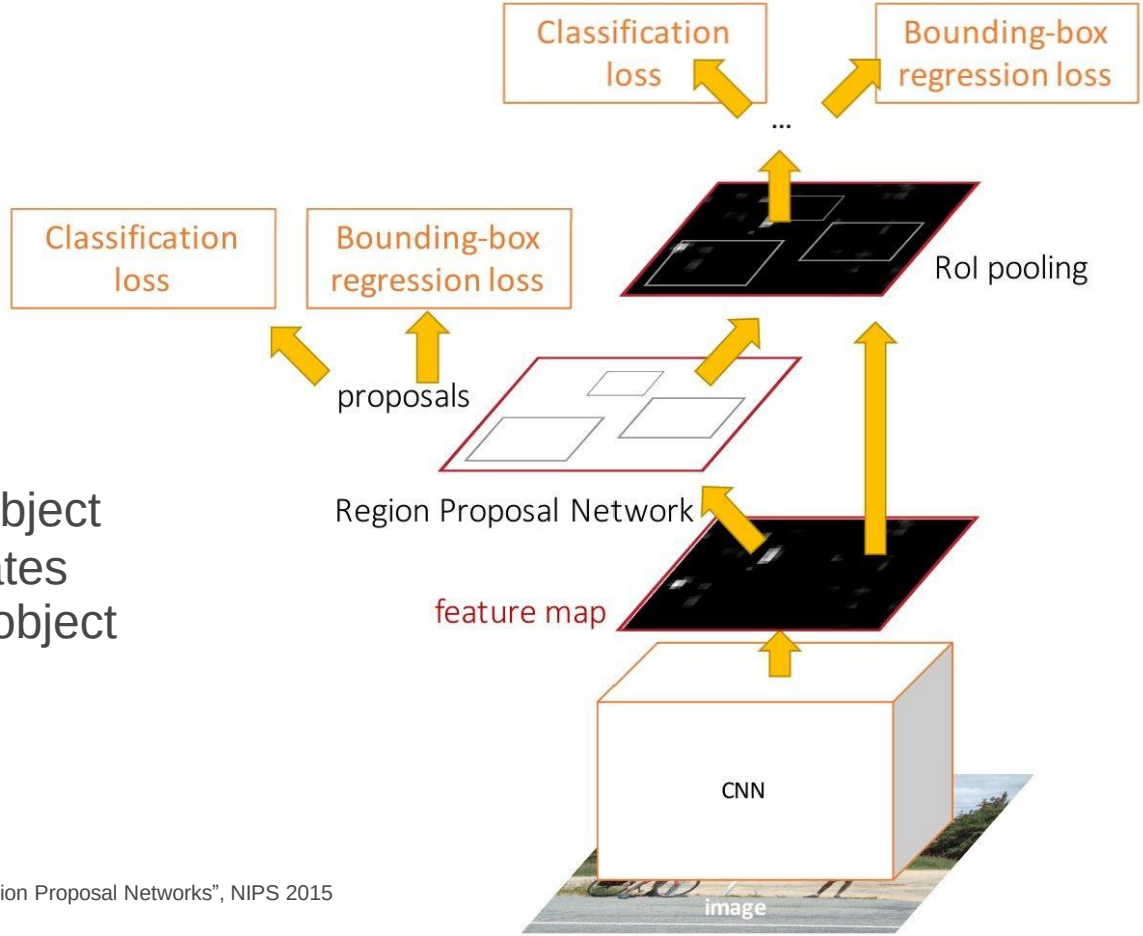
Faster R-CNN:

Make CNN do proposals!

Insert **Region Proposal Network (RPN)** to predict proposals from features

Jointly train with 4 losses:

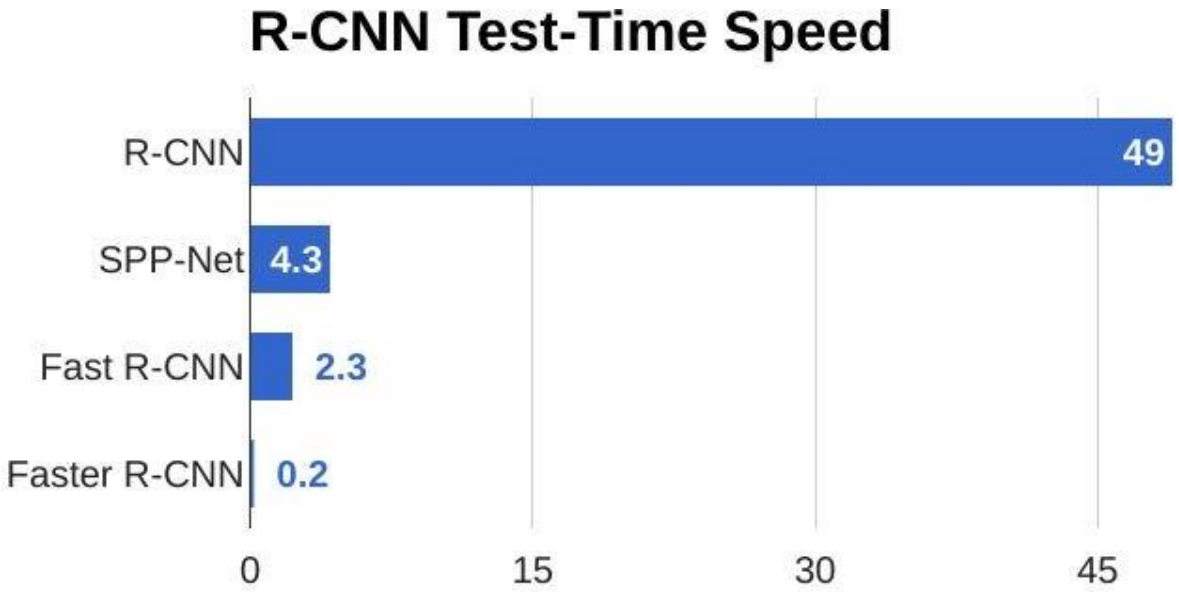
1. RPN classify object / not object
2. RPN regress box coordinates
3. Final classification score (object classes)
4. Final box coordinates



Ren et al, "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks", NIPS 2015

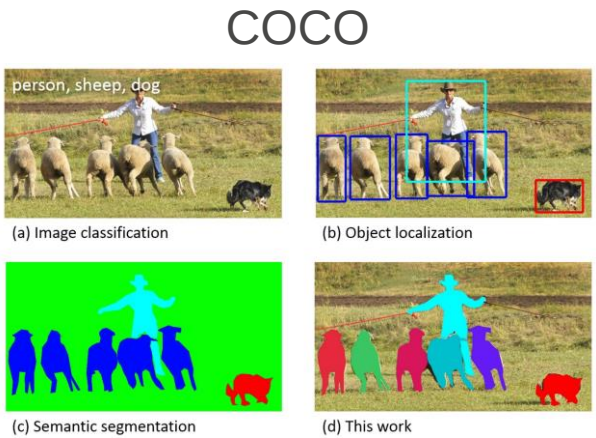
Faster R-CNN:

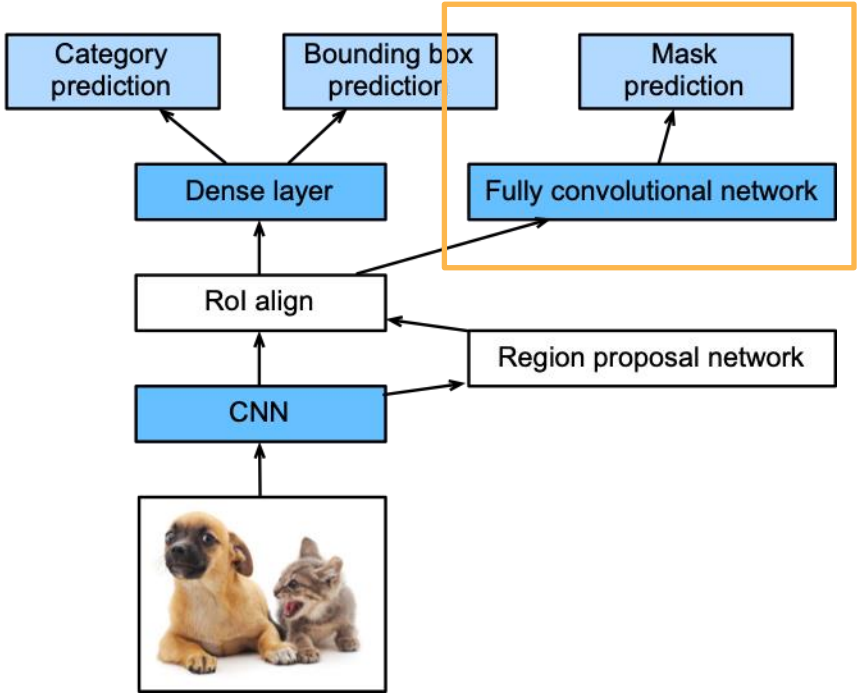
Make CNN do proposals!



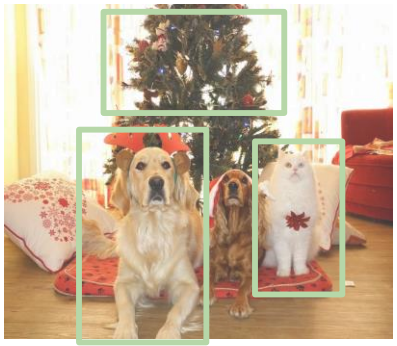
Mask R-CNN

- If pixel-level labels are available, add an additional loss (FCN) to take into account them

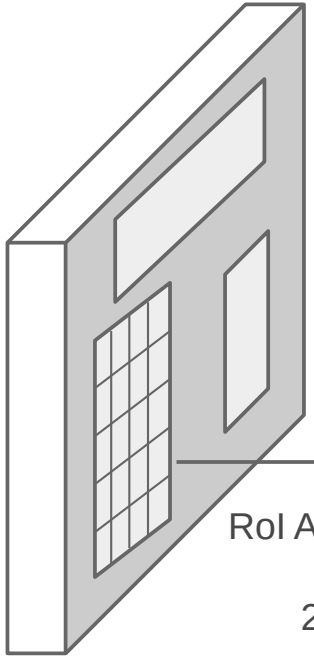




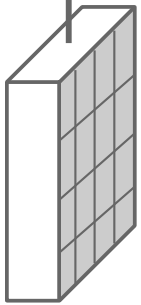
Mask R-CNN



CNN

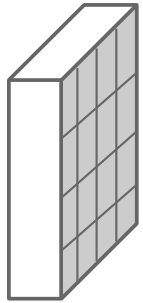


RoI Align



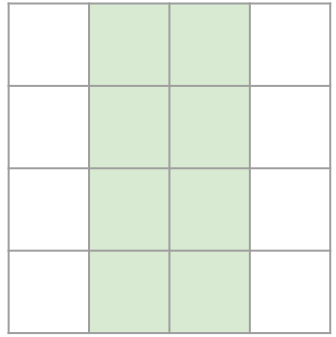
256 x 14 x 14

Conv



256 x 14 x 14

Conv



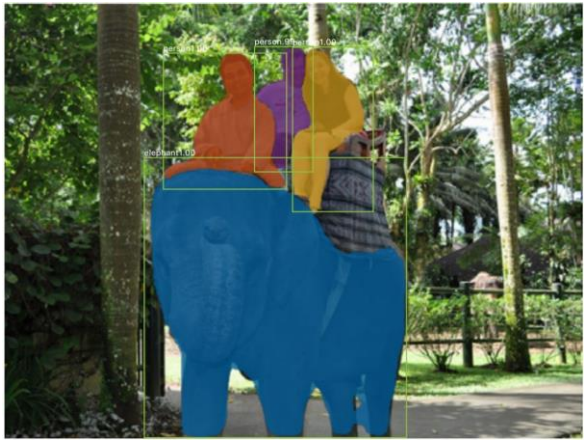
Predict a mask for each of C classes

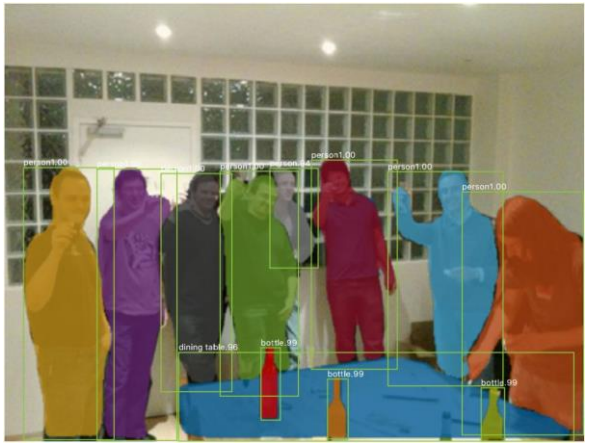
C x 14 x 14

Classification Scores: C
Box coordinates (per class): 4 * C

He et al, “Mask R-CNN”, arXiv 2017

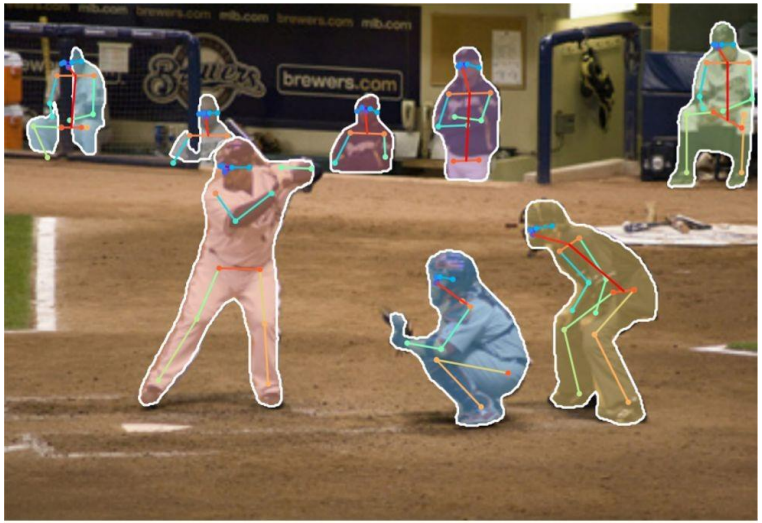
Mask R-CNN: Very Good Results!

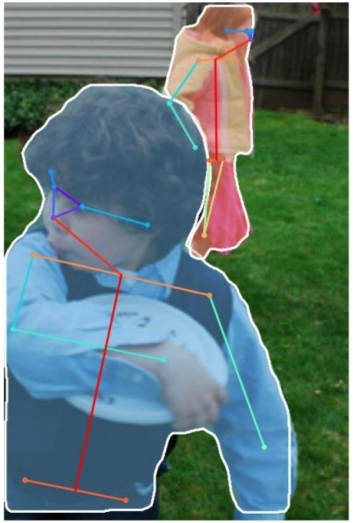




He et al, “Mask R-CNN”, arXiv 2017

Mask R-CNN Also does pose



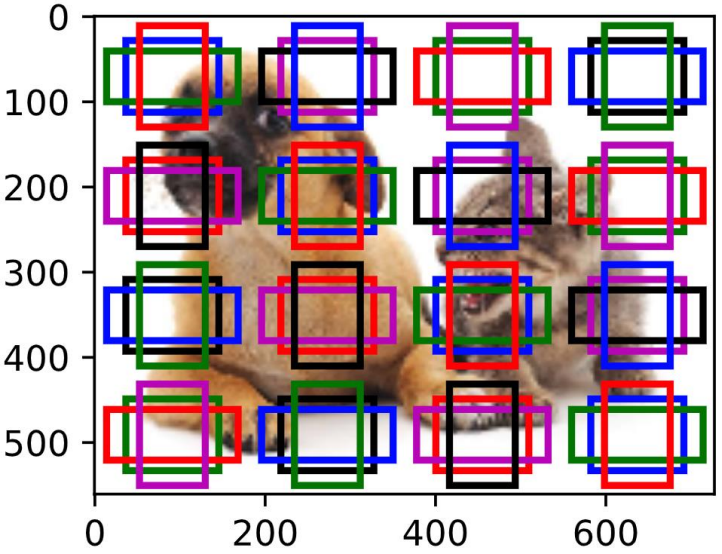


He et al, “Mask R-CNN”, arXiv 2017

Single Shot Multibox Detection (SSD)

Generate Anchor Boxes

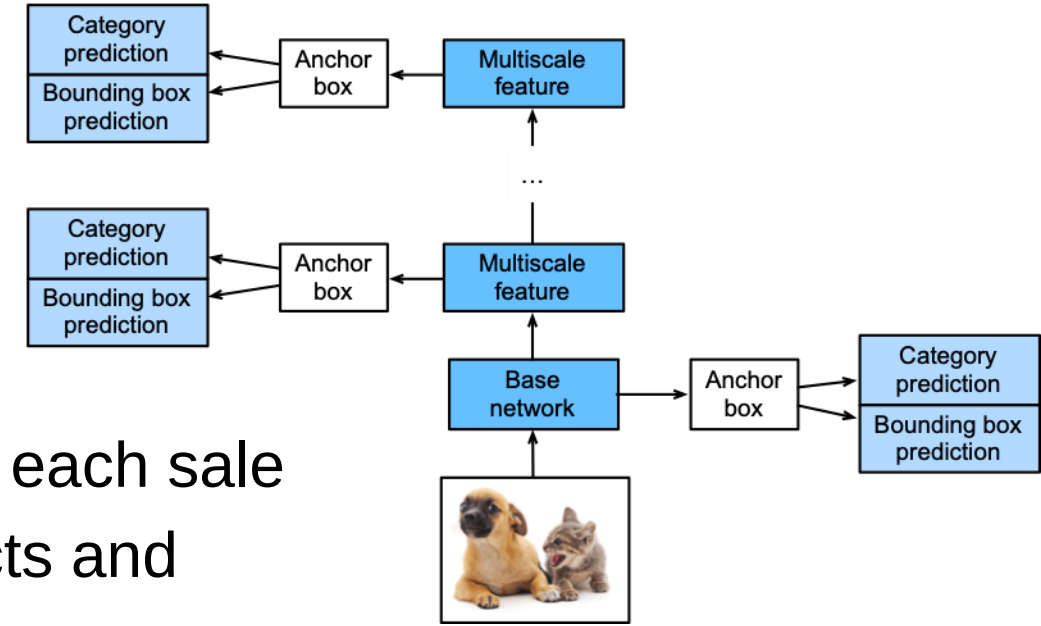
- For each pixel, generate multiple anchor boxes centered at this pixel
- Given *n* sizes $s_1, \ldots, s_n$ and *m* ratios $r_1, \ldots, r_m$, generate *n+m-1* anchor boxes



$$(s_1, r_1), (s_2, r_1), \ldots, (s_n, r_1), (s_1, r_2), \ldots, (s_1, r_m)$$

SSD Model

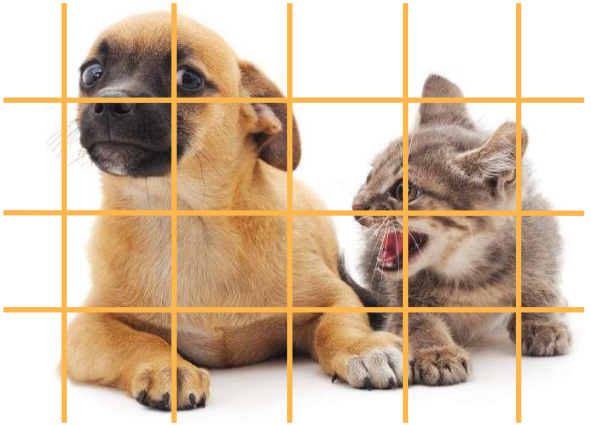
- A base network to extract feature, followed by conv-blocks to halve width and height
- Generate anchor boxes at each sale
  - Bottom for small objects and top for large objects
- Predict class and bounding box for each anchor box



You Only Look Once (YOLO)

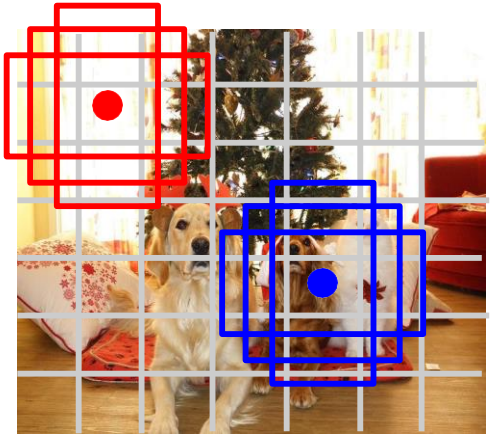
YOLO

- Anchor boxes are highly overlapped in SSD
- YOLO cuts the input image uniformly into *S x S* anchor boxes
- Each anchor box predicts *B* bounding boxes
- V2 and V3 add more improvements



Detection without Proposals: YOLO / SSD

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)

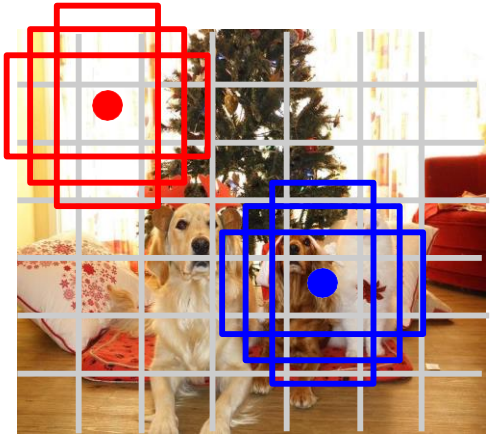
Output:
7 x 7 x (5 * B + C)

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

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

End-to-End Object Detection with Transformers

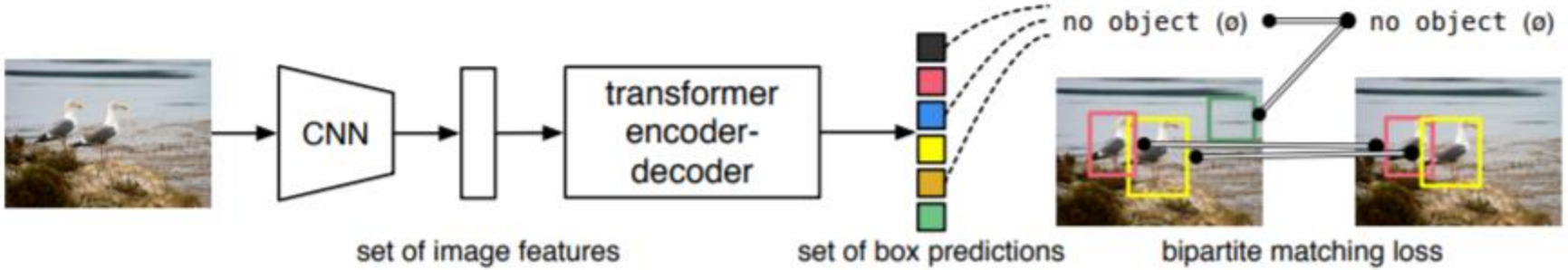


Fig. 1: DETR directly predicts (in parallel) the final set of detections by combining a common CNN with a transformer architecture. During training, bipartite matching uniquely assigns predictions with ground truth boxes. Prediction with no match should yield a "`no object`" (∅) class prediction.

End-to-End Object Detection with Transformers

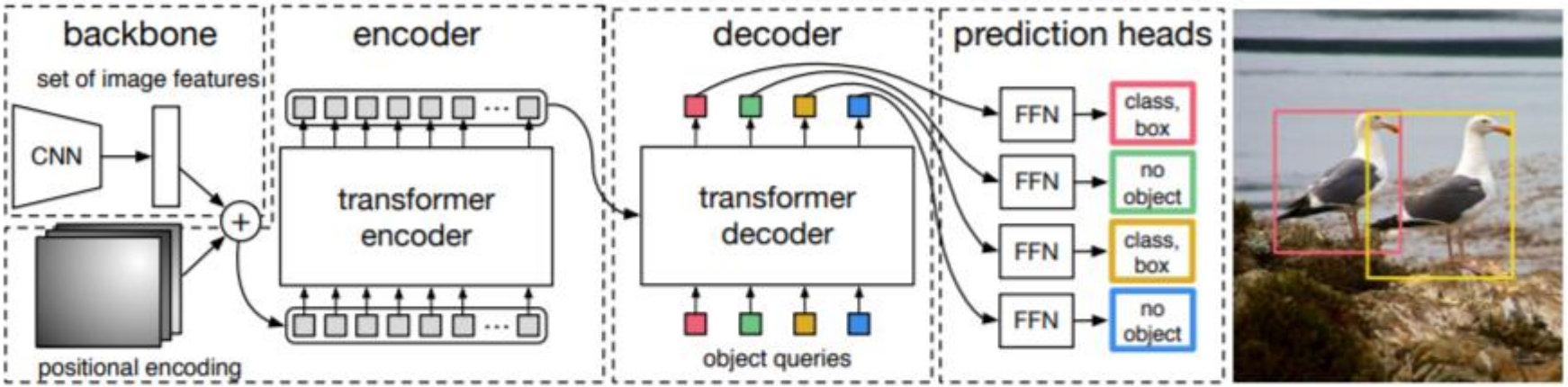


Fig. 2: DETR uses a conventional CNN backbone to learn a 2D representation of an input image. The model flattens it and supplements it with a positional encoding before passing it into a transformer encoder. A transformer decoder then takes as input a small fixed number of learned positional embeddings, which we call *object queries*, and additionally attends to the encoder output. We pass each output embedding of the decoder to a shared feed forward network (FFN) that predicts either a detection (class and bounding box) or a "`no object`" class.

PIX2SEQ: A LANGUAGE MODELING FRAMEWORK FOR OBJECT DETECTION

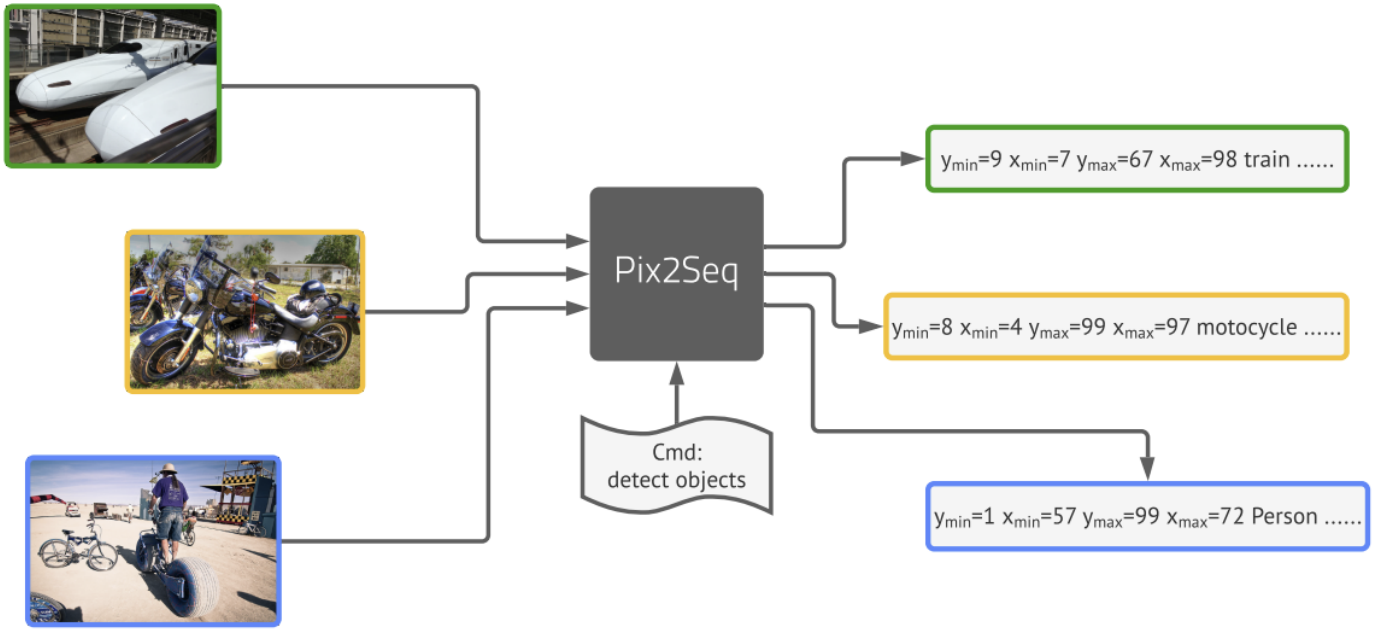


Figure 1: Illustration of Pix2Seq framework for object detection. The neural net perceives an image and generates a sequence of tokens that correspond to bounding boxes and class labels.

PIX2SEQ: A LANGUAGE MODELING FRAMEWORK FOR OBJECT DETECTION

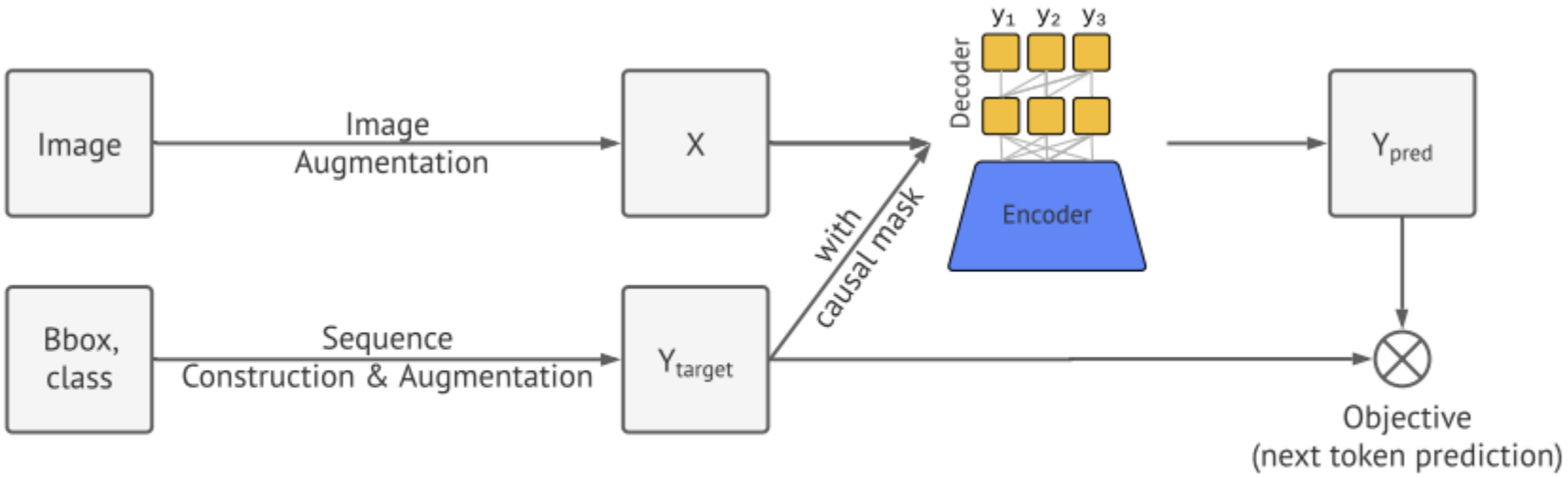


Figure 2: Major components of the Pix2Seq learning framework.

PIX2SEQ: A LANGUAGE MODELING FRAMEWORK FOR OBJECT DETECTION

Random ordering (multiple samples):

327 370 653 444 1001 544 135 987 338 1004 508 518 805 892 1004 0

544 135 987 338 1004 327 370 653 444 1001 508 518 805 892 1004 0

508 518 805 892 1004 544 135 987 338 1004 327 370 653 444 1001 0

Area ordering:

544 135 987 338 1004 508 518 805 892 1004 327 370 653 444 1001 0

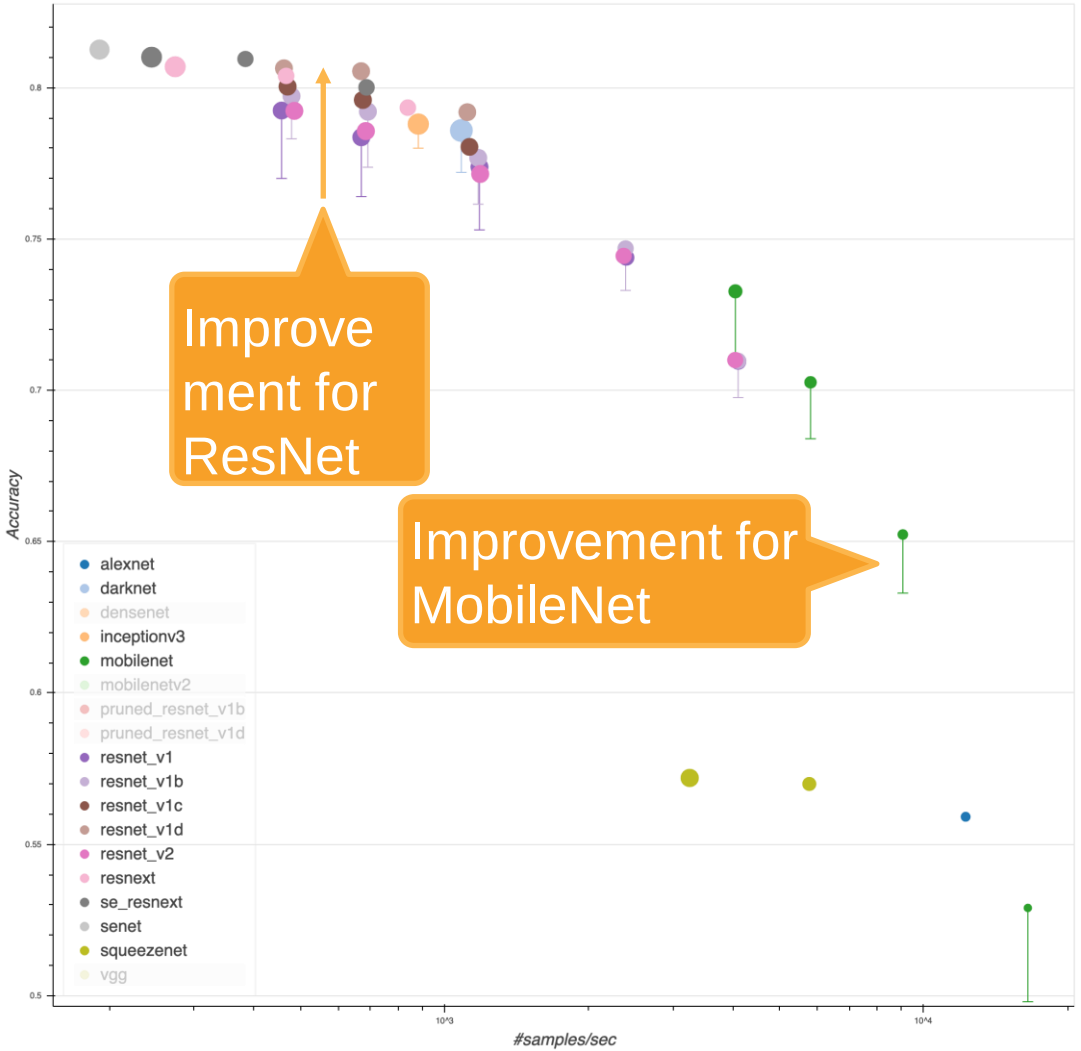
Dist2ori ordering:

544 135 987 338 1004 327 370 653 444 1001 508 518 805 892 1004 0

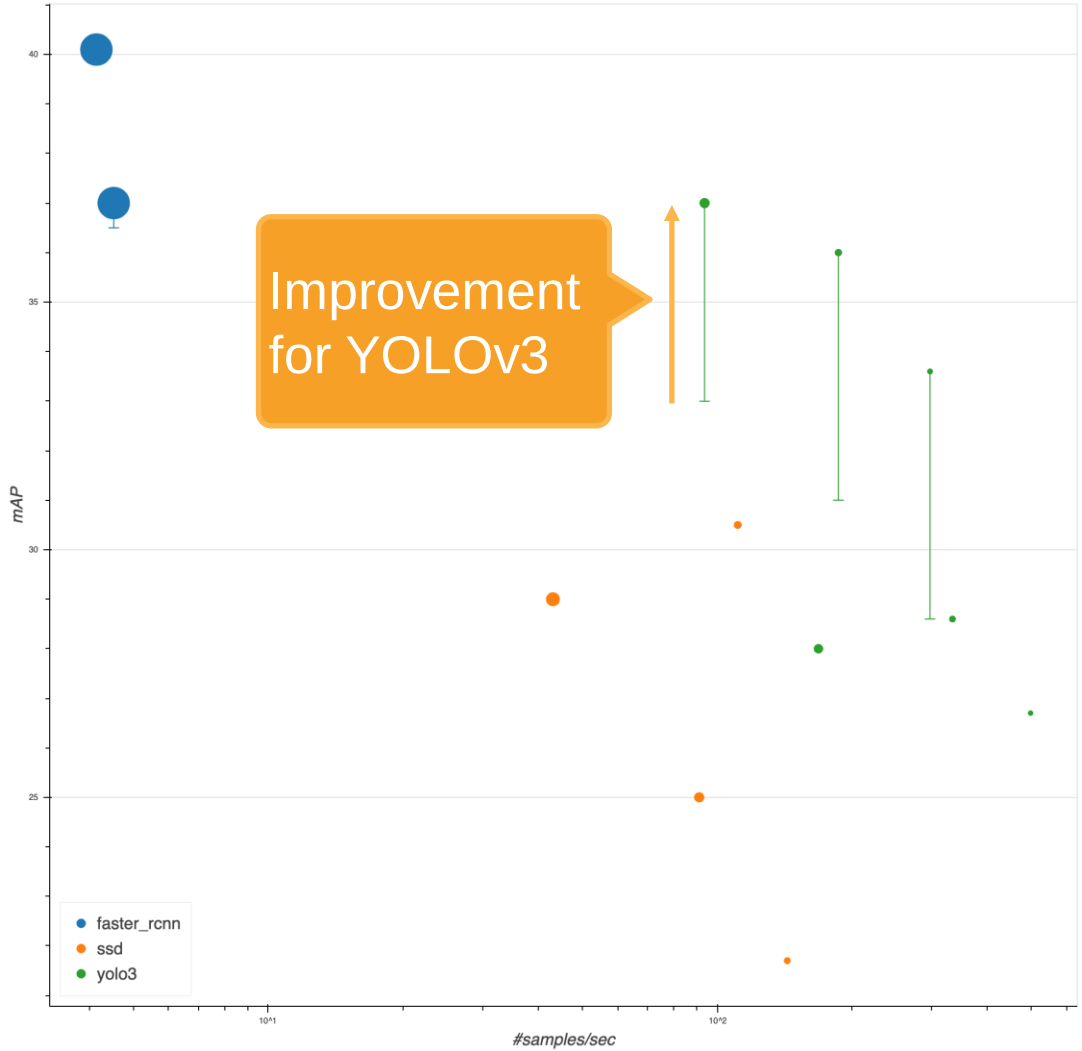
Figure 4: Examples of sequence construction with $n_{\text{bins}} = 1000$, and 0 is EOS token.

Tricks for Training

Various training tricks greatly improve image classification model accuracy



Apply to object detection models as well



Mixup Training Example

- Randomly select two examples *i* and *j*, sample a random number $\lambda \in [0,1]$
- Compute the mixed new example

$$x = \lambda x_i + (1-\lambda)x_j \; y = \lambda y_i + (1-\lambda)y_j$$

- Train on mixed examples

bittern	0
...	0
otter	0
...	0
analog_clock	1

* 0.9 +

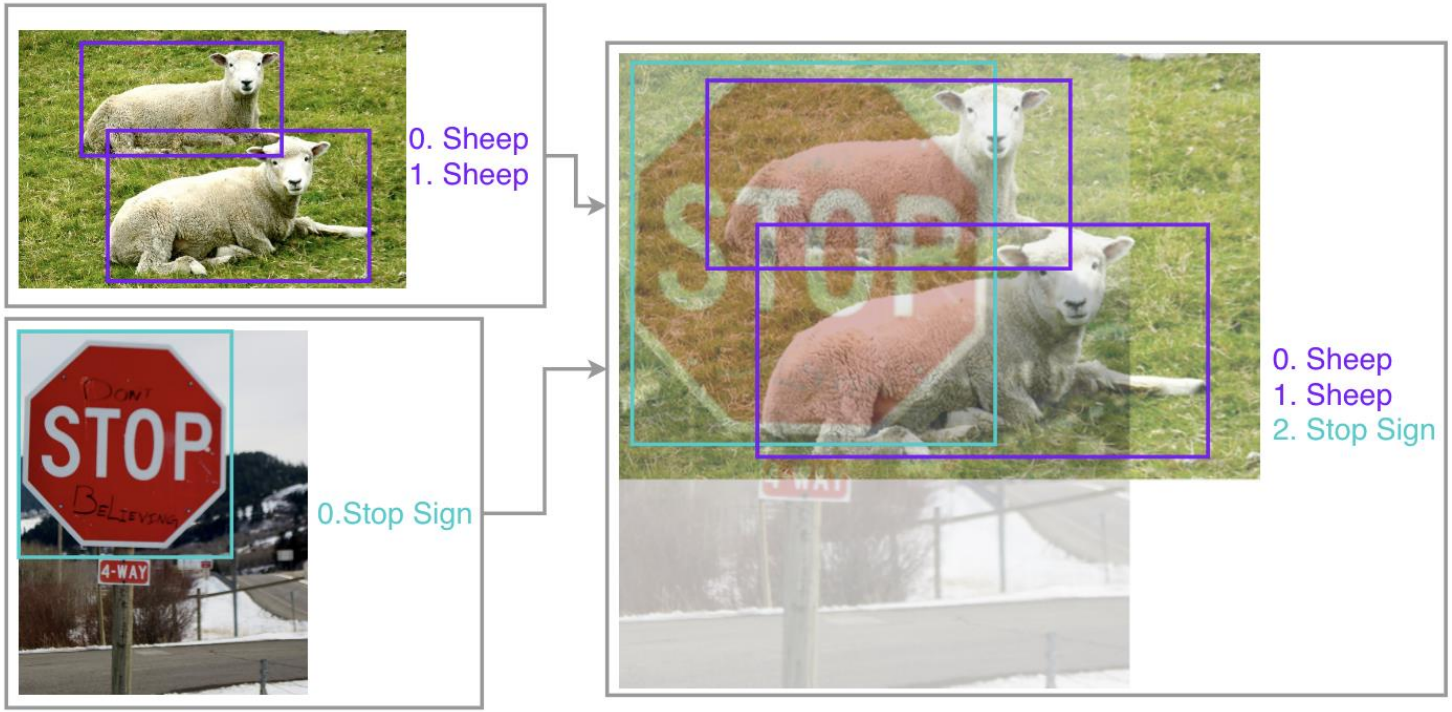
bittern	1
...	0
otter	0
...	0
analog_clock	0

* 0.1 =

bittern	0.1
...	0
otter	0
...	0
analog_clock	0.9

Mixup Training Example

- Apply to object detection as well



Label Smoothing

- Assume $\mathbf{y} \in \mathbb{R}^n$ is the one-hot encoding of label

$$y_i = \begin{cases} 1 & \textit{if belongs to class } i \\ 0 & \textit{otherwise} \end{cases}$$

- Approximating 0/1 values with softmax is hard
- The smoothed version

$$y_i = \begin{cases} 1-\epsilon & \textit{if belongs to class } i \\ \epsilon/(n-1) & \textit{otherwise} \end{cases}$$

  - Commonly use $\epsilon = 0.1$

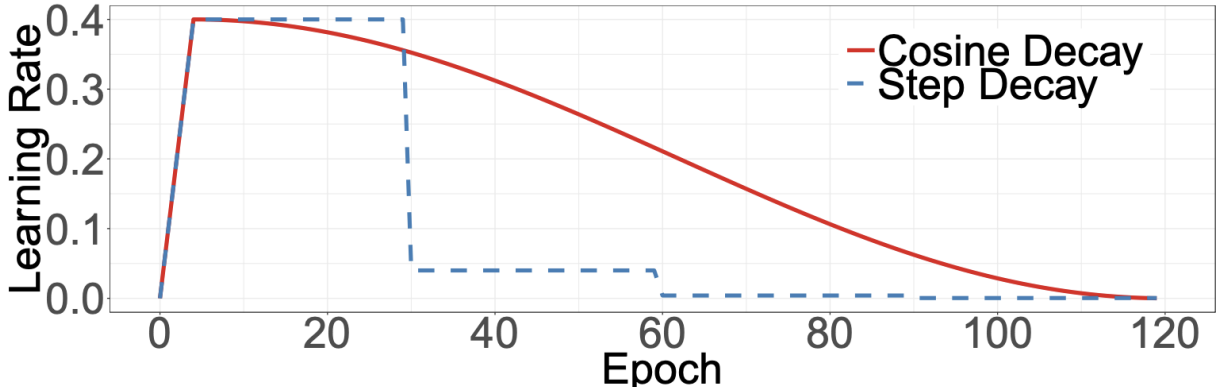
Learning Rate Warmup

- A large learning rate for randomly initialized parameters may cause numerical issue
- The warmup trick uses a small learning rate at beginning and then increases it to the initial value. For example:
  - If we choose the initial learning rate to be 0.1 and use 5 epochs for warmup
  - Start the learning rate with 0, linearly increases it to 0.1 in the first 5 epochs

Cosine Decay

- We need to decrease learning rate for SGD to converge
  - E.g. decreasing by 10x at epoch 30, 60, and 90
- Assume in total *T* iterations (batches), the cosine decay computes learning rate at iteration *t* by

$$\eta_t = 1/2\big(1 + \cos(t\pi/T)\big)\eta$$



Random Batch Shapes

- Images are resized to same shape in a batch, e.g. 224 width and 224 height
- We can vary this shape:
  - For each batch, choose a random width/height from 224 (7x32), 256 (8x32), 228 (9x32), …
  - Resize all images into this shape

Image Classification

Refinements	ResNet-50-D		Inception-V3		MobileNet	
	Top-1	Δ	Top-1	Δ	Top-1	Δ
Efficient	77.16		77.50		71.90	
+ cosine decay	77.91	+0.75	78.19	+0.69	72.83	+0.93
+ label smoothing	78.31	+0.4	78.40	+0.21	72.93	+0.1
+ mixup	**79.15**	+0.84	**78.77**	+0.37	**73.28**	+0.35

Hang et.al *Bag of Tricks for Image Classification with Convolutional Neural Networks*

Results for YOLOv3

Incremental Tricks	mAP	Δ	Cumu Δ
- data augmentation	64.26	-15.99	-15.99
baseline	80.25	0	0
+ synchronize BN	80.81	+0.56	+0.56
+ random training shapes	81.23	+0.42	+0.98
+ cosine lr schedule	81.69	+0.46	+1.44
+ class label smoothing	82.14	+0.45	+1.89
+ mixup	**83.68**	**+1.54**	**+3.43**

Zhi et al, *Bag of Freebies for Training Object Detection Neural Networks*

The End