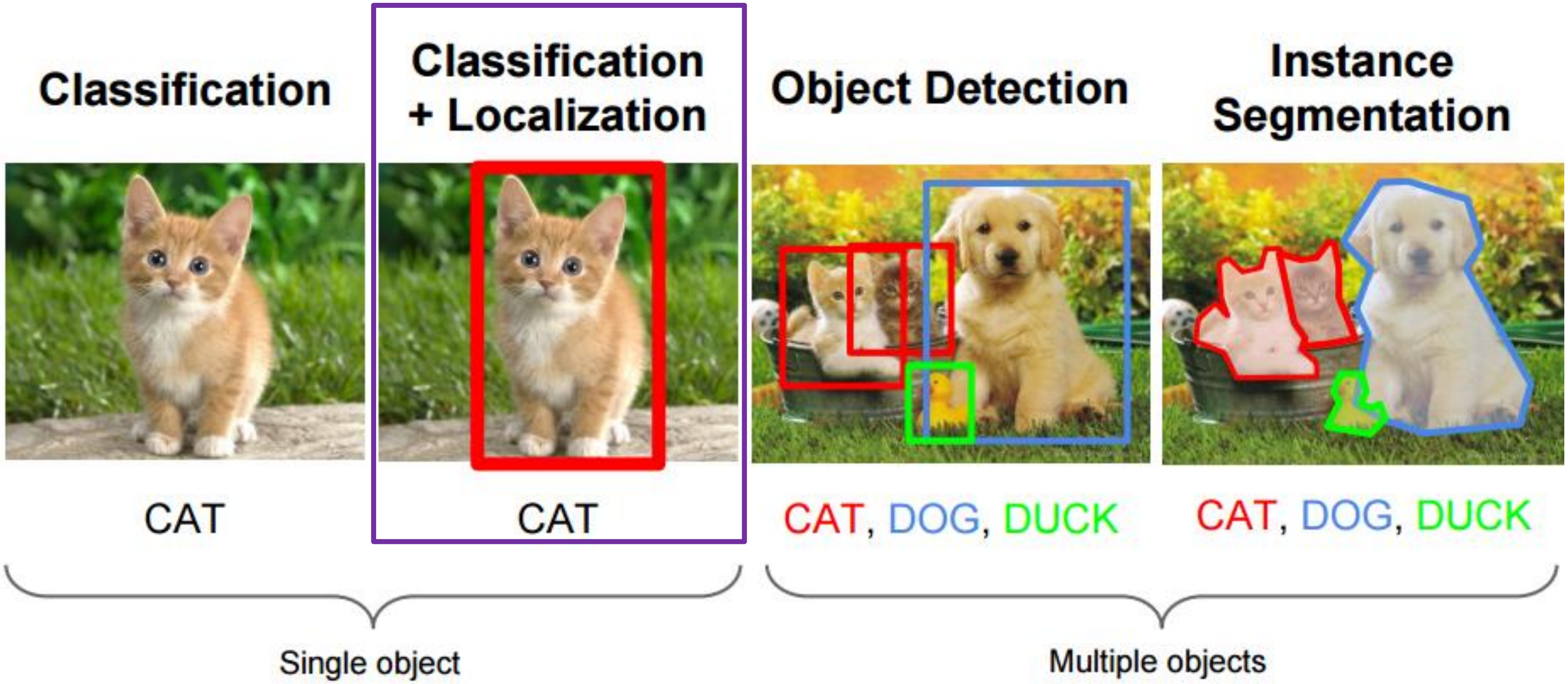
OBJECT DETECTION

HYUNG IL KOO

INTRODUCTION

Computer Vision Tasks



Classification + Localization

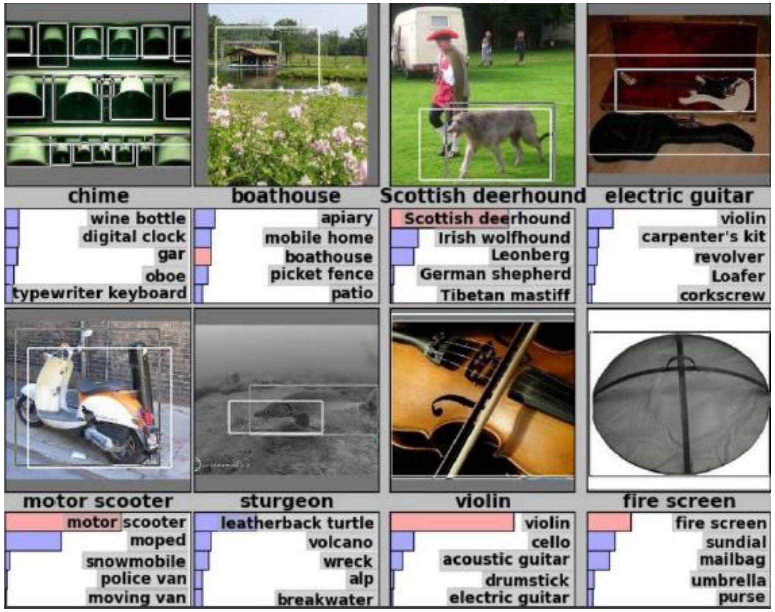
- Classification: C-classes
  - Input: image
  - Output: class label
  - Evaluation metric: accuracy

- Localization
  - Input: image
  - Output: box in the image (x, y, w, h)
  - Evaluation metric: IoU (intersection over union)

- Classification + Localization

Classification + Localization

- ImageNet
  - 1000 classes (same as classification)
  - Each image has 1 class, at least one bounding box
  - ~800 training images per class
  - Algorithm produces 5 (class, box) guesses
  - Example is correct if
    - at least one one guess has correct class, and
    - bounding box at least 0.5 intersection over union (IoU)



Idea #1: Localization as regression

Input: image

Neural Net

Output:
Box coordinates
(4 numbers)

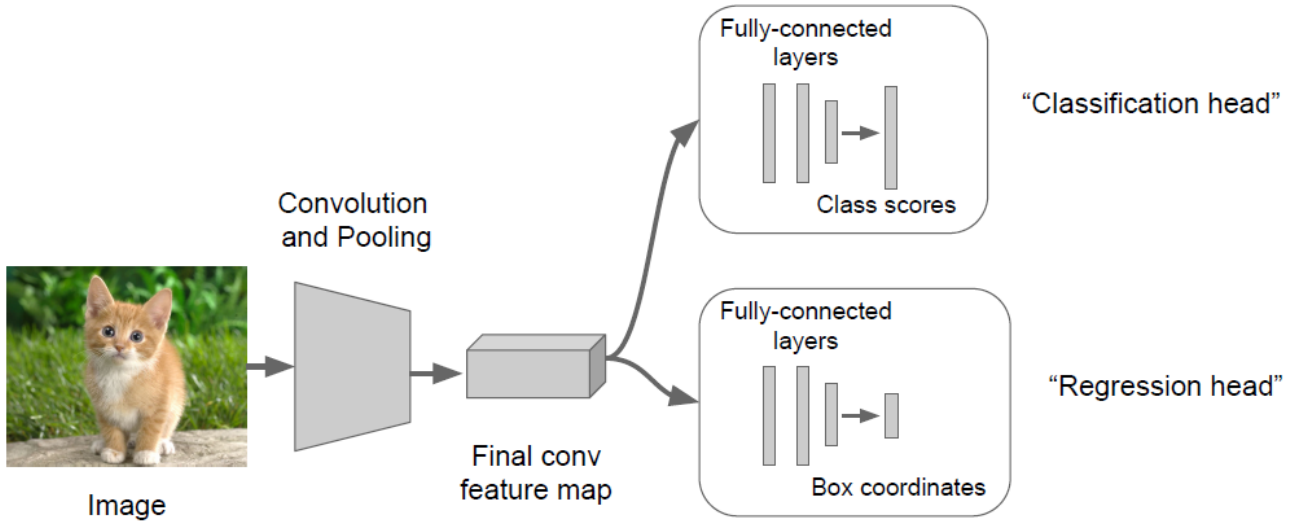
Correct output:
box coordinates
(4 numbers)

Loss:
L2 distance

Only one object,
simpler than detection

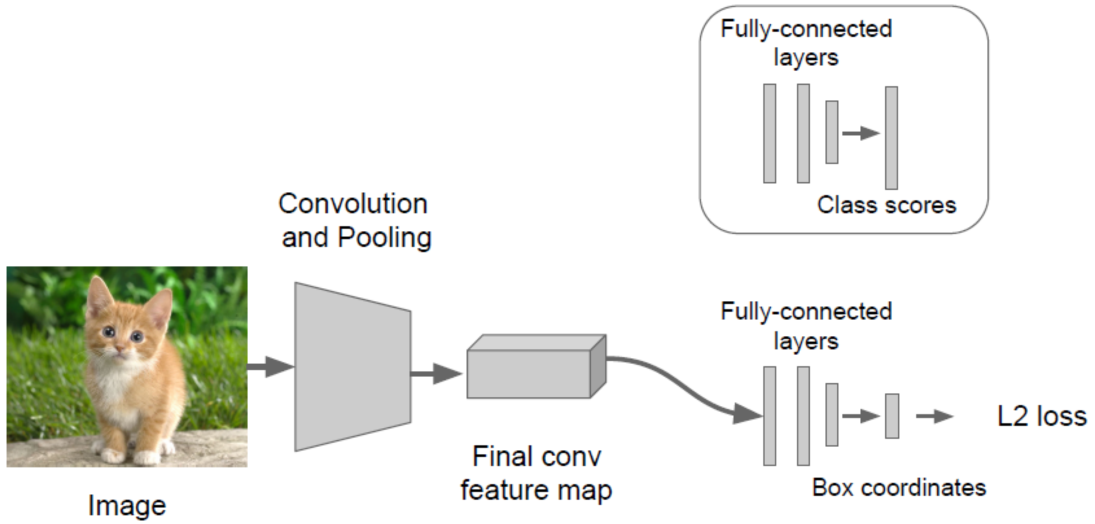
Idea #1: Localization as regression

- Steps
  - Train (or download) a classification model (AlexNet, VGG, GoogLeNet)
  - Attach new fully-connected "regression head" to the network



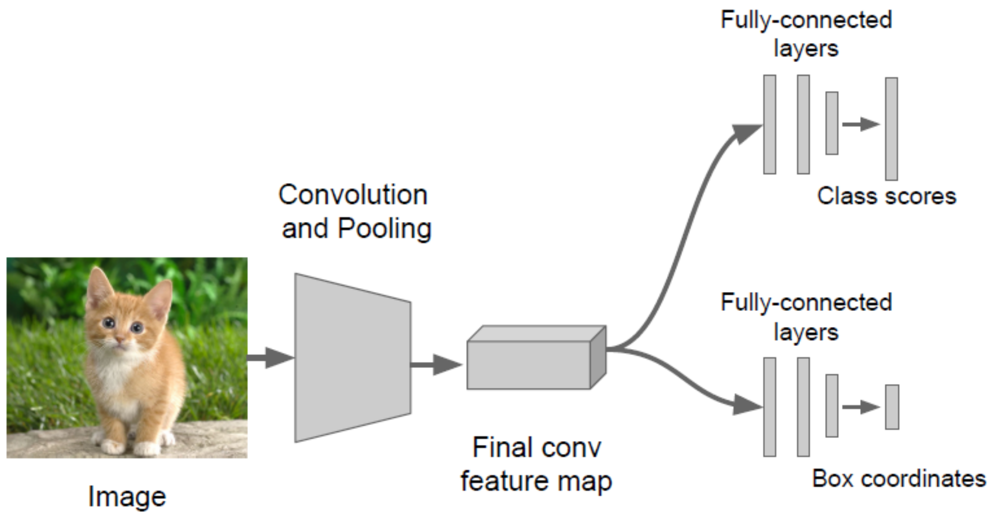
Idea #1: Localization as regression

- Steps
  - Train (or download) a classification model (AlexNet, VGG, GoogLeNet)
  - Attach new fully-connected "regression head" to the network
  - Train the regression head only with SGD and L2 loss

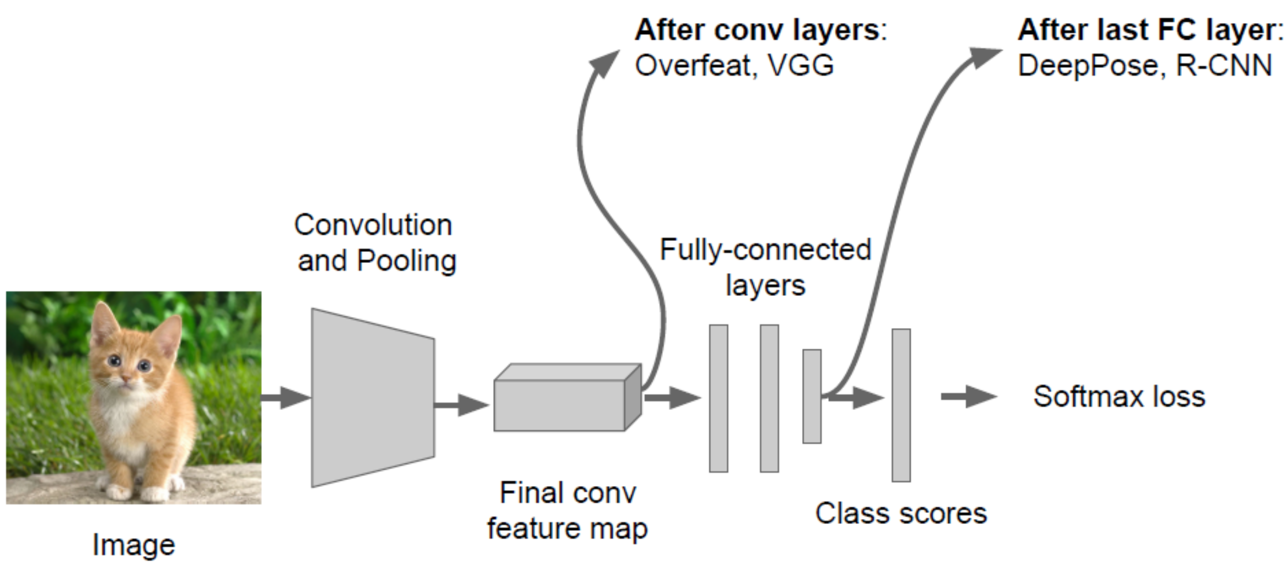


Idea #1: Localization as regression

- Steps
  - Train (or download) a classification model (AlexNet, VGG, GoogLeNet)
  - Attach new fully-connected “regression head” to the network
  - Train the regression head only with SGD and L2 loss
  - At test time use both heads



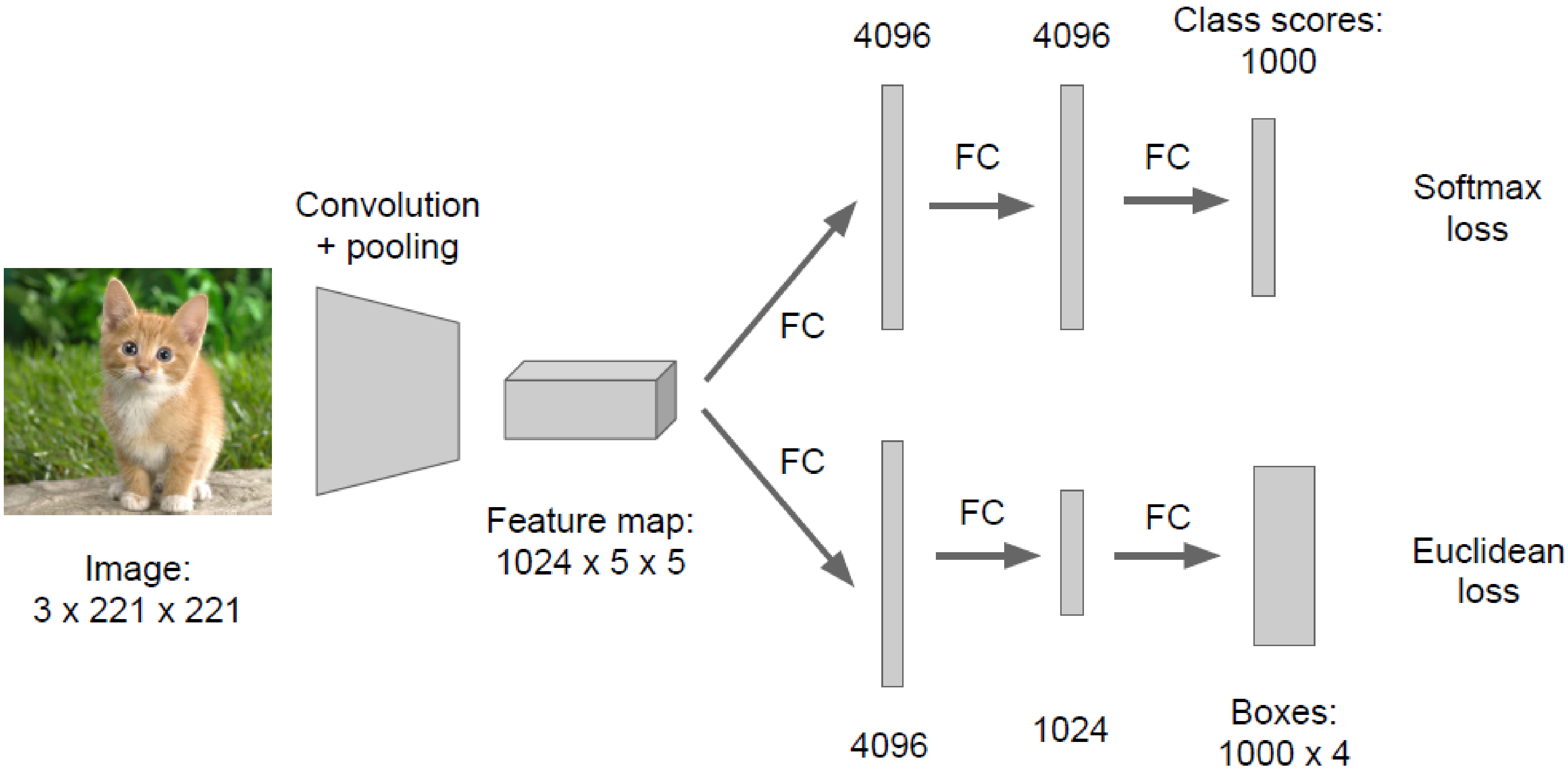
Where to attach the regression head?



Idea #2: Sliding Window

- Run classification + regression network at multiple locations on a high resolution image
- Combine classifier and regressor predictions across all scales for final prediction

Sliding Window: Overfeat

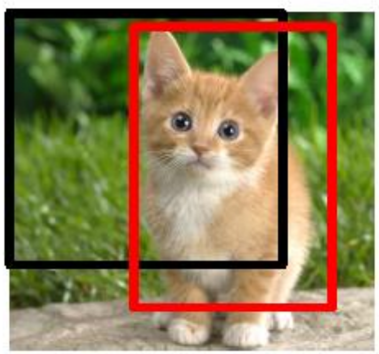


Sliding Window: Overfeat

Network input:
3 x 221 x 221

Larger image:
3 x 257 x 257

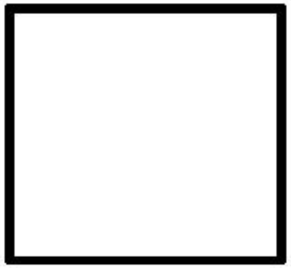
Network input:
3 x 221 x 221



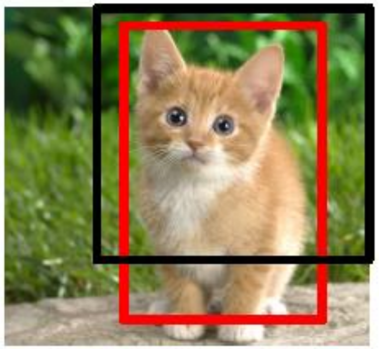
Larger image:
3 x 257 x 257

0.5	

Classification scores:
P(cat)



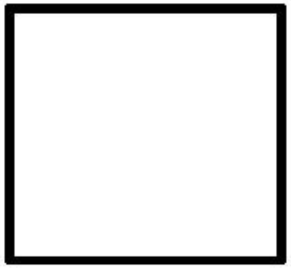
Network input:
3 x 221 x 221



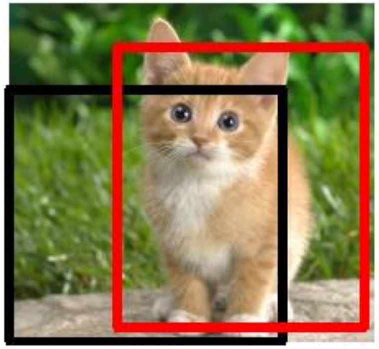
Larger image:
3 x 257 x 257

0.5	0.75

Classification scores:
P(cat)



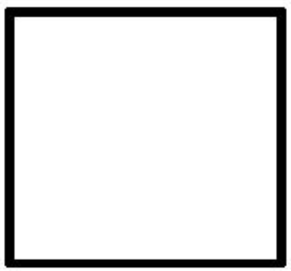
Network input:
3 x 221 x 221



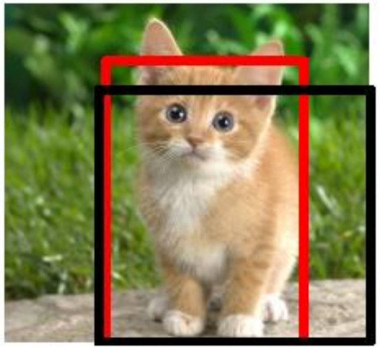
Larger image:
3 x 257 x 257

0.5	0.75
0.6	

Classification scores:
P(cat)



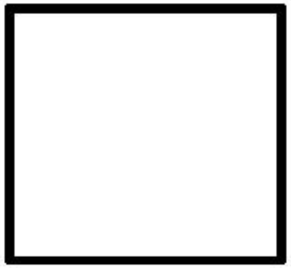
Network input:
3 x 221 x 221



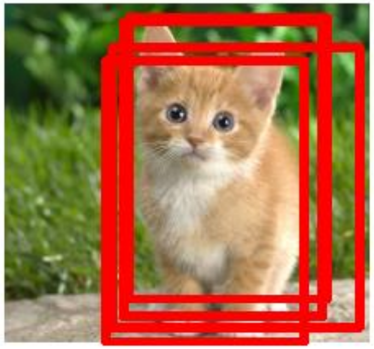
Larger image:
3 x 257 x 257

0.5	0.75
0.6	0.8

Classification scores:
P(cat)



Network input:
3 x 221 x 221

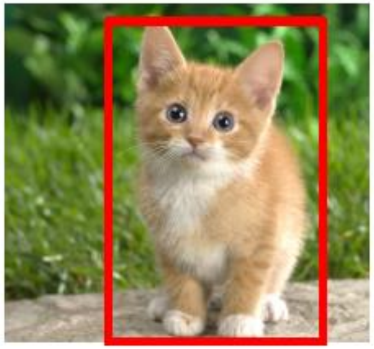


Larger image:
3 x 257 x 257

0.5	0.75
0.6	0.8

Classification scores:
P(cat)

Network input:
3 x 221 x 221



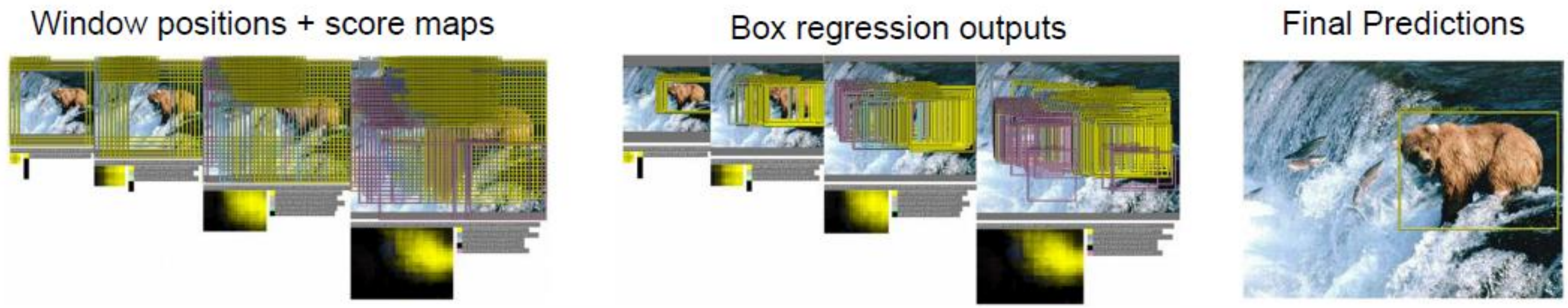
Larger image:
3 x 257 x 257

0.8

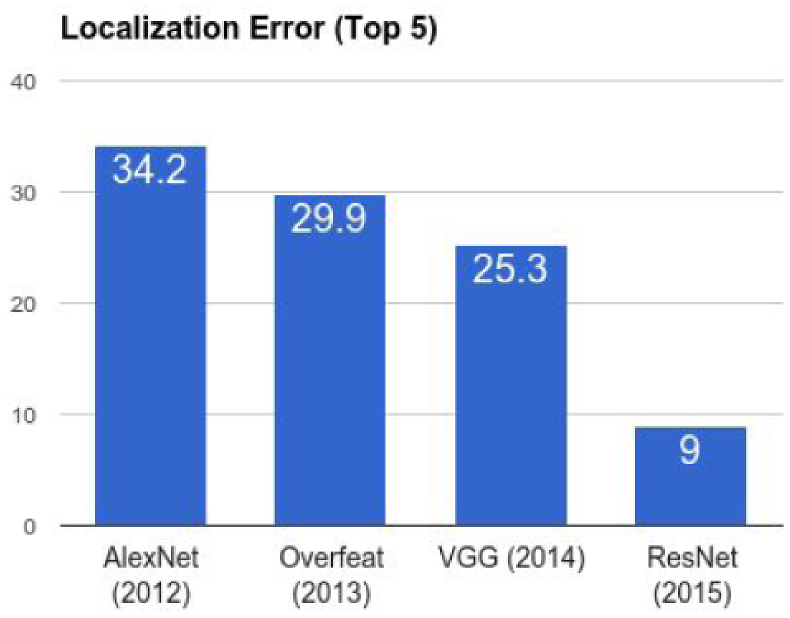
Classification score: P (cat)

Sliding Window: Overfeat

- In practice, use many sliding window location and multiple scales



ImageNet Classification + Localization



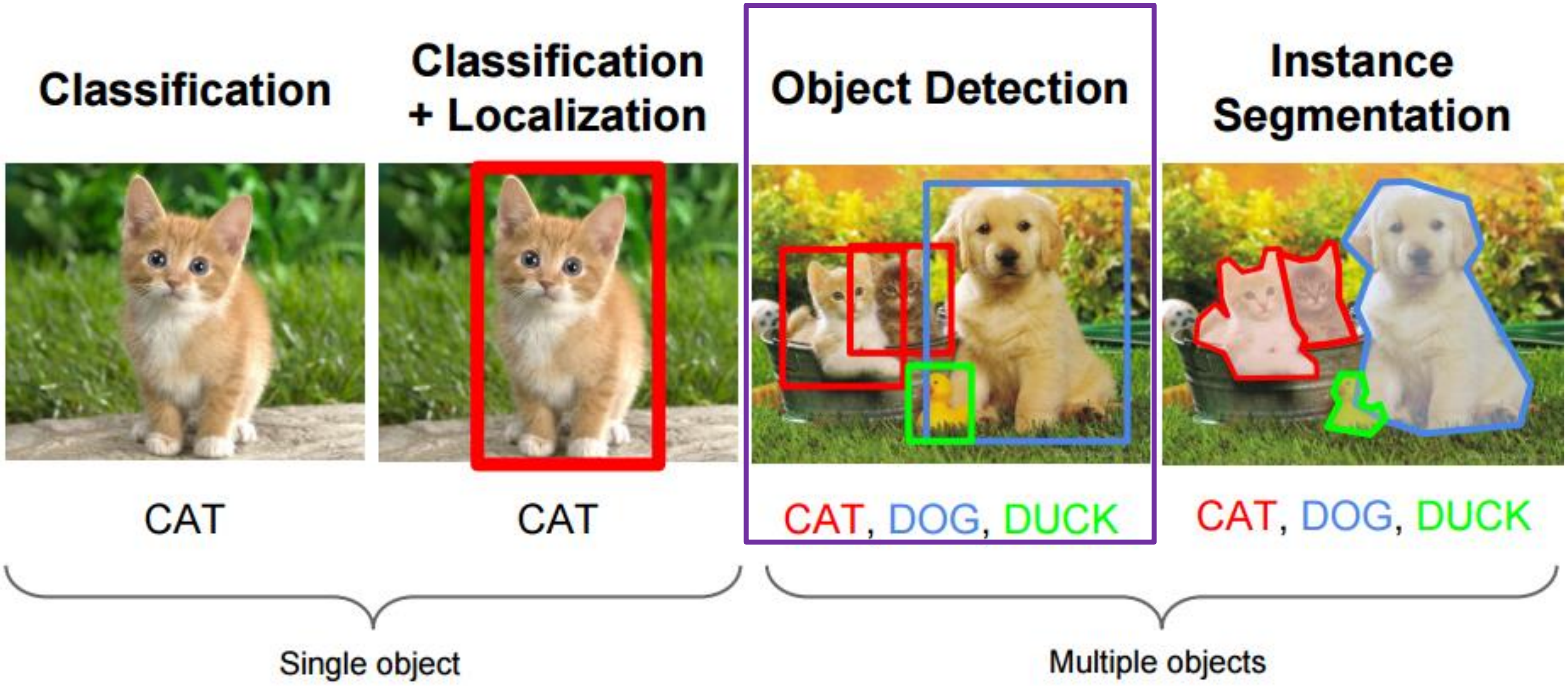
AlexNet: Localization method not published

Overfeat: Multiscale convolutional regression with box merging

VGG: Same as Overfeat, but fewer scales and locations; simpler method, gains all due to deeper features

ResNet: Different localization method (RPN) and much deeper features

Computer Vision Tasks



Detection as regression?

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

= 16 numbers

Detection as regression?

- Need variable sized outputs

CAT, (x, y, w, h)
CAT, (x, y, w, h)
....
CAT (x, y, w, h)

= many numbers

Detection as classification

- Detection as classification
  - Problem: Need to test many positions and scales
  - Solution: If your classifier is fast enough, just do it

- Detection with a CNN classifier
  - Problem: Need to test many positions and scales, and use a computationally demanding classifier
  - Solution: Only look at a tiny subset of possible positions

CAT? NO

DOG? NO

CAT? YES!

DOG? NO

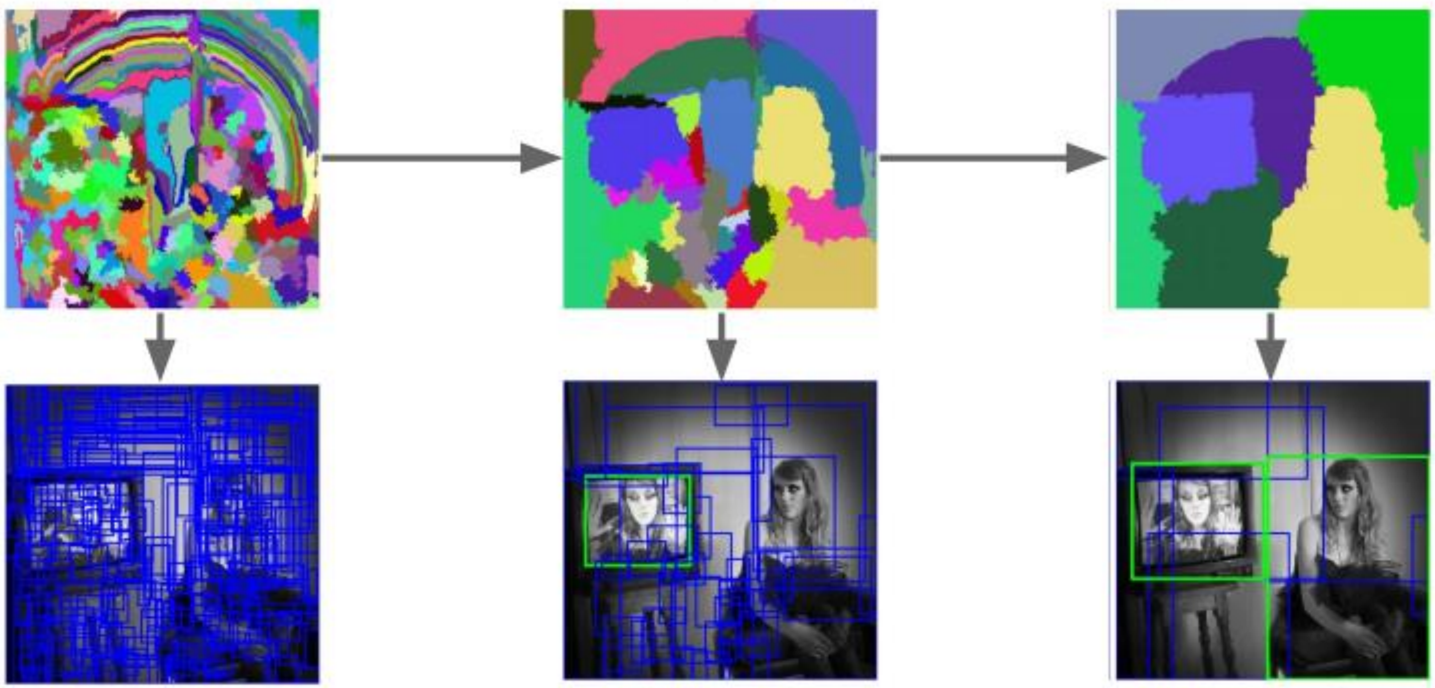
CAT? NO

DOG? NO

Region Proposals

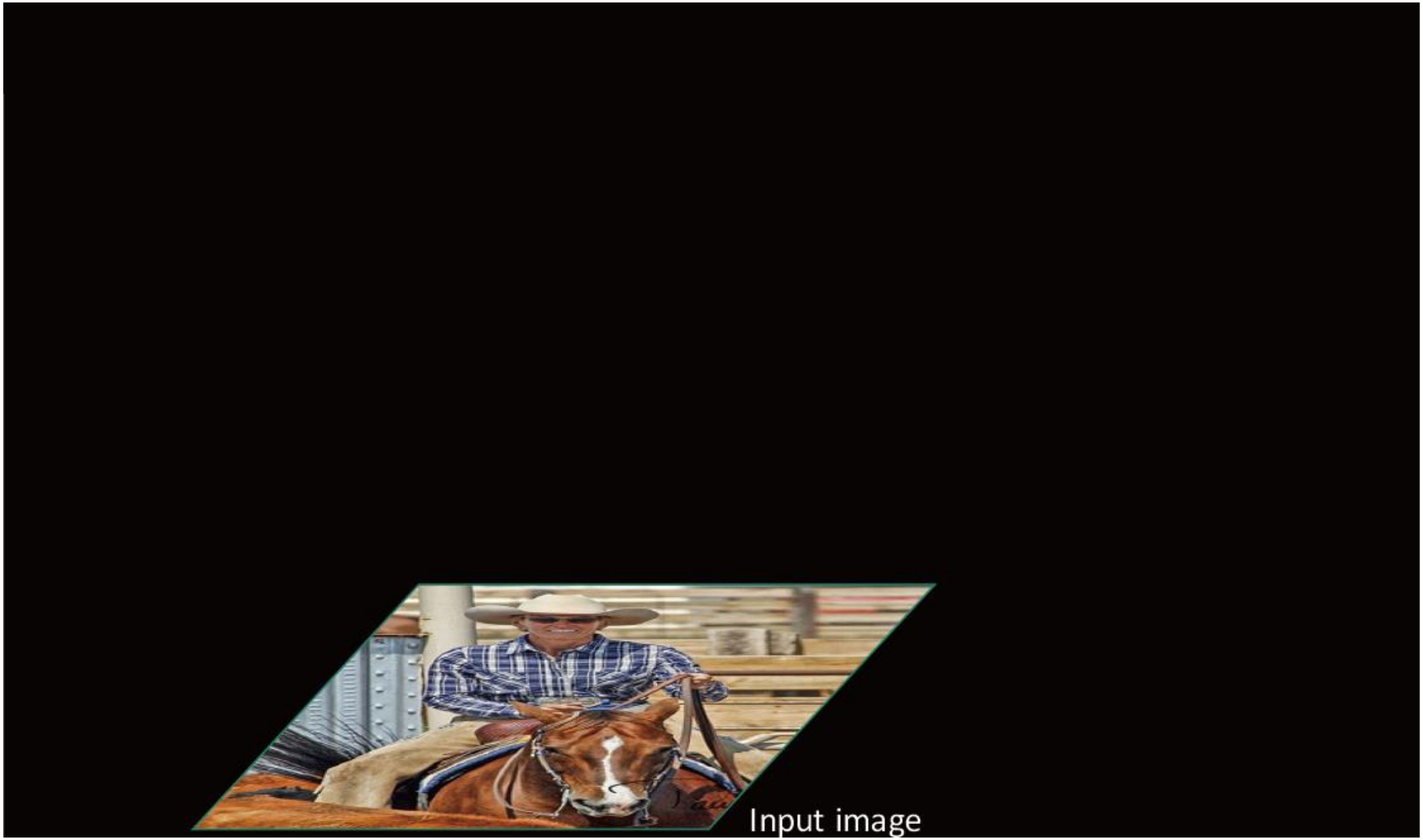
Bottom-up segmentation, merging regions at multiple scales

Convert regions to boxes



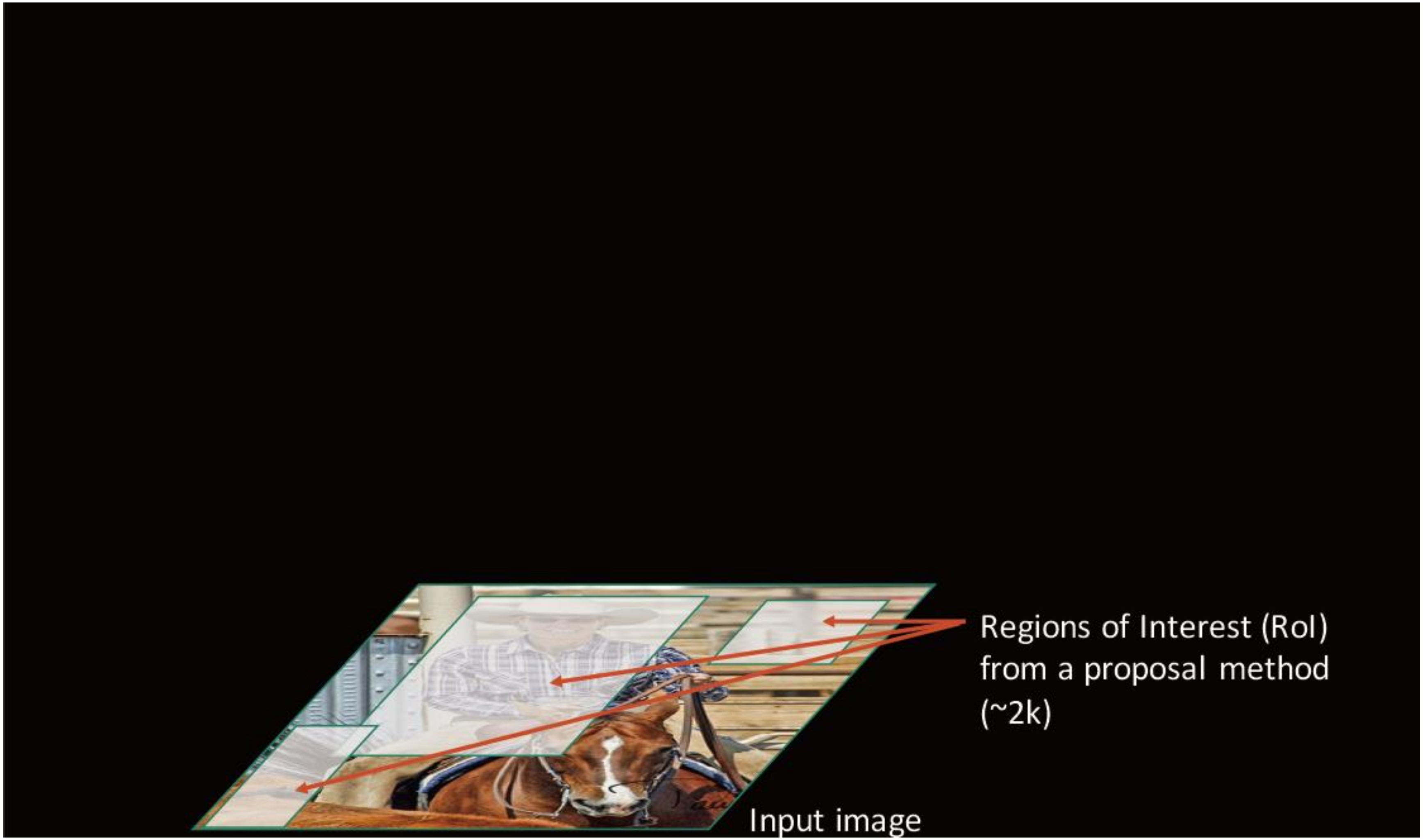
R-CNN
(REGIONS WITH CNN FEATURES)

R-CNN



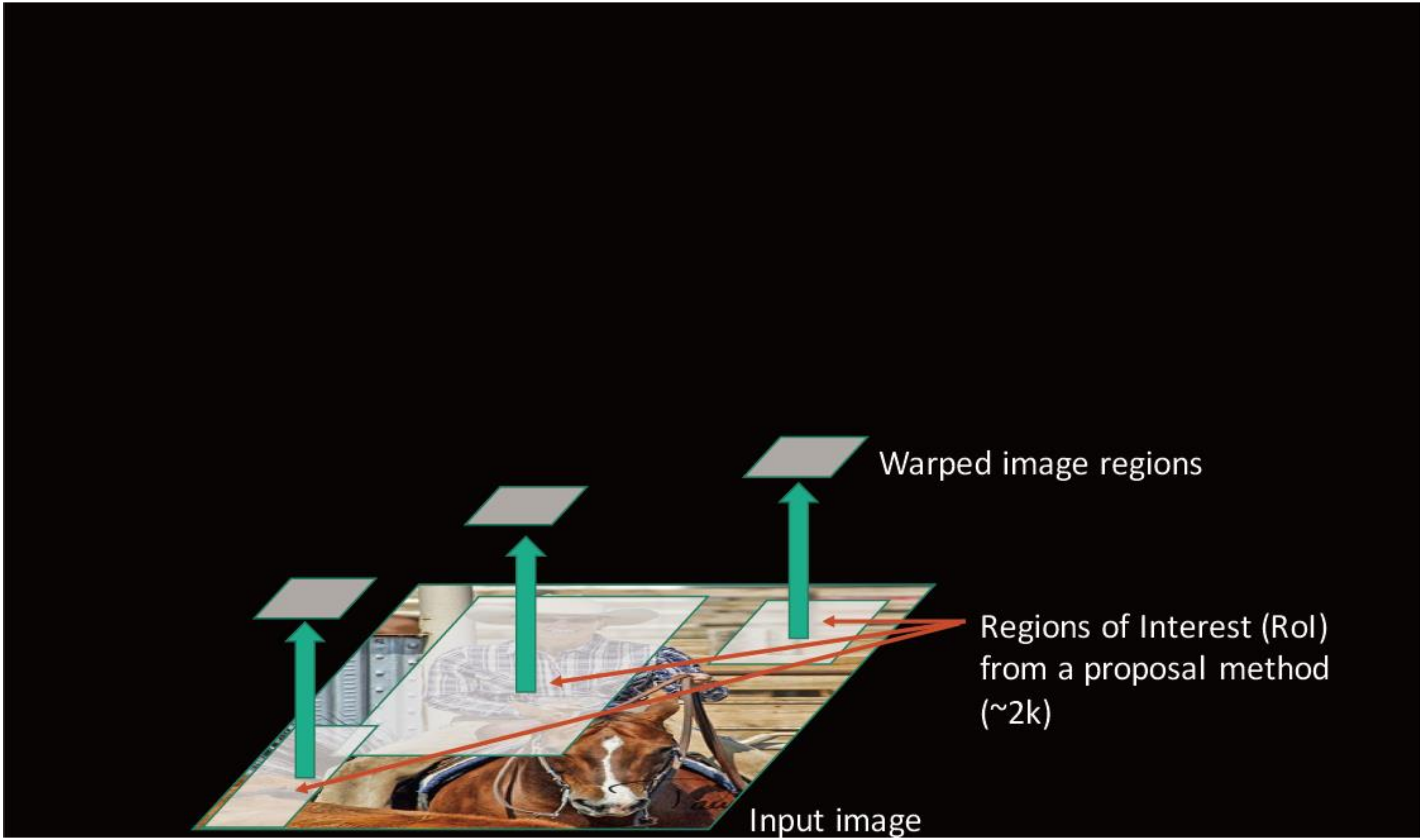
Slide credit: Ross Girschick

R-CNN



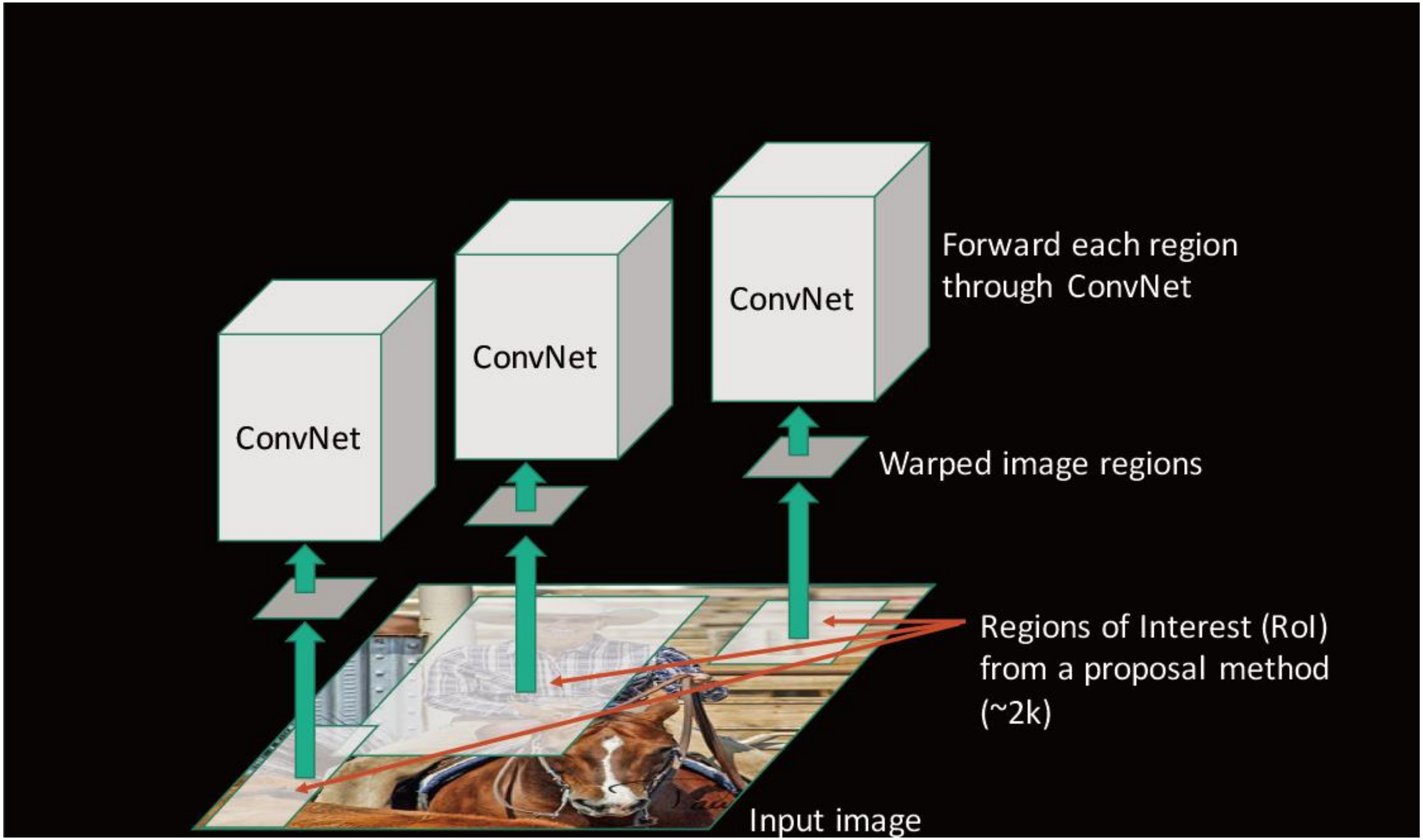
Slide credit: Ross Girschick

R-CNN



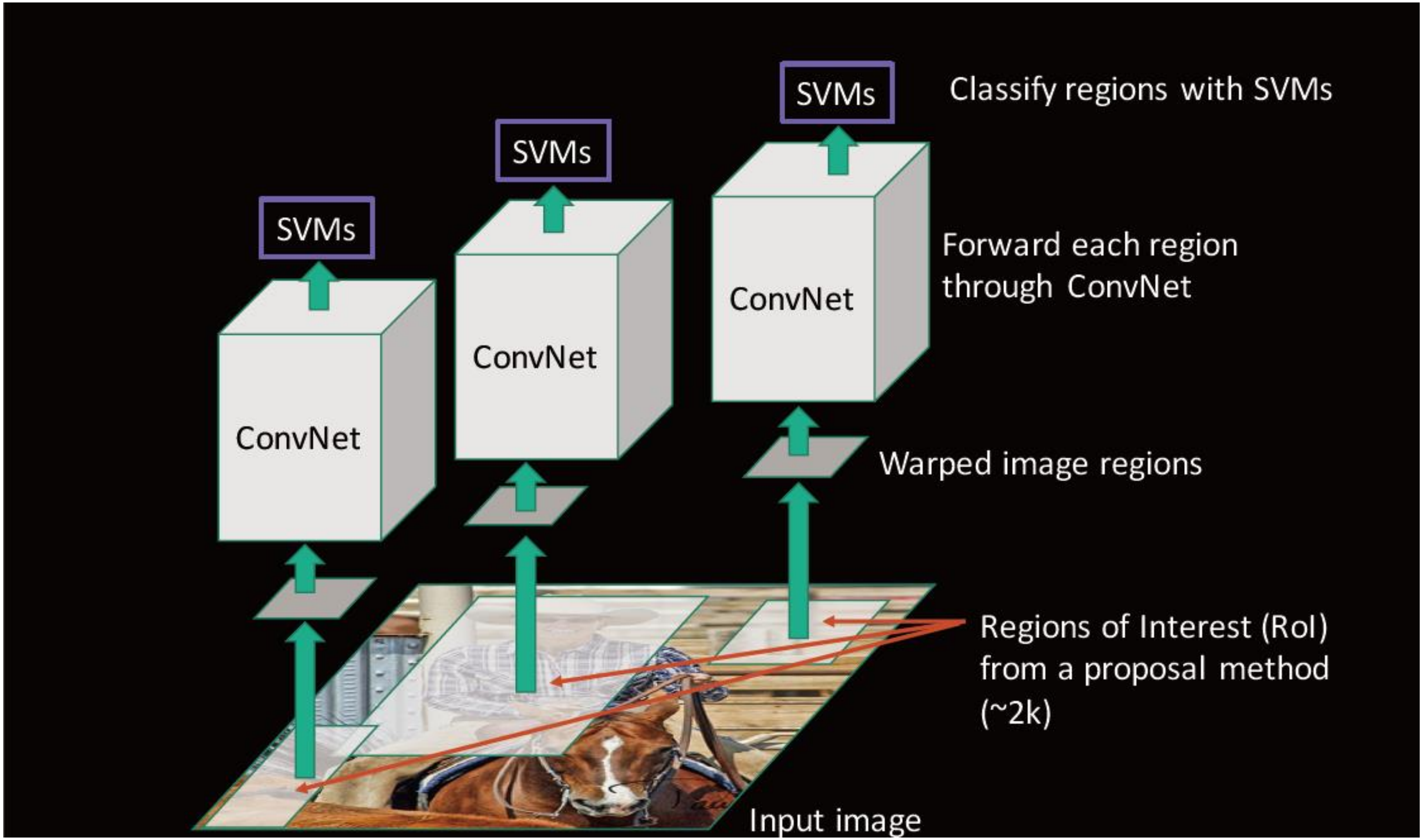
Slide credit: Ross Girschick

R-CNN



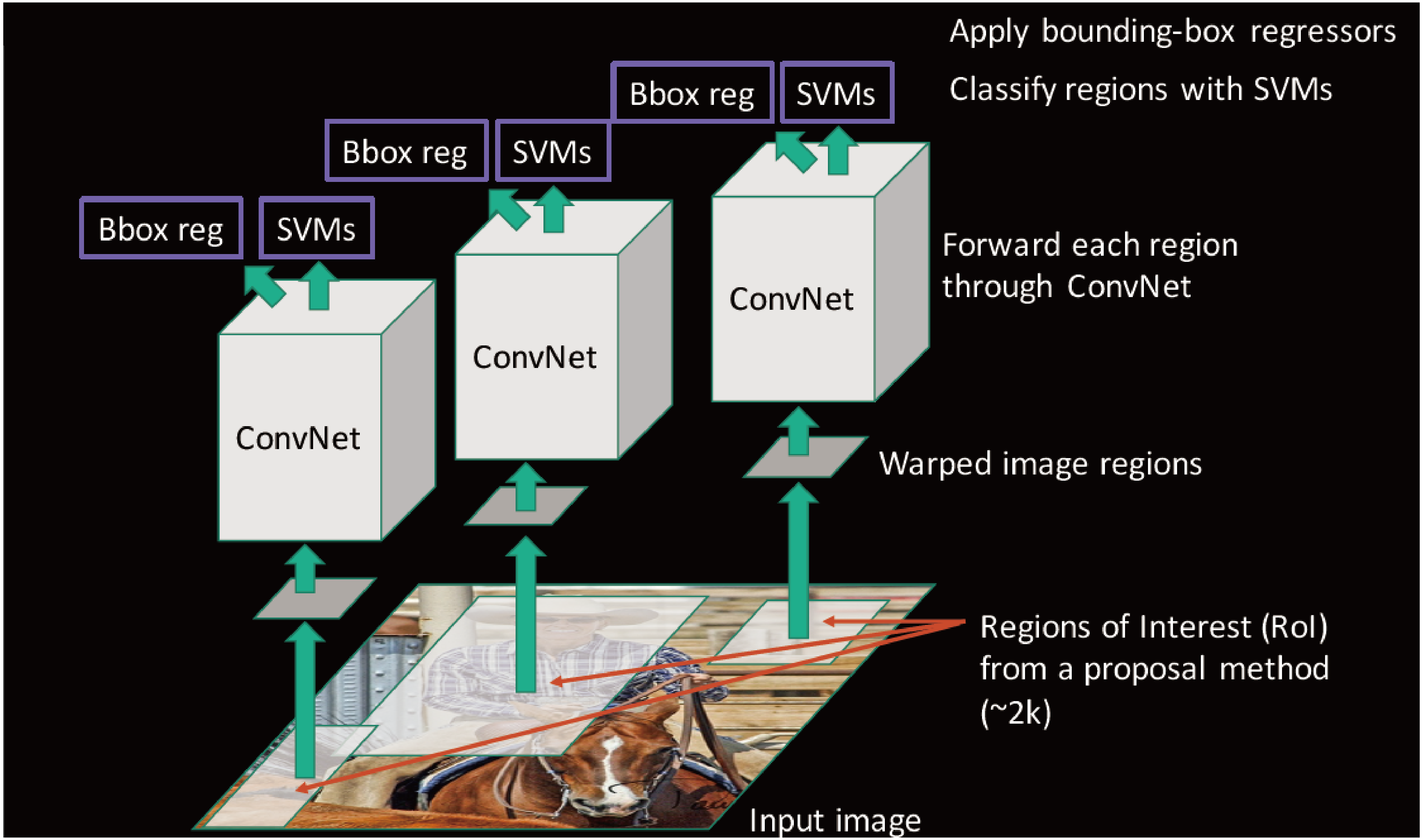
Slide credit: Ross Girschick

R-CNN



Slide credit: Ross Girschick

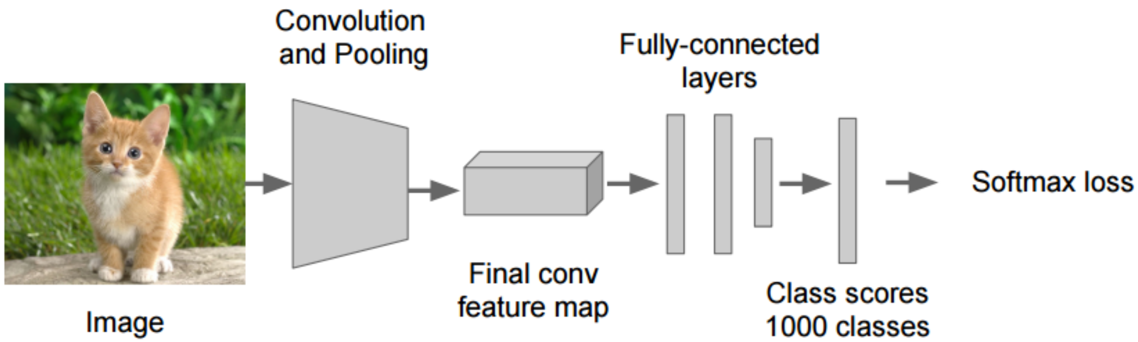
R-CNN



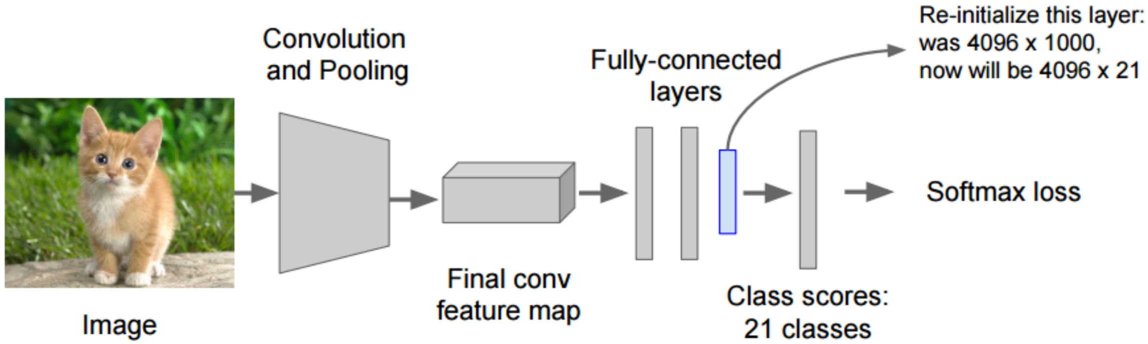
Slide credit: Ross Girschick

Training steps

- Step 1: Train a classification model for ImageNet (AlexNet)

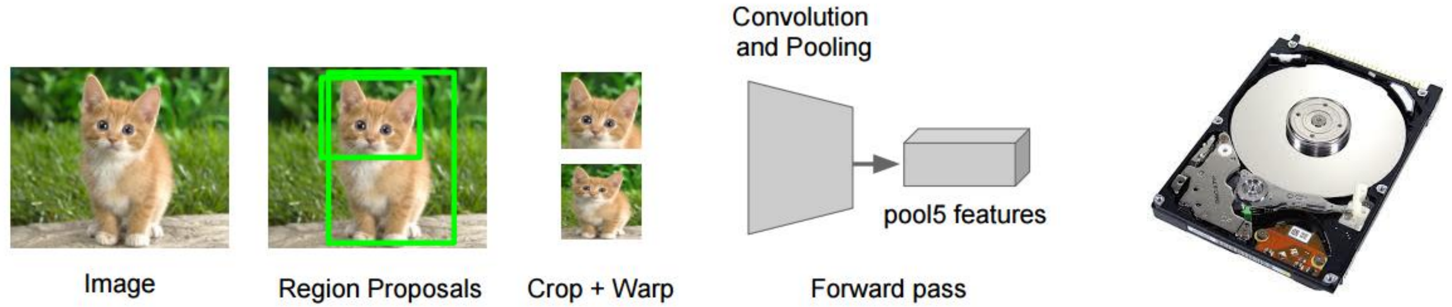


- Step 2: Fine-tune model for detection (20 object classes + backgrounds)



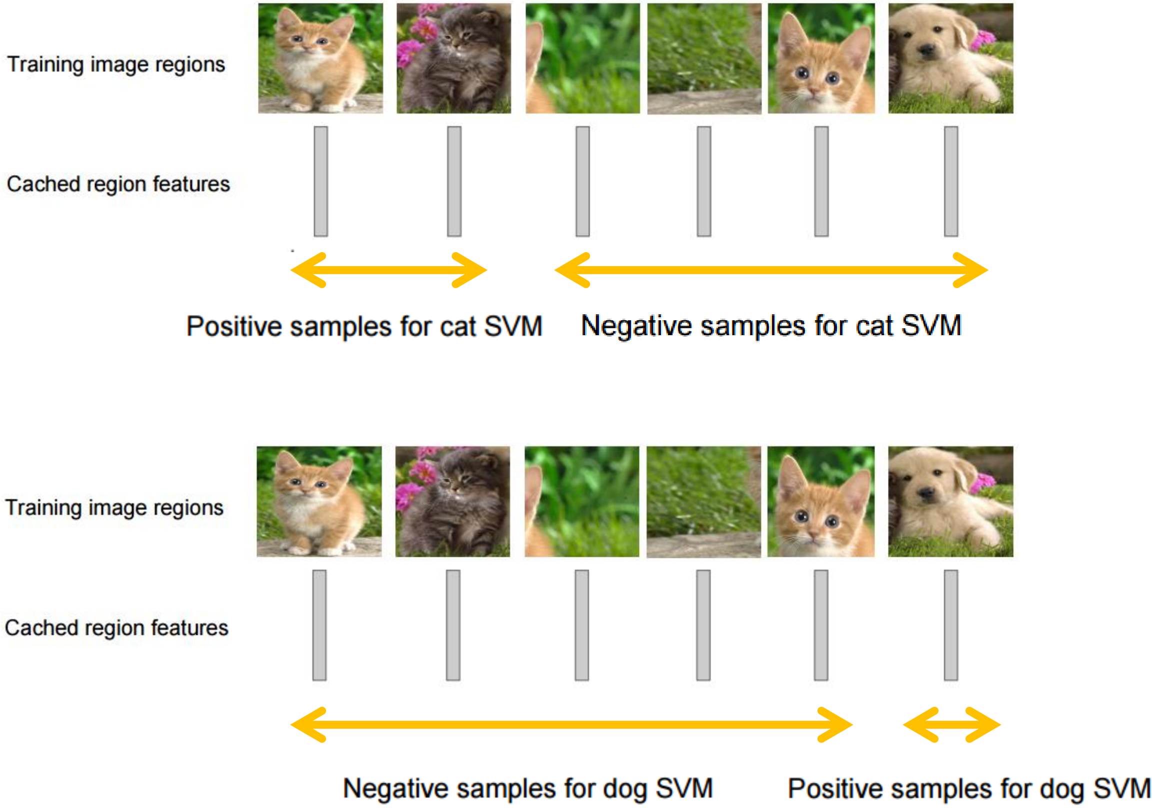
Training steps

- Step 3: Extract features
  - Extract region proposals for all images
  - For each region: warp to CNN input size, run forward through CNN, save pool5 features to disk (~ 200 GB)



Training steps

- Step 4: Train one binary SVM per class to classify region features

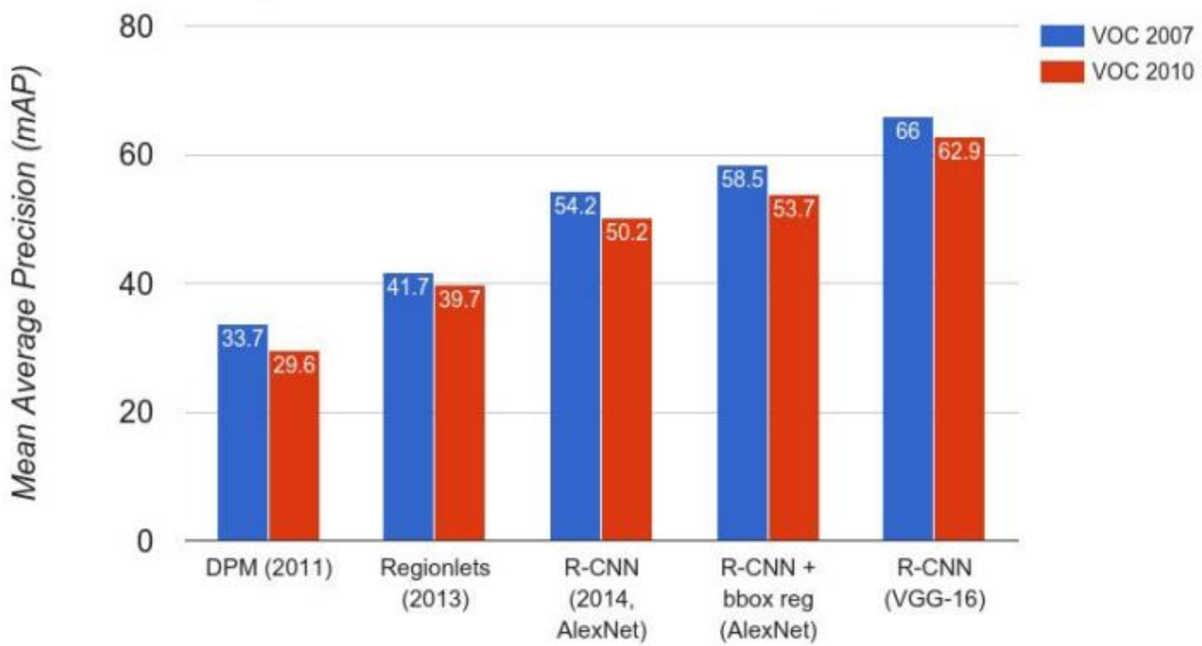


Training steps

- Step 5 (bounding-box regression): For each class, train a linear regression model to map from cached features to offsets to GT boxes to make up for “slightly wrong” proposals

Evaluation

- We use a metric called "mean average precision" (mAP)
- Compute average precision (AP) separately for each class, then average over classes



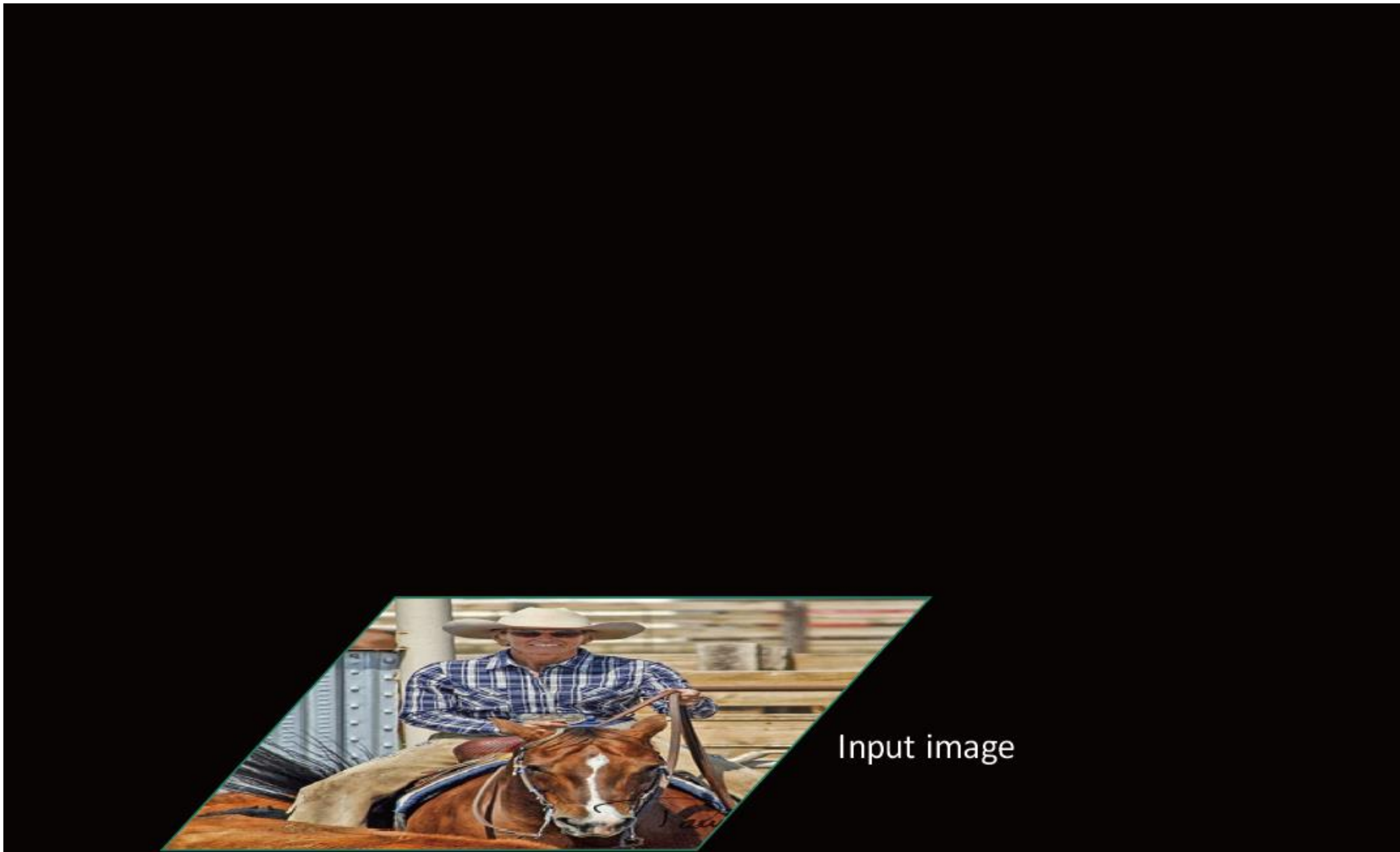
Limitations of R-CNN

- 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
  - Fixed by SPP net [He et al. ECCV14]

Slide credit: Ross Girschick

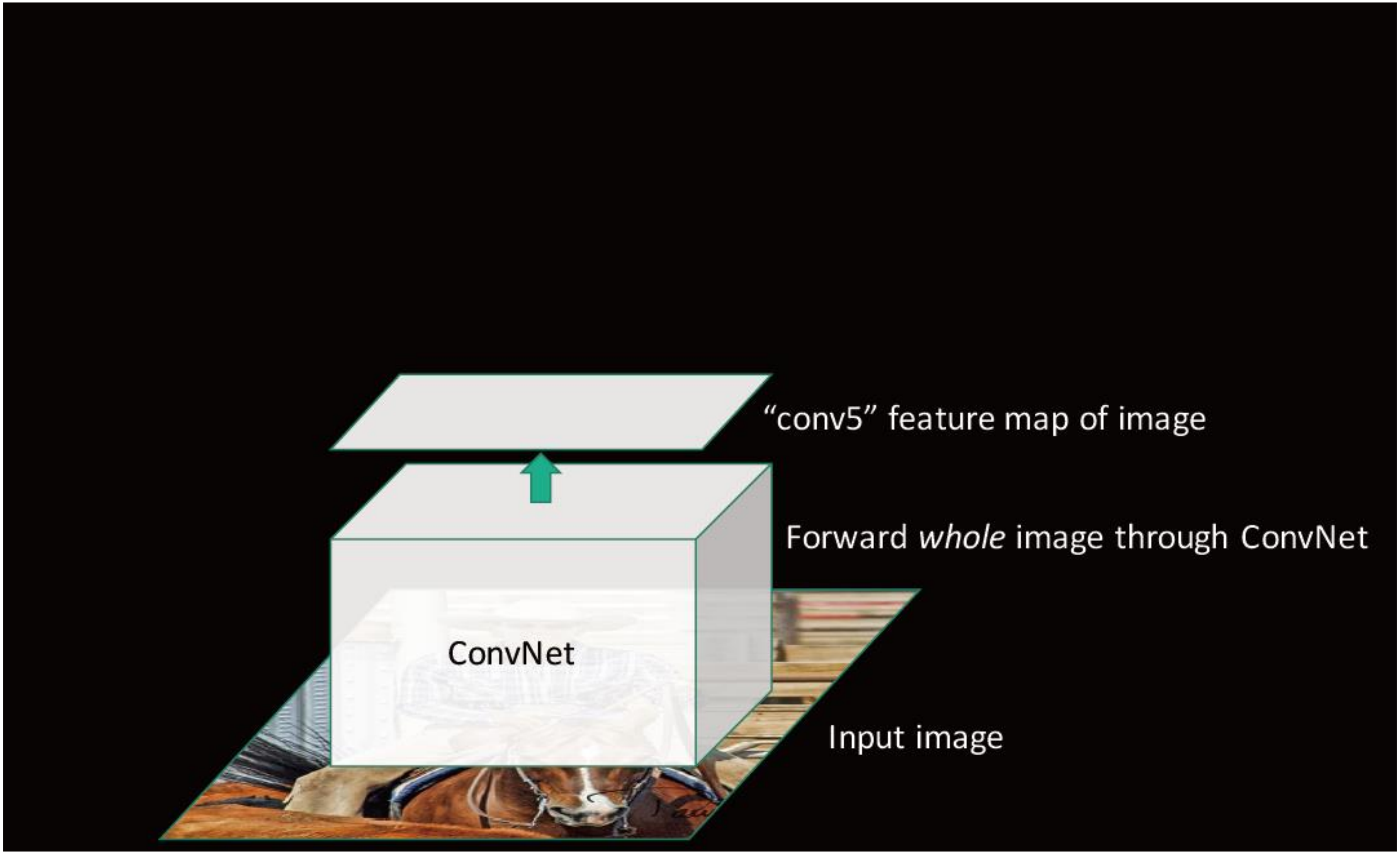
SPATIAL PYRAMID POOLING–NET

SPP net



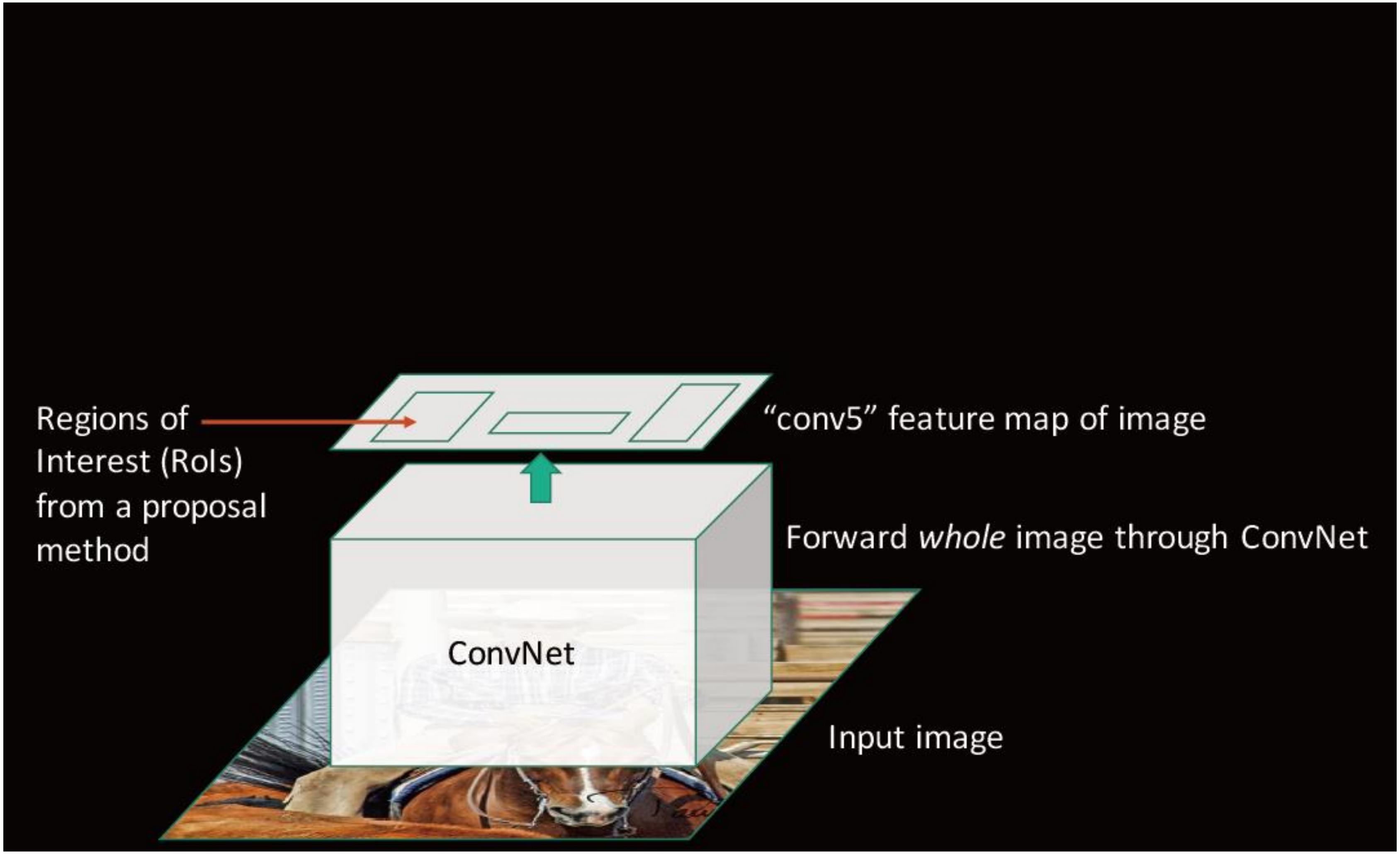
Slide credit: Ross Girschick

SPP net



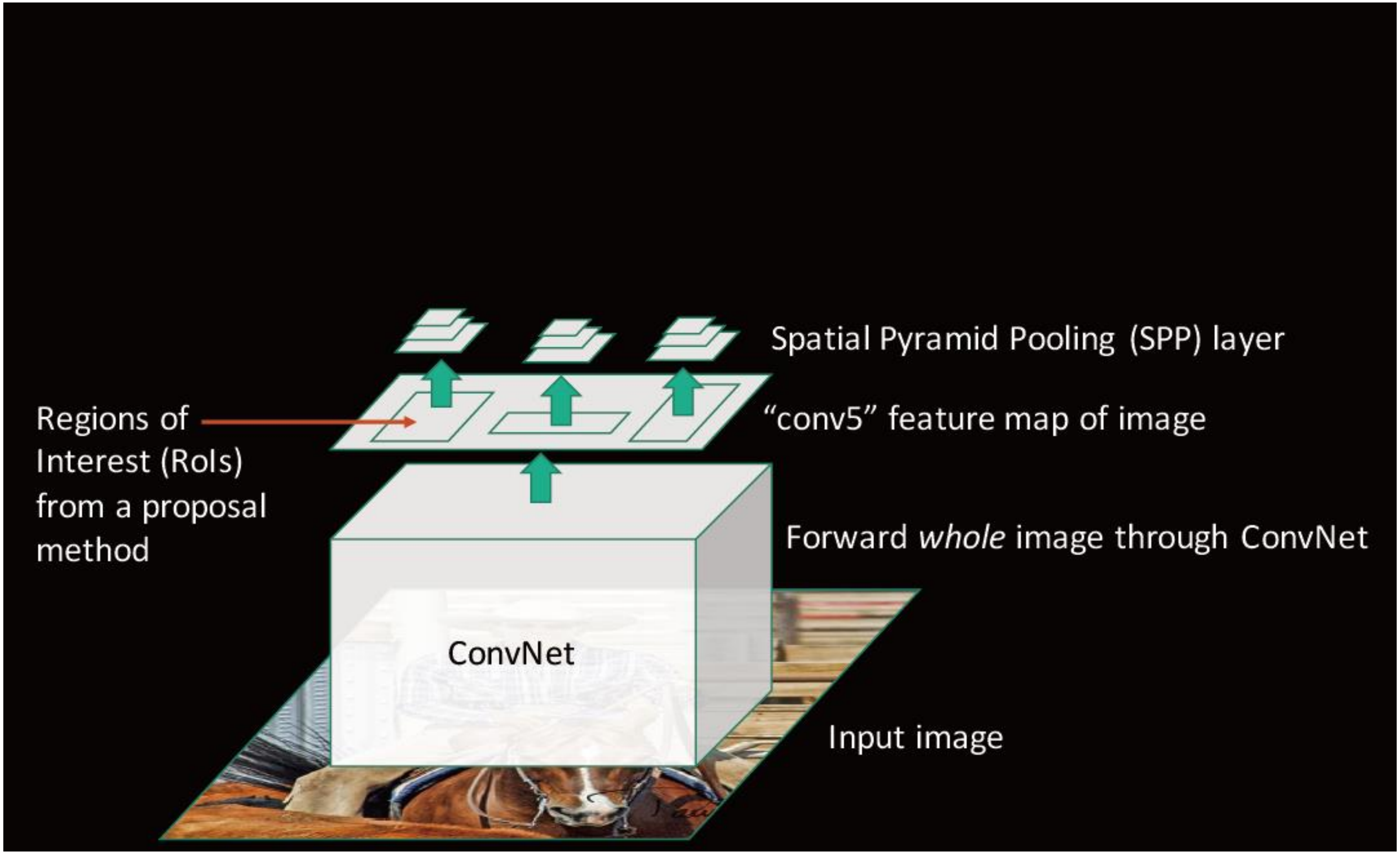
Slide credit: Ross Girschick

SPP net



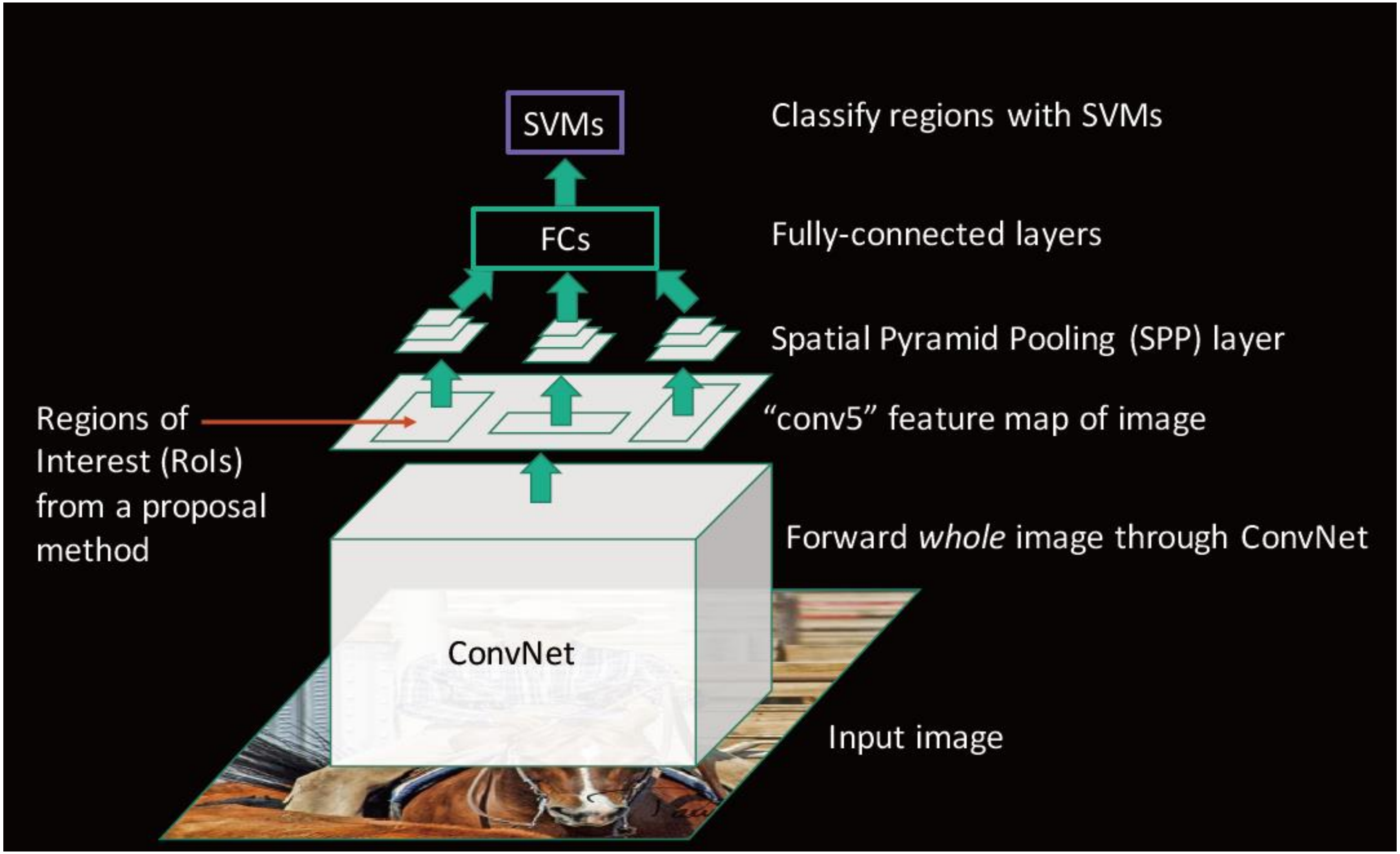
Regions of Interest (RoIs) from a proposal method
"conv5" feature map of image
Forward *whole* image through ConvNet
ConvNet
Input image

SPP net

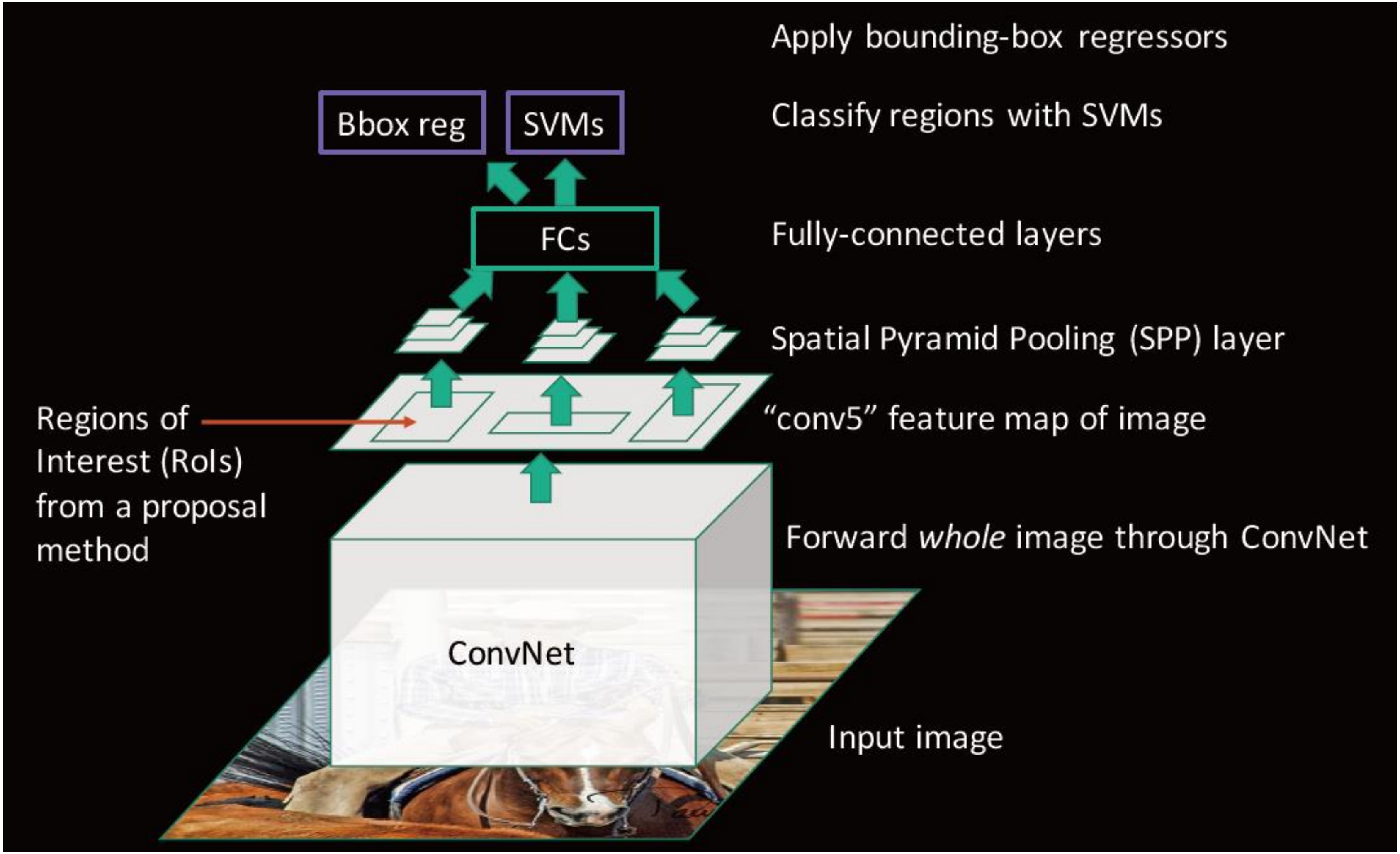


Spatial Pyramid Pooling (SPP) layer
Regions of Interest (RoIs) from a proposal method
"conv5" feature map of image
Forward *whole* image through ConvNet
ConvNet
Input image

SPP net


SVMs
Classify regions with SVMs
FCs
Fully-connected layers
Spatial Pyramid Pooling (SPP) layer
Regions of Interest (RoIs) from a proposal method
"conv5" feature map of image
Forward *whole* image through ConvNet
ConvNet
Input image

SPP net

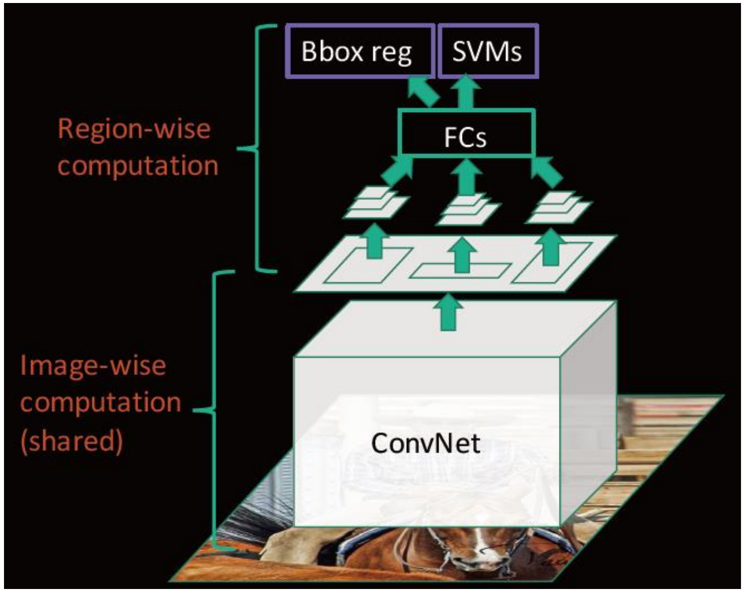


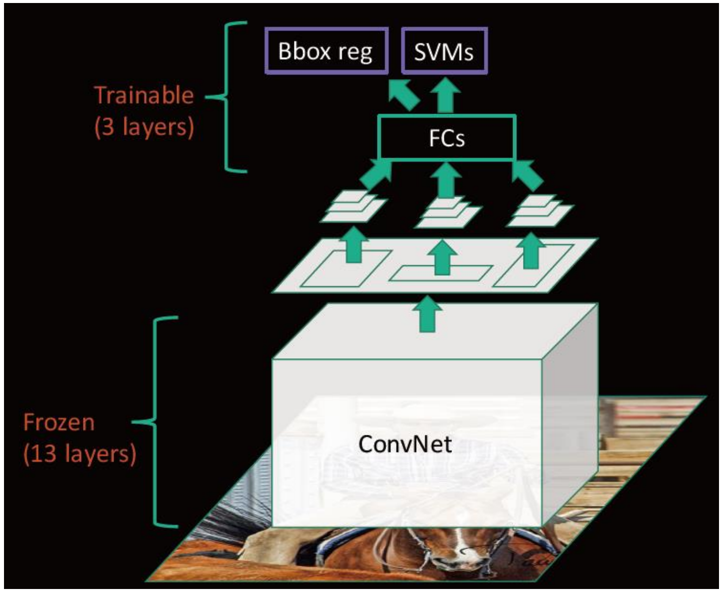
Slide credit: Ross Girschick

SPP net

- What's good?
  - It makes testing fast

- What's wrong?
  - Ad hoc training objectives
  - Training is slow (25h), takes a lot of disk space
  - Cannot update parameters below SPP layer during training





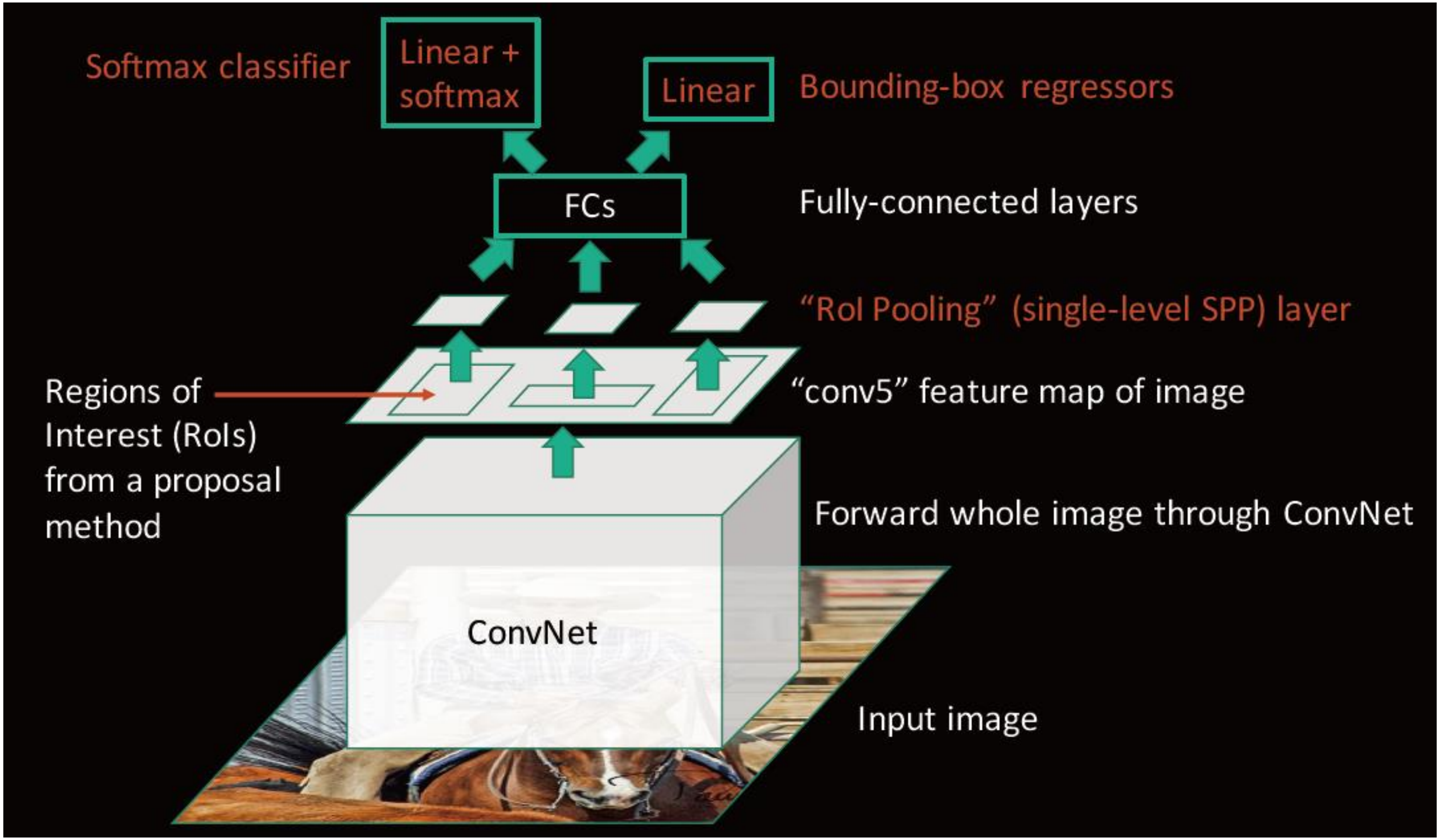
FAST R-CNN

Fast R-CNN

- Fast test time, like SPP-net
- One network, trained in one stage
- Higher mean average precision than R-CNN and SPP net

Slide credit: Ross Girschick

Fast R-CNN (test time)

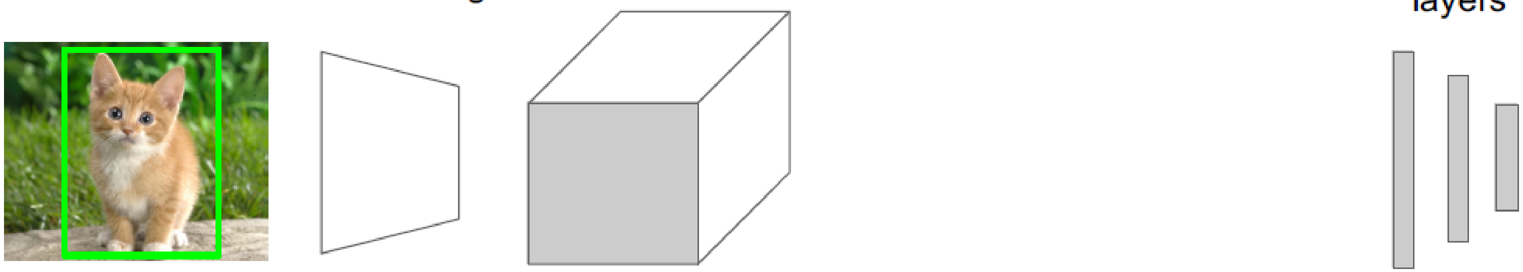


Slide credit: Ross Girschick

ROI pooling layer

Convolution and Pooling

Fully-connected layers

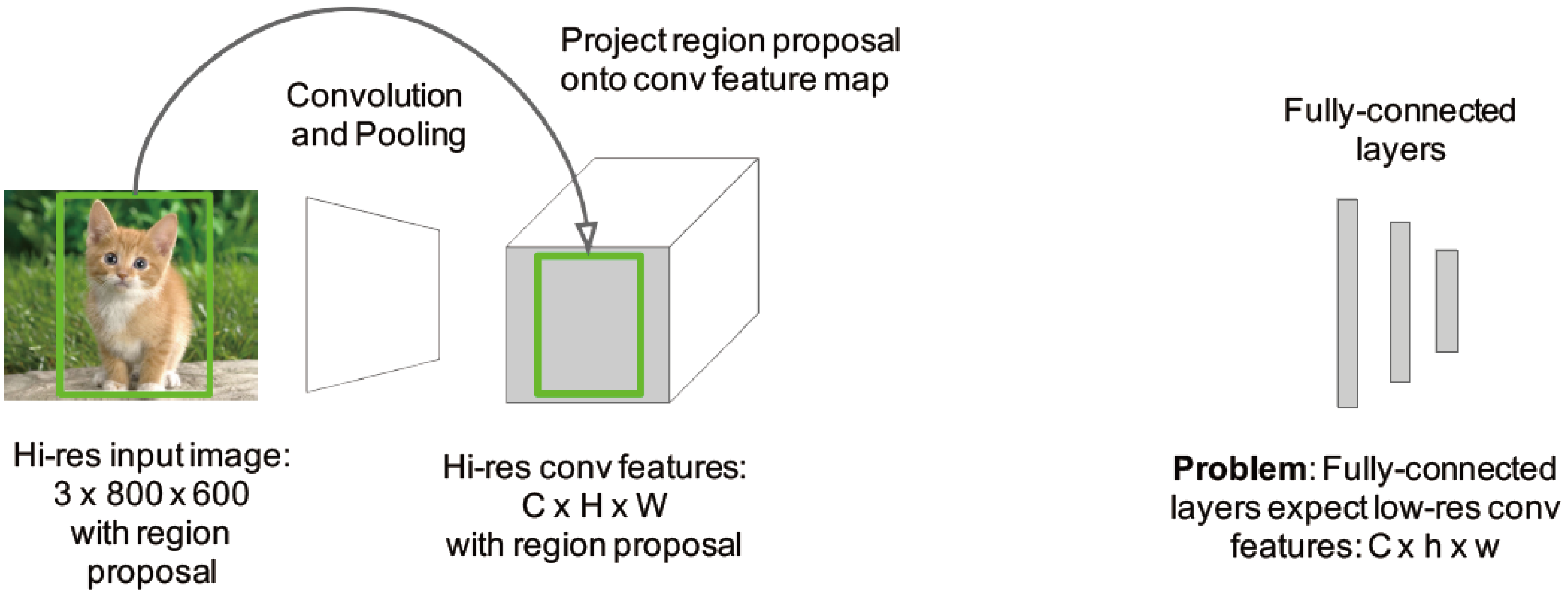


Hi-res input image:
3 x 800 x 600
with region
proposal

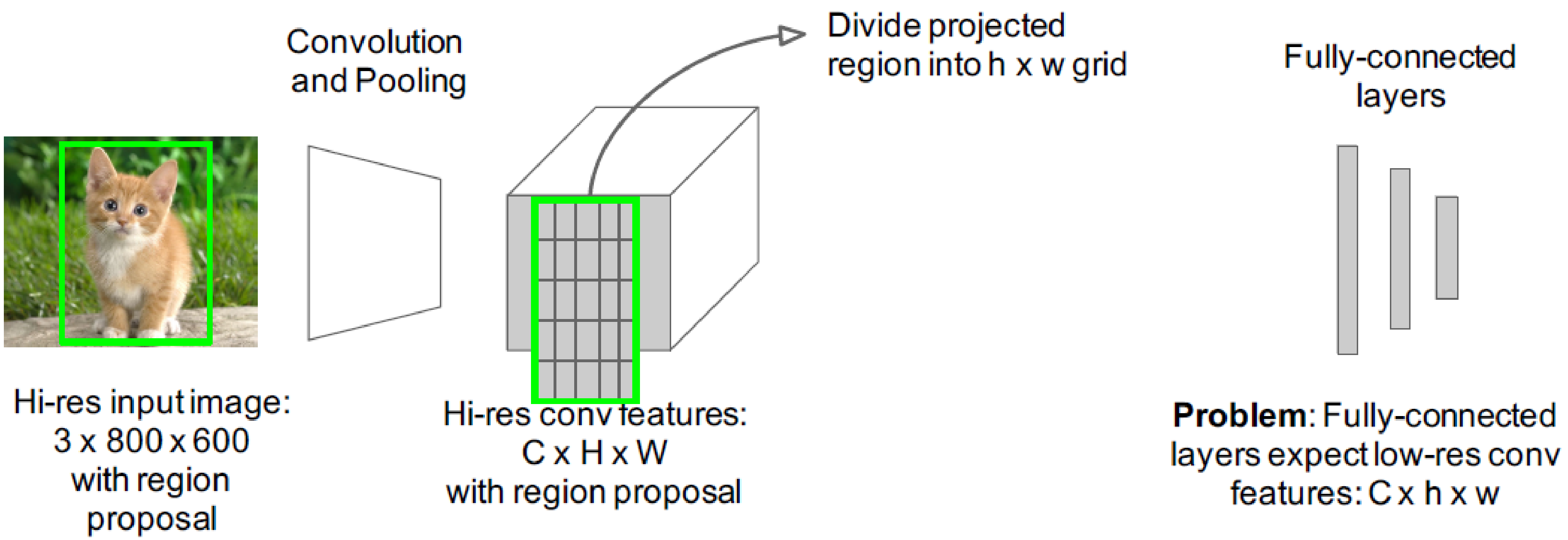
Hi-res conv features:
C x H x W
with region proposal

Problem: Fully-connected layers expect low-res conv features: C x h x w

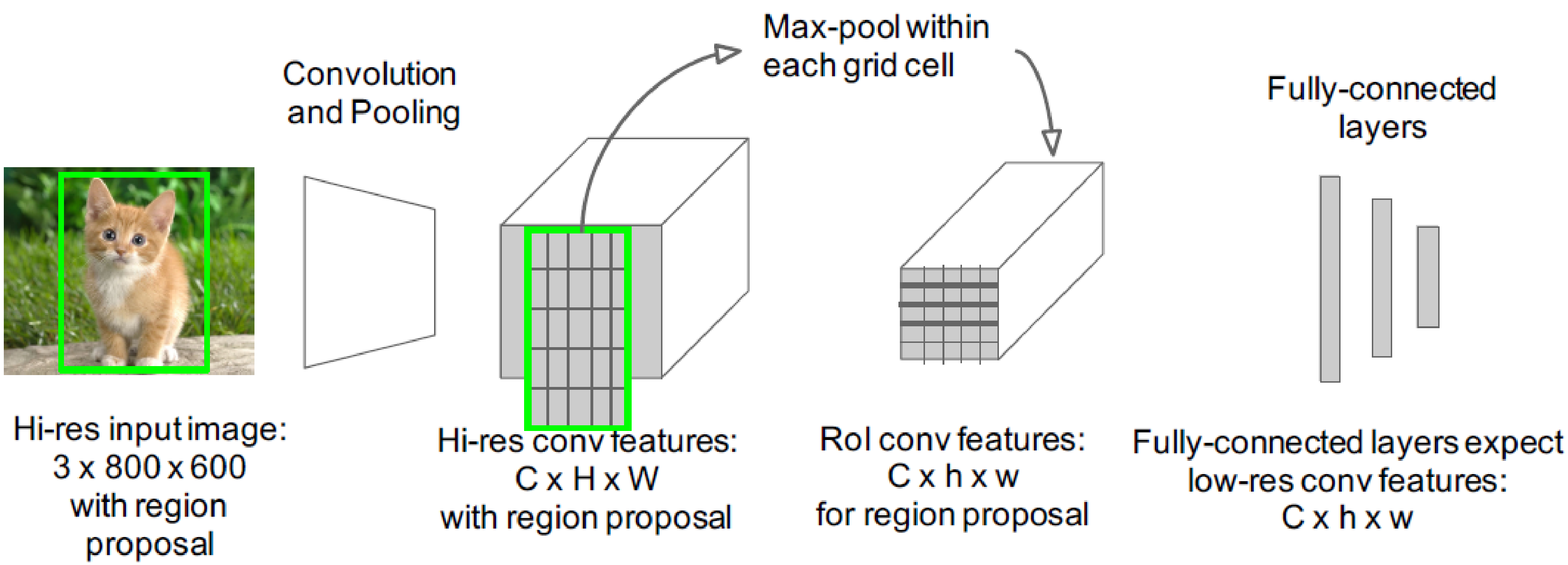
ROI pooling layer



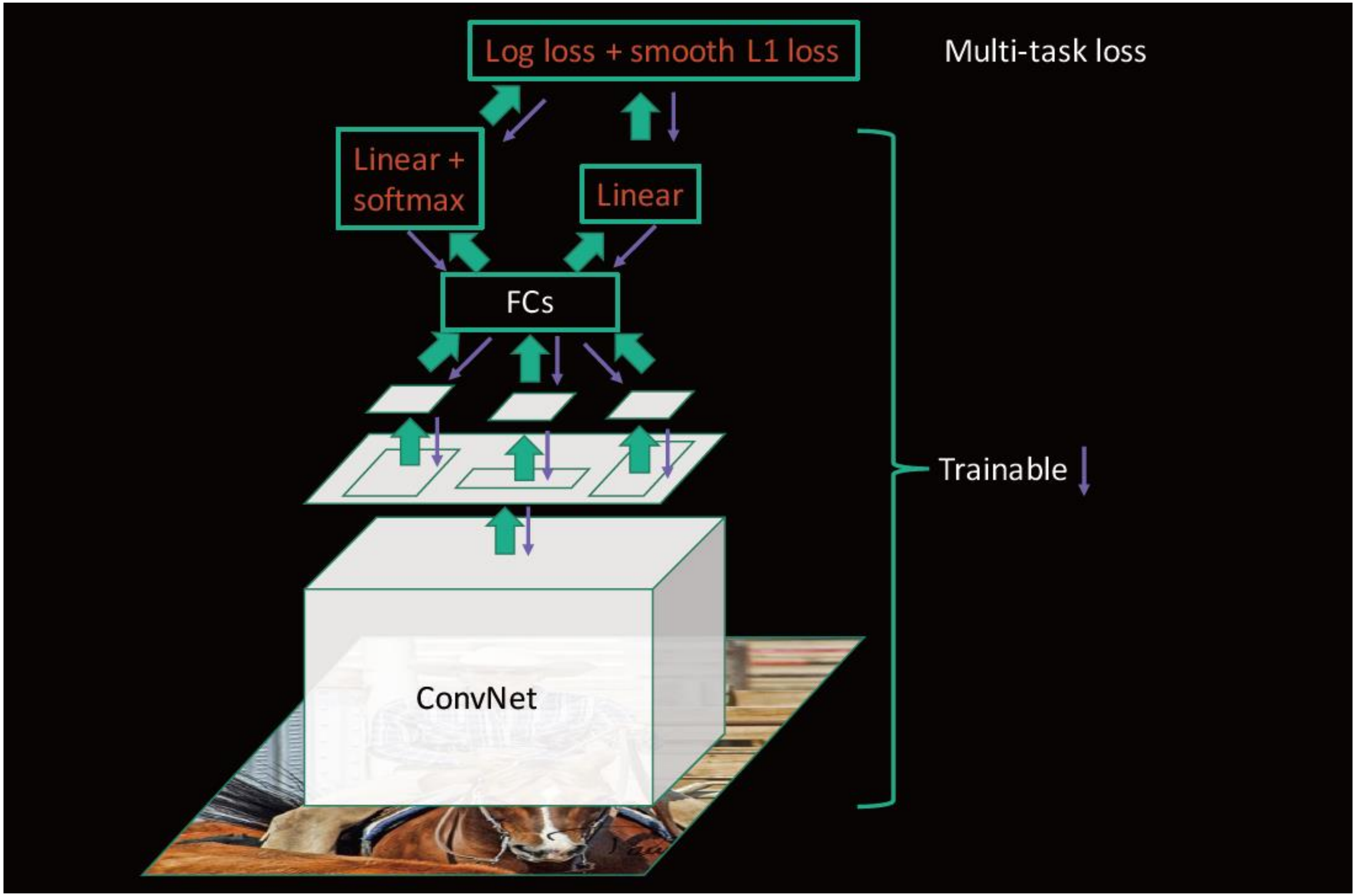
ROI pooling layer



ROI pooling layer



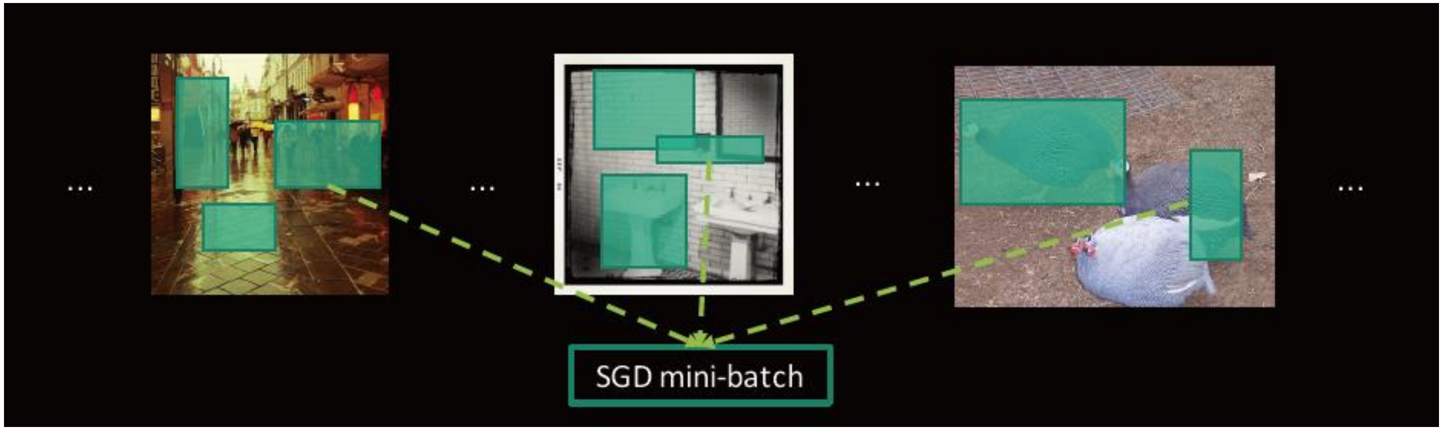
Fast R-CNN (training time)



Slide credit: Ross Girschick

Fast R-CNN training

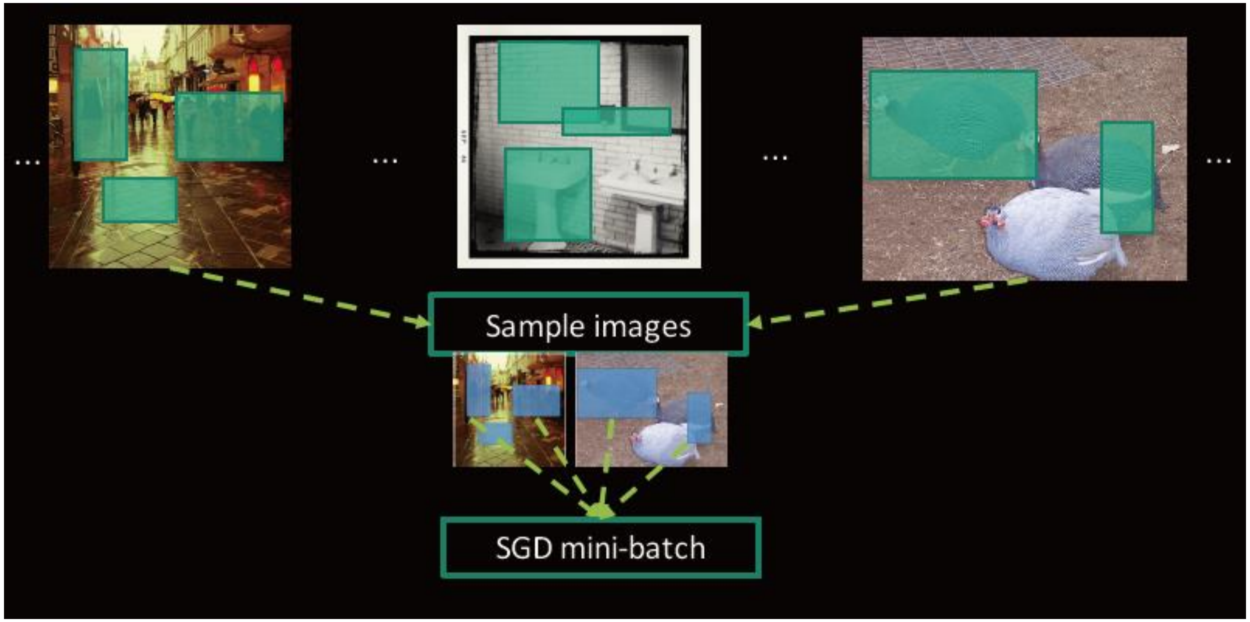
- Slow R-CNN and SPP-net use region-wise sampling to make mini-batches
  - Sample 128 example RoIs uniformly at random
  - Examples will come from different images with high probability



Slide credit: Ross Girschick

Fast R-CNN training

- Solution: use hierarchical sampling to build mini-batches

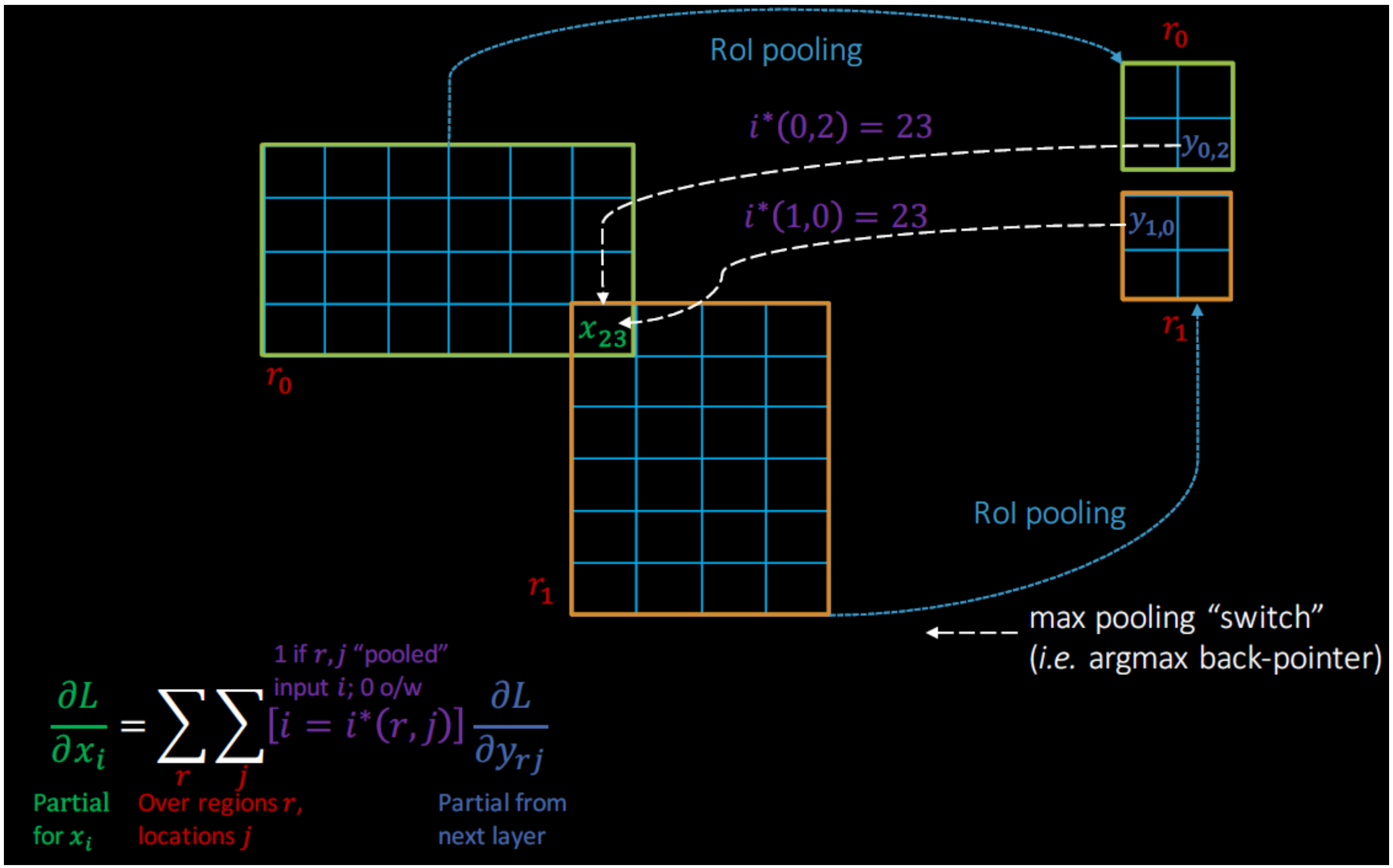


- Sample a small number of images (2)
- Sample many examples from each image (64)

Slide credit: Ross Girschick

Fast R-CNN training

- Differentiable ROI pooling



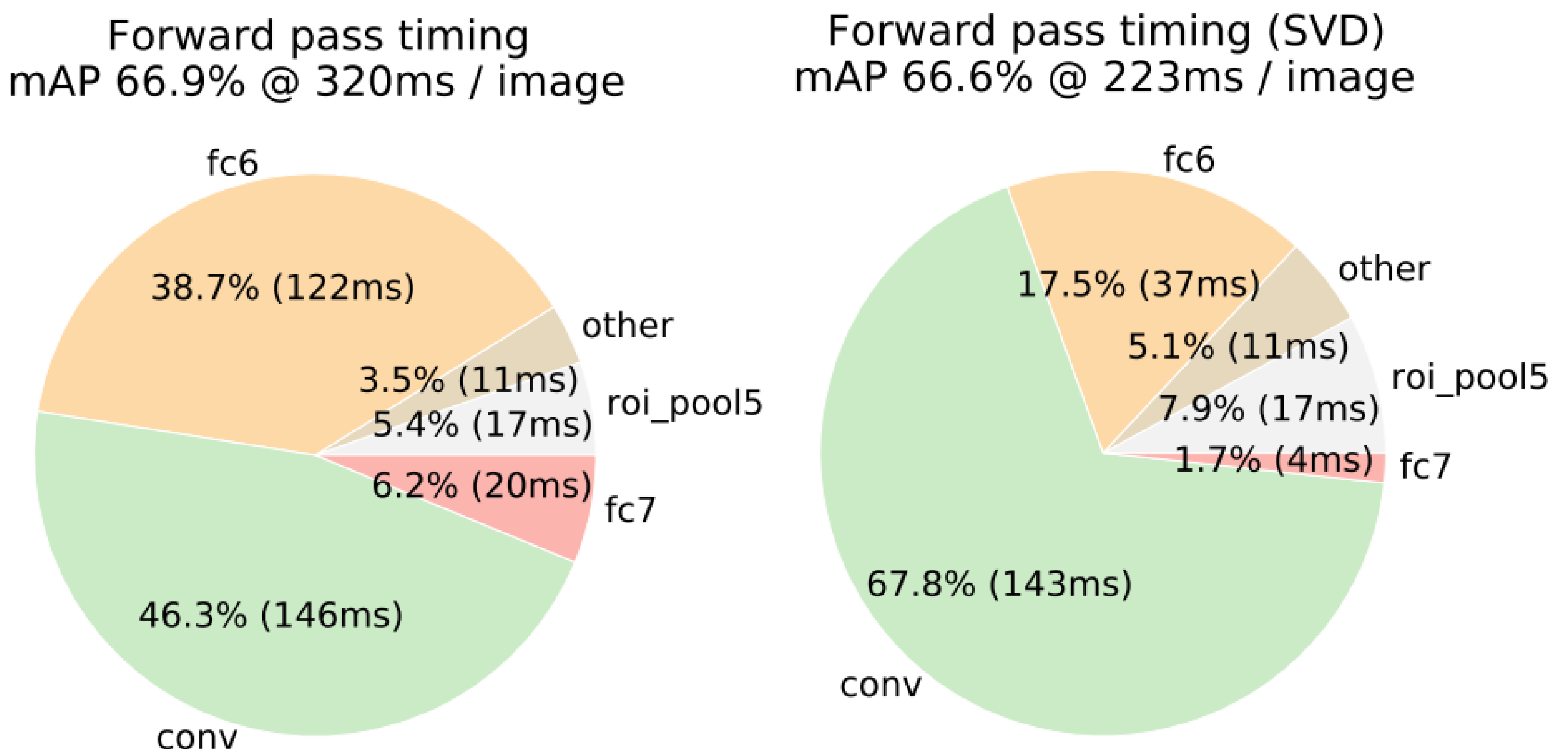
Main results

	Fast R-CNN	R-CNN	SPP-net
Train time (h)	**9.5**	84	25
Speedup	**8.8x**	1x	3.4x
Test time/image	**0.32s**	47.0s	2.3s
Test speedup	**146x**	1x	20x
mAP	**66.9**	66.0	63.1

Timings exclude object proposal time, which is equal for all methods. All methods use VGG16 from Simonyan and Zisserman.

Slide credit: Ross Girschick

Further test-time speedups



Slide credit: Ross Girschick

Fast R-CNN

- Pros
  - End-to-end training of deep ConvNets for detection
  - Fast training times
- Cons
  - Out-of-network region proposals
    - Selective search: 2s/image

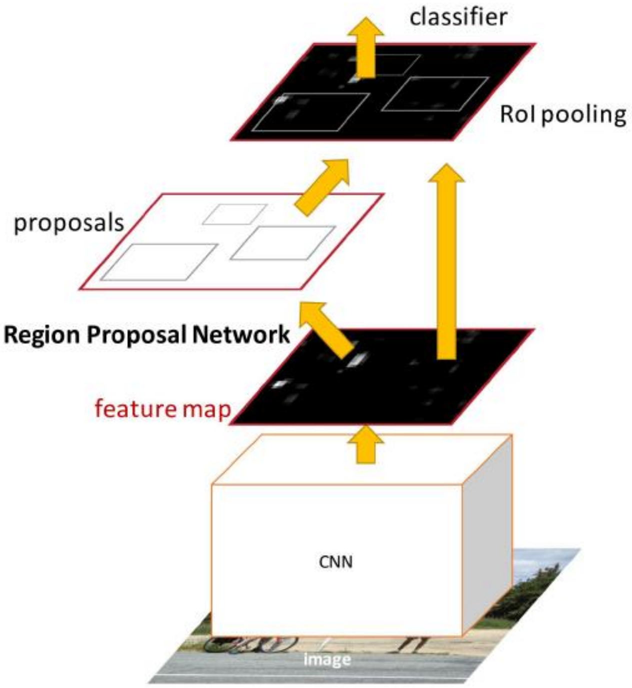
- Solution
  - Test-time speeds don't include region proposals
  - Just make the CNN do region proposals too!

Slide credit: Ross Girschick

FASTER R-CNN

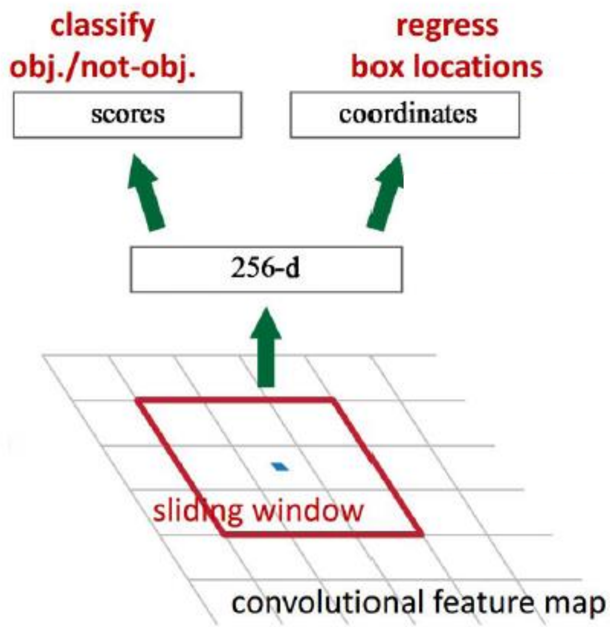
Faster RCNN

- Insert a **Region Proposal Network (RPN)** after the last convolutional layer
- RPN trained to produce region proposals directly; no need for external region proposals!
- After RPN, use RoI Pooling and an upstream classifier and bbox regressor just like Fast R-CNN



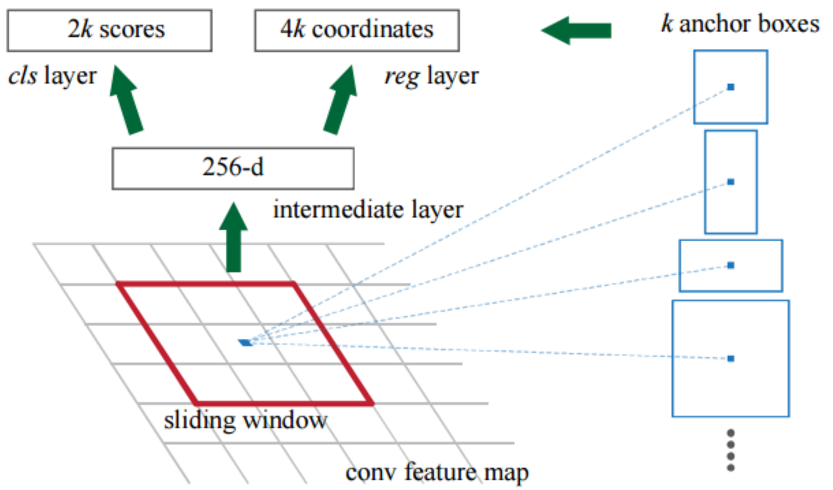
Faster R-CNN: RPN

- 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



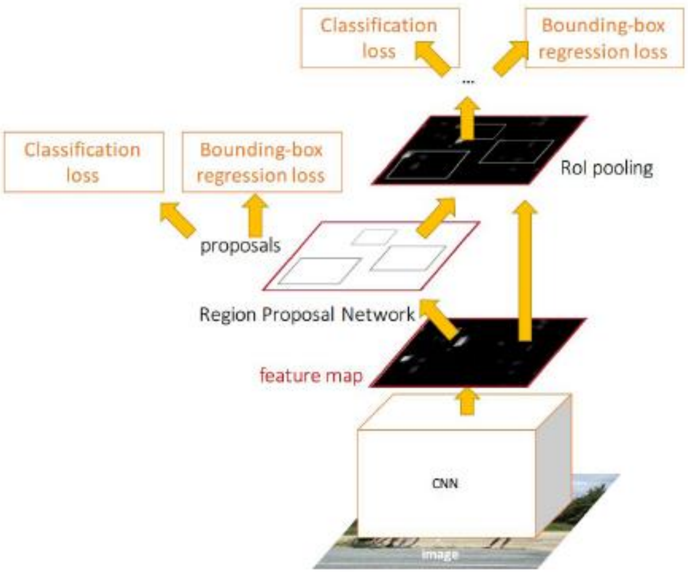
Faster R-CNN

- Use k **(=9) anchor boxes** at each location
- Anchors are translation invariant: use the same ones at every location
- Regression gives offsets from anchor boxes
- Classification gives the probability that each (regressed) anchor shows an object



Faster R-CNN: training

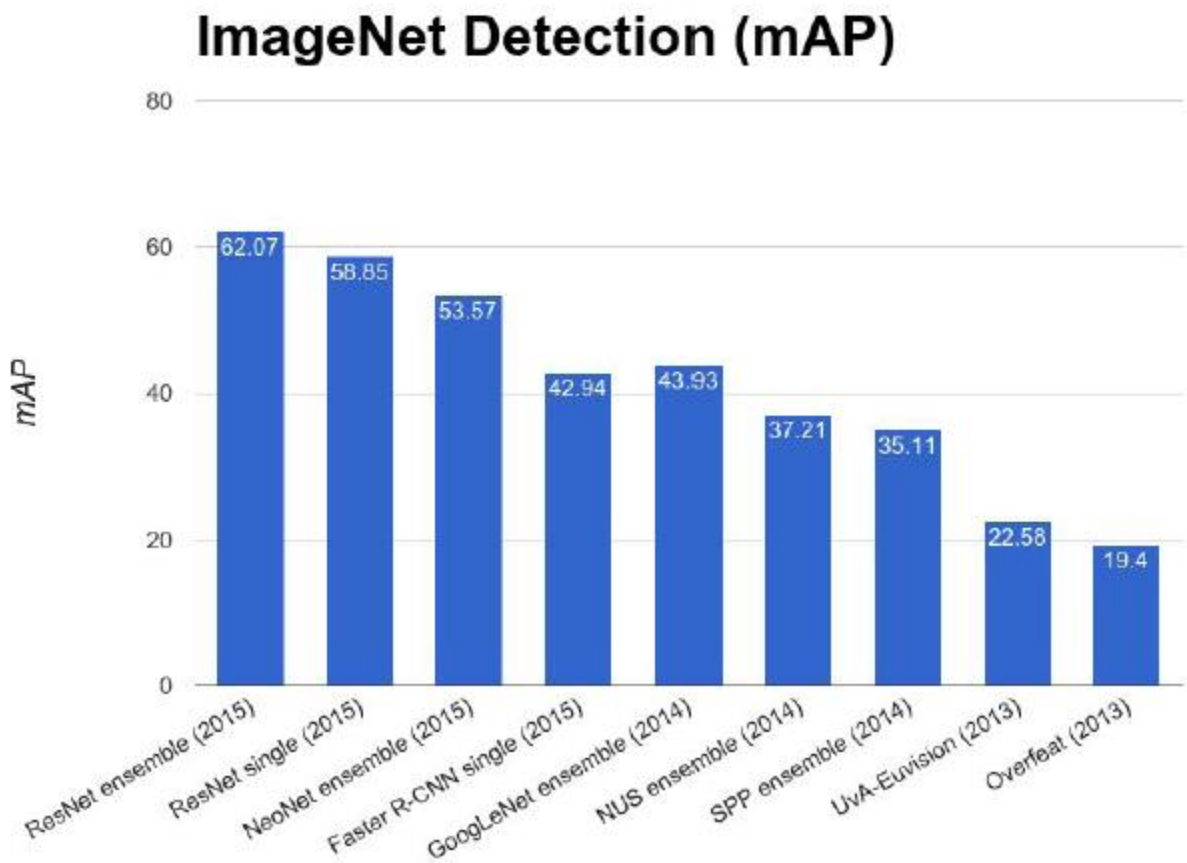
- Four loss functions
  - RPN classification (anchor good / bad)
  - RPN regression (anchor -> proposal)
  - Fast R-CNN classification (over classes)
  - Fast R-CNN regression (proposal -> box)



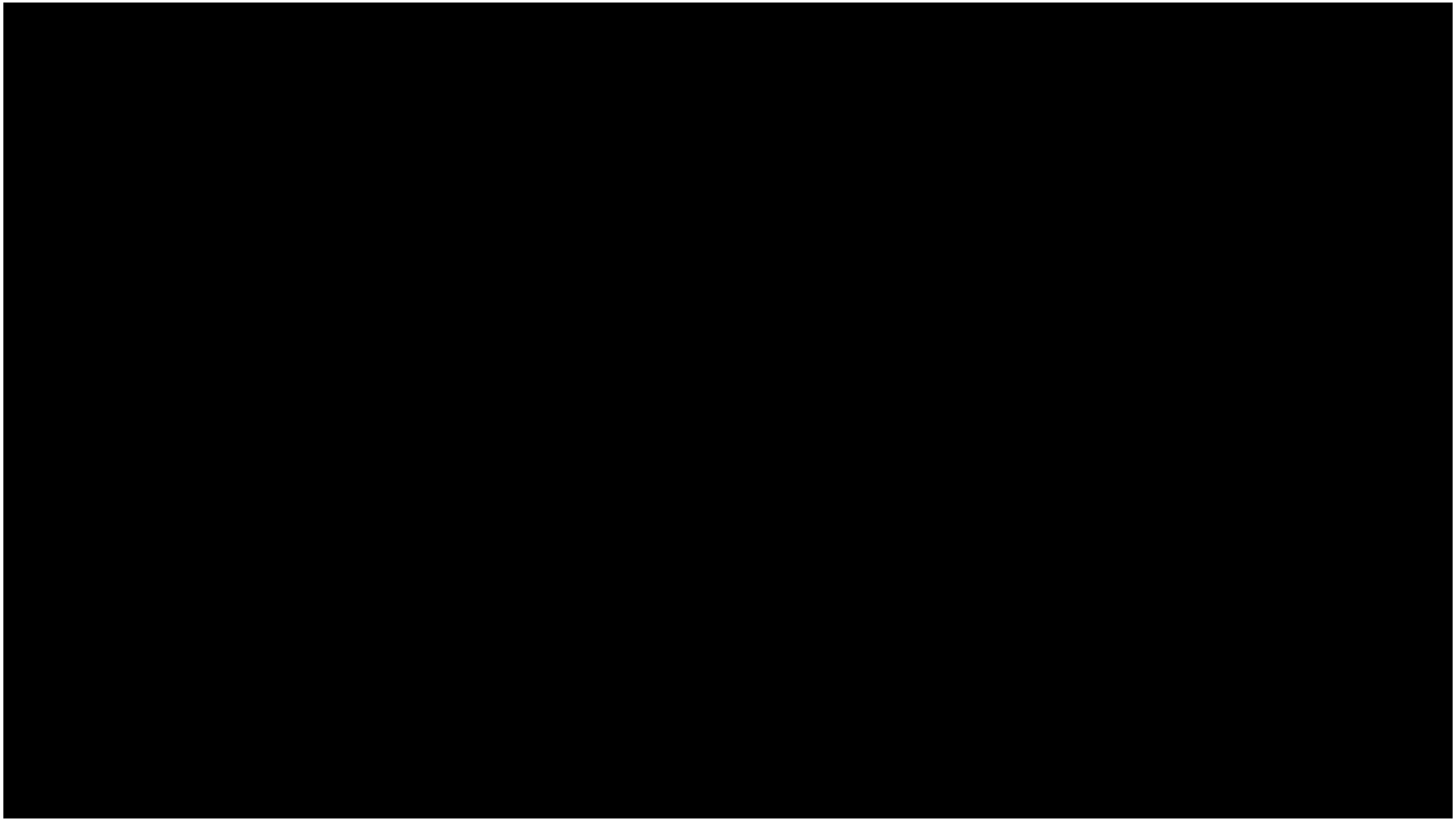
Results

	R-CNN	Fast R-CNN	Faster R-CNN
Test time per image (with proposals)	50 seconds	2 seconds	**0.2 seconds**
Speedup	1x	25x	**250x**
mAP (VOC 2007)	66.0	**66.9**	**66.9**

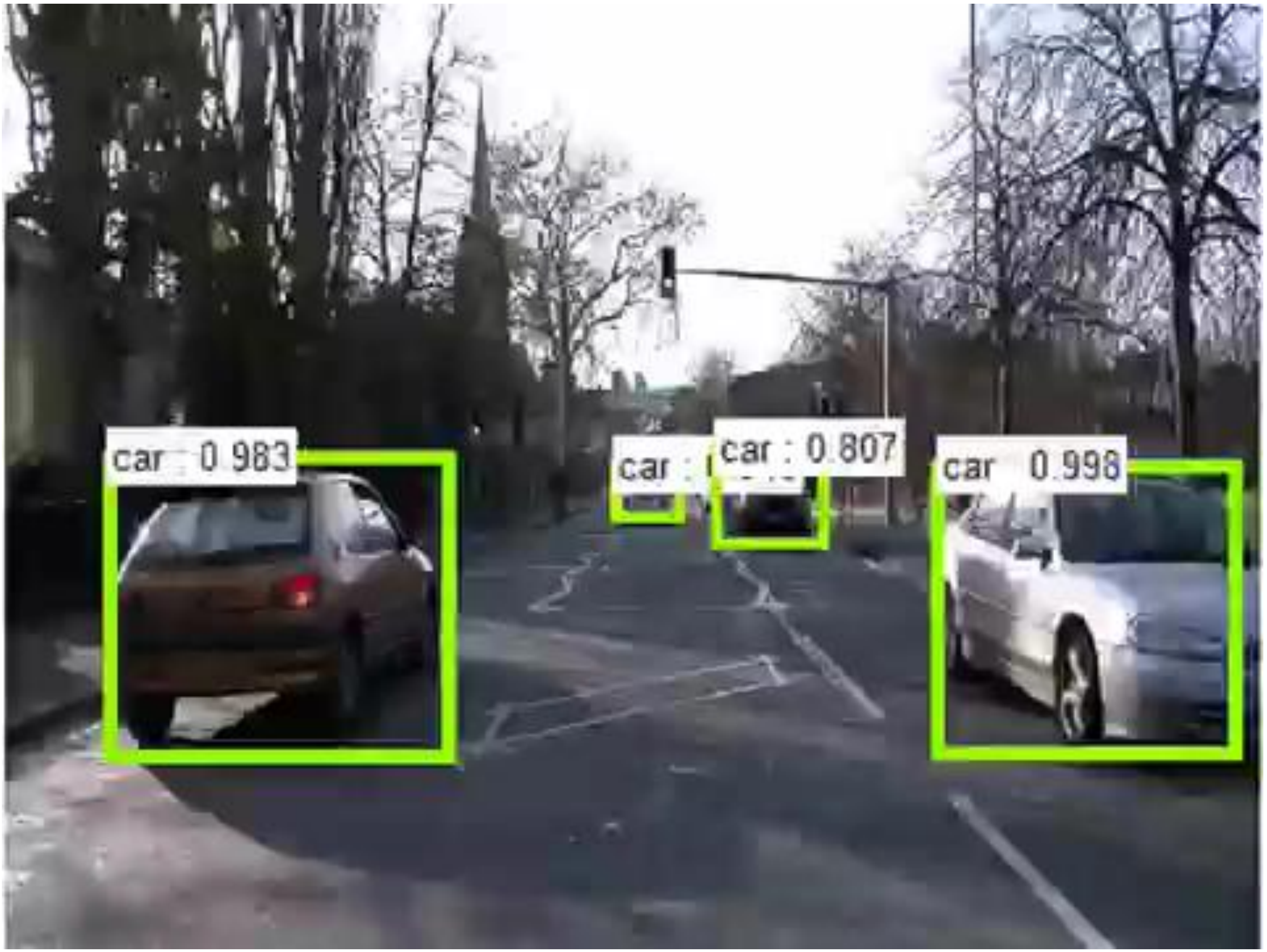
ImageNet Detection 2013 – 2015



Results



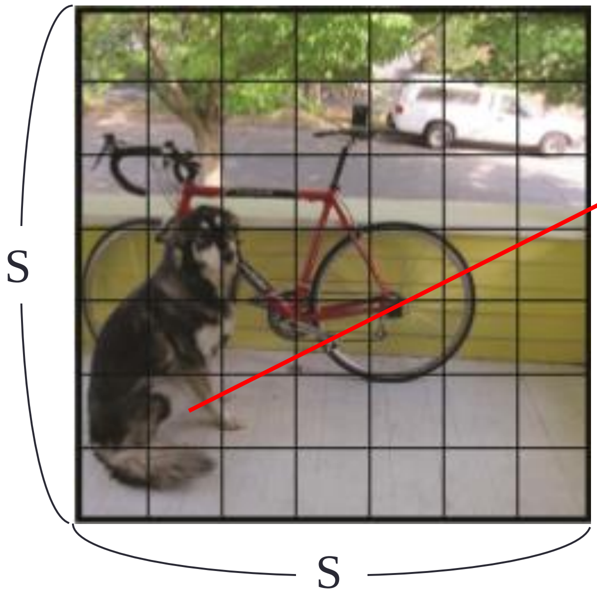
Object detection in the wild by Faster R-CNN + ResNet



YOLO:
YOU ONLY LOOK ONCE

YOLO algorithm

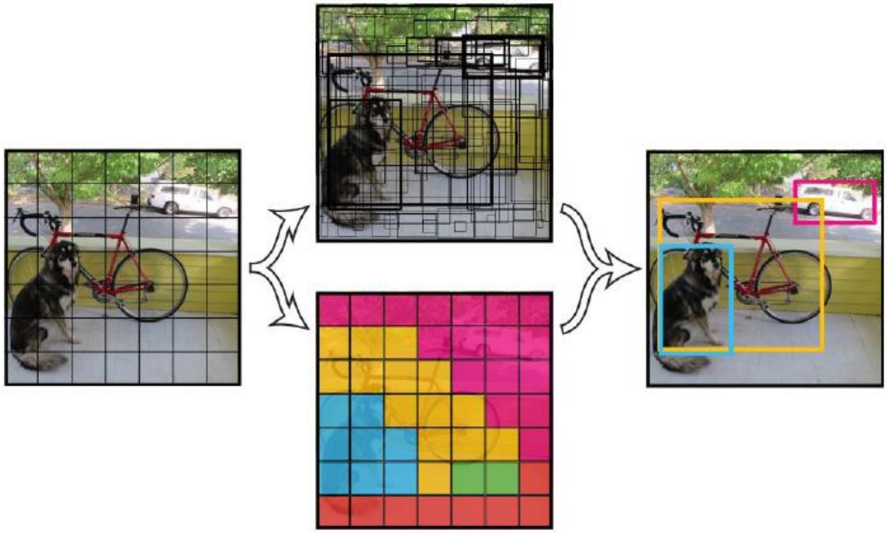
- Input & Output
  - Input : 448×448×3 resized image
  - Output : 7×7×30 tensor (S×S× (B×P + C))



\# Bounding box (B) : 2
\# Predictions of bounding box (P) : 5
(x, y, h, w, confidence)
\# Class probabilities (C) : 20

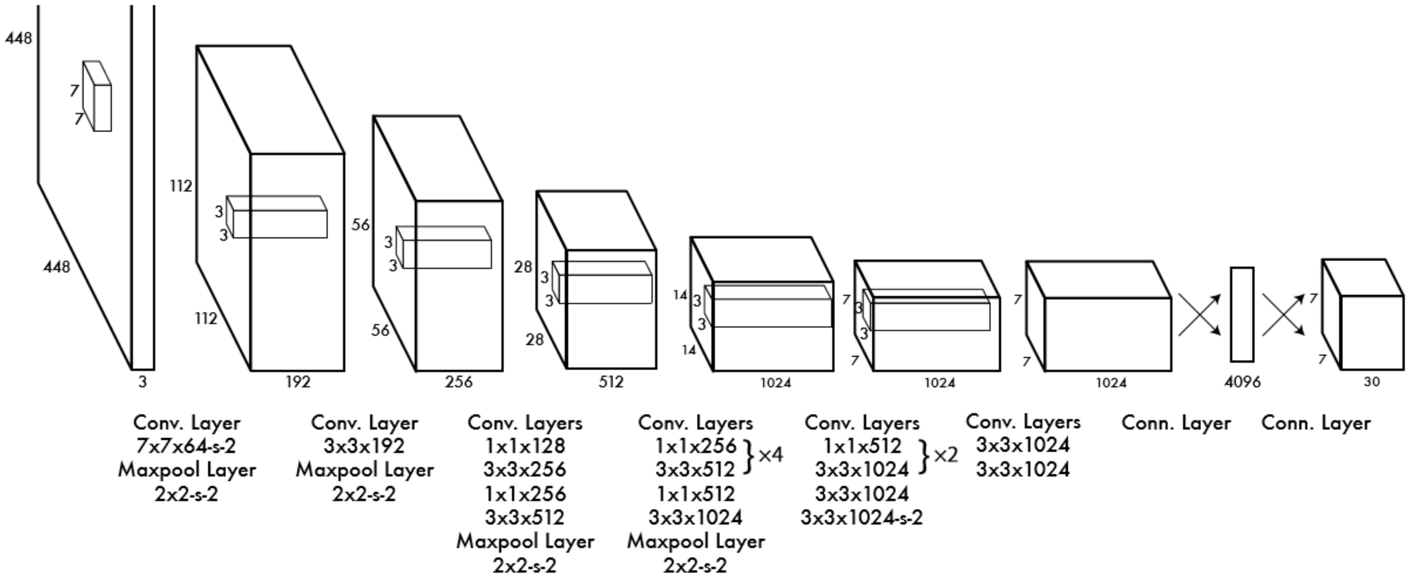
YOLO algorithm

- Divide image into S x S grid
- Within each grid cell predict:
  - B Boxes: 4 coordinates + confidence
  - Class scores: C numbers
- Regression from image to $7 \times 7 \times (5 \times B + C)$ tensor



YOLO algorithm

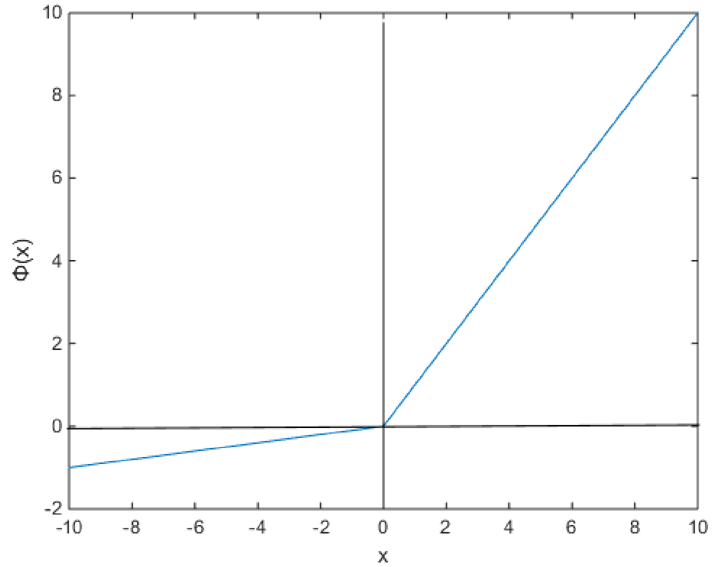
- Network architecture
  - Similar to GoogLeNet model
  - 1×1 reduction layers instead of Inception layer
  - Use leaky rectified linear activation function



YOLO algorithm

- Leaky rectified linear activation function

$$\phi(x) = \begin{cases} x, & \text{if } x > 0 \\ 0.1x, & \text{otherwise} \end{cases}$$



YOLO algorithm

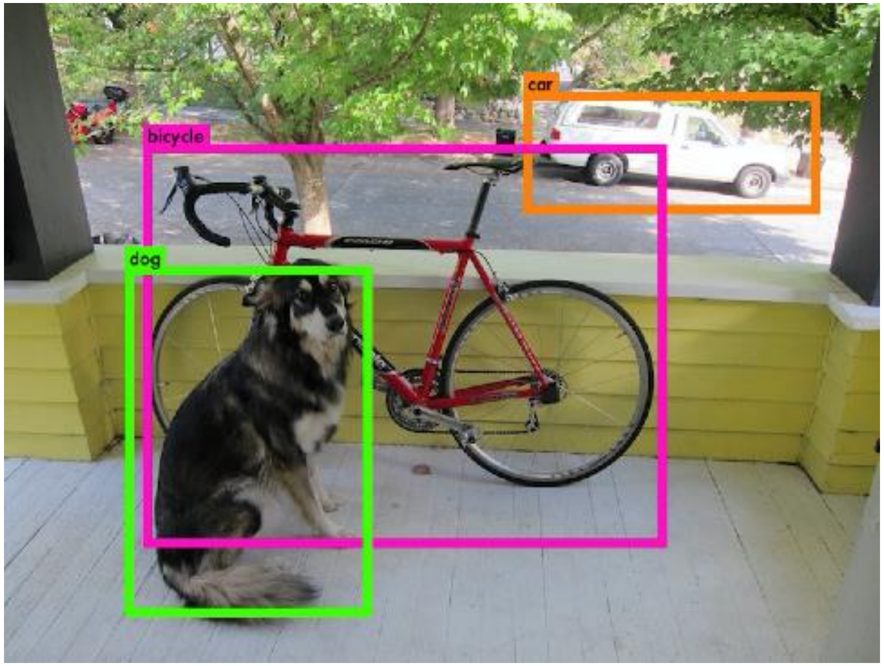
- Loss function

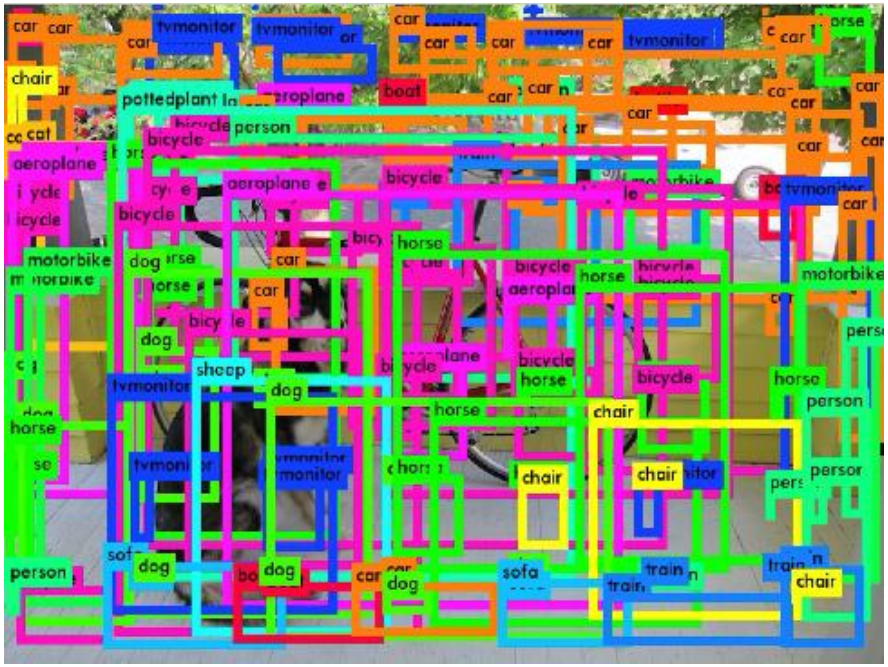
$$E(\theta) = \lambda_{coord} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbf{1}_{i,j}^{obj} [(x_i - \hat{x}_i)^2 + (y_i - \hat{y}_i)^2] + \lambda_{coord} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbf{1}_{i,j}^{obj} \left[\left(\sqrt{w_i} - \sqrt{\hat{w}_i}\right)^2 + \left(\sqrt{h_i} - \sqrt{\hat{h}_i}\right)^2 \right]$$

$$+ \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbf{1}_{i,j}^{obj} \left(C_i - \hat{C}_i\right)^2 + \lambda_{noobj} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbf{1}_{i,j}^{noobj} \left(C_i - \hat{C}_i\right)^2 + \sum_{i=0}^{S^2} \mathbf{1}_{i,j}^{obj} \sum_{c \in classes}^{B} (p_i(c) - \hat{p}_i(c))^2$$

YOLO algorithm

- Thresholding





th = 0.2

th = 0

YOLO: You Only Look Once

- Faster than Faster R-CNN, but not as good

Real-Time Detectors	Train	mAP	FPS
100Hz DPM [30]	2007	16.0	100
30Hz DPM [30]	2007	26.1	30
Fast YOLO	2007+2012	52.7	**155**
YOLO	2007+2012	**63.4**	45
Less Than Real-Time			
Fastest DPM [37]	2007	30.4	15
R-CNN Minus R [20]	2007	53.5	6
Fast R-CNN [14]	2007+2012	70.0	0.5
Faster R-CNN VGG-16[27]	2007+2012	73.2	7
Faster R-CNN ZF [27]	2007+2012	62.1	18

Demo Videos

SUMMARY

Object Detection Summary

- Find a variable number of objects by classifying image regions
- Before CNNs:
  - dense multiscale sliding window (HoG, DPM)
- R-CNN:
  - Selective Search + CNN classification / regression
- Fast R-CNN:
  - Swap order of convolutions and region extraction
- Faster R-CNN:
  - Compute region proposals within the network

Code links

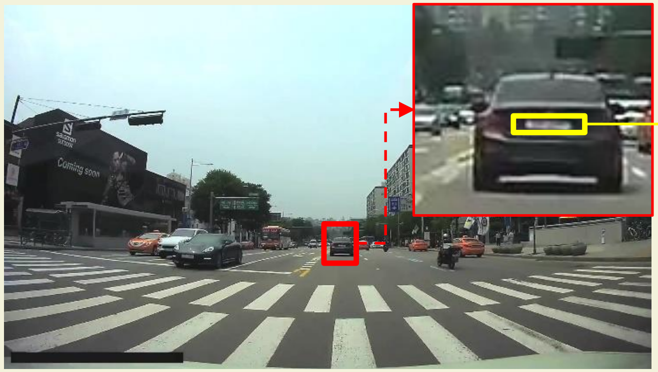
- R-CNN
  - Caffe + Matlab (https://github.com/rbgirshick/rcnn)
- Faster R-CNN
  - Caffe + Matlab (https://github.com/rbgirshick/fast-rcnn)
- Faster R-CNN
  - Caffe + Matlab (https://github.com/ShaoqingRen/faster_rcnn)
  - Caffe + Python (https://github.com/rbgirshick/py-faster-rcnn)
- YOLO
  - http://pjreddie.com/darknet/yolo/

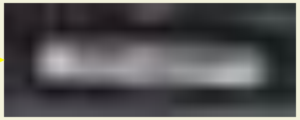
BACKUPS

CAR LICENSE PLATE DETECTION

1. 개요

목표 : 특수목적 차량(경찰차) 의 블랙박스 영상에서 번호판 검출





→ License Plate?

해상도 문제로 인해 영상에서 바로 번호판을 검출하는 것은 어려움

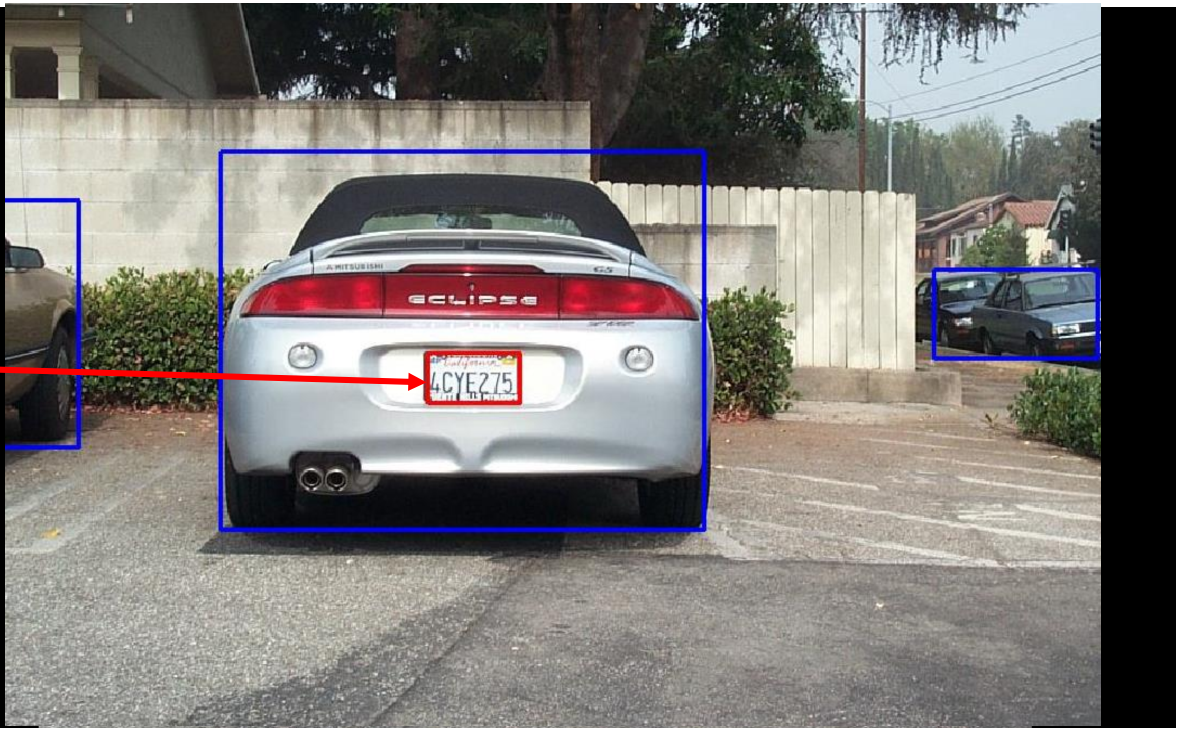
번호판 검출을 위한 방법

차량 검출 → (차량 영역) → 자동 줌 → (고해상도 영상) → 번호판 검출

2. 제안하는 방법

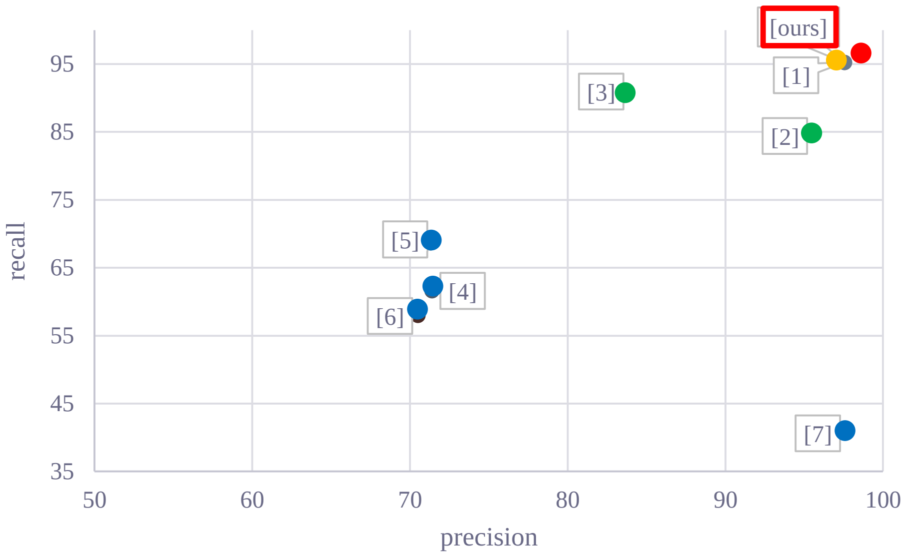
License Plate Detection
(CNN)

- 각 Region Proposal중 Best region 분류
- 높은 성능으로 검출



3. 성능 비교

Correct detection : Intersection over Union (IoU) $\geq$ 0.5



	Precision (%)	Recall (%)
[ours]	**98.39**	**96.83**
[1]	97.56	95.24
[2]	95.5	84.8
[3]	83.73	90.48
[4]	71.4	61.6
[5]	71.3	68.7
[6]	70.5	58
[7]	97.6	40.67

*Dataset : Caltech car_markus (http://www.vision.caltech.edu/Image_Datasets/cars_markus/cars_markus.tar)

[1] : character-based + CNN

[2], [3] : character-based

[4], [5], [6], [7] : edge-based

4. 결과


UCLA
BUICK
2LOX542
1LWY942
4CYE275
2PGT989
CALIFORNIA
EZEZ BEH
CHEVROLET
1SPP999

THE IMAGENET CHALLENGE

Backpack

Flute

Strawberry

Traffic light

Backpack

Matchstick

Bathing cap

Sea lion

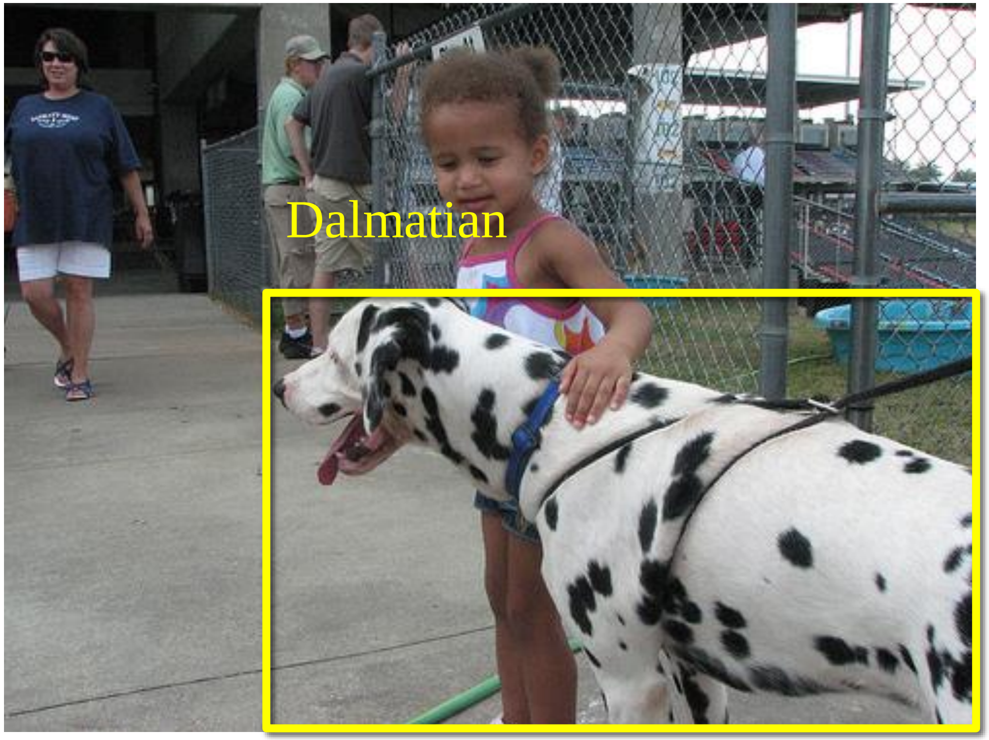
Racket

Large-scale recognition

Large-scale recognition

Large Scale Visual Recognition Challenge (ILSVRC) 2010-2012

1000 object classes **1,431,167 images**



http://image-net.org/challenges/LSVRC/{2010,2011,2012}

ILSVRC Task 1: Classification

Steel drum

ILSVRC Task 1: Classification

Steel drum

Output:
Scale
T-shirt
<u>Steel drum</u>
Drumstick
Mud turtle

✔

Output:
Scale
T-shirt
Giant panda
Drumstick
Mud turtle

✗

ILSVRC Task 1: Classification

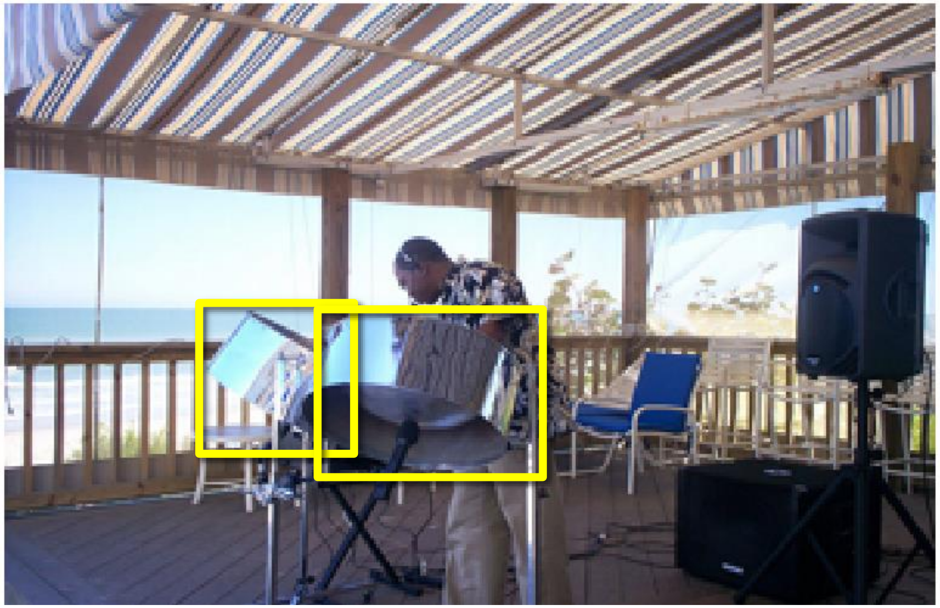
Steel drum

Output:		Output:	
Scale T-shirt Steel drum Drumstick Mud turtle	✔	Scale T-shirt Giant panda Drumstick Mud turtle	✗

$$\text{Accuracy} = \frac{1}{N} \sum_{\substack{N \\ \text{images}}} 1[\text{correct on image i}]$$

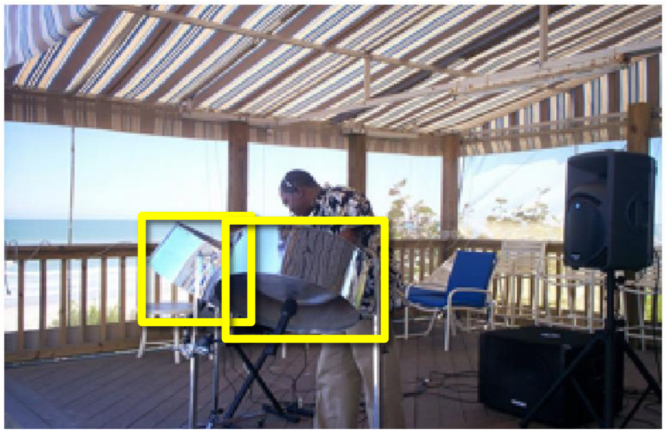
ILSVRC Task 2: Classification + Localization

Steel drum

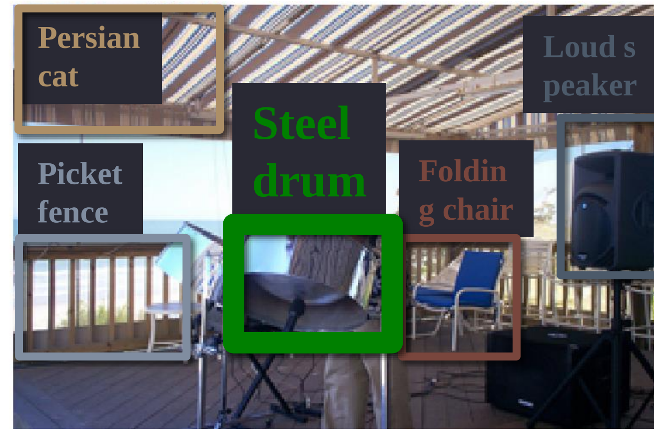


ILSVRC Task 2: Classification + Localization

Steel drum

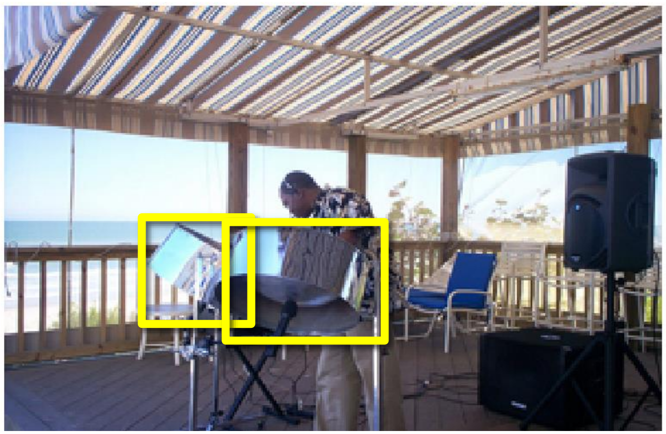


Output



ILSVRC Task 2: Classification + Localization

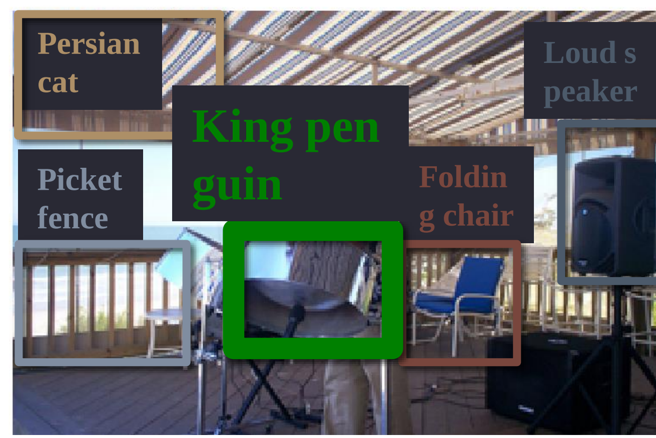
Steel drum



Output

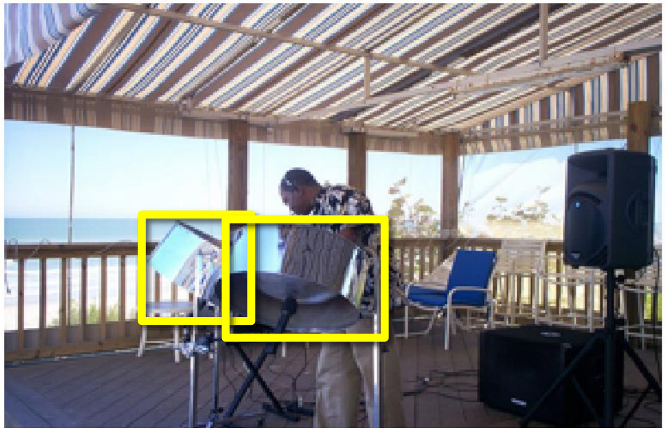
Output (bad localization)

Output (bad classification)



ILSVRC Task 2: Classification + Localization

Steel drum



Output

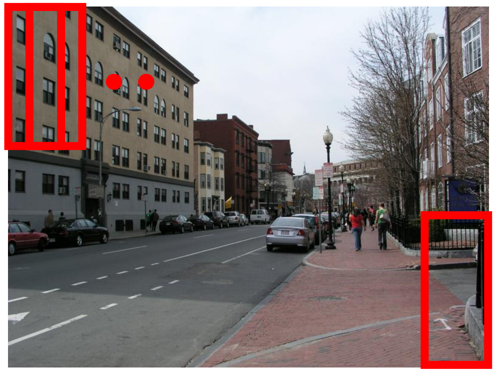
$$\text{Accuracy} = \frac{1}{N} \sum_{\text{N-images}} 1[\text{correct on image i}]$$

Classification + Localization

Team name	Filename	Error (5 guesses)	Description
SuperVision	test-rect-preds-144-cloc-141-146.2009-131-137-145-	0.335463	Using extra training data for classification from ImageNet Fall 2011 release
SuperVision	test-rect-preds-144-cloc-131-137-145-135-145f.txt	0.341905	Using only supplied training data
OXFORD_VGG	test_adhocmix_detection.txt	0.500342	Re-ranked DPM detection over Mixed selection from High-Level SVM scores and Baseline Scores, decision is performed by looking at the validation performance
OXFORD_VGG	test_finecls_detection_bestbbox.txt	0.50139	Re-ranked DPM detection over High-Level SVM Scores
OXFORD_VGG	test_finecls_detection_firstbbox.txt	0.522189	Re-ranked DPM detection over High-Level SVM Scores - First bbox selection heuristic

SLIDING WINDOW SCHEME

Localization prob. → classification prob.



Each window is separately classified