

Automatic Object Detection

Jianping Fan
Dept of Computer Science
UNC-Charlotte

Course Website:

<http://webpages.uncc.edu/jfan/itcs5152.html>

Advanced Approaches for Object Detection

- Object Detection
- The RCNN Object Detector (2014)
- The Fast RCNN Object Detector (2015)
- The Faster RCNN Object Detector (2016)

Computer Vision Tasks

Classification



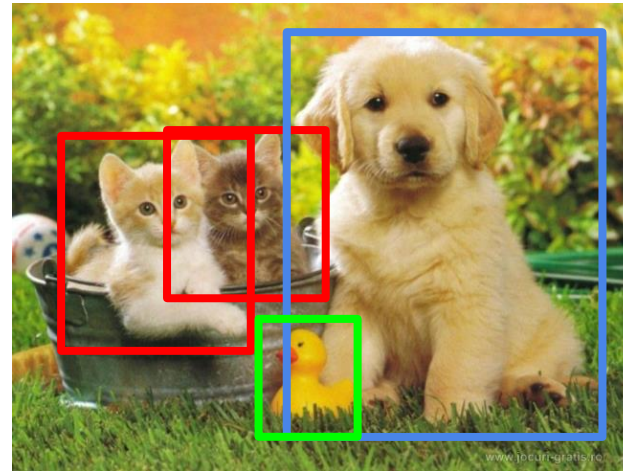
CAT

**Classification
+ Localization**



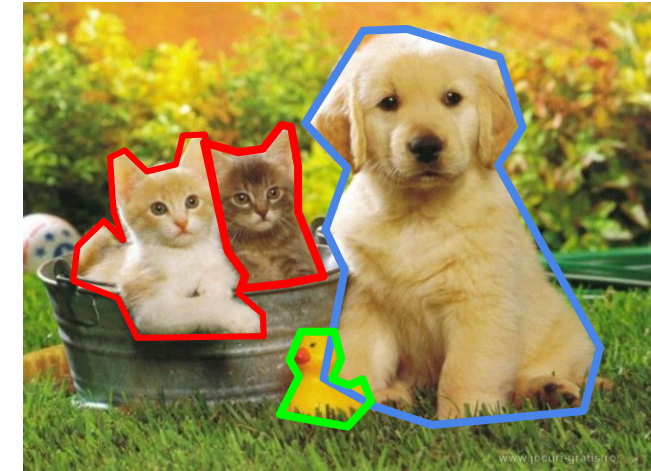
CAT

Object Detection



CAT, DOG, DUCK

**Instance
Segmentation**



CAT, DOG, DUCK

Single object

Multiple objects

Computer Vision Tasks

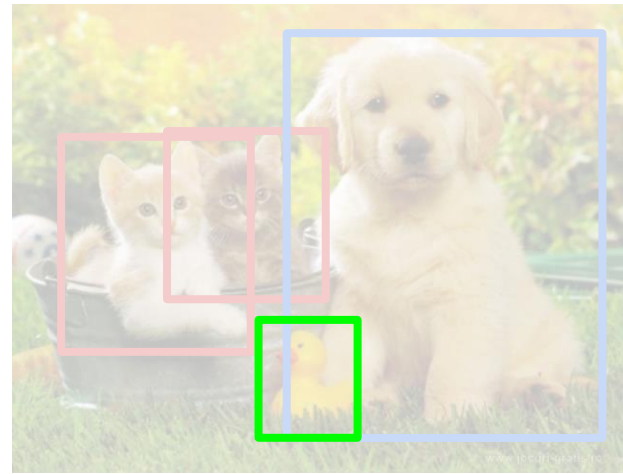
Classification



**Classification
+ Localization**



Object Detection



Instance
Segmentation



Classification + Localization: Task

Classification: C classes

Input: Image

Output: Class label

Evaluation metric: Accuracy



→ CAT

Localization:

Input: Image

Output: Box in the image (x, y, w, h)

Evaluation metric: Intersection over Union



→ (x, y, w, h)

Classification + Localization: Do both

Classification + Localization: ImageNet

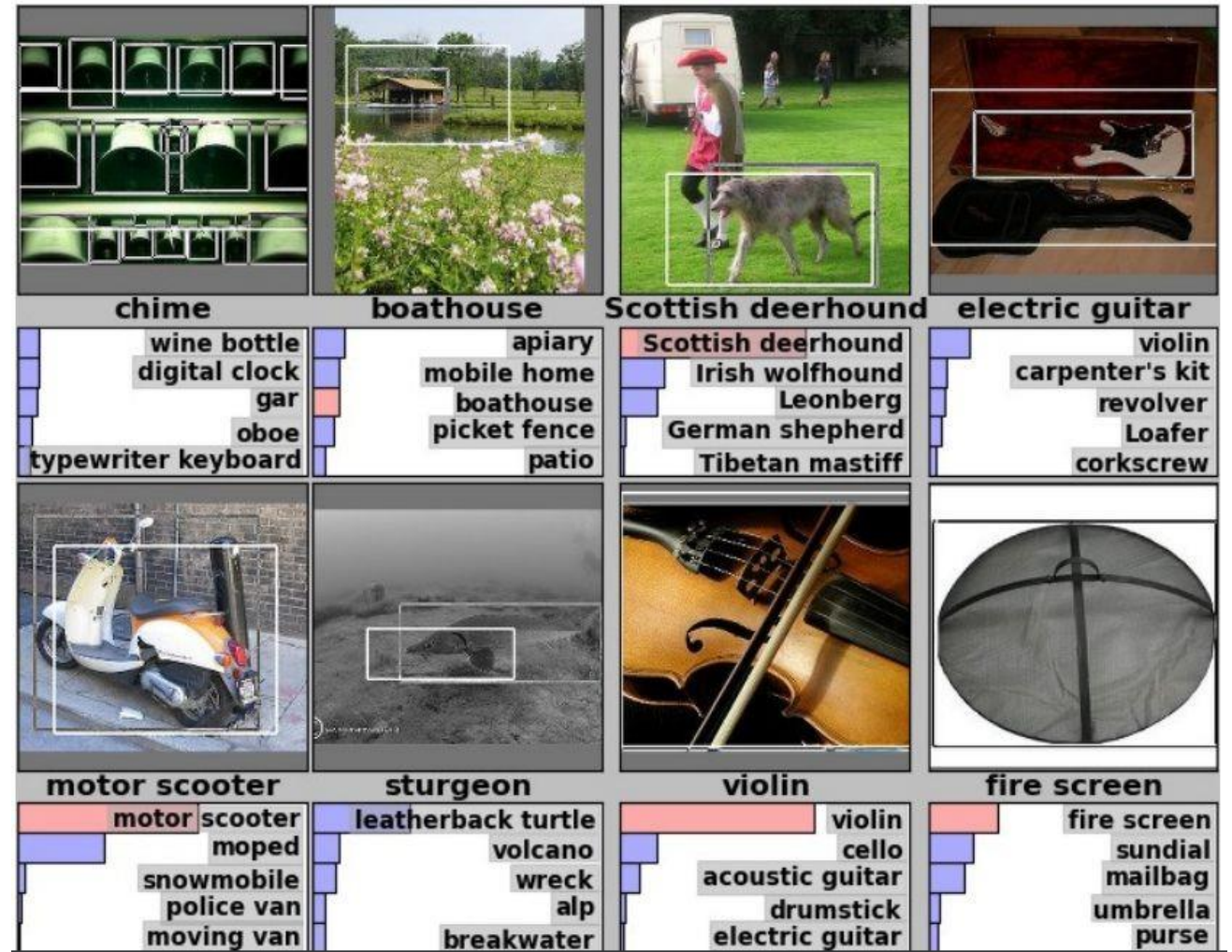
1000 classes (same as classification)

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 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)

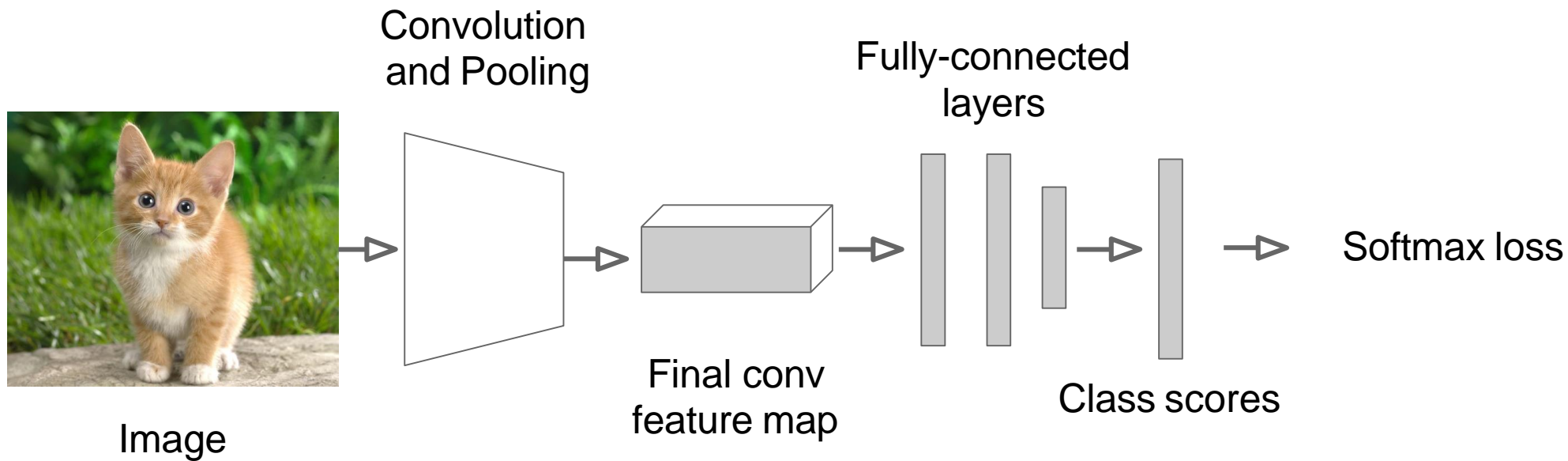
Correct output:
box coordinates
(4 numbers)

Loss:
L2 distance

Only one object,
simpler than detection

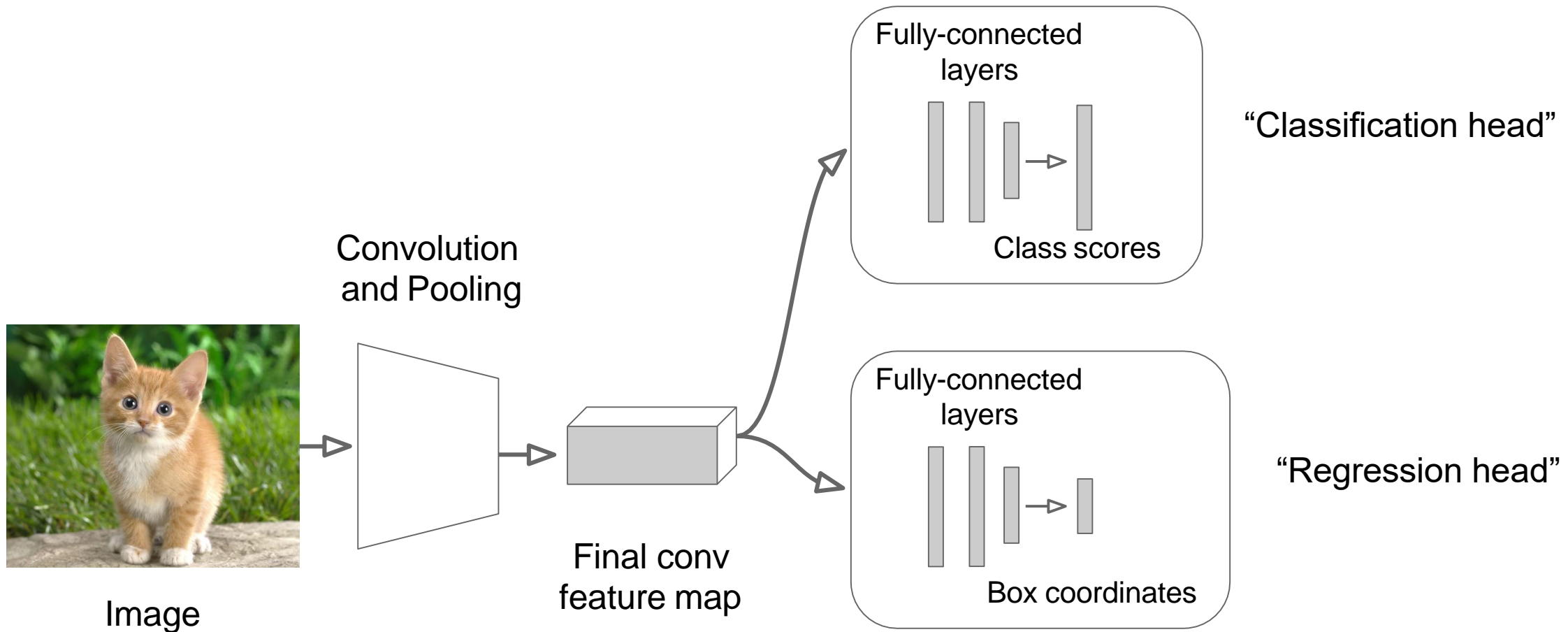
Simple Recipe for Classification + Localization

Step 1: Train (or download) a classification model (AlexNet, VGG, GoogLeNet)



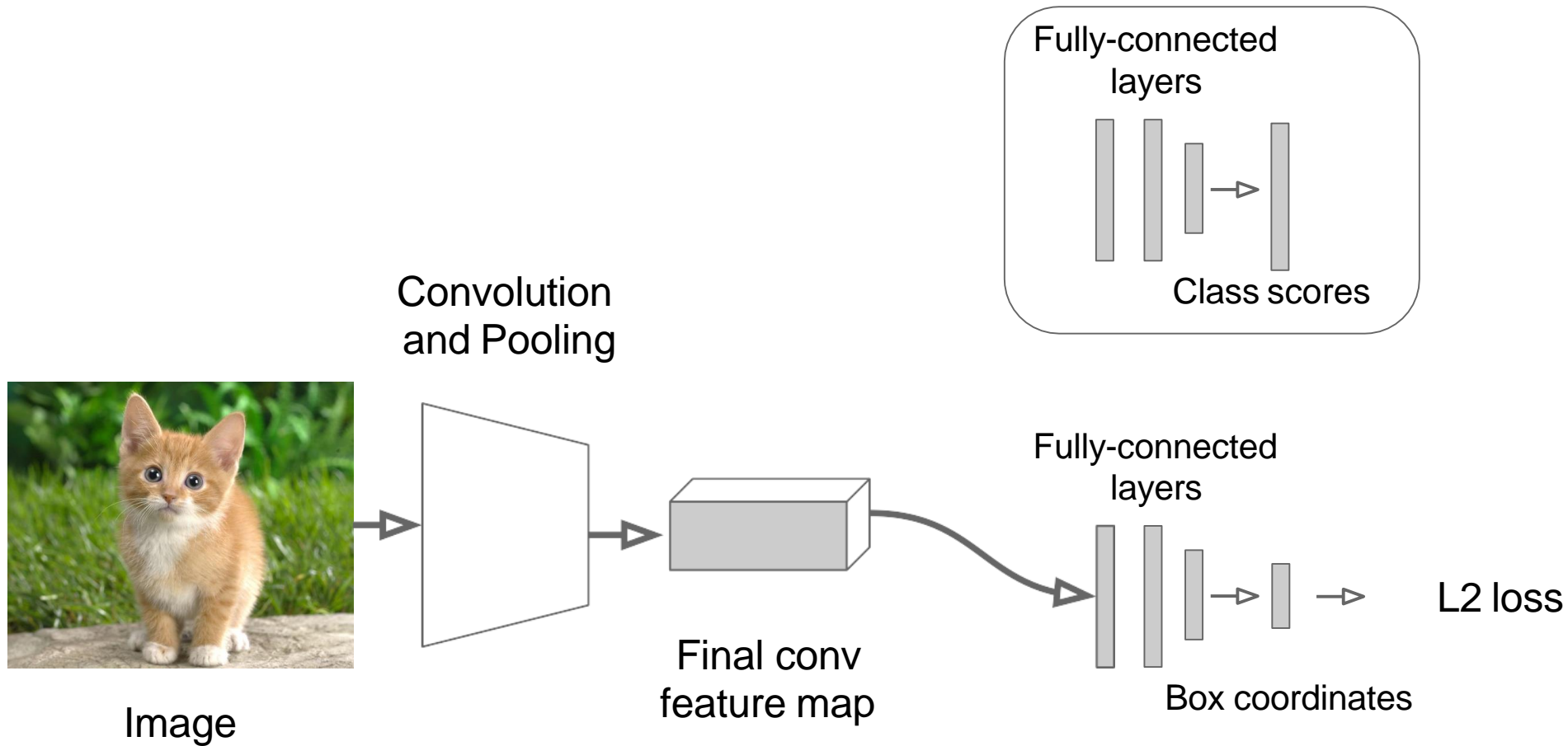
Simple Recipe for Classification + Localization

Step 2: Attach new fully-connected “regression head” to the network



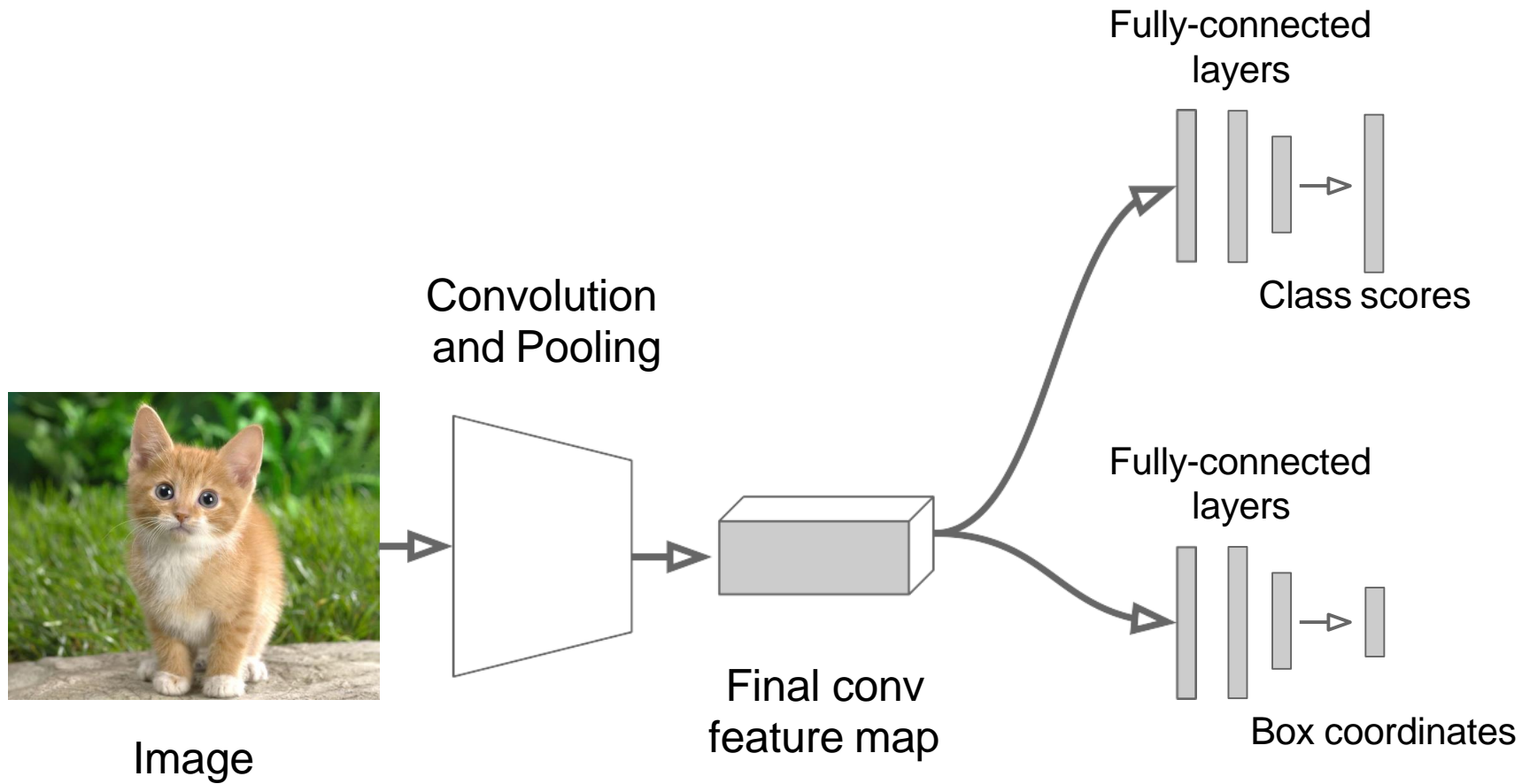
Simple Recipe for Classification + Localization

Step 3: Train the regression head only with SGD and L2 loss



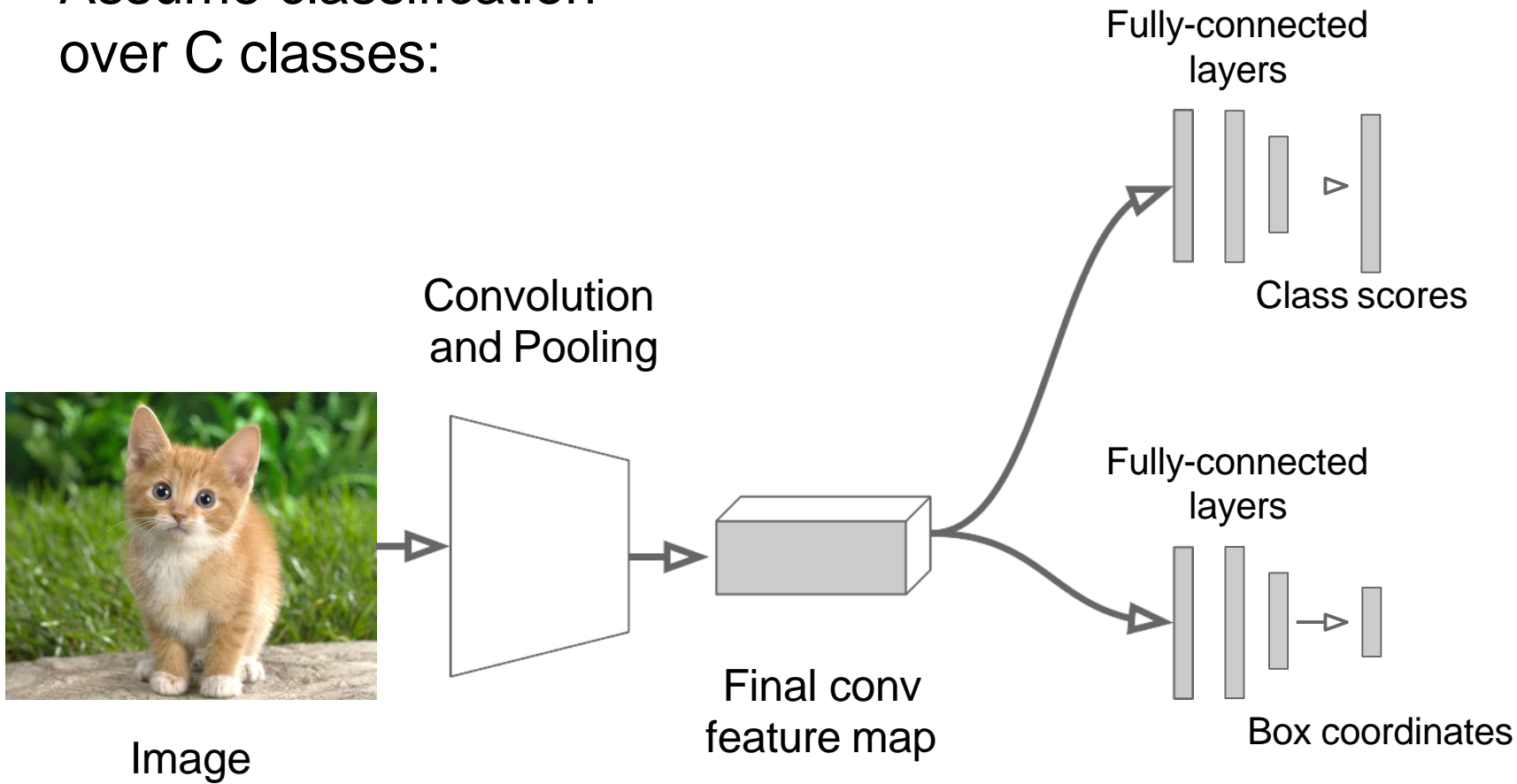
Simple Recipe for Classification + Localization

Step 4: At test time use both heads



Per-class vs class agnostic regression

Assume classification over C classes:



Classification head:
 C numbers
(one per class)

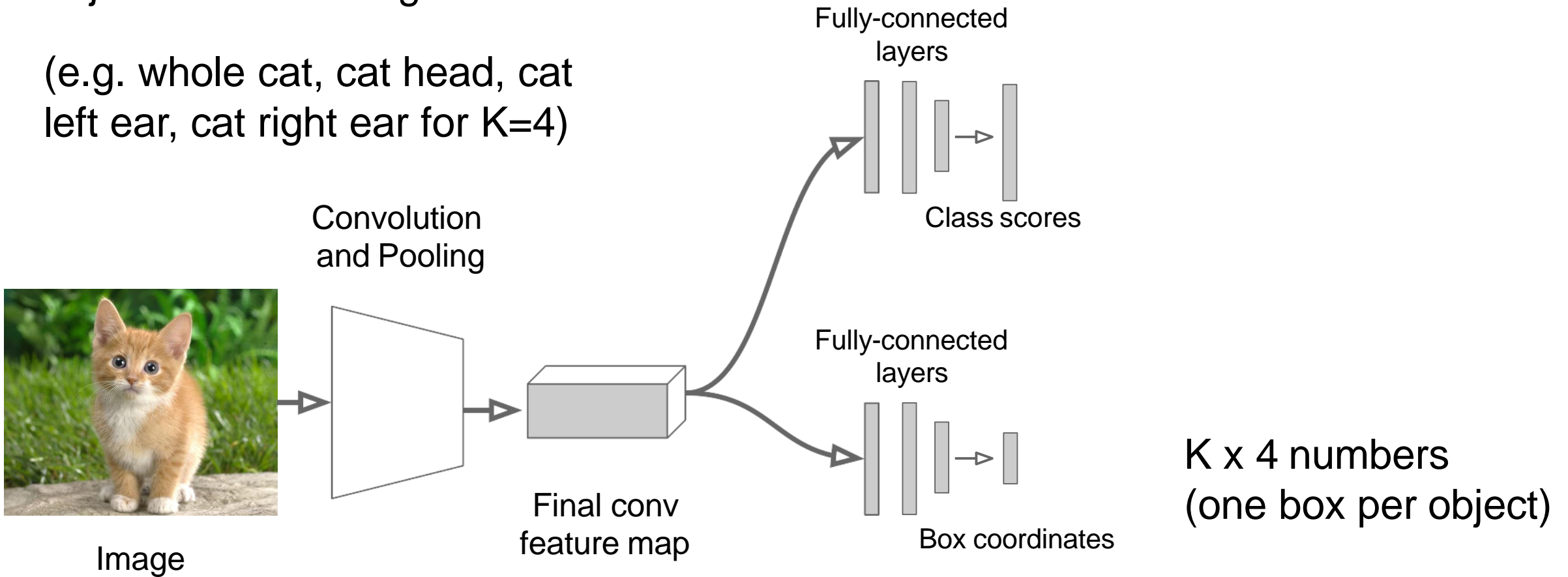
Class agnostic:
4 numbers
(one box)

Class specific:
 $C \times 4$ numbers
(one box per class)

Aside: Localizing multiple objects

Want to localize **exactly** K objects in each image

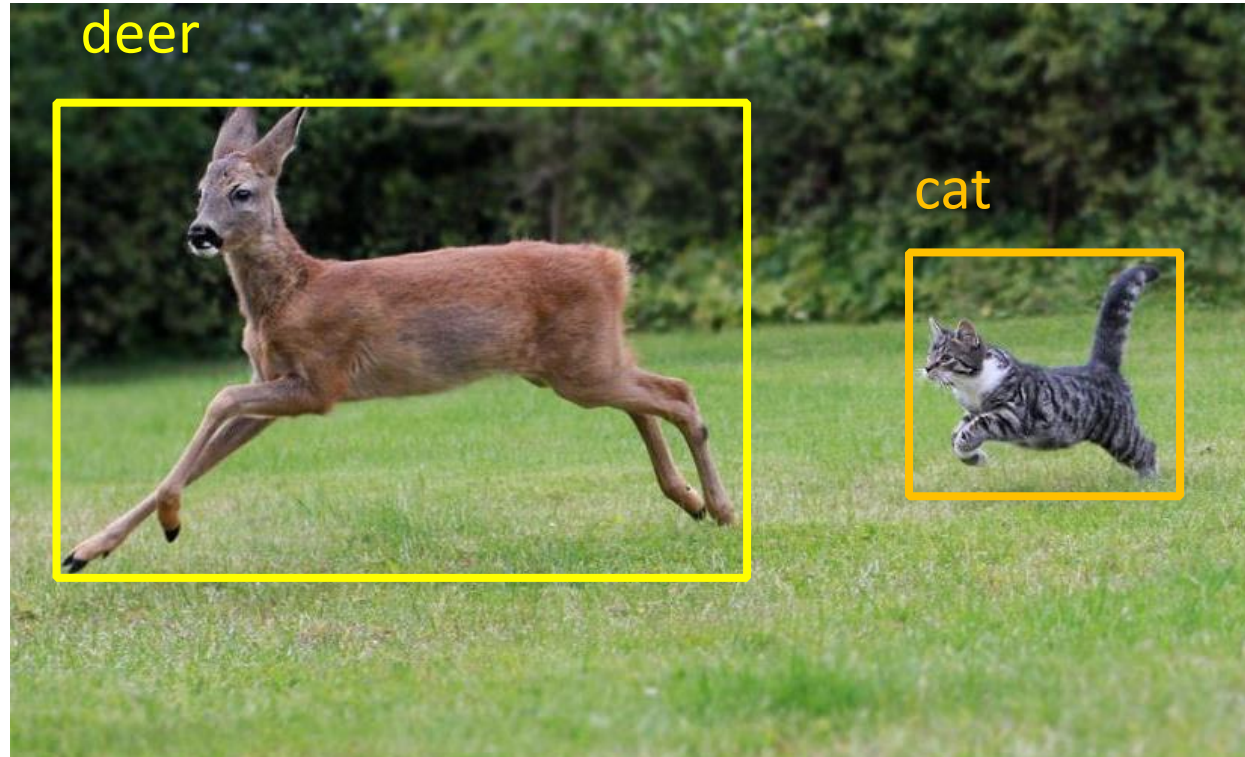
(e.g. whole cat, cat head, cat left ear, cat right ear for $K=4$)



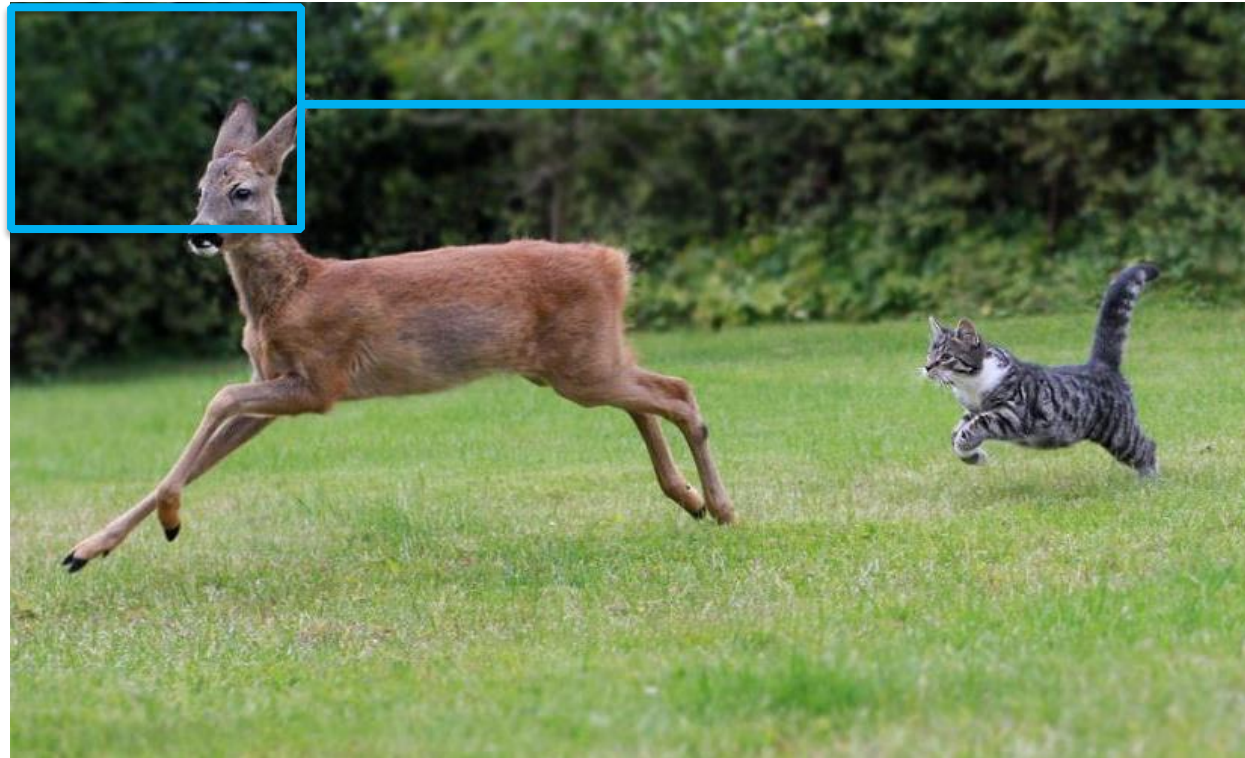
Idea #2: Sliding Window

- Run classification + regression network at multiple locations on a high-resolution image
- Convert fully-connected layers into convolutional layers for efficient computation
- Combine classifier and regressor predictions across all scales for final prediction

Object Detection: *sliding windows following by classification*

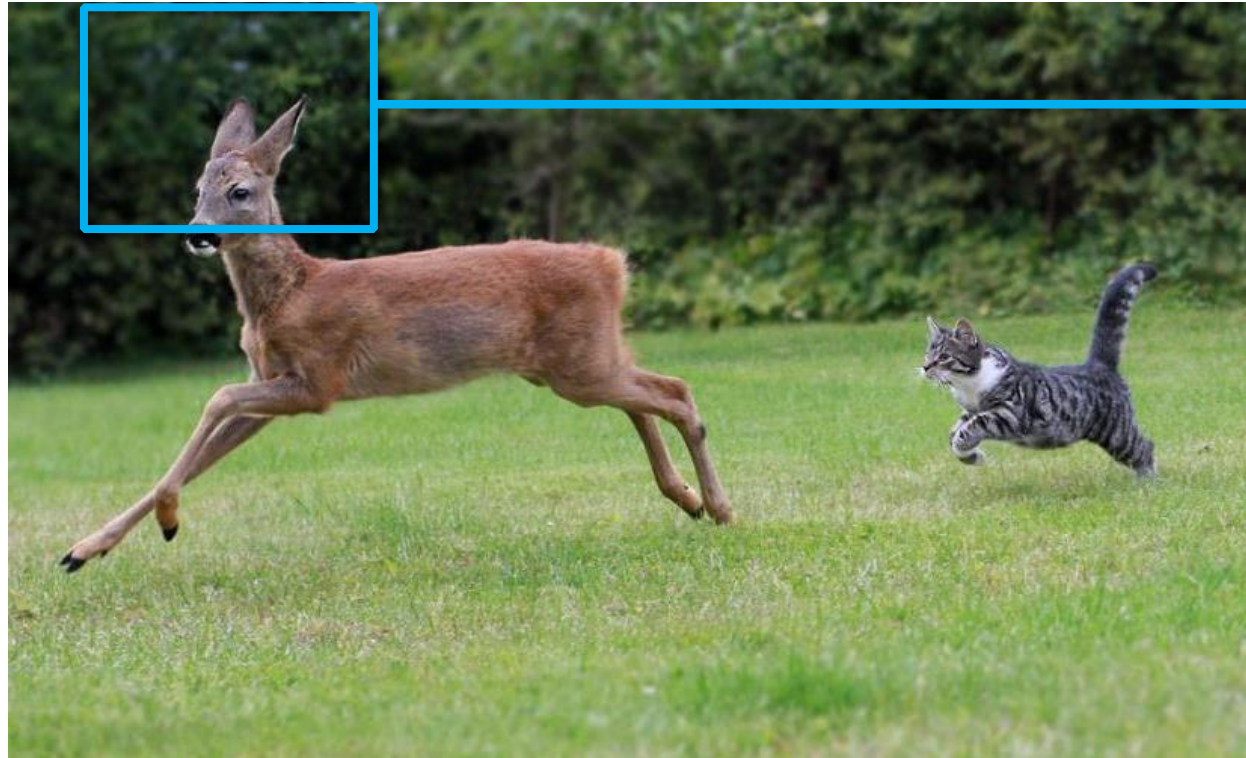


Object Detection as Classification of sliding windows (proposals)



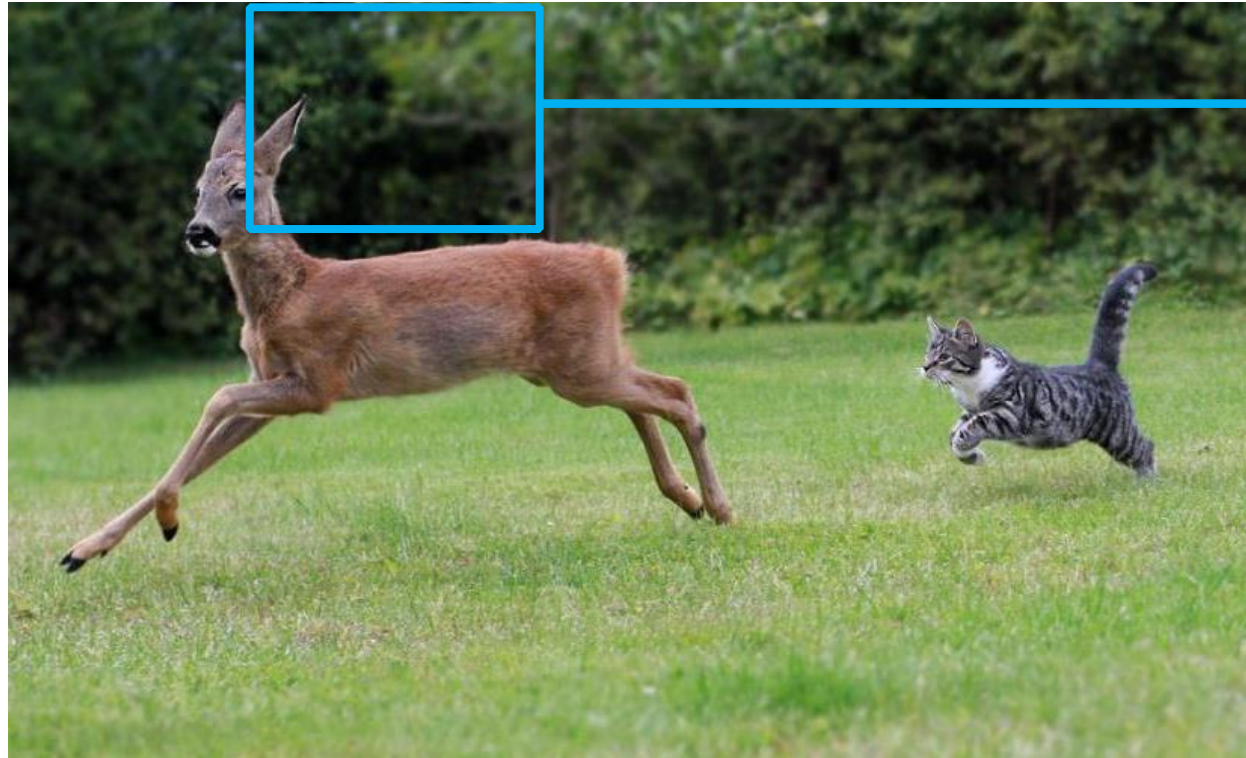
deer?
cat?
background?

Object Detection as Classification of sliding windows (proposals)



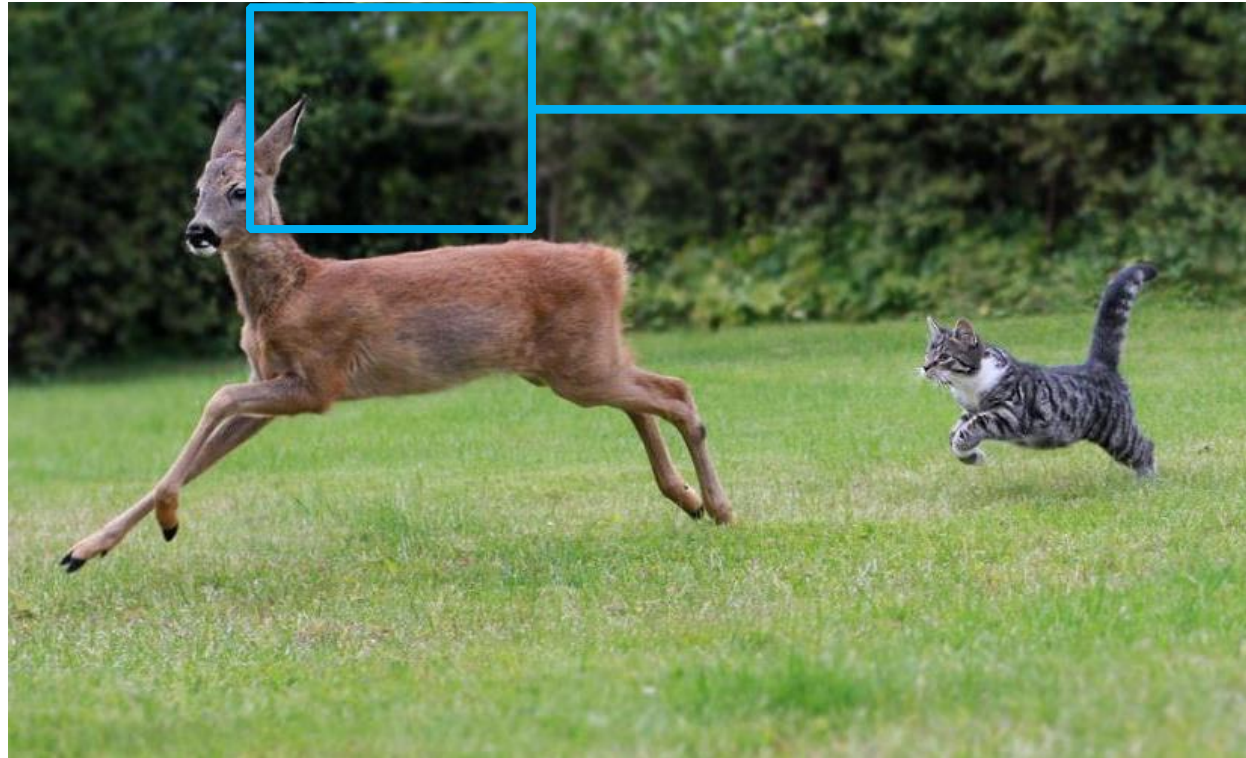
deer?
cat?
background?

Object Detection as Classification of sliding windows (proposals)



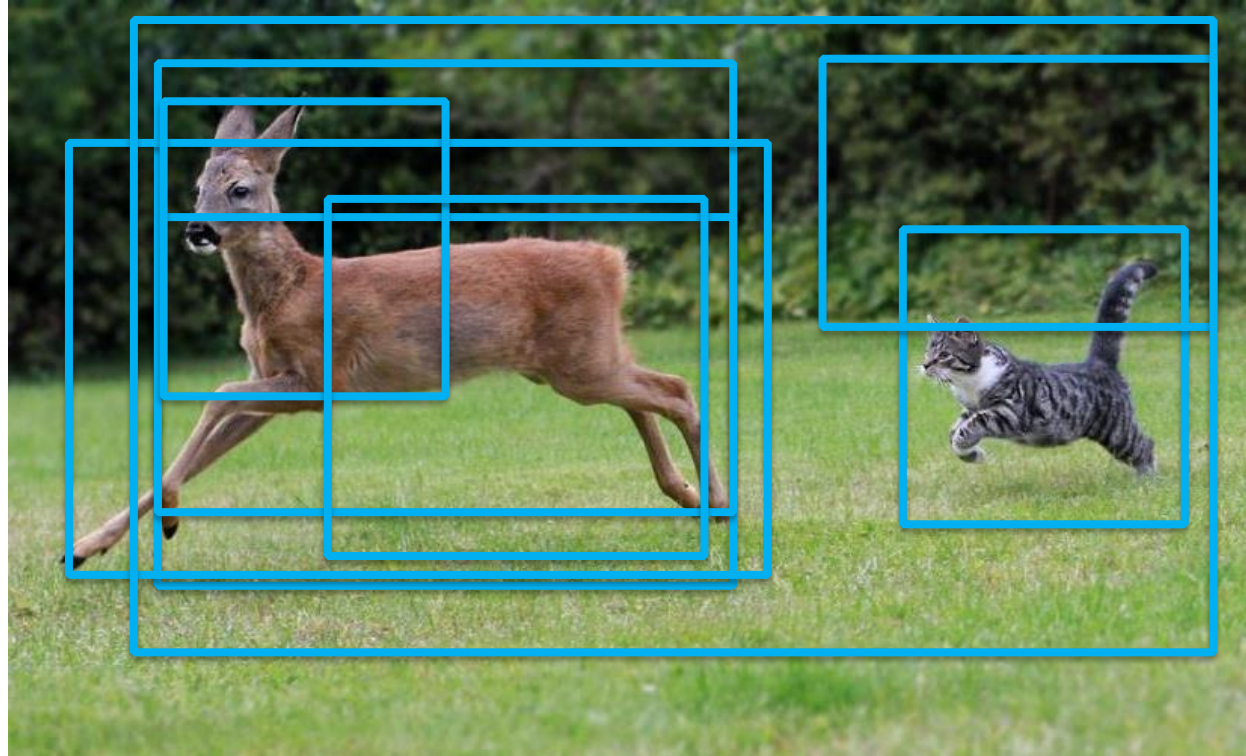
deer?
cat?
background?

Object Detection as Classification of sliding windows (proposals)



deer?
cat?
background?

Object Detection as Classification with Box Proposals



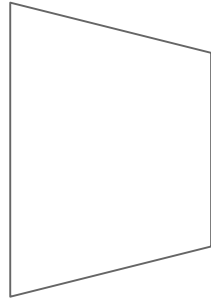
Sliding Window: Overfeat

Winner of ILSVRC 2013
localization challenge

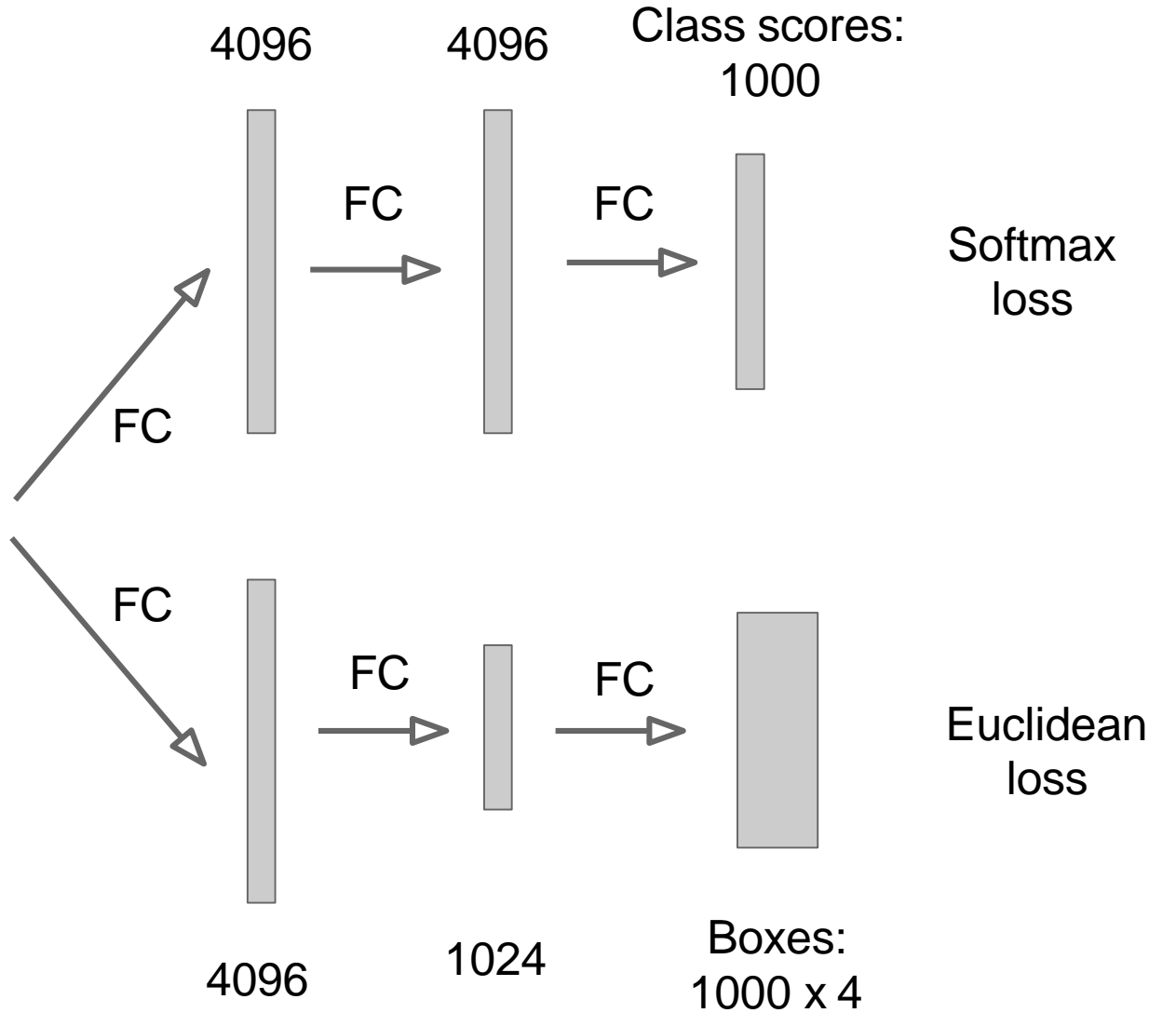


Image:
3 x 221 x 221

Convolution
+ pooling



Feature map:
1024 x 5 x 5



Sermanet et al, "Integrated Recognition, Localization and
Detection using Convolutional Networks", ICLR 2014

Sliding Window: Overfeat

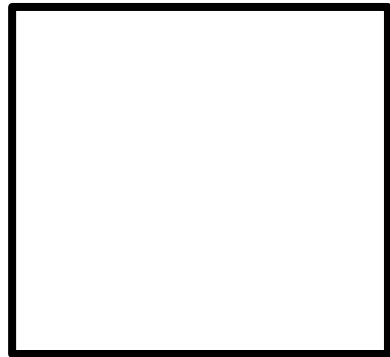


Network input:
3 x 221 x 221

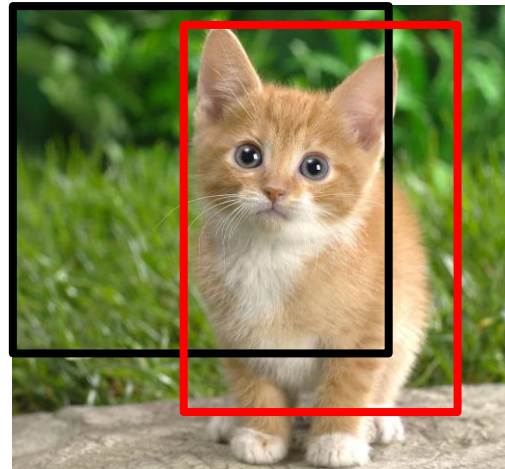


Larger image:
3 x 257 x 257

Sliding Window: Overfeat



Network input:
 $3 \times 221 \times 221$

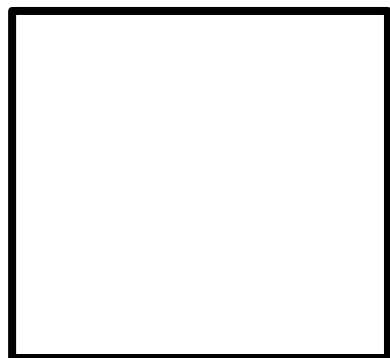


Larger image:
 $3 \times 257 \times 257$

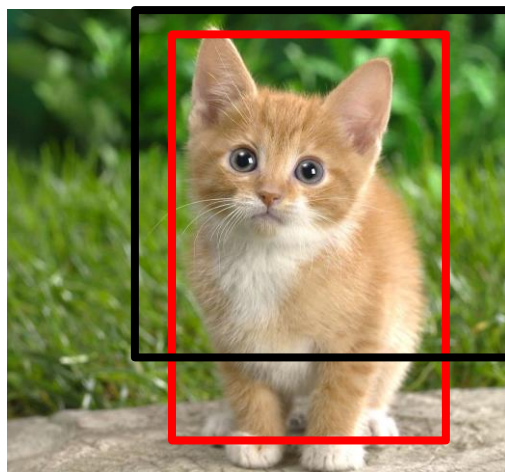
0.5	

Classification scores:
 $P(\text{cat})$

Sliding Window: Overfeat



Network input:
 $3 \times 221 \times 221$

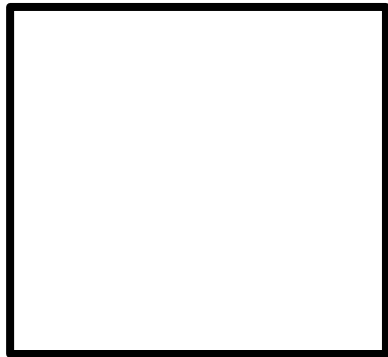


Larger image:
 $3 \times 257 \times 257$

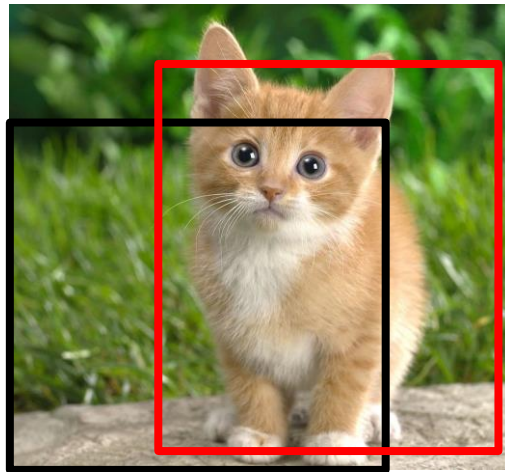
0.5	0.75

Classification scores:
 $P(\text{cat})$

Sliding Window: Overfeat



Network input:
3 x 221 x 221

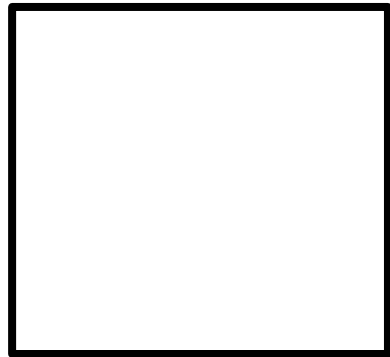


Larger image:
3 x 257 x 257

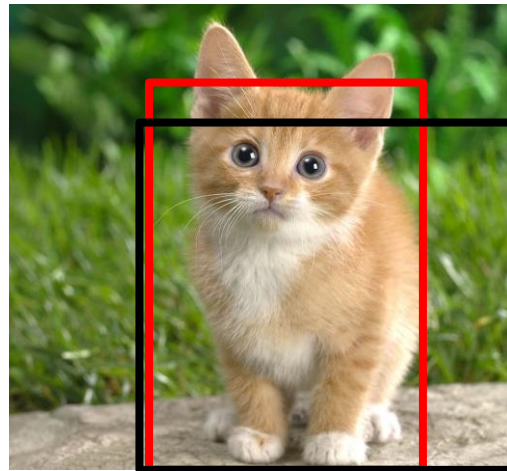
0.5	0.75
0.6	

Classification scores:
P(cat)

Sliding Window: Overfeat



Network input:
3 x 221 x 221

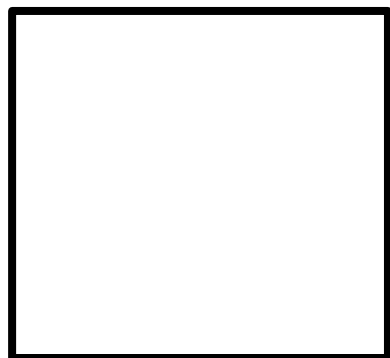


Larger image:
3 x 257 x 257

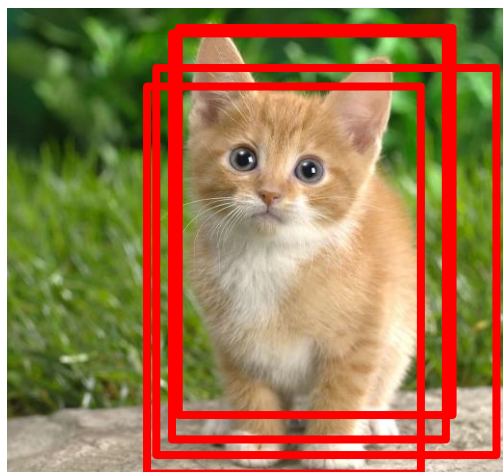
0.5	0.75
0.6	0.8

Classification scores:
P(cat)

Sliding Window: Overfeat



Network input:
3 x 221 x 221



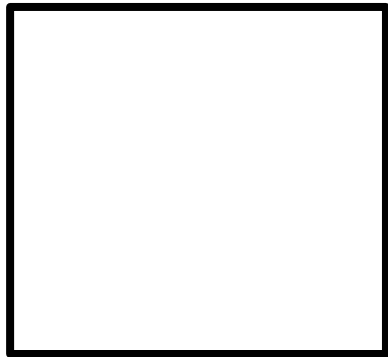
Larger image:
3 x 257 x 257

0.5	0.75
0.6	0.8

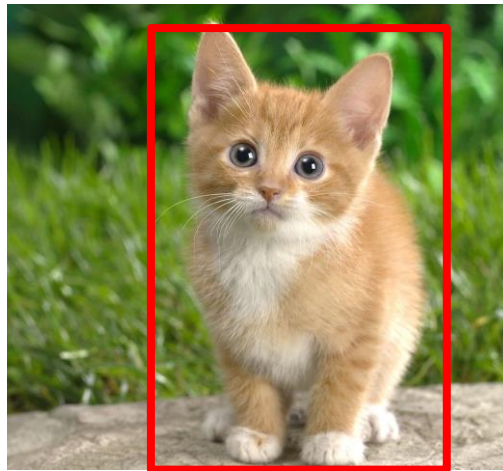
Classification scores:
P(cat)

Sliding Window: Overfeat

Greedily merge boxes and scores (details in paper)



Network input:
3 x 221 x 221



Larger image:
3 x 257 x 257

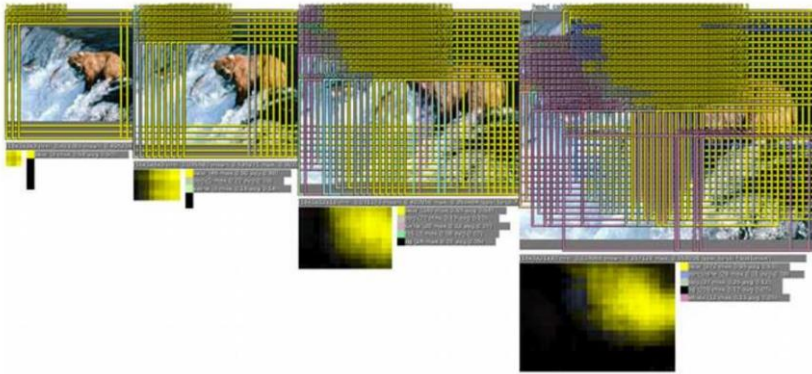
0.8

Classification score: P
(cat)

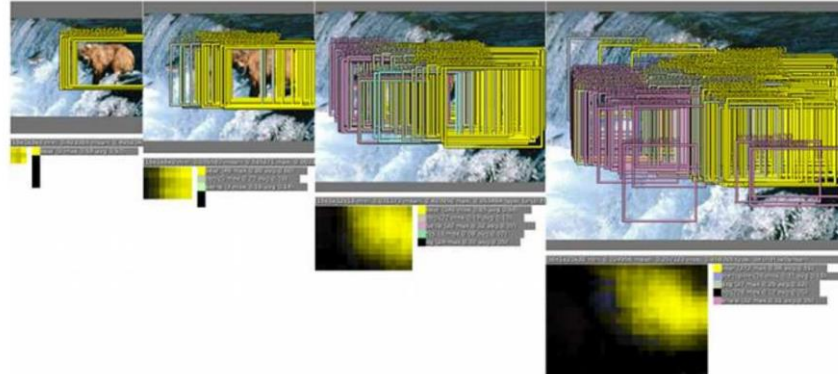
Sliding Window: Overfeat

In practice use many sliding window locations and multiple scales

Window positions + score maps



Box regression outputs



Final Predictions

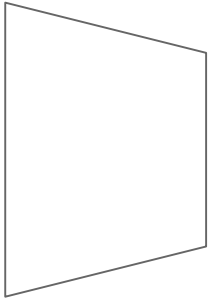


Efficient Sliding Window: Overfeat

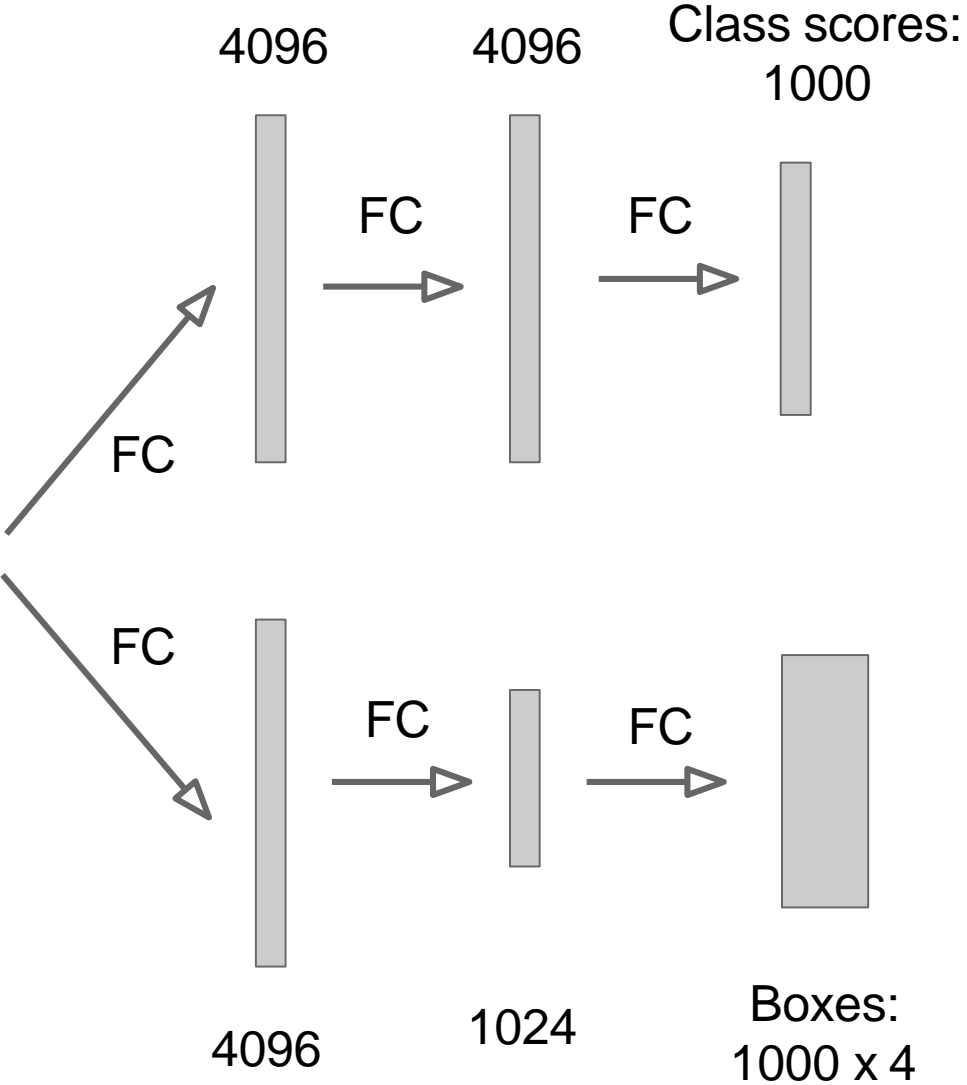


Image:
3 x 221 x 221

Convolution
+ pooling

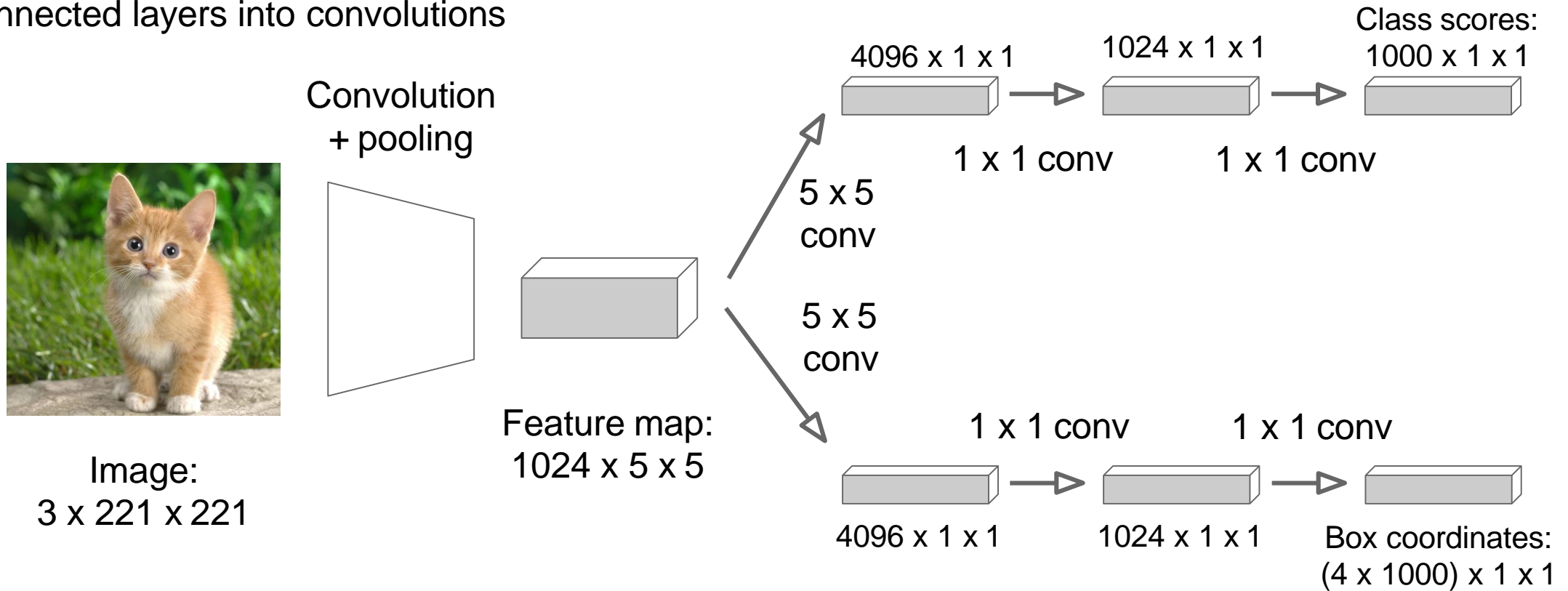


Feature map:
1024 x 5 x 5



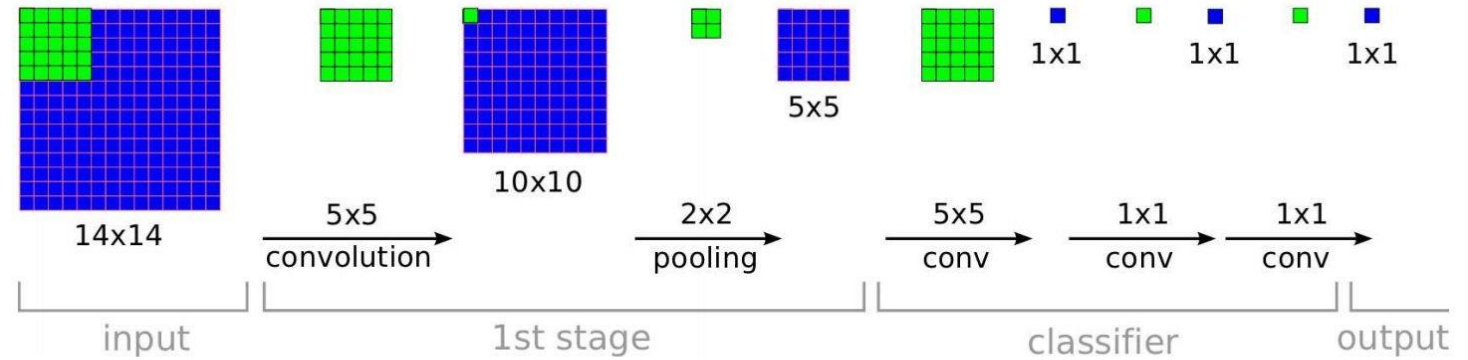
Efficient Sliding Window: Overfeat

Efficient sliding window by converting fully-connected layers into convolutions

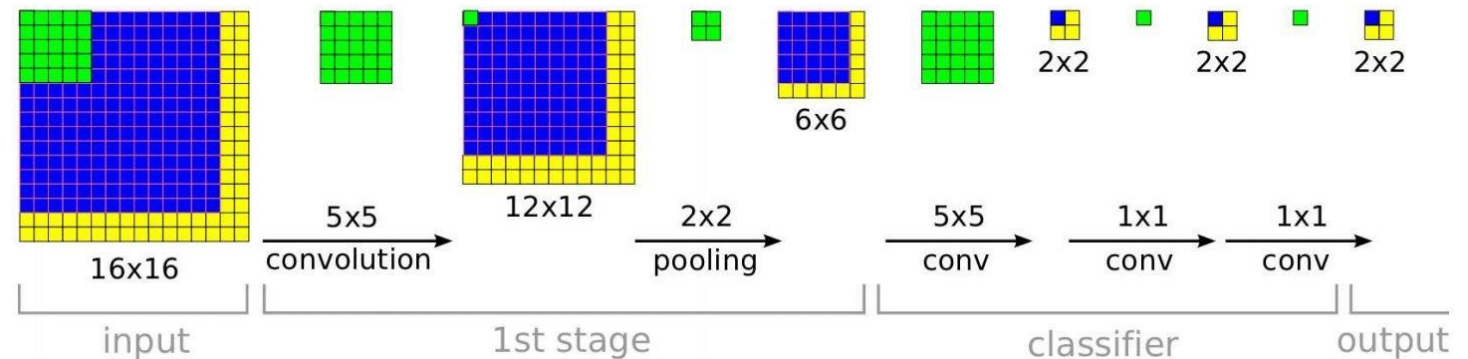


Efficient Sliding Window: Overfeat

Training time: Small image, 1 x 1 classifier output



Test time: Larger image, 2 x 2 classifier output, only extra compute at yellow regions



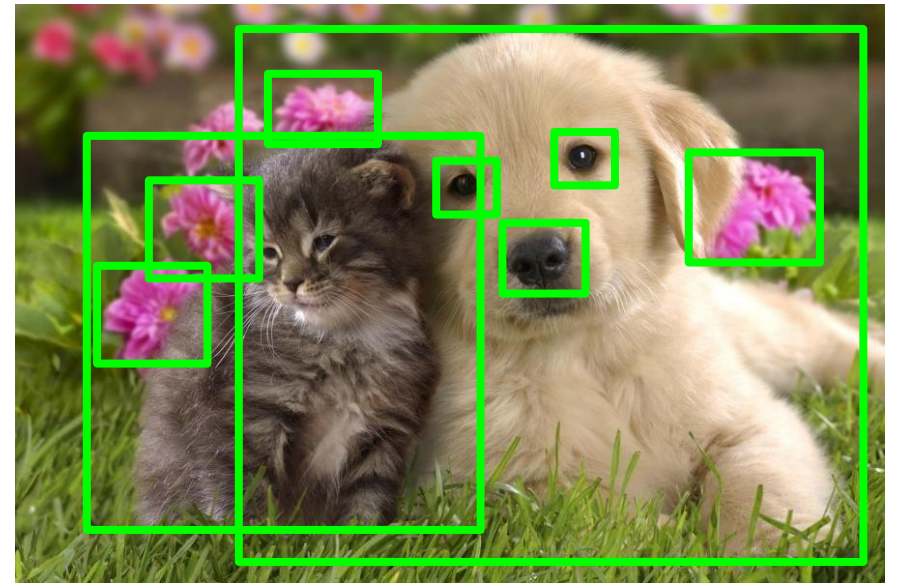
Detection as Classification

Problem: Need to test many positions and scales,
and use a computationally demanding classifier (CNN)

Solution: Only look at a tiny subset of possible positions

Region Proposals

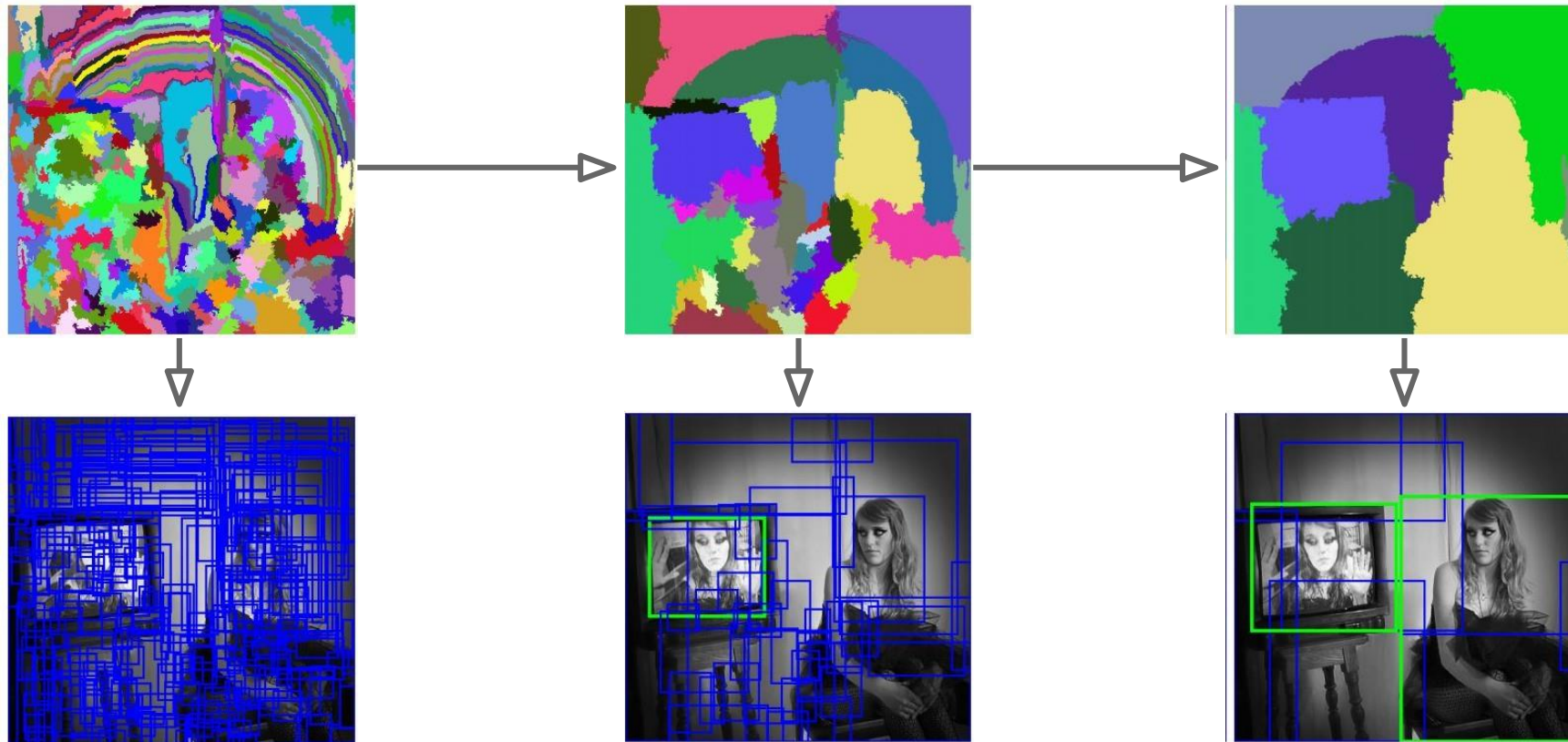
- Find “blobby” image regions that are likely to contain objects
- “Class-agnostic” object detector
- Look for “blob-like” regions



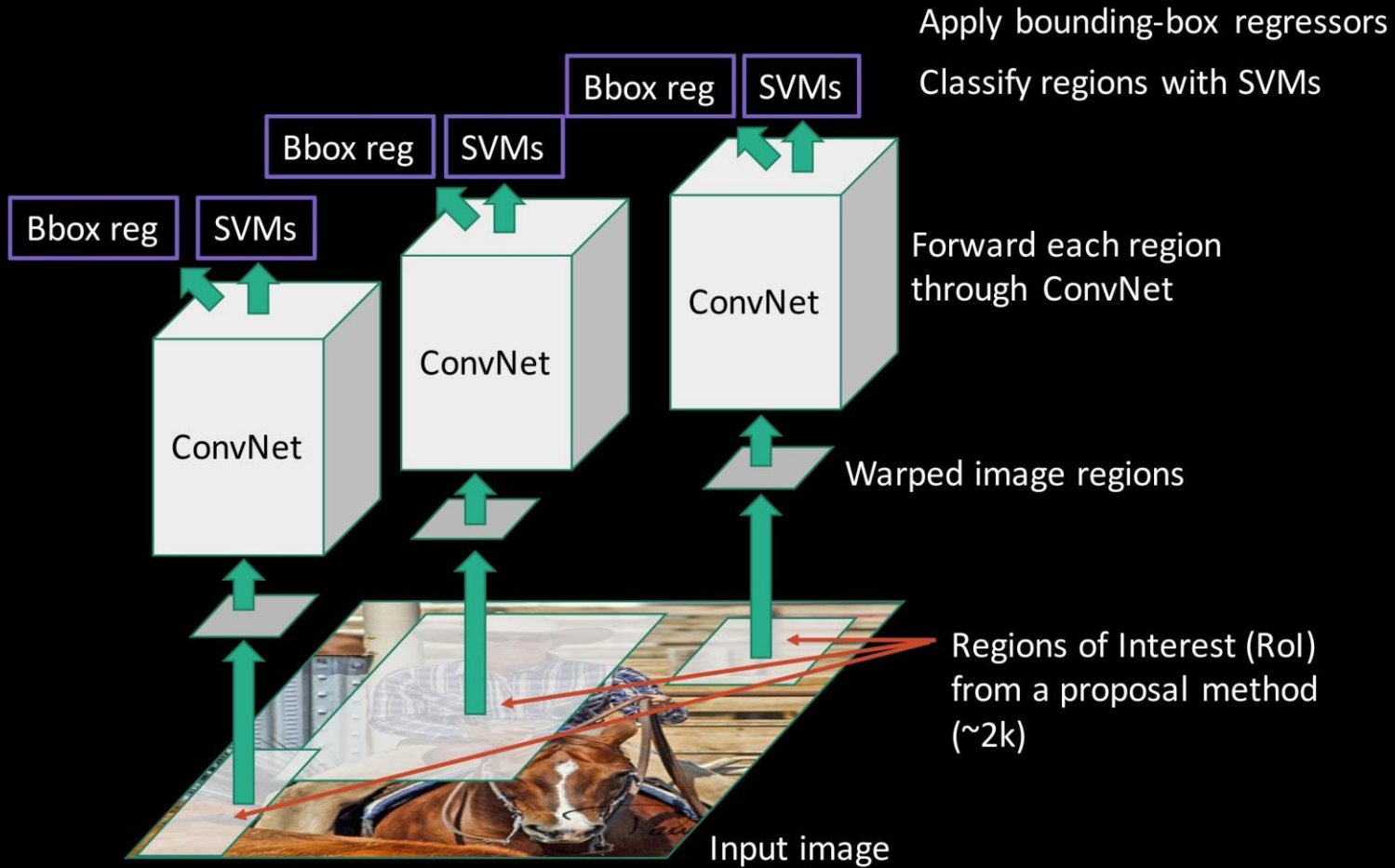
Region Proposals: Selective Search

Bottom-up segmentation, merging regions at multiple scales

Convert
regions
to boxes



Putting it together: R-CNN



Girshick et al. CVPR14.

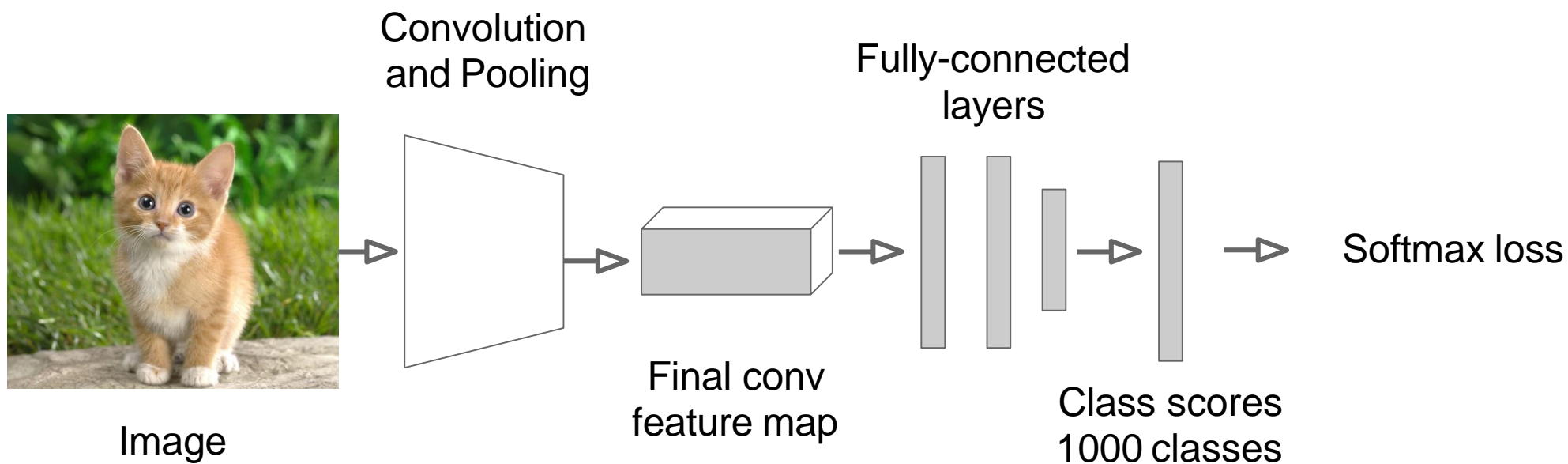
Post hoc component

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

Slide credit: Ross Girshick

R-CNN Training

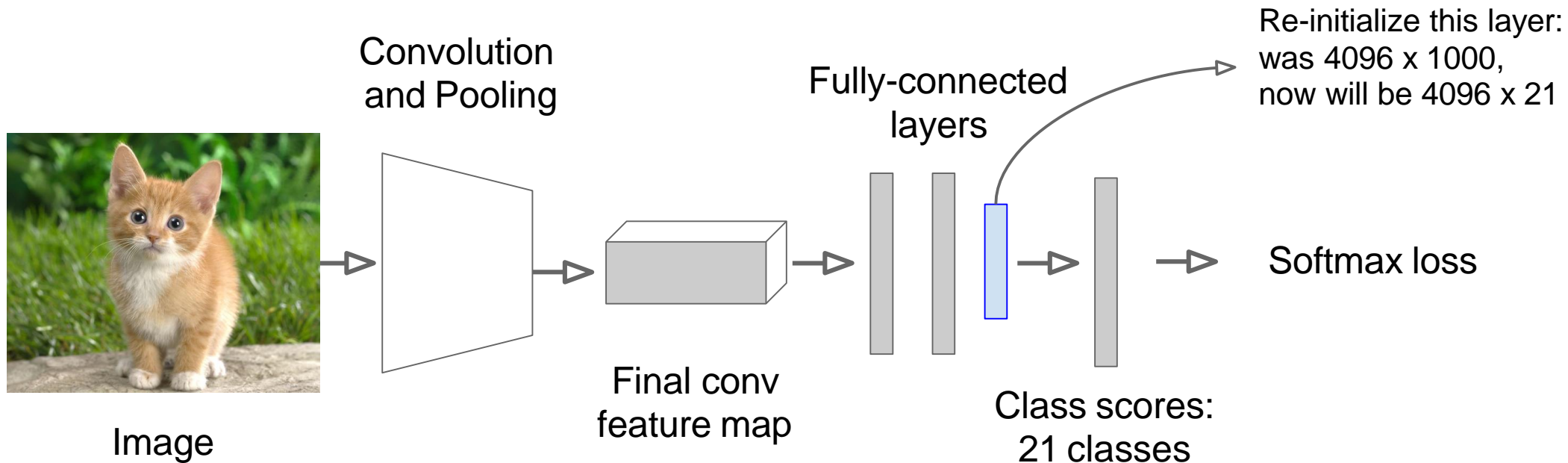
Step 1: Train (or download) a classification model for ImageNet (AlexNet)



R-CNN Training

Step 2: Fine-tune model for detection

- Instead of 1000 ImageNet classes, want 20 object classes + background
- Throw away final fully-connected layer, reinitialize from scratch
- Keep training model using positive / negative regions from detection images



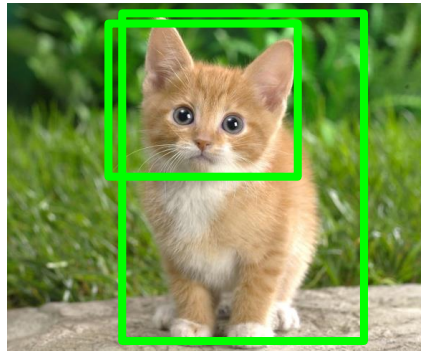
R-CNN Training

Step 3: Extract features

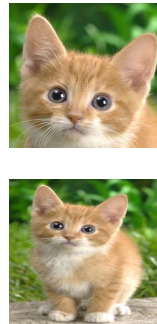
- Extract region proposals for all images
- For each region: warp to CNN input size, run forward through CNN, save pool5 features to disk
- Have a big hard drive: features are ~200GB for PASCAL dataset!



Image

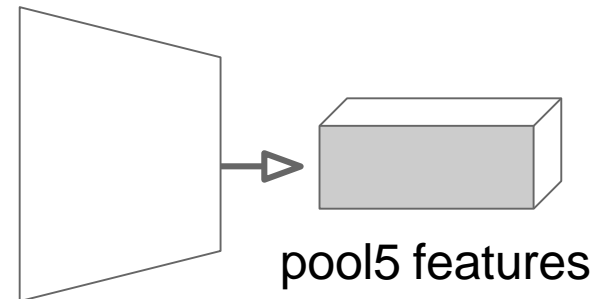


Region Proposals

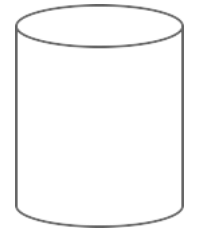


Crop + Warp

Convolution
and Pooling



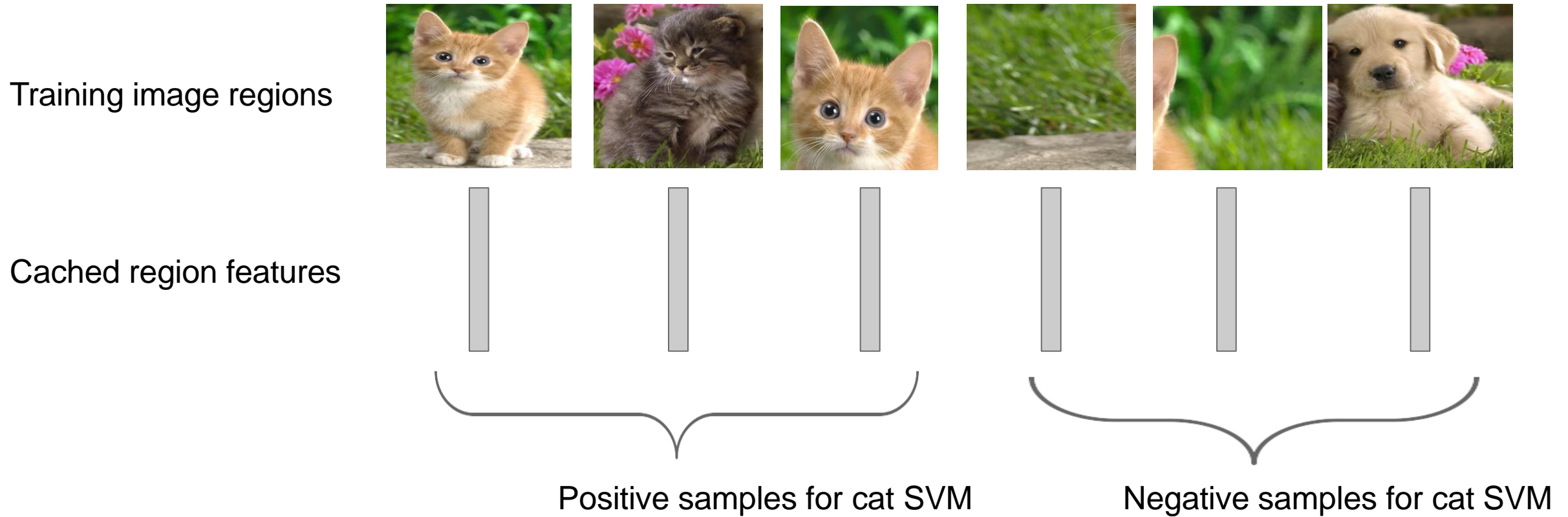
Forward pass



Save to disk

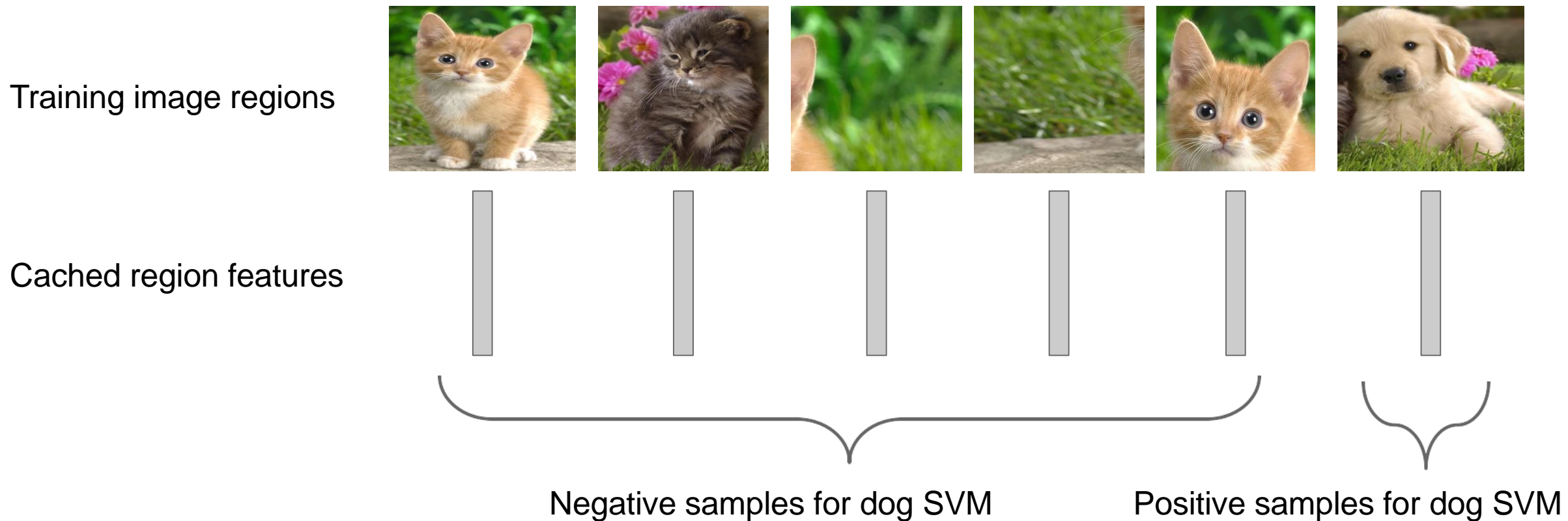
R-CNN Training

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



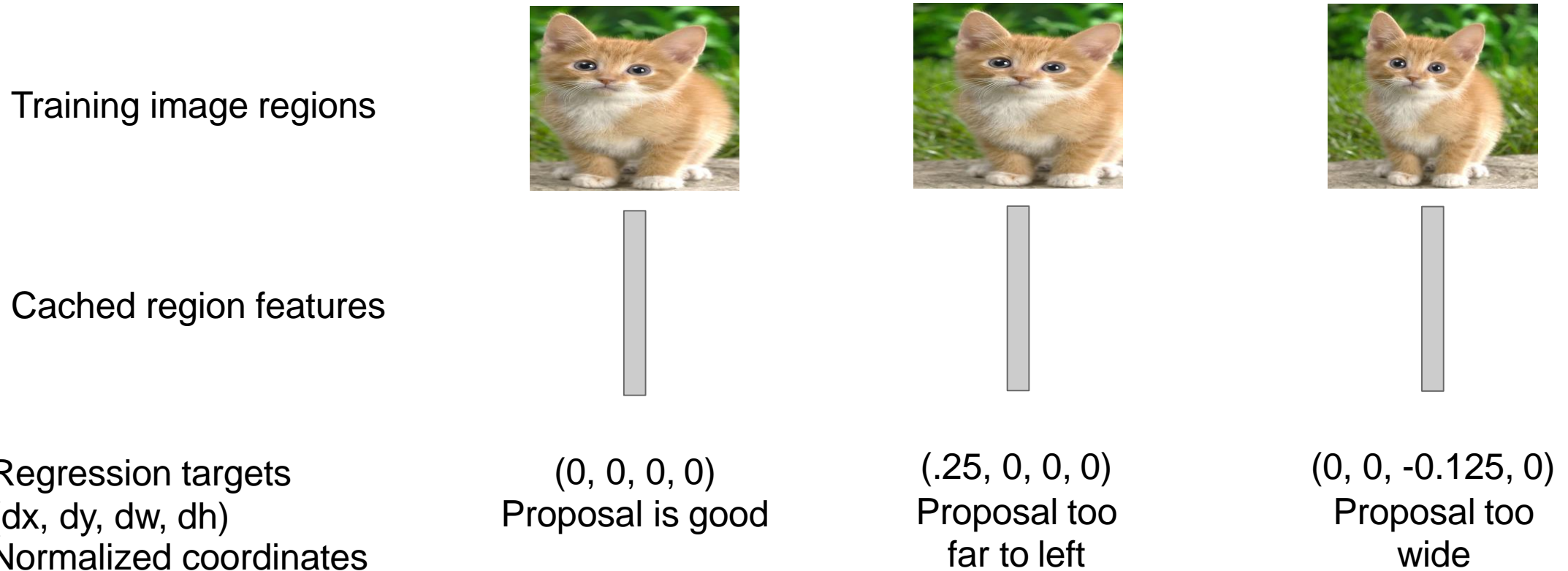
R-CNN Training

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



R-CNN Training

Step 5 (bbox 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



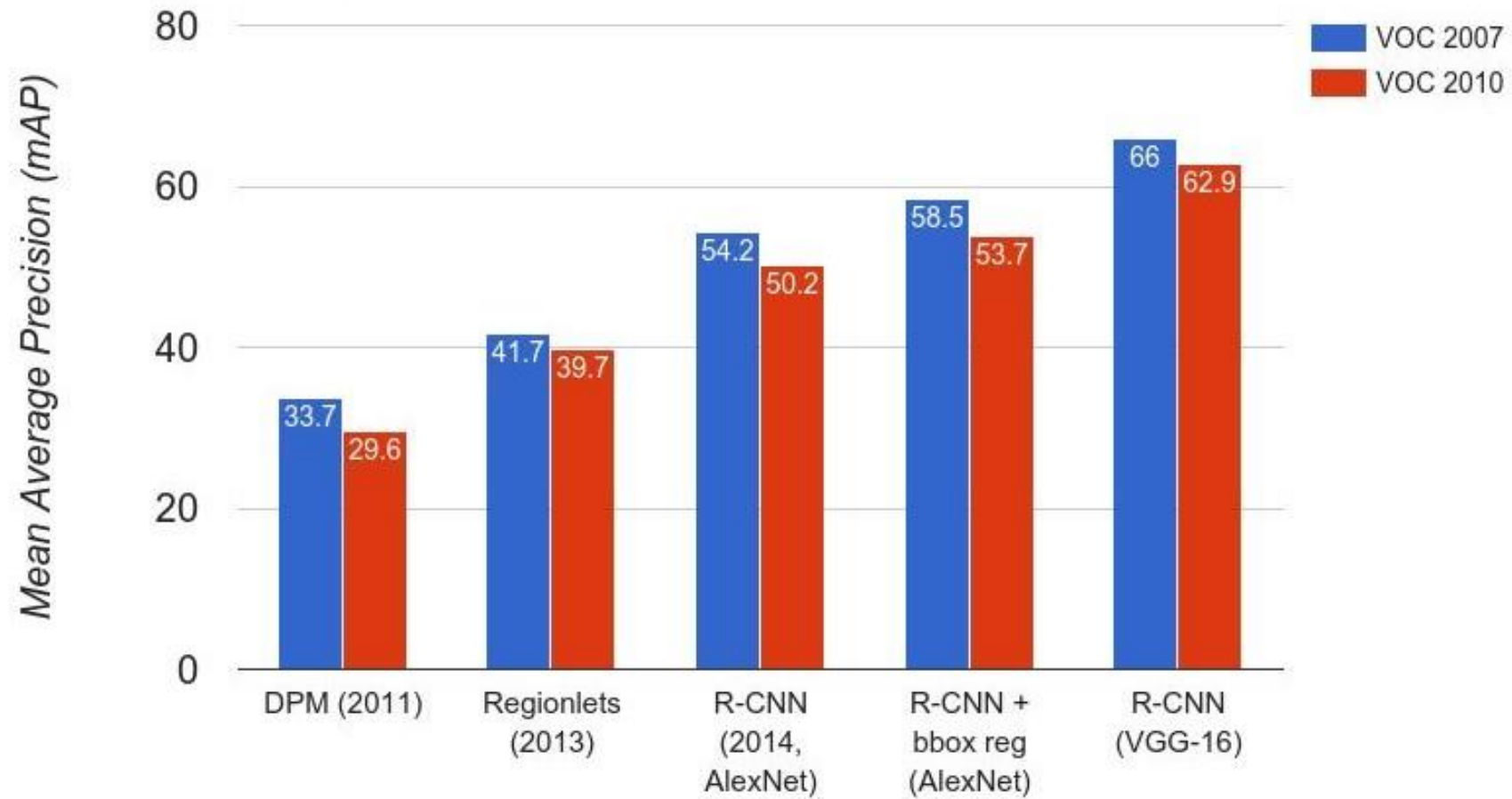
Object Detection: Datasets

	PASCAL VOC (2010)	ImageNet Detection (ILSVRC 2014)	MS-COCO (2014)
Number of classes	20	200	80
Number of images (train + val)	~20k	~470k	~120k
Mean objects per image	2.4	1.1	7.2

Object Detection: Evaluation

- We use a metric called “mean average precision” (mAP)
- Compute average precision (AP) separately for each class, then average over classes
- A detection is a true positive if it has IoU with a ground-truth box greater than some threshold (usually 0.5) (mAP@0.5)
- Combine all detections from all test images to draw a precision / recall curve for each class; AP is area under the curve
- TL;DR mAP is a number from 0 to 100; high is good

R-CNN Results



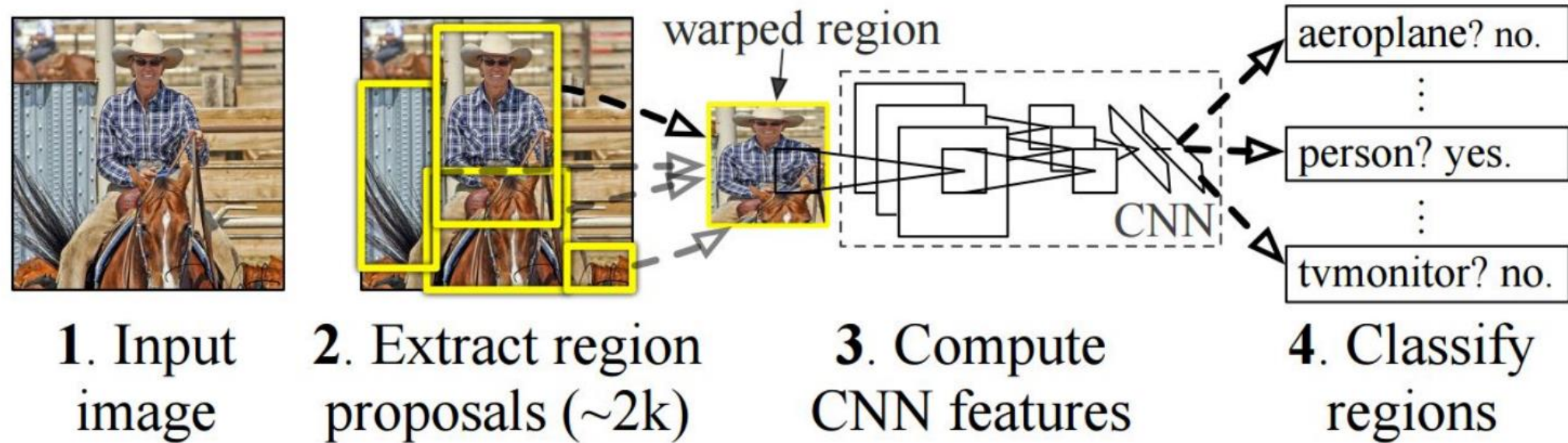
Wang et al, "Regionlets for Generic Object Detection", ICCV 2013

R-CNN Problems

1. Slow at test-time: need to run full forward pass of CNN for each region proposal
2. SVMs and regressors are post-hoc: CNN features not updated in response to SVMs and regressors
3. Complex multistage training pipeline

RCNN

R-CNN: *Regions with CNN features*



<https://people.eecs.berkeley.edu/~rbg/papers/r-cnn-cvpr.pdf>

Rich feature hierarchies for accurate object detection and semantic segmentation.

Girshick et al. CVPR 2014.

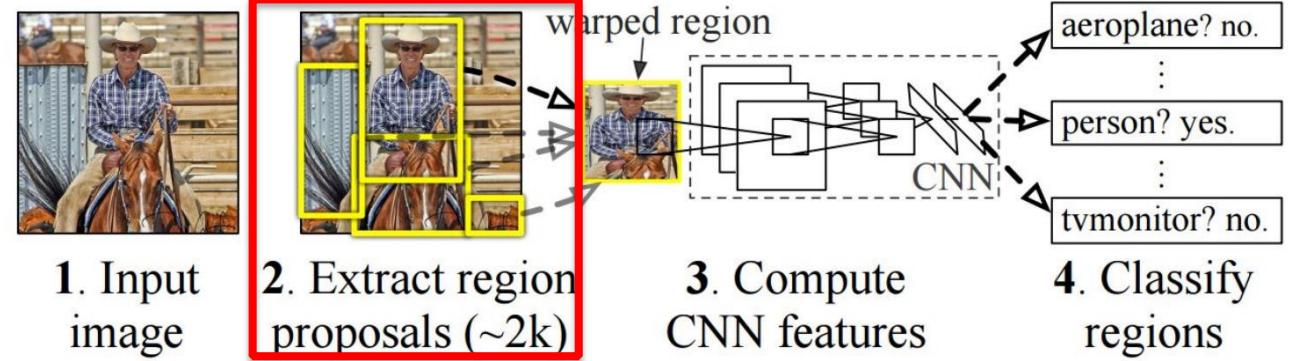
RCNN

First stage: generate category-independent region proposals.

- 2000 Region proposals for every image

Selective Search: combine the strength of both an exhaustive search and segmentation.
Uijlings et al. IJCV 2013.
ref

R-CNN: *Regions with CNN features*



RCNN

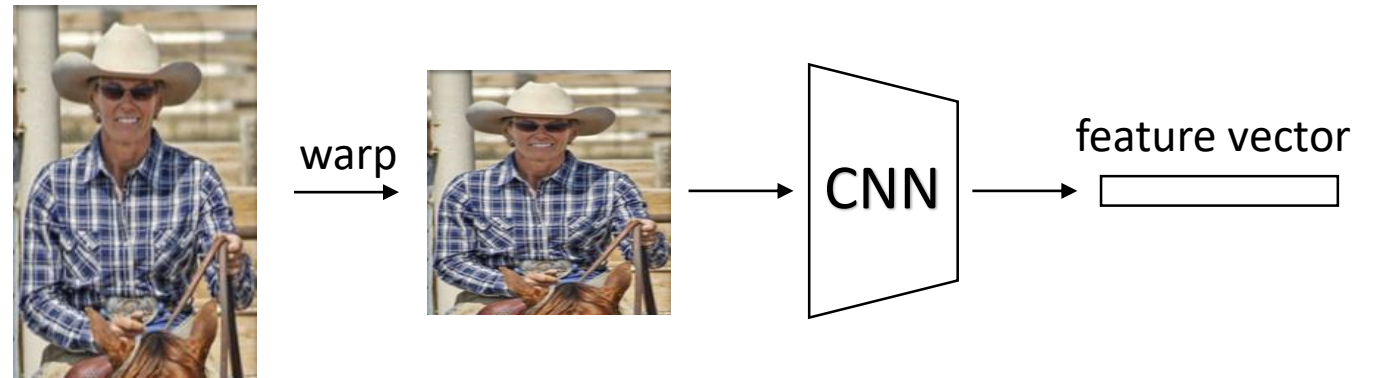
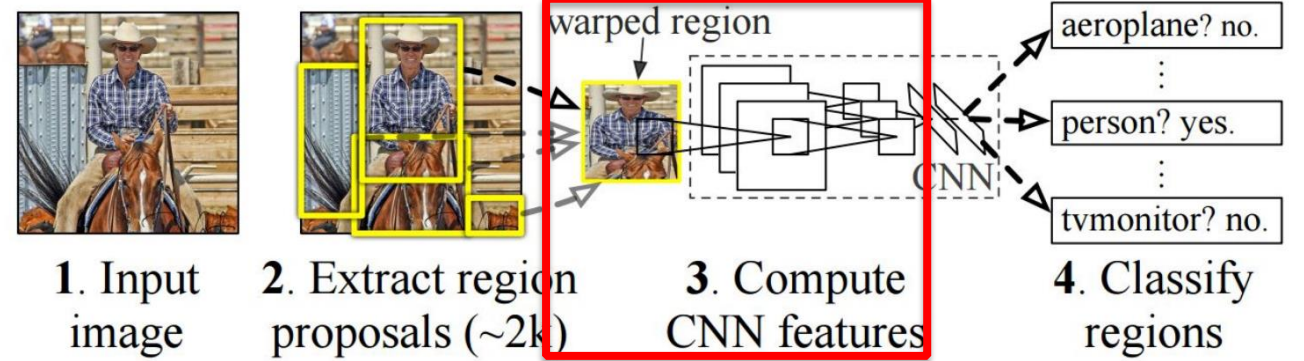
First stage: generate category-independent region proposals.

- 2000 Region proposals for every image

Second stage: extracts a fixed-length feature vector from each region.

- a 4096-dimensional feature vector from each region proposal

R-CNN: *Regions with CNN features*



Arbitrary rectangles?
A fixed size input? 227 x 227

5 conv layers + 2 fully connected layers

RCNN

First stage: generate category-independent region proposals.

- 2000 Region proposals for every image

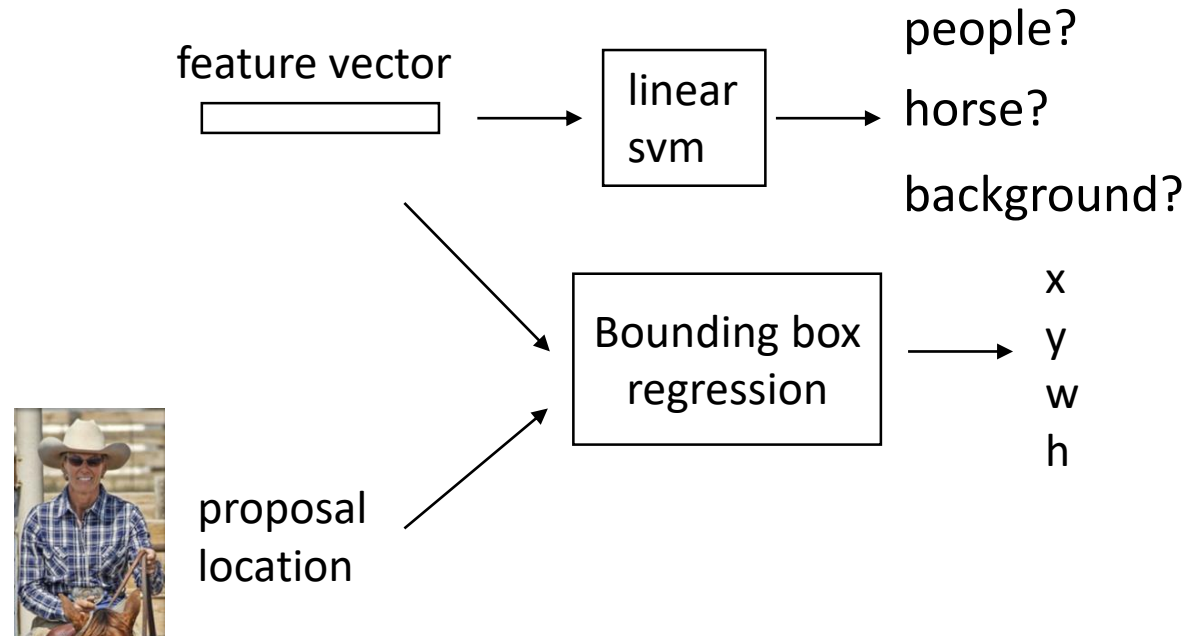
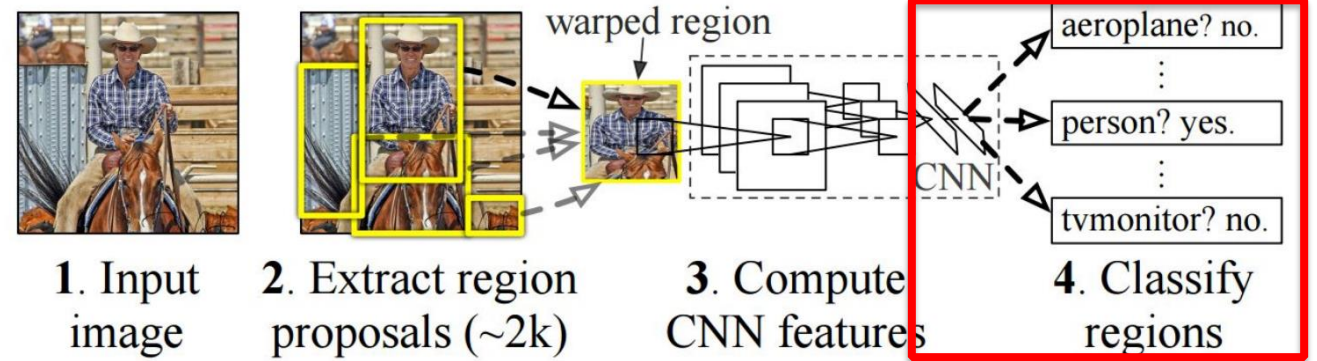
Second stage: extracts a fixed-length feature vector from each region.

- a 4096-dimensional feature vector from each region proposal

Third stage: a set of class-specific linear SVMs.

- object category and location

R-CNN: *Regions with CNN features*



RCNN

- Simple and scalable.
- improves mAP.

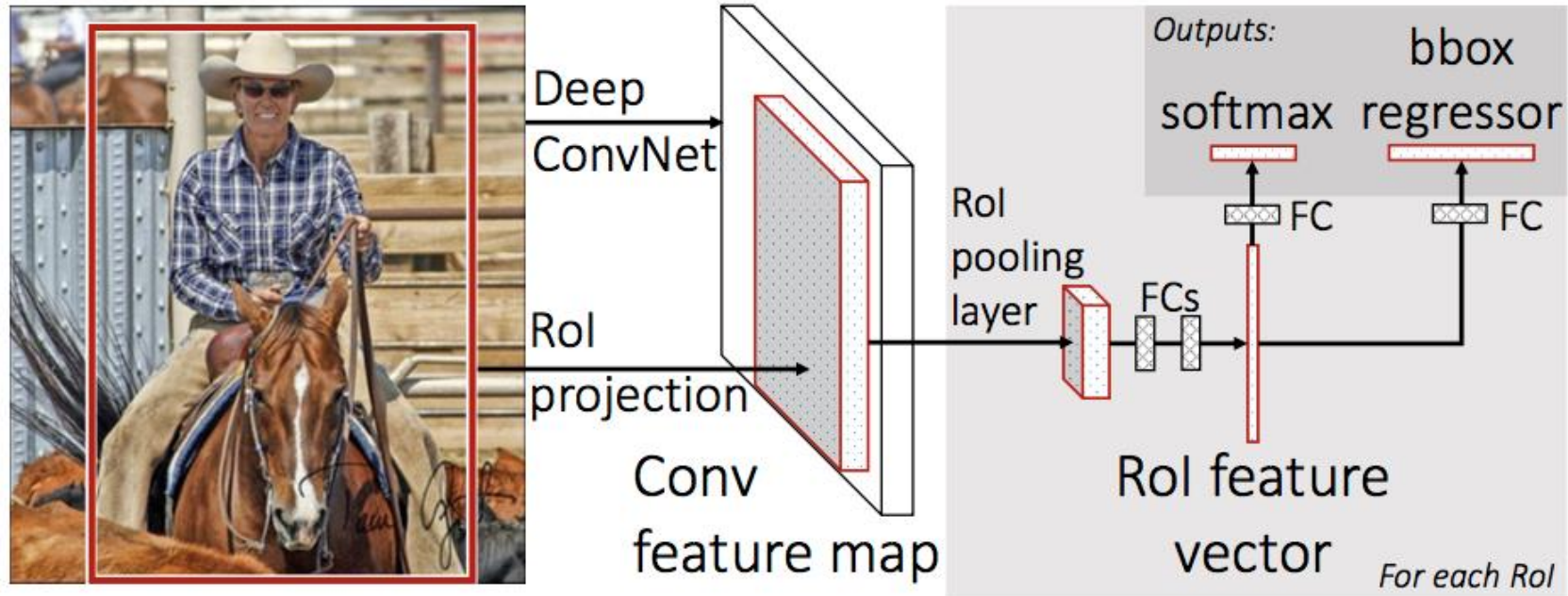
- A multistage pipeline.
- Training is expensive in space and time (features are extracted from each region proposal in each image and written into disk).
- Object detection is slow.



Fast-RCNN

?

Fast-RCNN

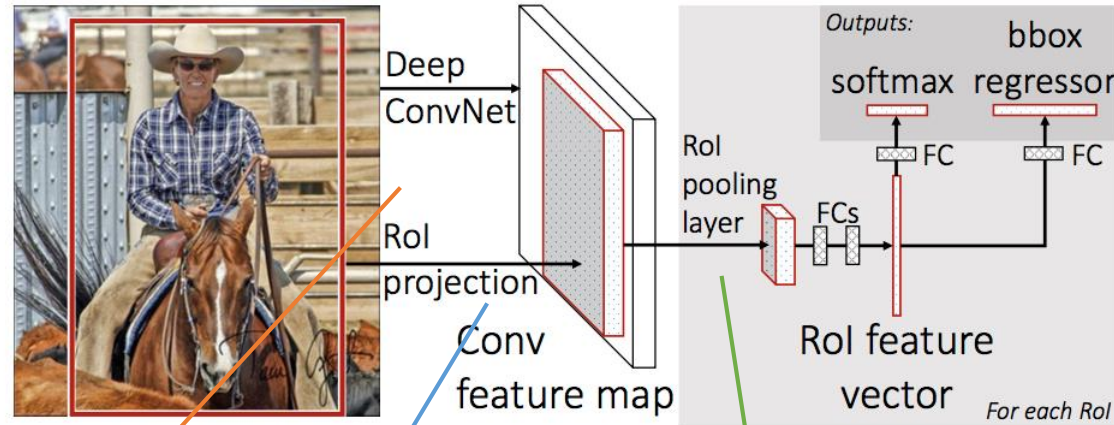


<https://arxiv.org/abs/1504.08083>

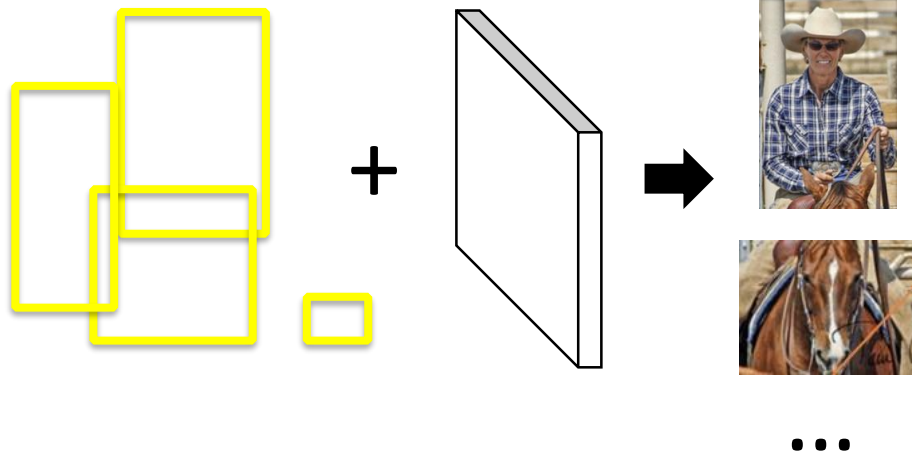
Fast R-CNN. Girshick. ICCV 2015.

Idea: No need to recompute features for every box independently

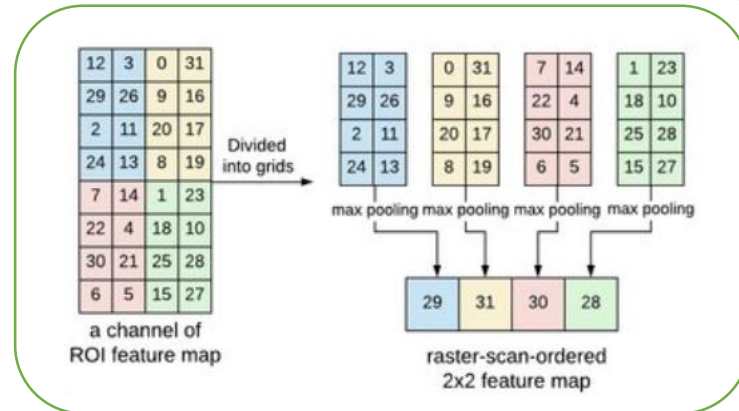
Fast-RCNN



Process the whole image with several convolutional (*conv*) and max pooling layers to produce a conv feature map.



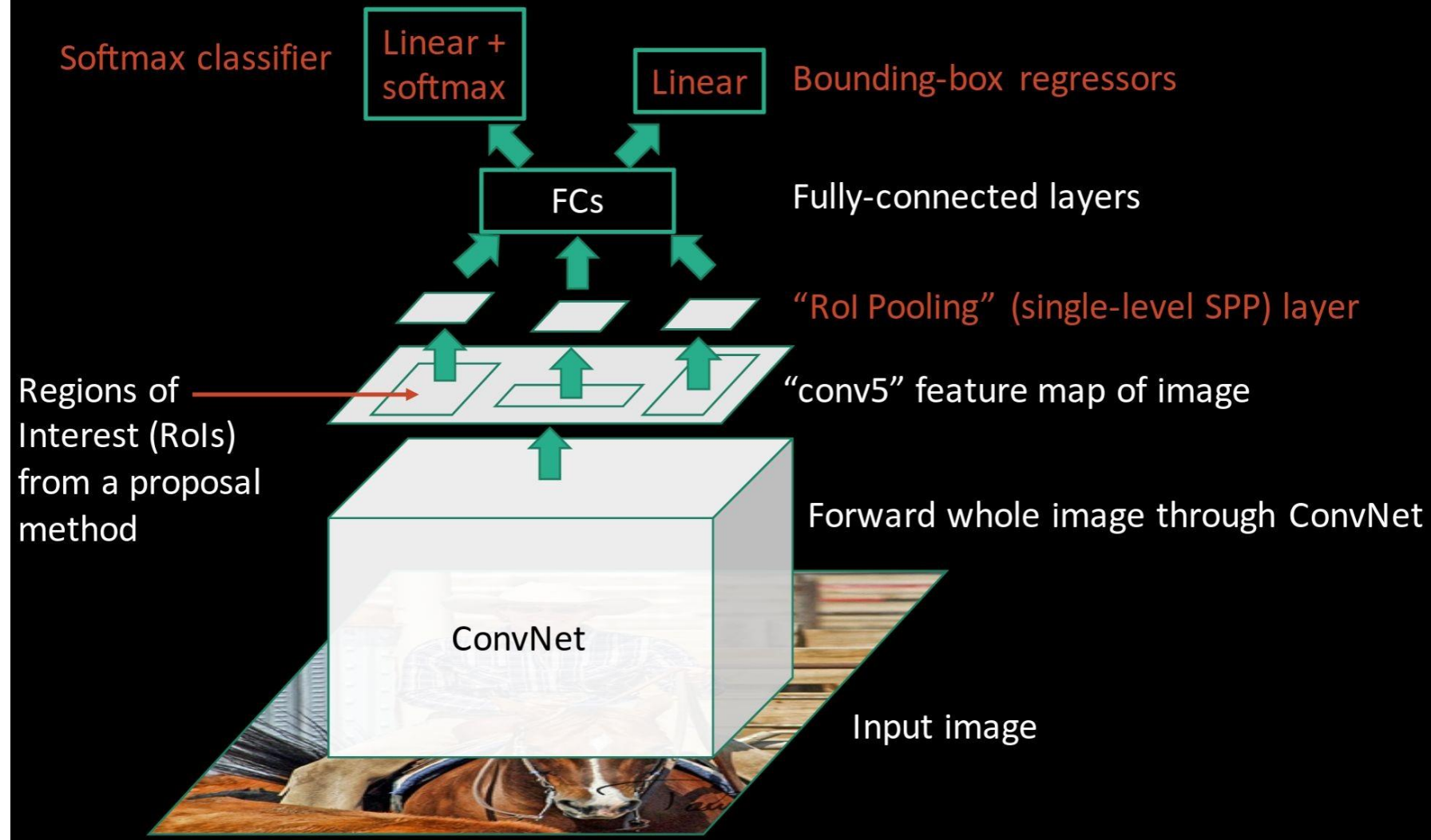
a region of interest (*RoI*) pooling layer extracts a **fixed-length** feature vector from the region feature map.



FC+ softmax $K + 1$ categories

FC+ regressor four real-valued numbers for each of the K object classes.

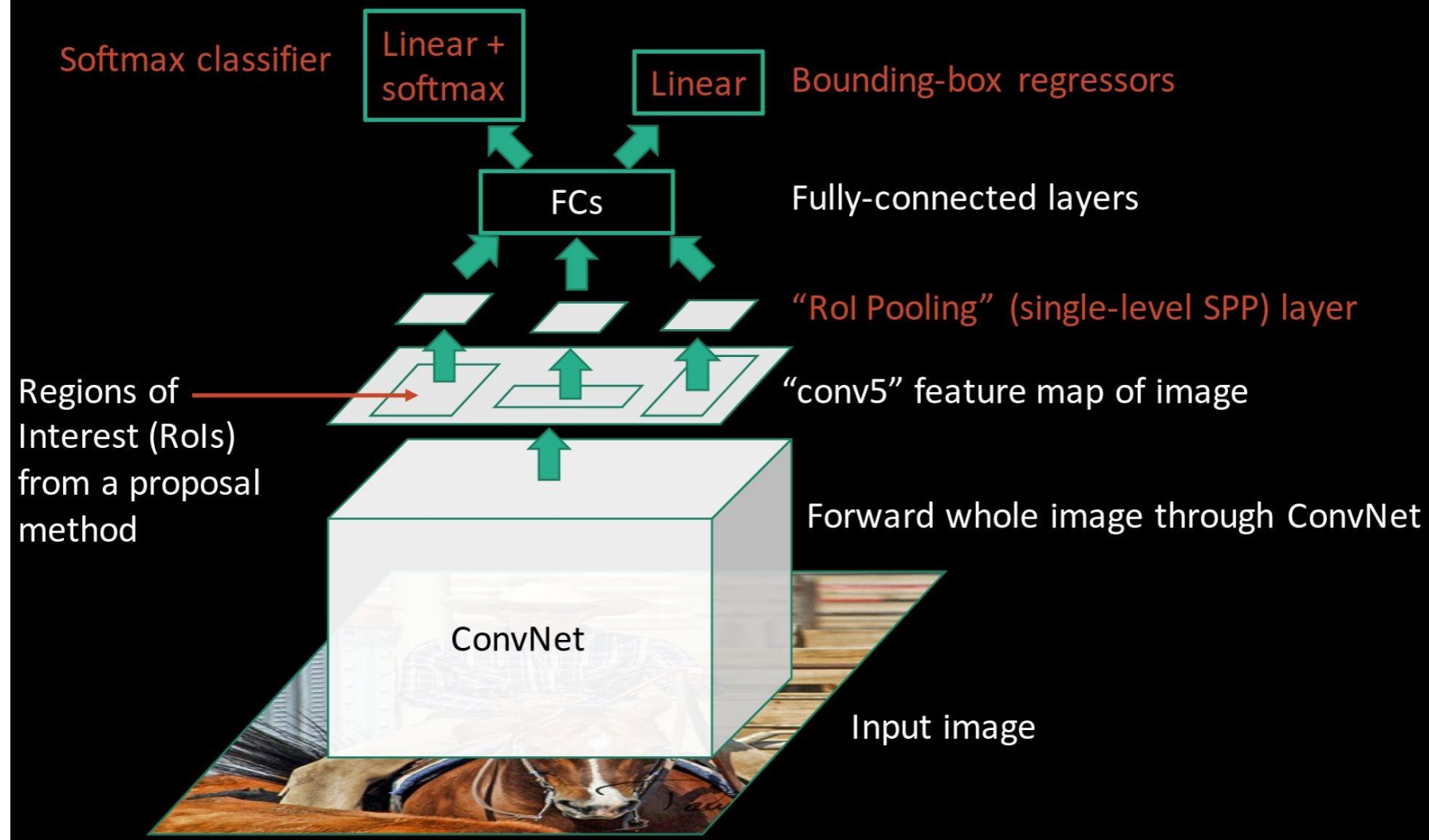
Fast R-CNN (test time)



Girschick, "Fast R-CNN", ICCV 2015

Slide credit: Ross Girschick

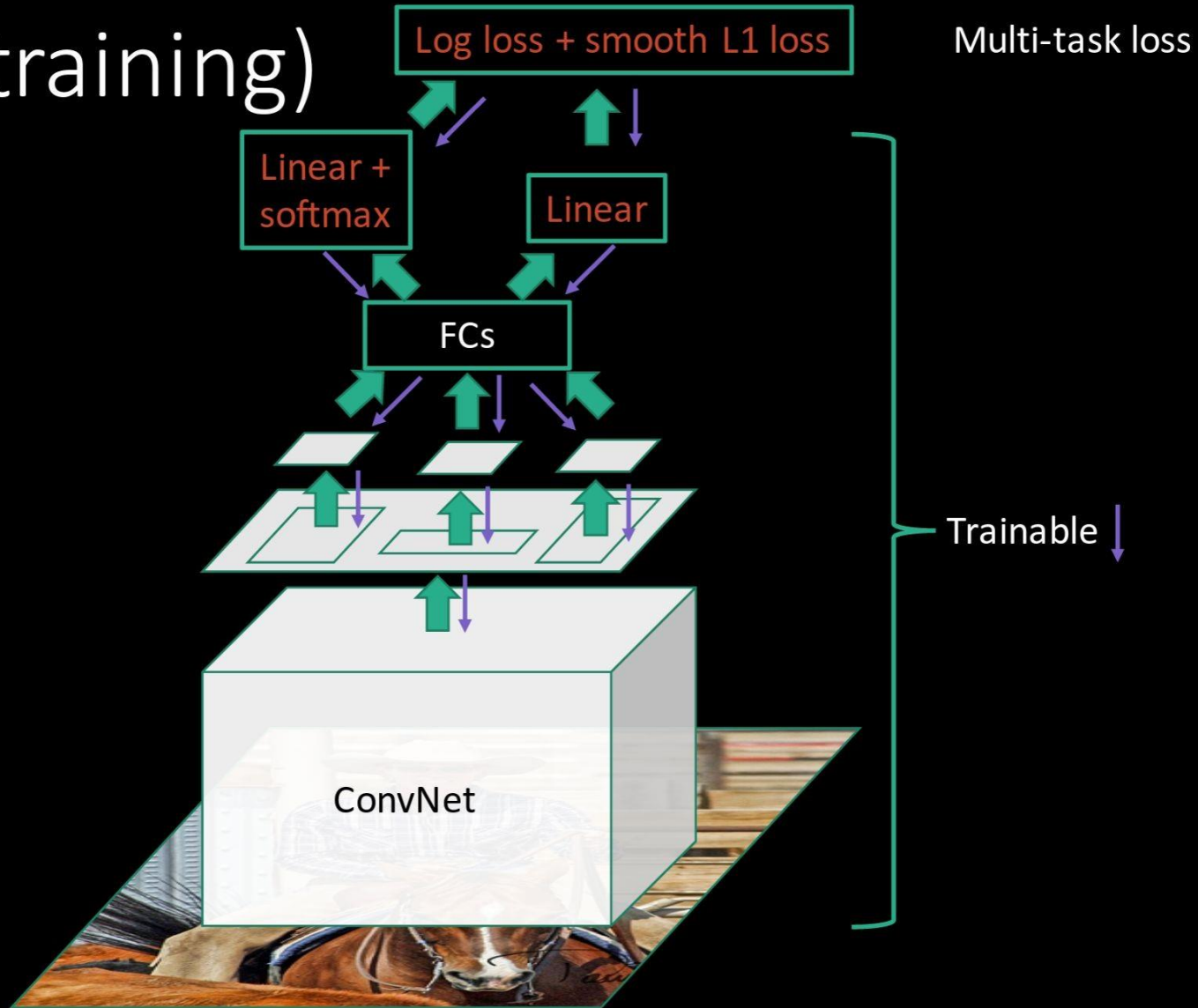
Fast R-CNN (test time)



R-CNN Problem #1:
Slow at test-time due to independent forward passes of the CNN

Solution:
Share computation of convolutional layers between proposals for an image

Fast R-CNN (training)



R-CNN Problem #2:

Post-hoc training: CNN not updated in response to final classifiers and regressors

R-CNN Problem #3:

Complex training pipeline

Solution:

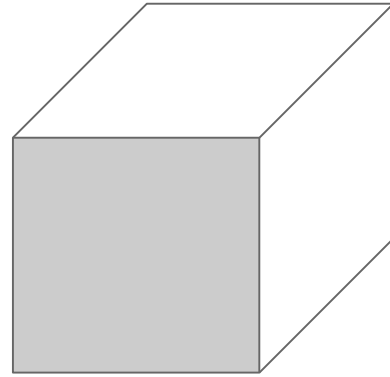
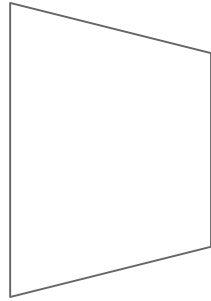
Just train the whole system end-to-end all at once!

Slide credit: Ross Girshick

Fast R-CNN: Region of Interest Pooling



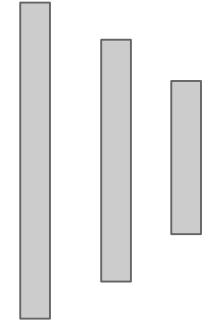
Convolution
and Pooling



Hi-res input image:
 $3 \times 800 \times 600$
with region
proposal

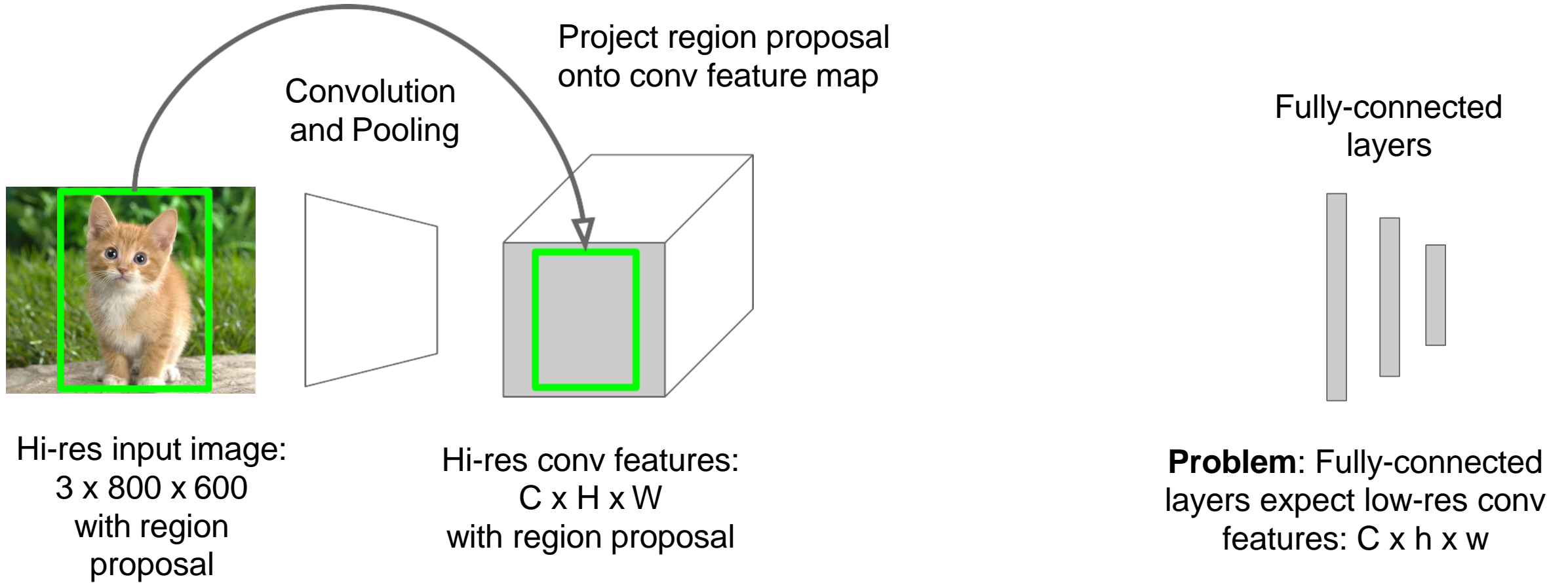
Hi-res conv features:
 $C \times H \times W$
with region proposal

Fully-connected
layers

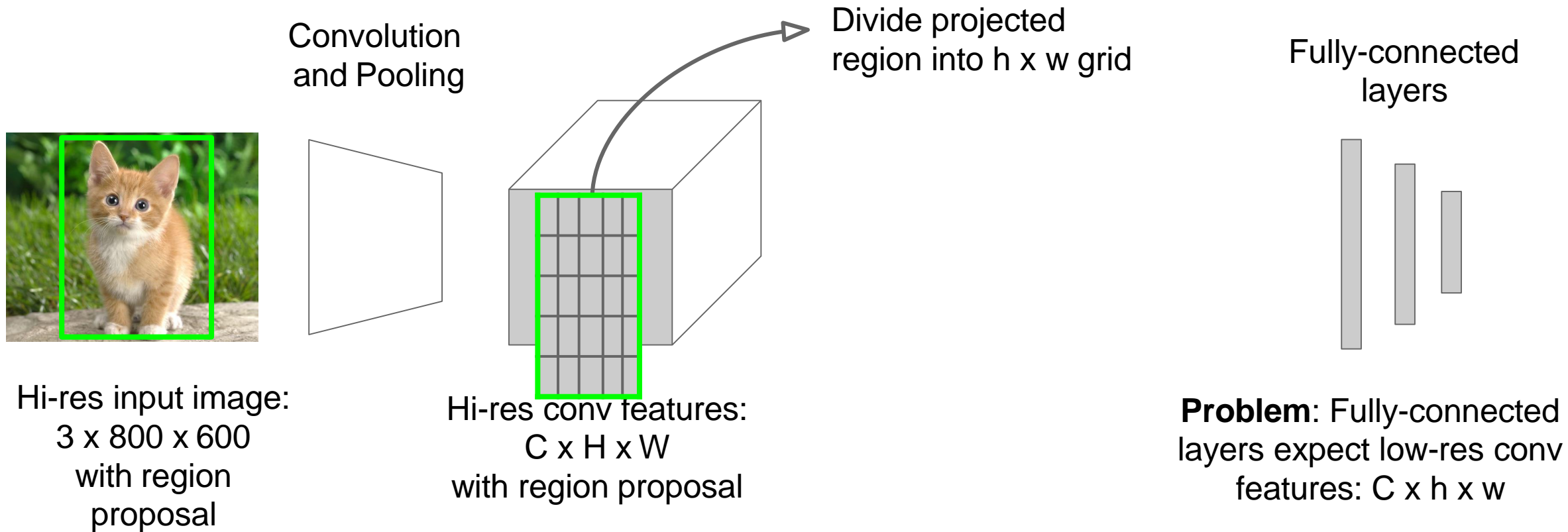


Problem: Fully-connected
layers expect low-res conv
features: $C \times h \times w$

Fast R-CNN: Region of Interest Pooling



Fast R-CNN: Region of Interest Pooling

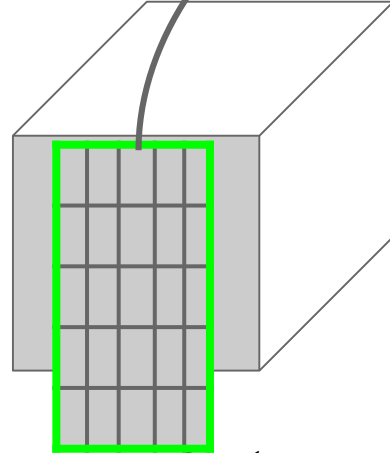
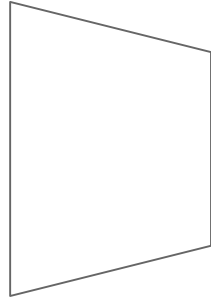


Fast R-CNN: Region of Interest Pooling



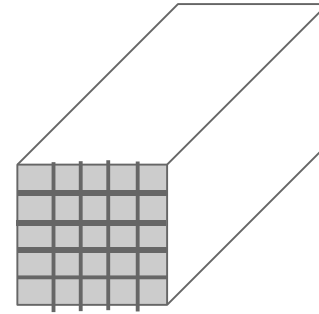
Hi-res input image:
 $3 \times 800 \times 600$
with region
proposal

Convolution
and Pooling



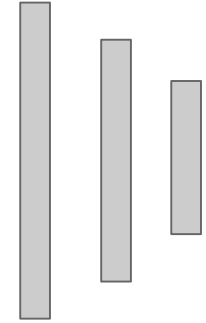
Hi-res conv features:
 $C \times H \times W$
with region proposal

Max-pool within
each grid cell



RoI conv features:
 $C \times h \times w$
for region proposal

Fully-connected
layers



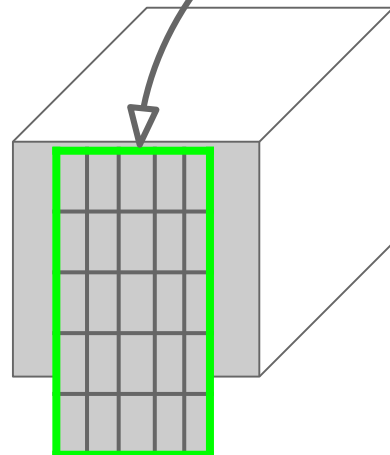
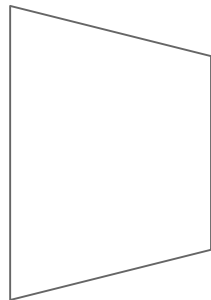
Fully-connected layers expect
low-res conv features:
 $C \times h \times w$

Fast R-CNN: Region of Interest Pooling



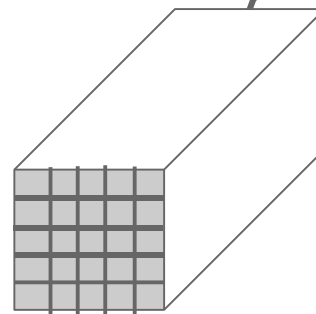
Hi-res input image:
 $3 \times 800 \times 600$
with region
proposal

Convolution
and Pooling



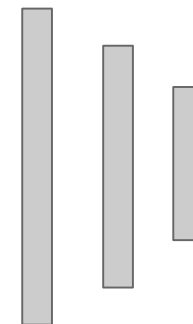
Hi-res conv features:
 $C \times H \times W$
with region proposal

Can back propagate
similar to max pooling



RoI conv features:
 $C \times h \times w$
for region proposal

Fully-connected
layers



Fully-connected layers expect
low-res conv features:
 $C \times h \times w$

Fast R-CNN Results

Faster!

	R-CNN	Fast R-CNN
Training Time:	84 hours	9.5 hours
(Speedup)	1x	8.8x

Using VGG-16 CNN on Pascal VOC 2007 dataset

Fast R-CNN Results

	R-CNN	Fast R-CNN
Training Time:	84 hours	9.5 hours
(Speedup)	1x	8.8x
Test time per image	47 seconds	0.32 seconds
(Speedup)	1x	146x

Faster!

FASTER!

Using VGG-16 CNN on Pascal VOC 2007 dataset

Fast R-CNN Results

	R-CNN	Fast R-CNN
Faster!	84 hours	9.5 hours
(Speedup)	1x	8.8x
FASTER!	47 seconds	0.32 seconds
(Speedup)	1x	146x
Better!	66.0	66.9
mAP (VOC 2007)		

Using VGG-16 CNN on Pascal VOC 2007 dataset

Fast R-CNN Problem:

Test-time speeds don't include region proposals

	R-CNN	Fast R-CNN
Test time per image	47 seconds	0.32 seconds
(Speedup)	1x	146x
Test time per image with Selective Search	50 seconds	2 seconds
(Speedup)	1x	25x

Fast R-CNN Problem Solution:

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

	R-CNN	Fast R-CNN
Test time per image	47 seconds	0.32 seconds
(Speedup)	1x	146x
Test time per image with Selective Search	50 seconds	2 seconds
(Speedup)	1x	25x

RCNN

- Simple and scalable.
- improves mAP.
- A multistage pipeline.
- Training is expensive in space and time (features are extracted from each region proposal in each image and written into disk).
- Object detection is slow.



Fast-RCNN

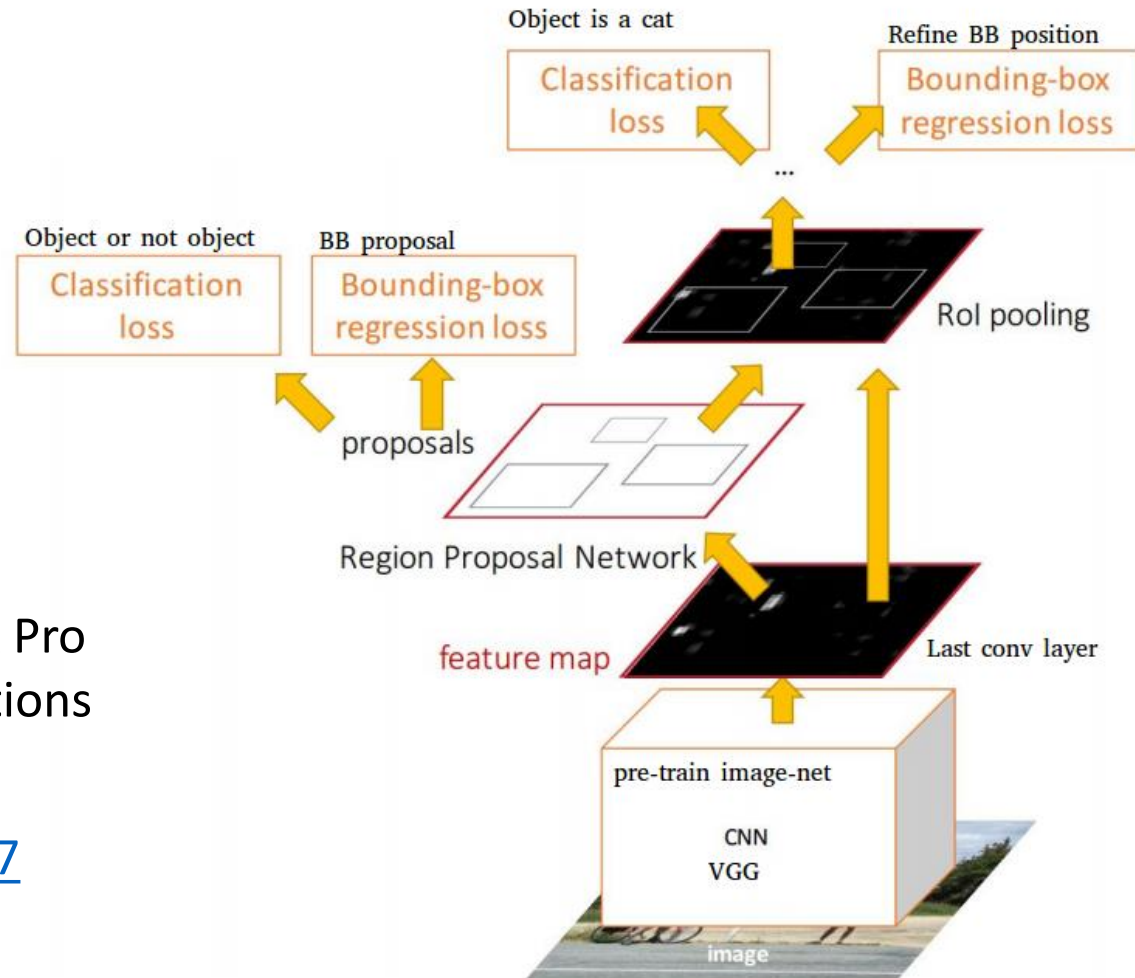
- Higher mAP.
- Single stage, end-to-end training.
- No disk storage is required for feature caching.
- proposals are the computational bottleneck in detection systems.



Faster-RCNN

?

Faster-RCNN



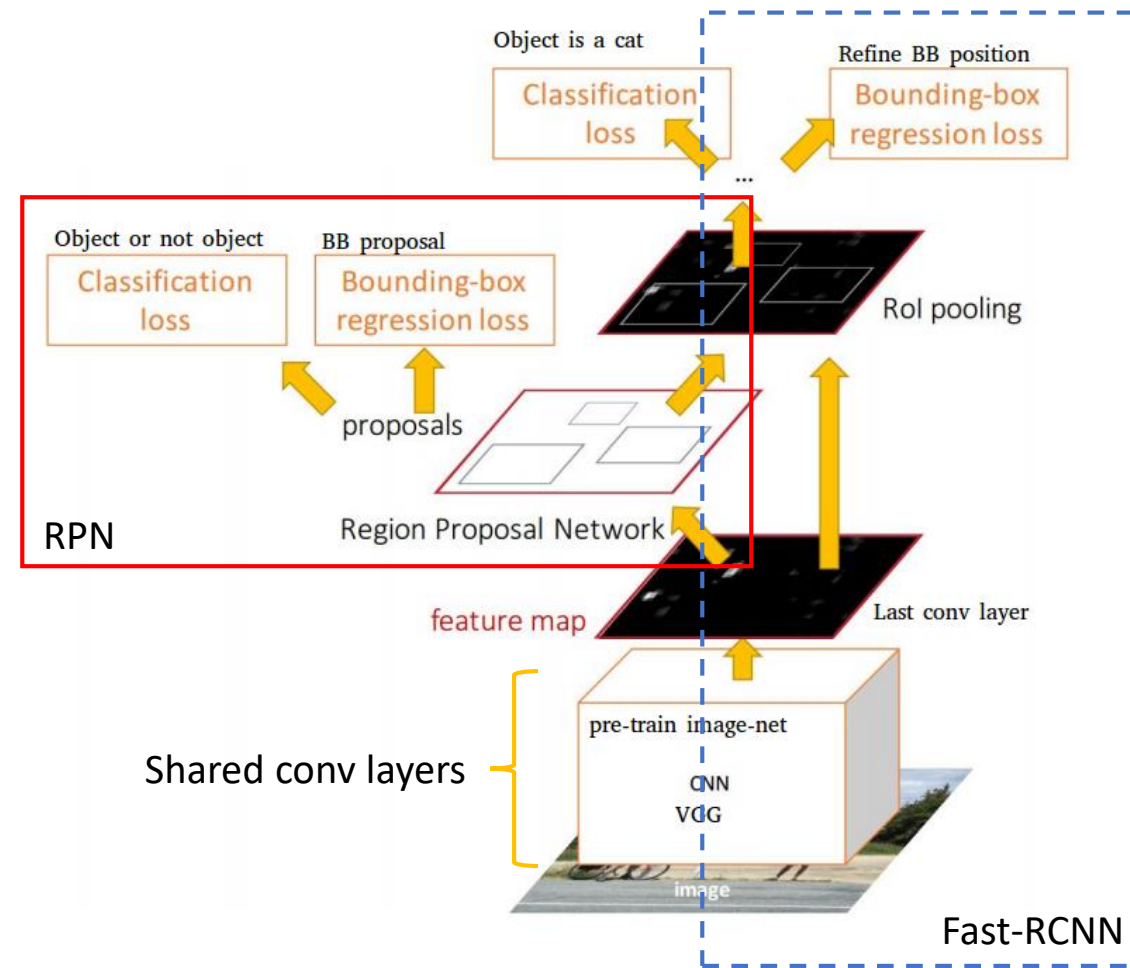
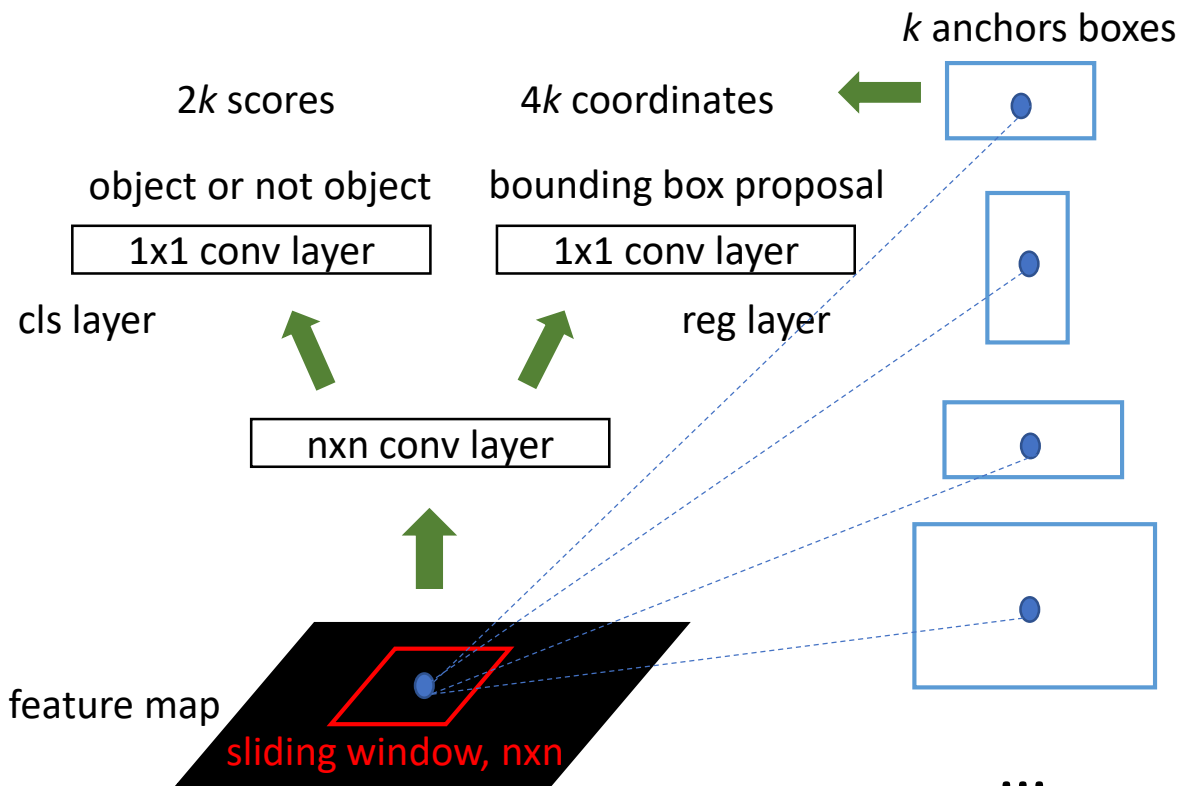
Idea: Integrate the Bounding Box Proposals as part of the CNN predictions

<https://arxiv.org/abs/1506.01497>

Ren et al. NIPS 2015.

Faster-RCNN

Region Proposal Networks:



RCNN

- Simple and scalable.
- improves mAP.
- A multistage pipeline.
- Training is expensive in space and time (features are extracted from each region proposal in each image and written into disk).
- Object detection is slow.



Fast-RCNN

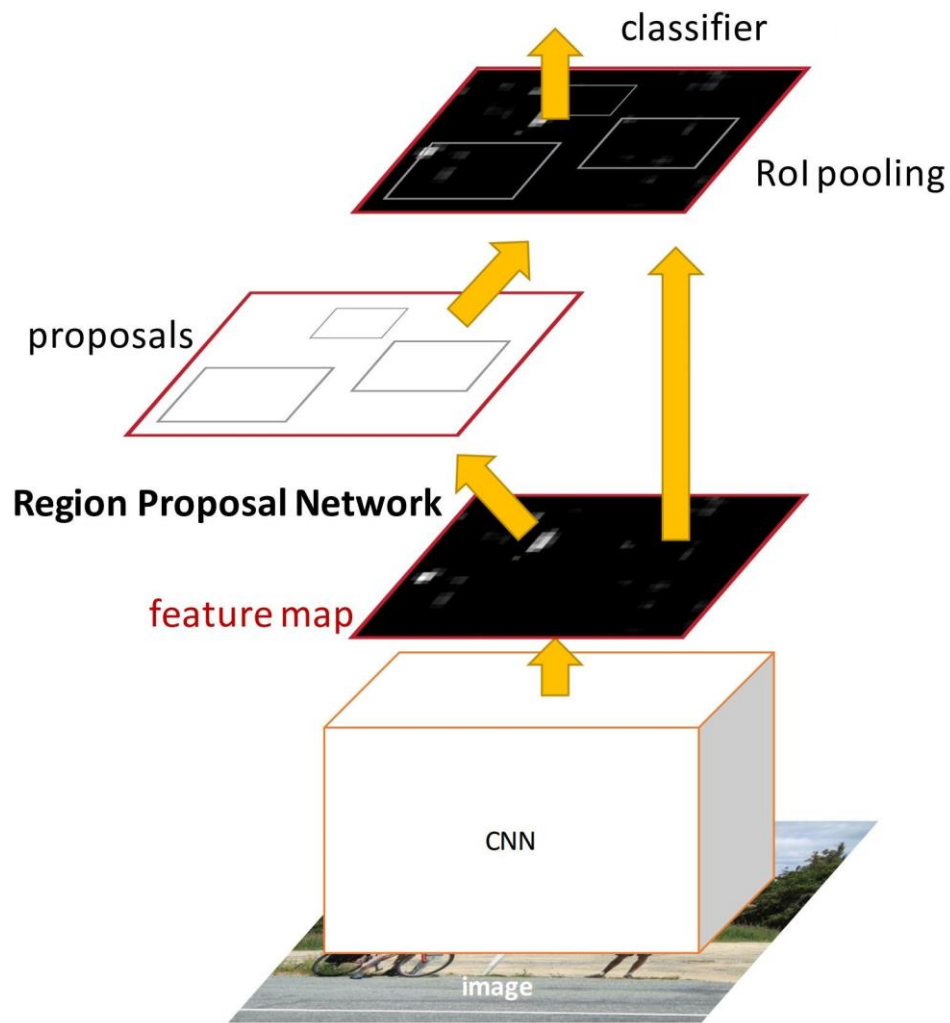
- Higher mAP.
- Single stage, end-to-end training.
- No disk storage is required for feature caching.
- proposals are the computational bottleneck in detection systems.



Faster-RCNN

- compute proposals with a deep convolutional neural network --*Region Proposal Network (RPN)*
- merge RPN and Fast R-CNN into a single network, enabling nearly cost-free region proposals.





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

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

Slide credit: Ross Girshick

Faster R-CNN: Region Proposal Network

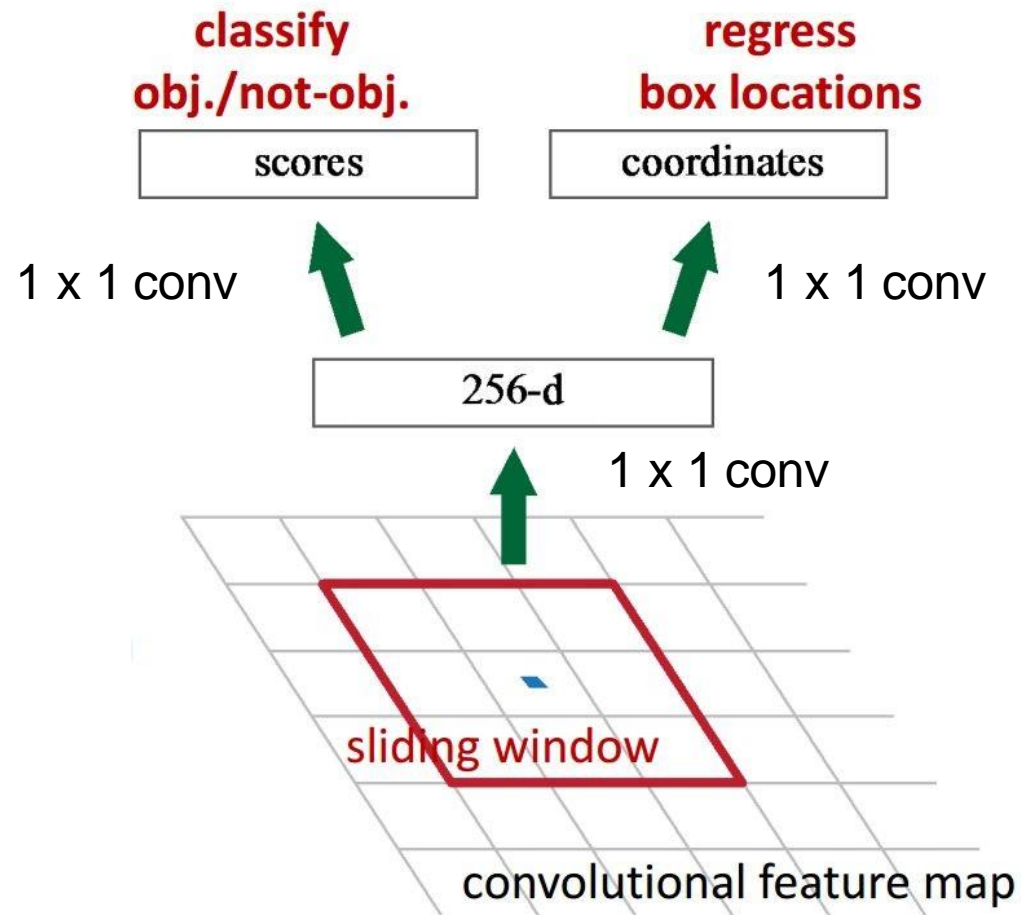
Slide a small window on the feature map

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



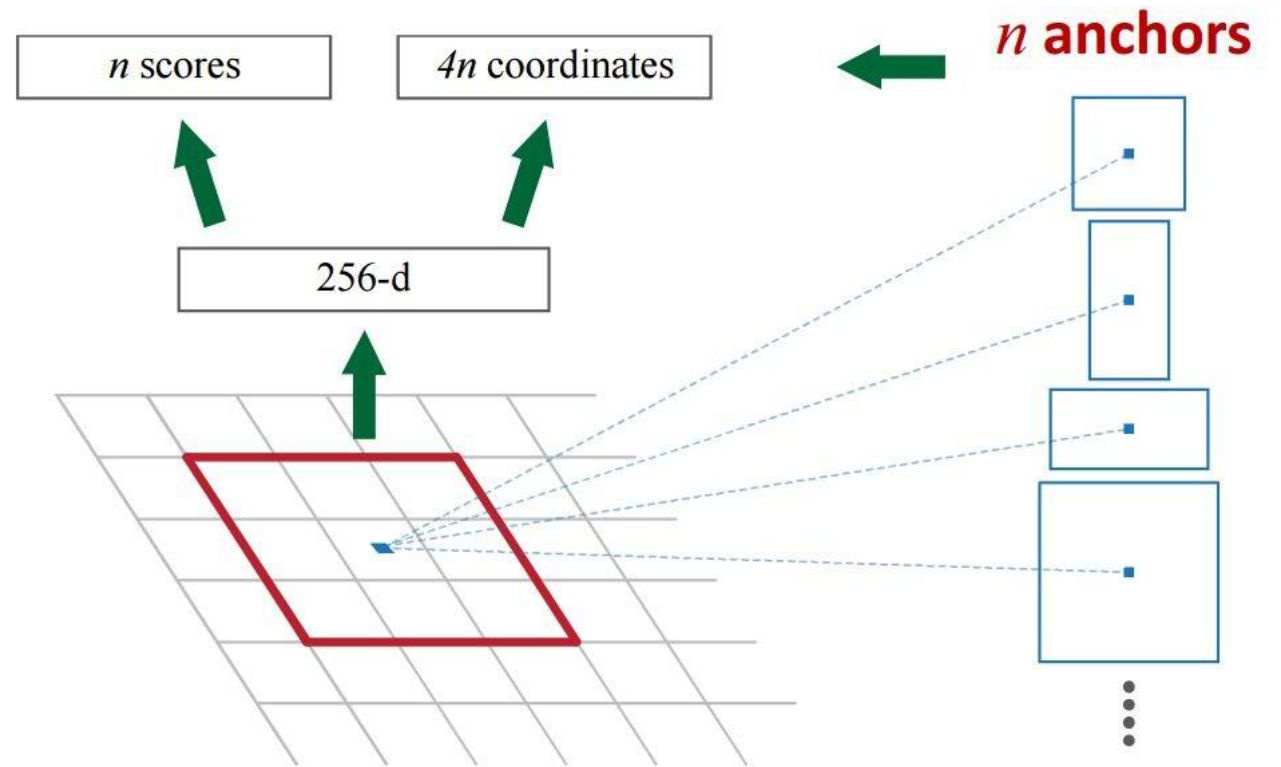
Faster R-CNN: Region Proposal Network

Use **N 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

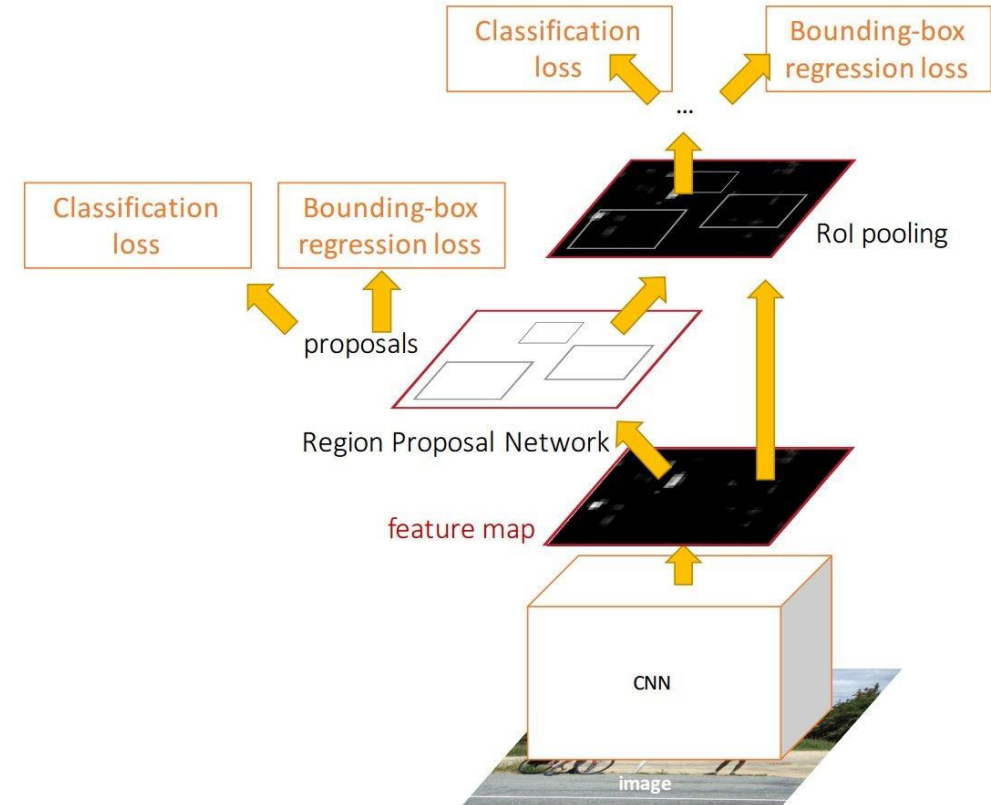
In the paper: Ugly pipeline

- Use alternating optimization to train RPN, then Fast R-CNN with RPN proposals, etc.
- More complex than it has to be

Since publication: Joint training!

One network, four losses

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



Faster R-CNN: 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