CSE578: Computer Vision

Spring 2019:

Deep Learning for Object Detection

Anoop M. Namboodiri

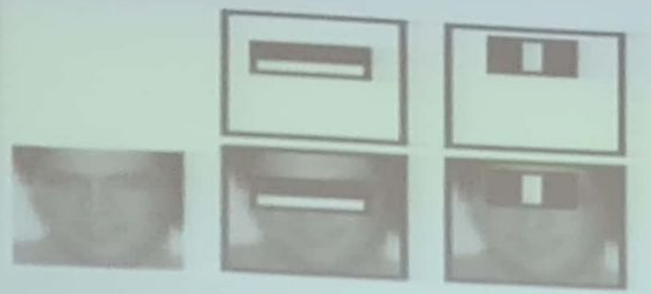
Center for Visual Information Technology

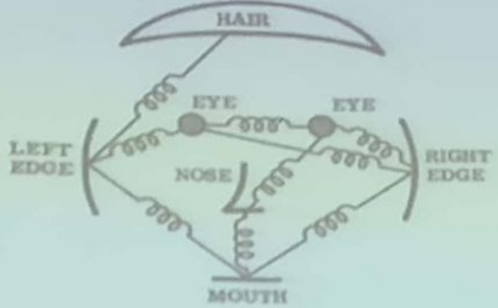
IIIT Hyderabad, INDIA

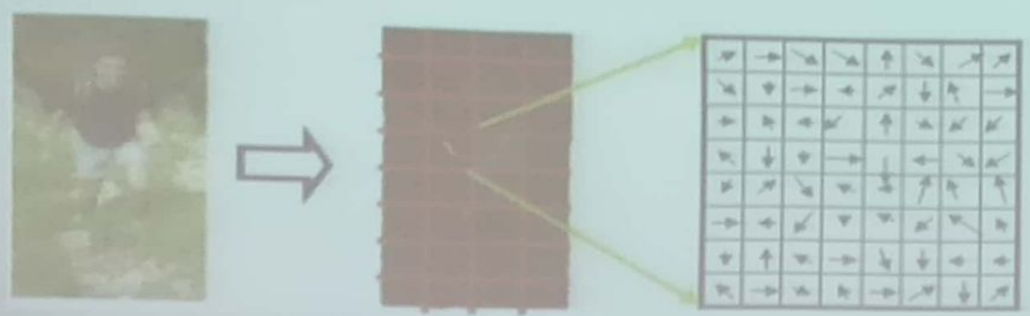
[Slides Credit: Justin Johnson, Andrej Karpathy, Fei-Fei Li]

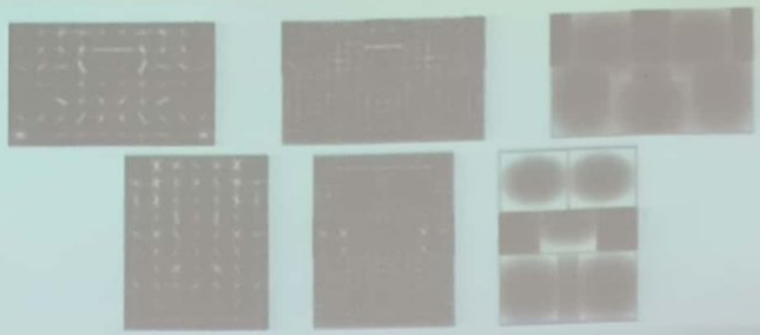
Object Detection: A Recap

- Classify each window in an image
  - Too many windows: Need speed and accuracy





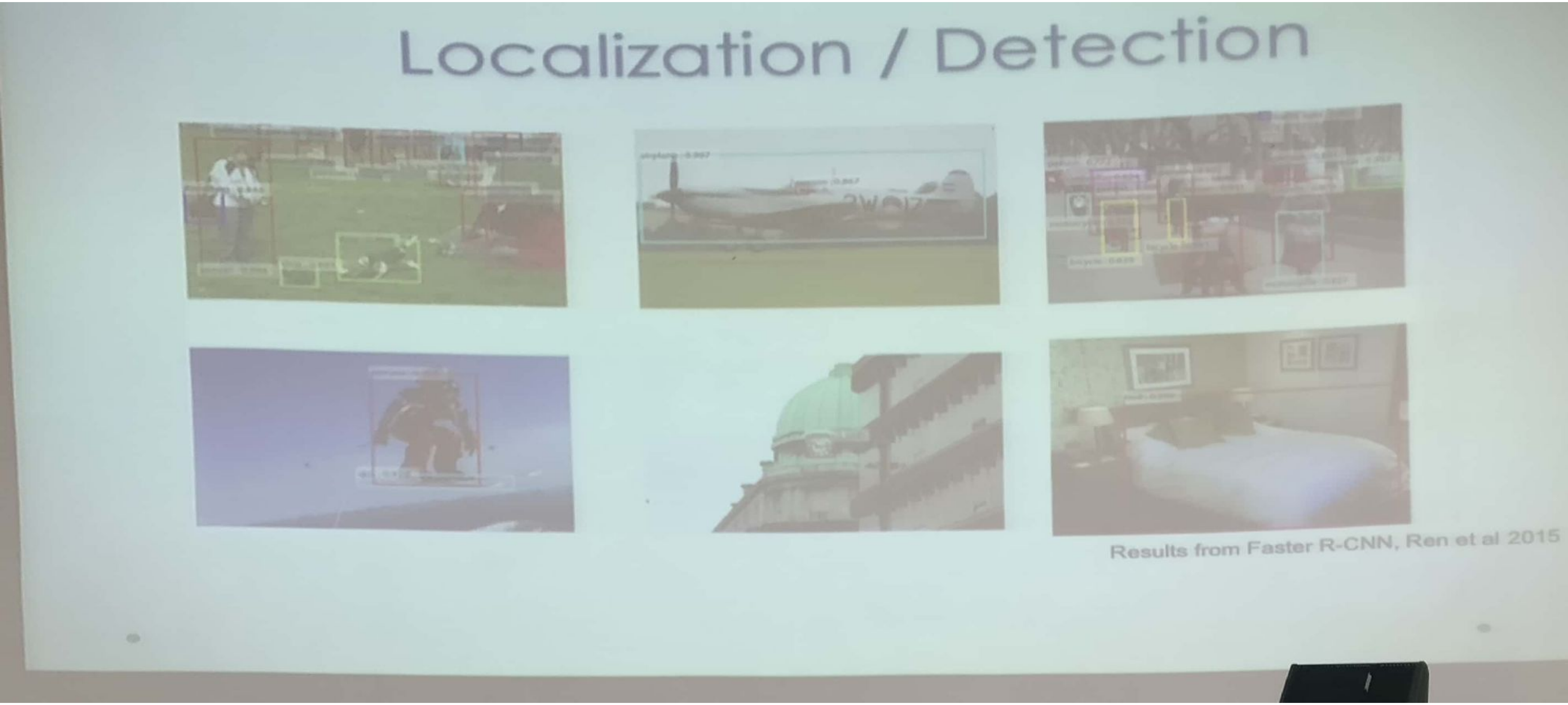




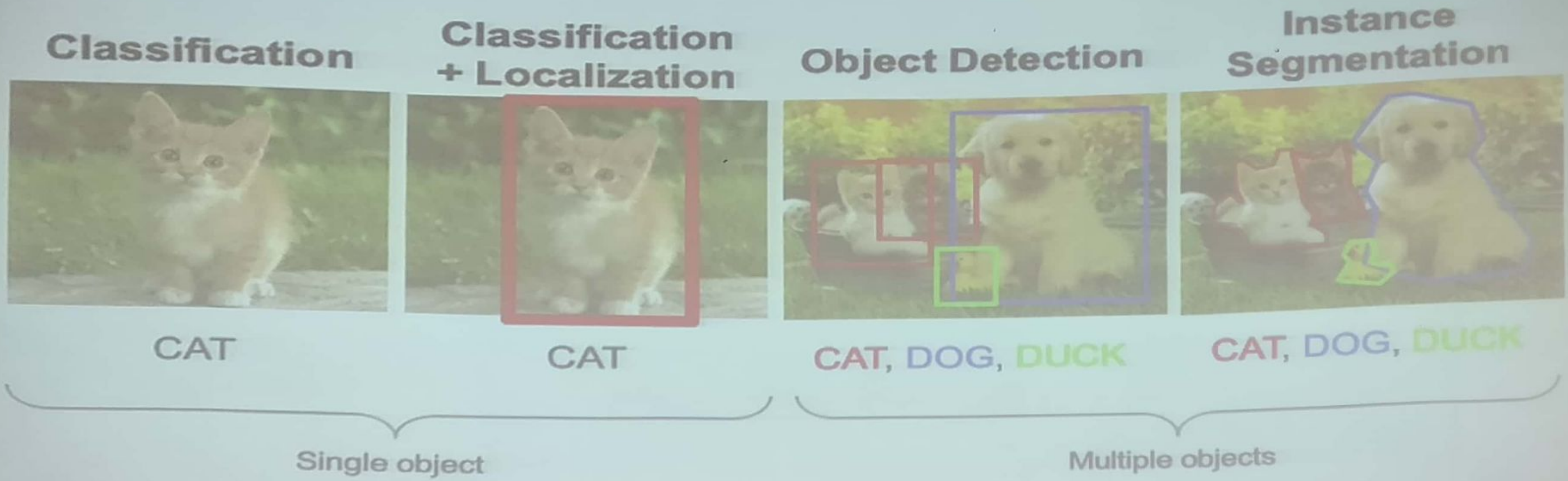
Scanned by CamScanner

Deep Learning based Detection

- Use deep Learnt features for detection
  - Cannot afford to do for every window
- Classify select windows (Region Proposals)
  - RCNN, Fast RCNN, Faster RCNN, Mask RCNN
- Directly predict bounding boxes
  - YOLO V1, V2, V3
- Deeplab V3, Xception
- Dialated Residual Networks


Localization / Detection
Results from Faster R-CNN, Ren et al 2015

Computer Vision Tasks



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

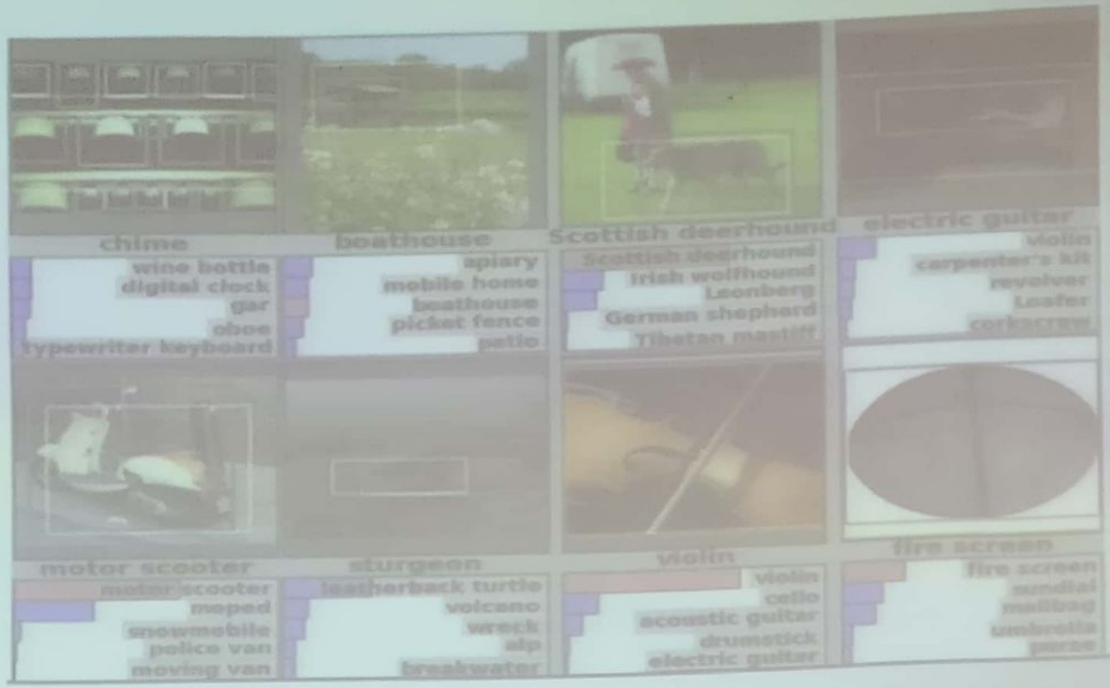
1000 classes (same as classification)

Each image has 1 class, at least one bounding box

~800 training images per class

Algorithm produces 5 (class, box) guesses

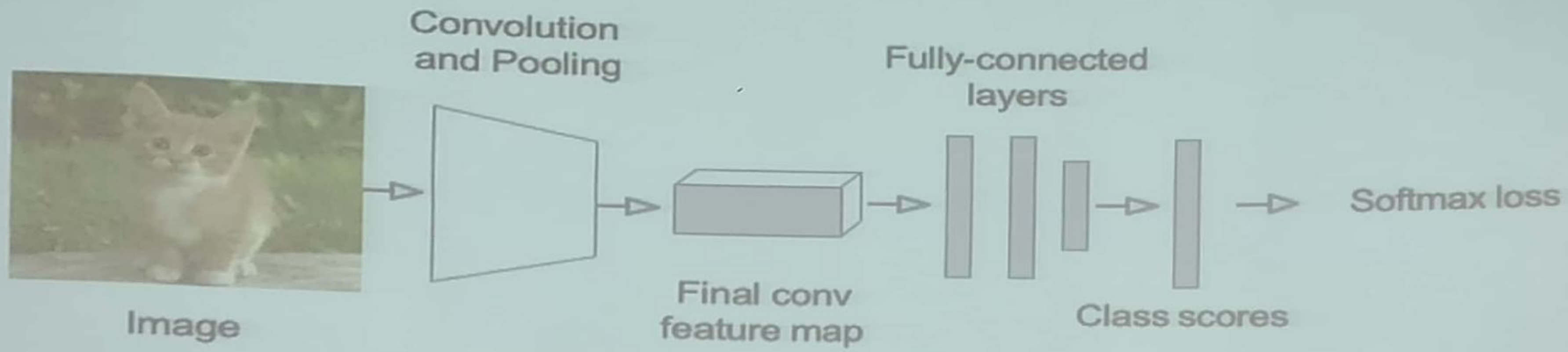
Example is correct if at least one one guess has correct class AND bounding box at least 0.5 intersection over union (IoU)



Krizhevsky et. al. 2012

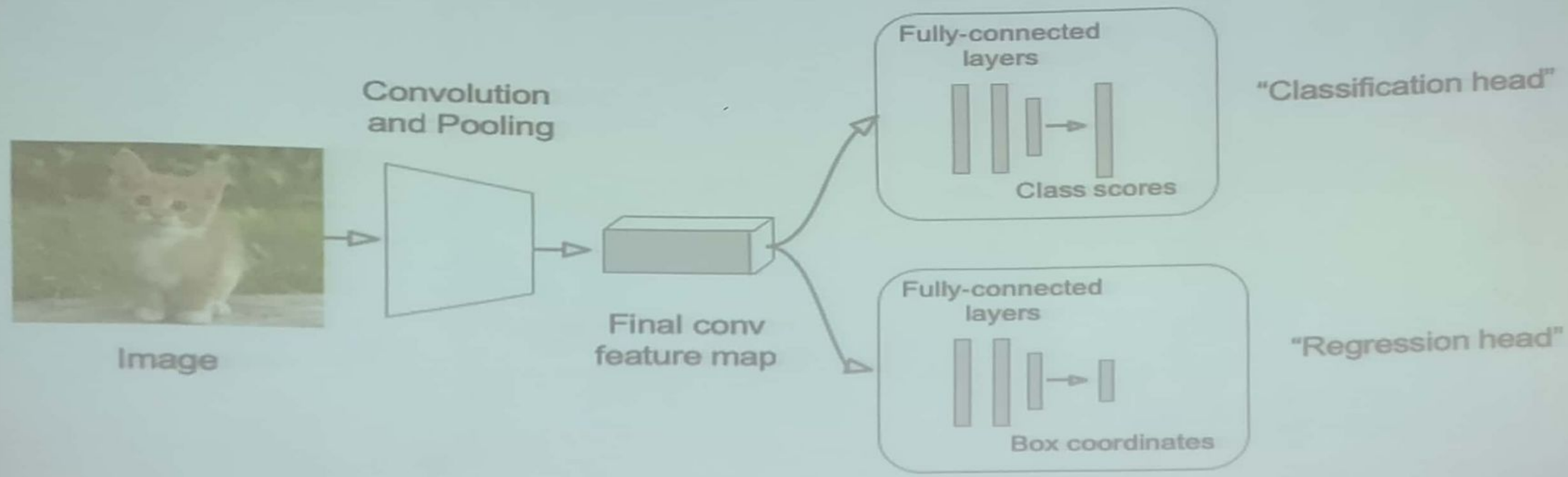
Simple Recipe for Classification + Localization

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



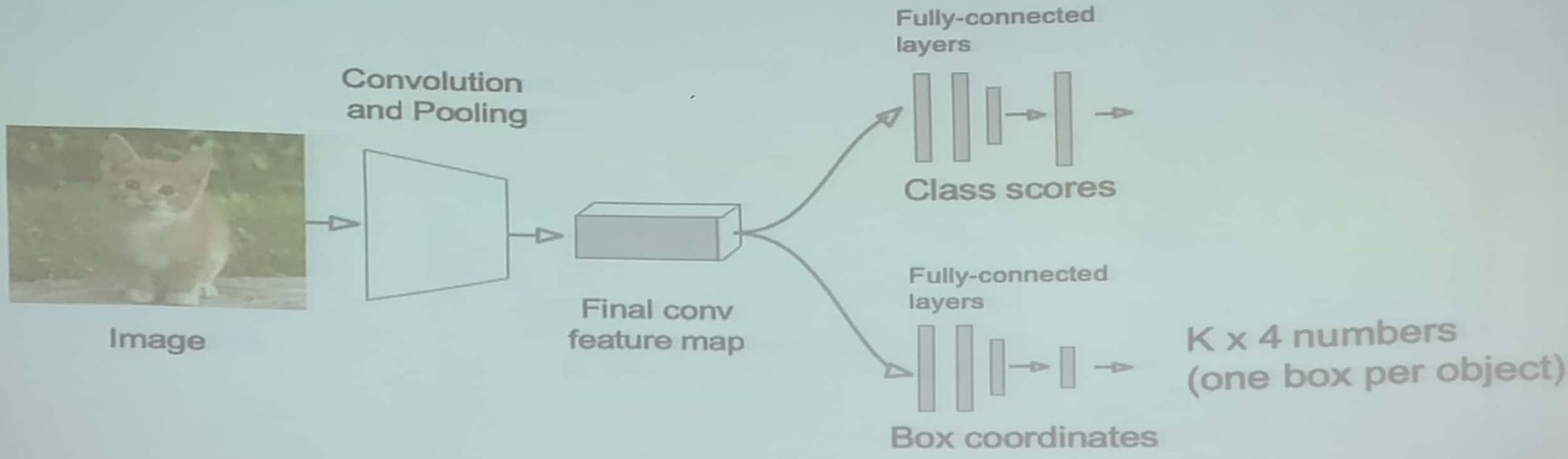
Simple Recipe for Classification + Localization

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



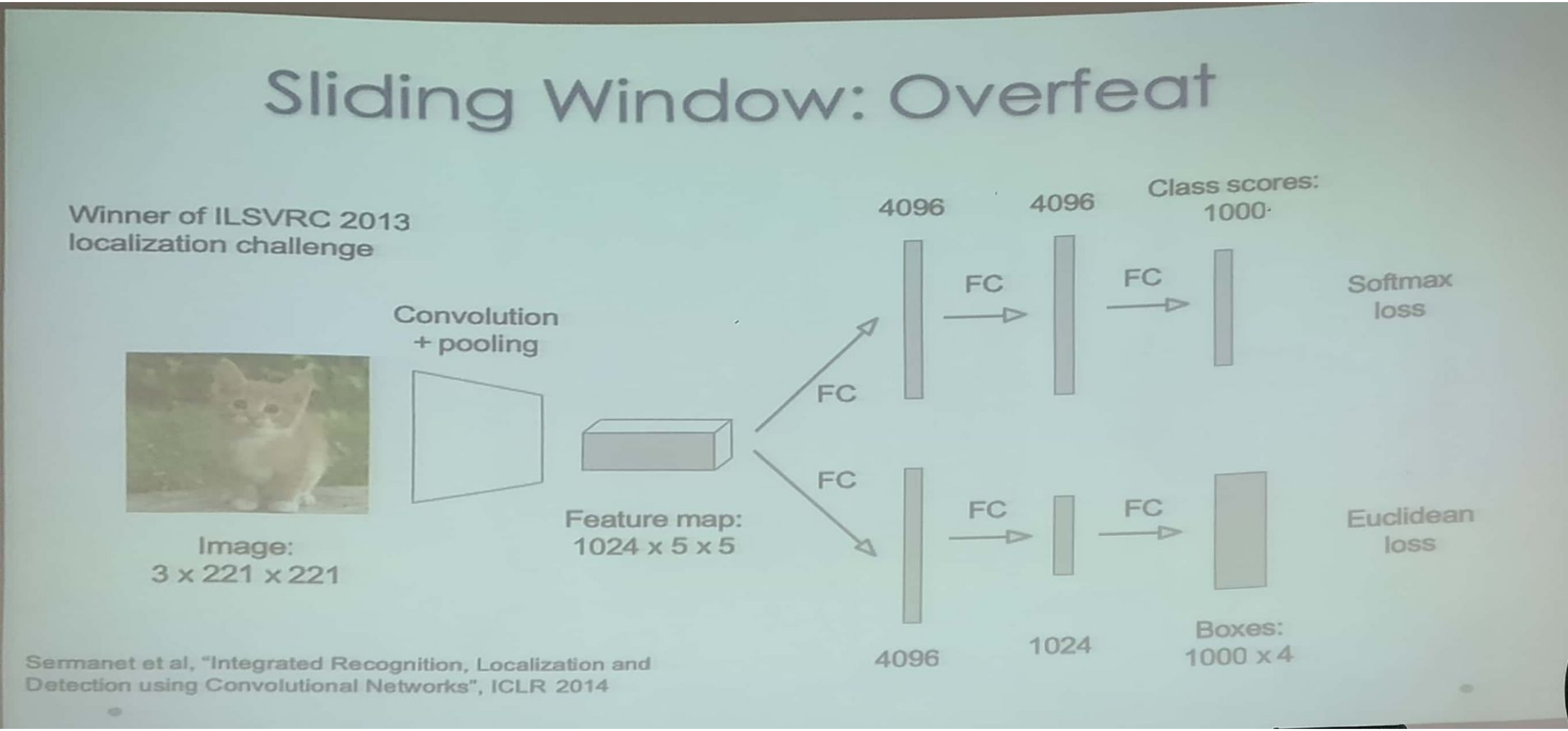
Localizing Multiple Objects

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

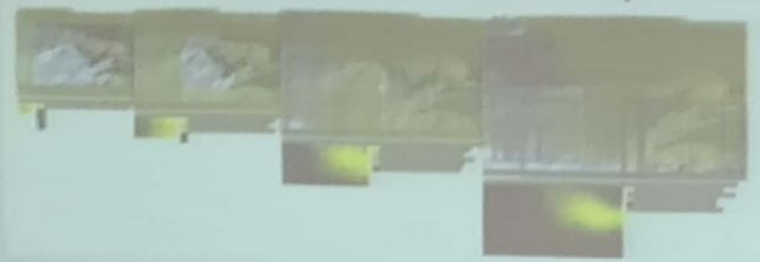
- Run classification + regression network at multiple locations on a high-resolution image
- Convert fully-connected layers into convolutional layers for efficient computation
- Combine classifier and regressor predictions across all scales for final prediction


Sliding Window: Overfeat
Winner of ILSVRC 2013
localization challenge
Convolution
+ pooling
Image:
3 x 221 x 221
Feature map:
1024 x 5 x 5
FC
FC
4096
FC
4096
FC
Class scores:
1000
Softmax
loss
FC
FC
4096
1024
Boxes:
1000 x 4
Euclidean
loss
Sermanet et al, "Integrated Recognition, Localization and
Detection using Convolutional Networks", ICLR 2014

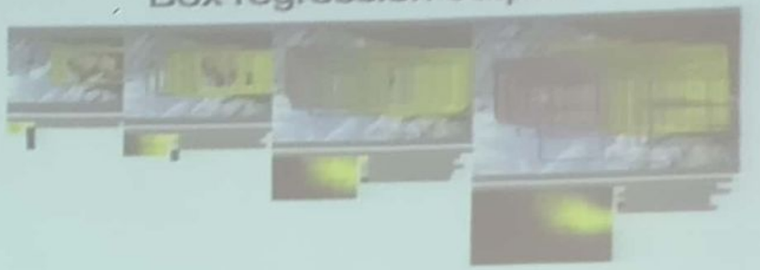
Sliding Window: Overfeat

In practice use many sliding window
locations and multiple scales

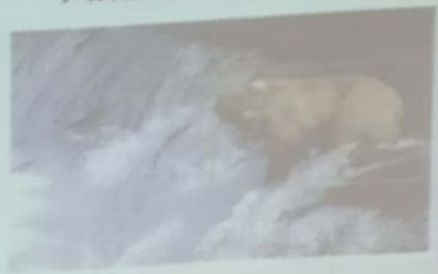
Window positions + score maps



Box regression outputs



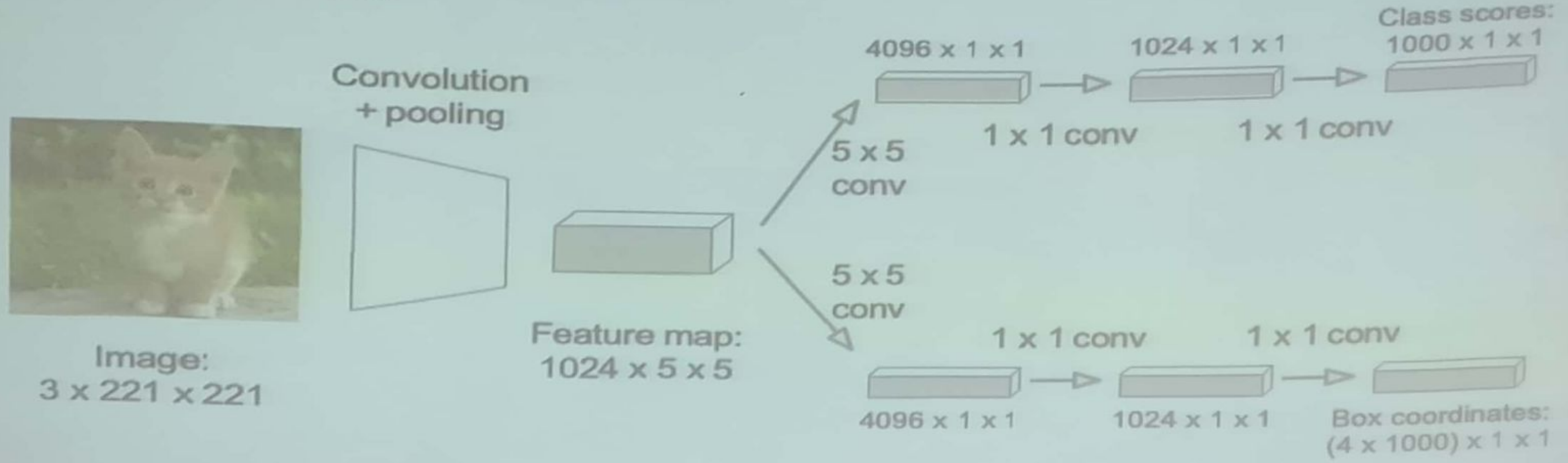
Final Predictions



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

Efficient Sliding Window: Overfeat

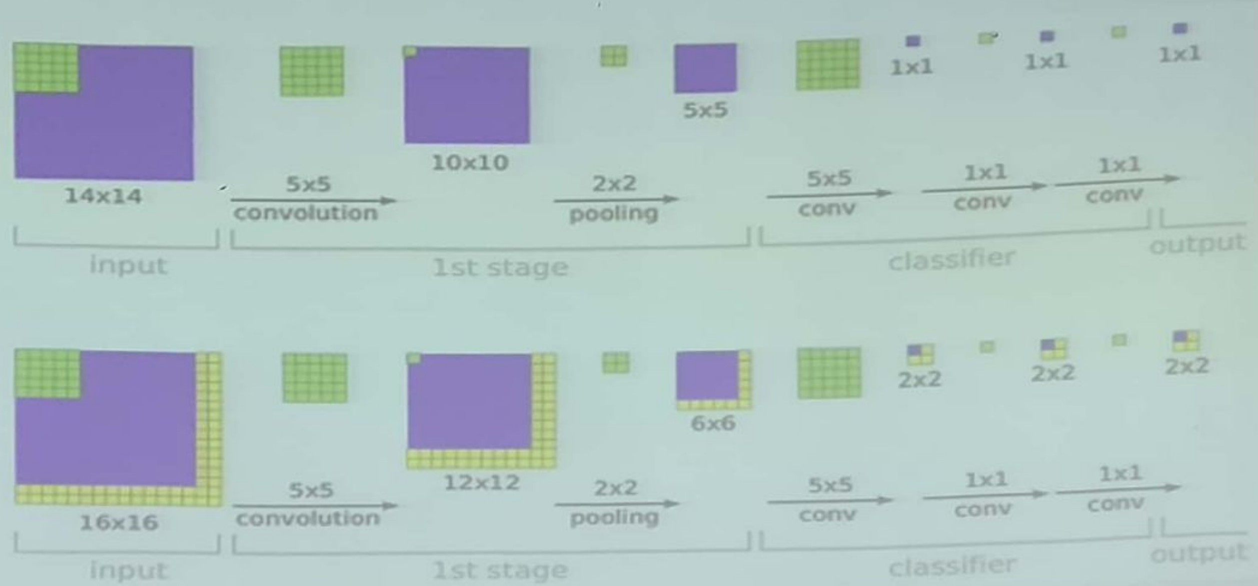
Efficient sliding window by converting fully-connected layers into convolutions



Efficient Sliding Window: Overfeat

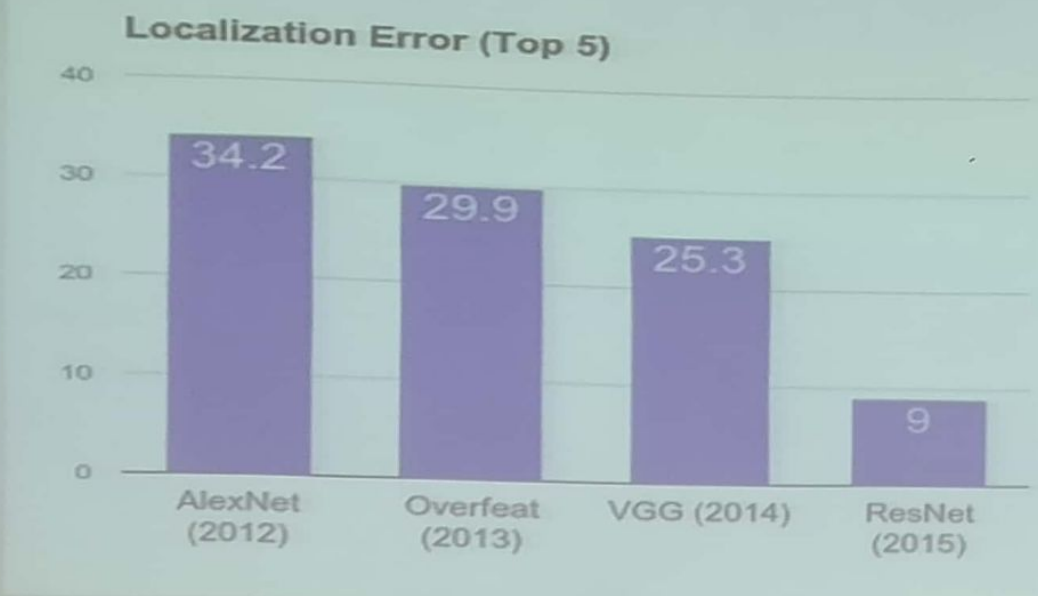
Training time: Small image, 1 x 1 classifier output

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



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

ImageNet Classification + Localization



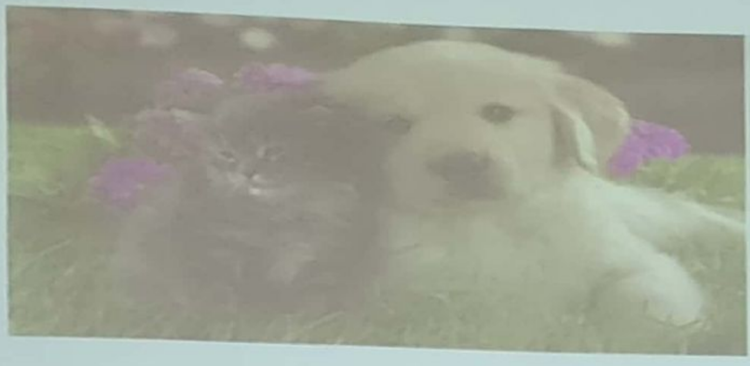
AlexNet: Localization method not published

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

Detection as Regression?



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

= 8 numbers

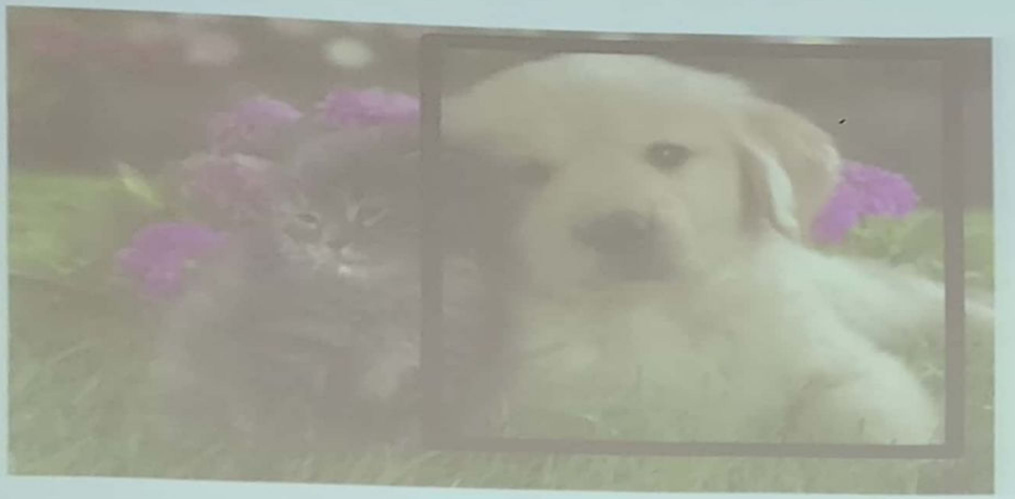
Detection as Classification

CAT? NO

DOG? YES!

Scanned by CamScanner

Detection as Classification



CAT? NO

DOG? YES!

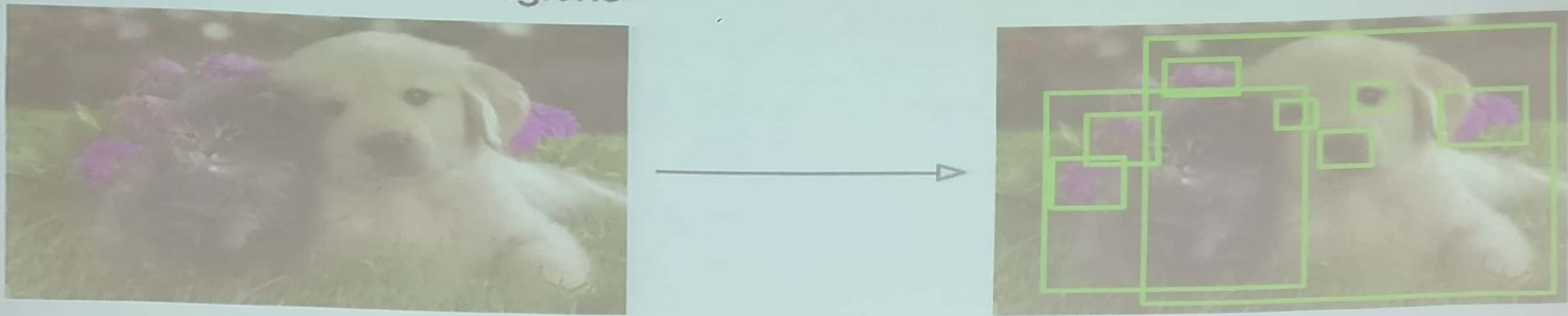
Detection as Classification

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

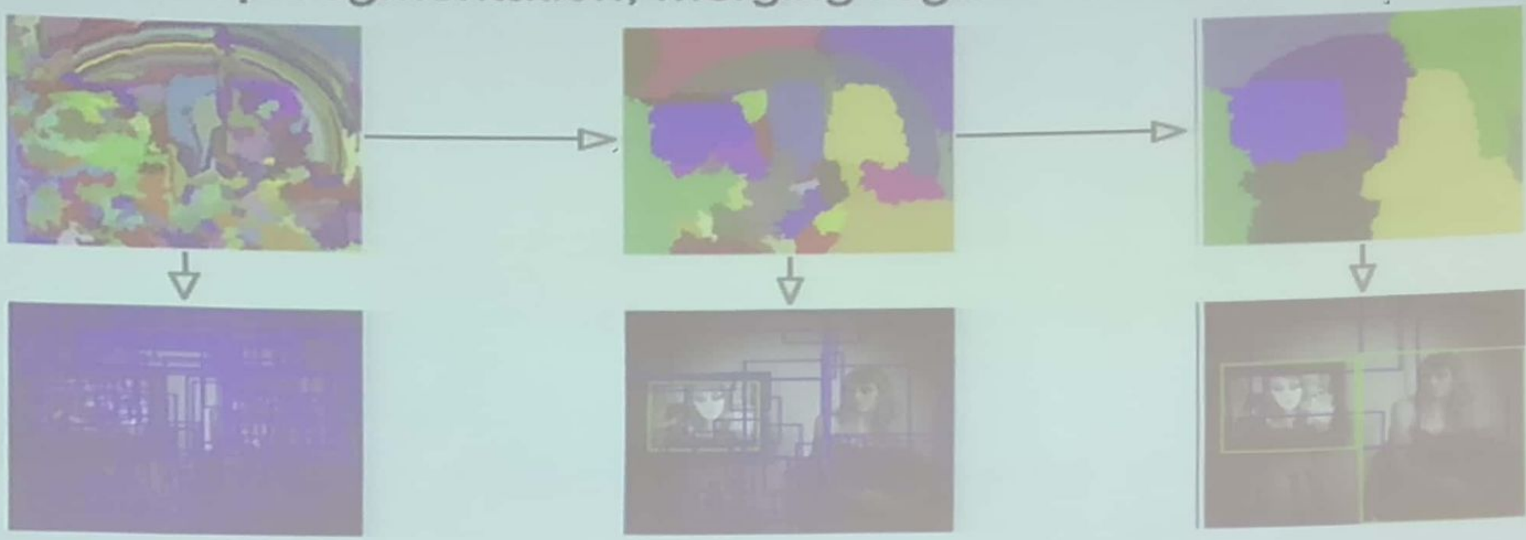
- Find “blobby” image regions that are likely to contain objects
- “Class-agnostic” object detector
- Look for “blob-like” regions



Region Proposals: Selective Search

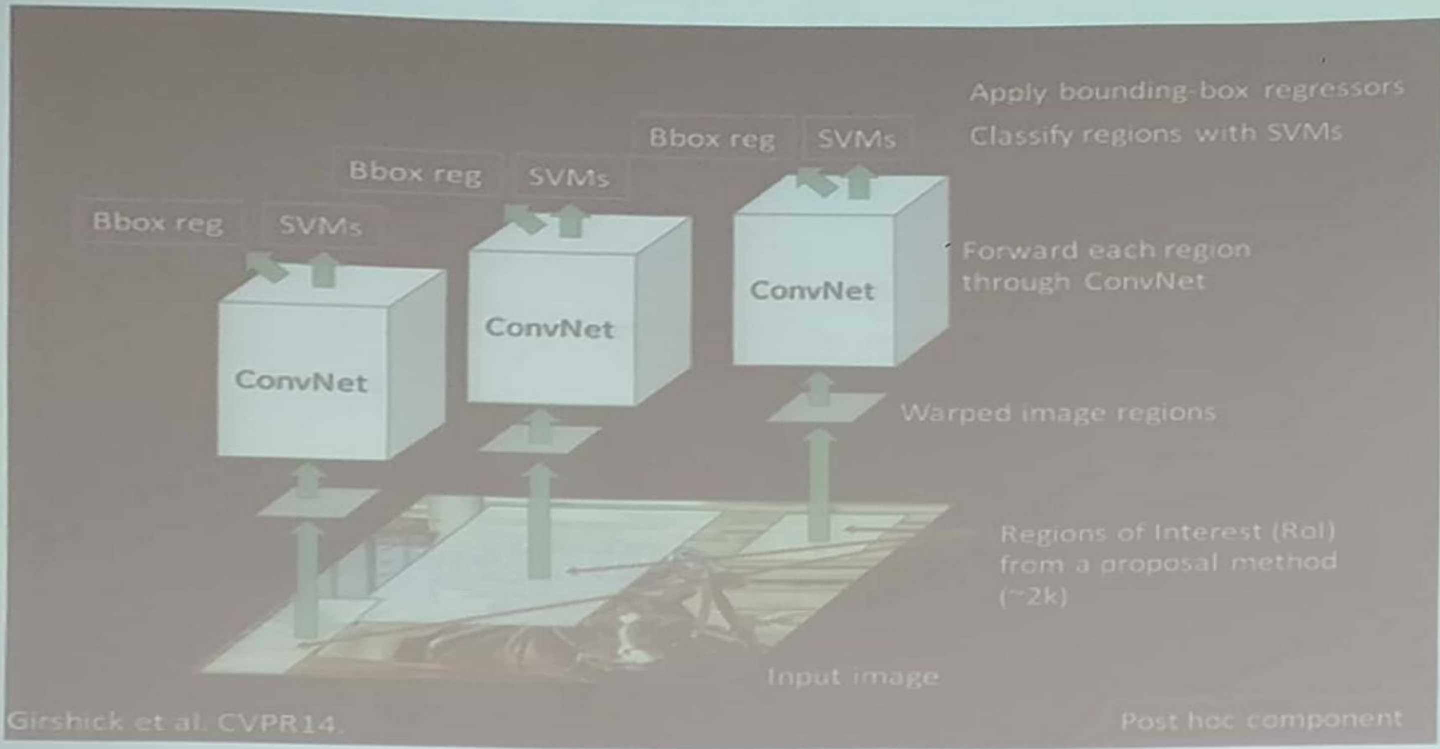
Bottom-up segmentation, merging regions at multiple scales

Convert regions to boxes



Uijlings et al, "Selective Search for Object Recognition", IJCV 2013

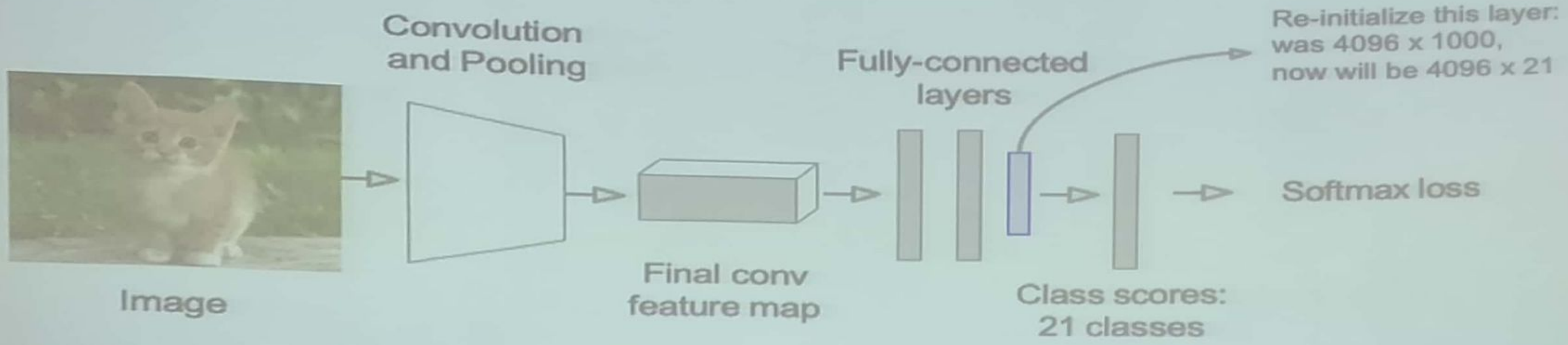
Putting it Together: R-CNN



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

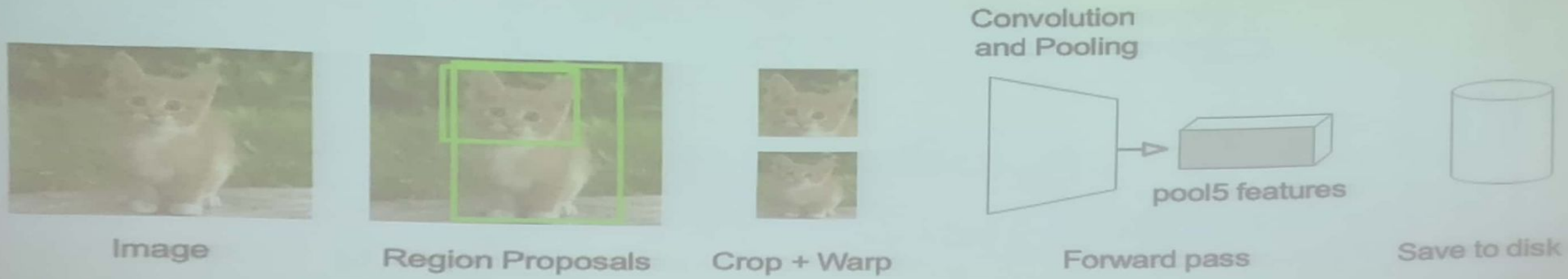
R-CNN Training

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
- In hard drive? Features are ~200GB for PASCAL dataset!



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

Training image regions

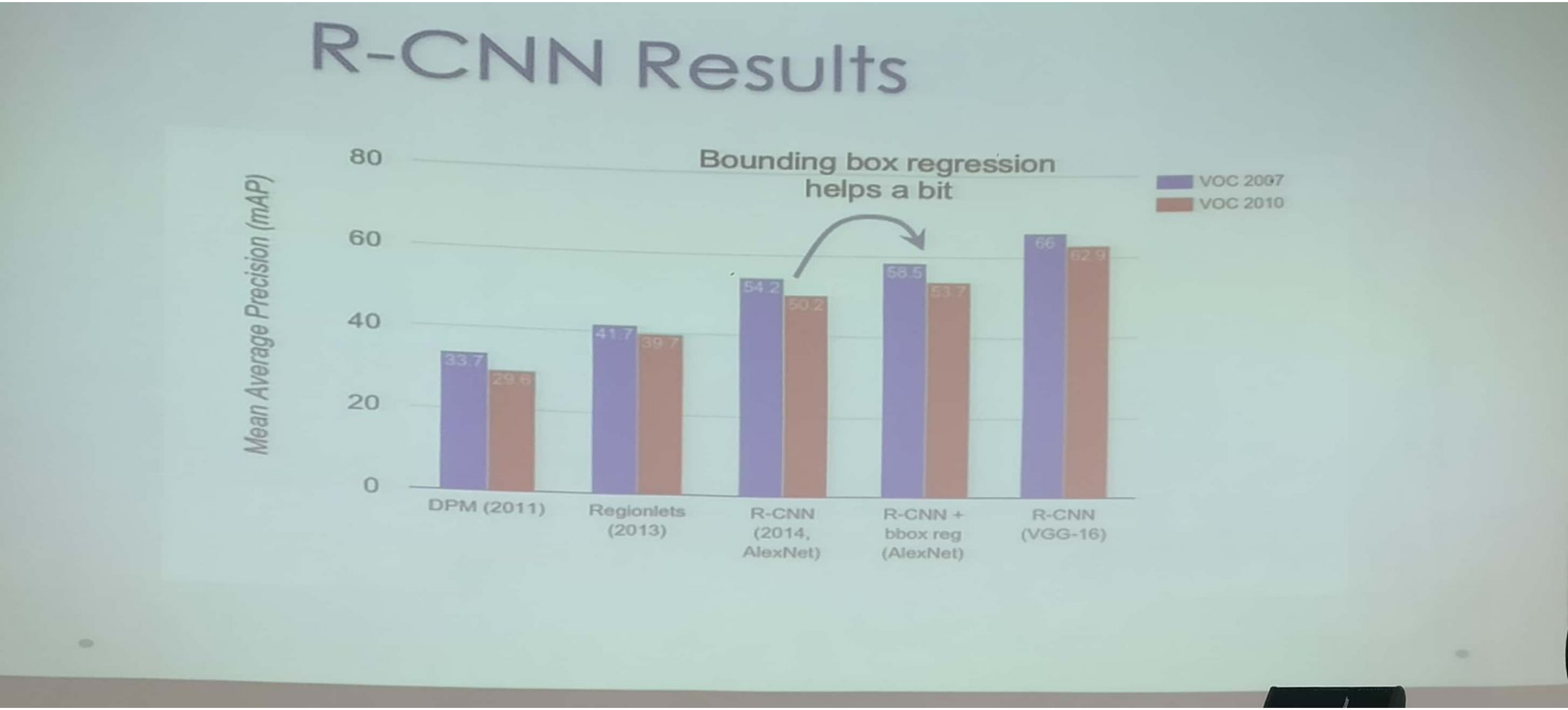
Cached region features

Regression targets
(dx, dy, dw, dh)
Normalized coordinates

(0, 0, 0, 0) Proposal is good	(.25, 0, 0, 0) Proposal too far to left	(0, 0, -0.125, 0) Proposal too wide

Object Detection: Evaluation

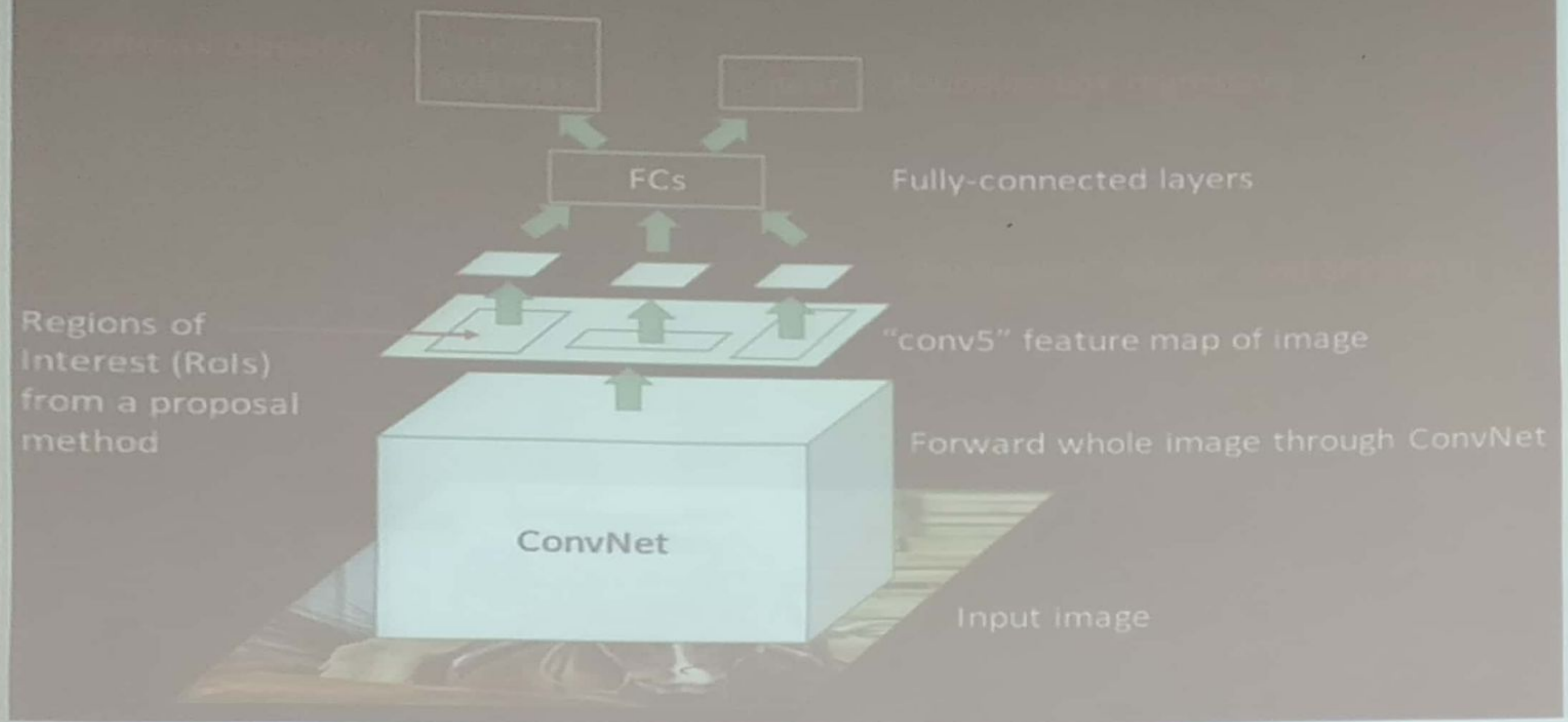
- Popular metric: "mean average precision" (mAP)
- Precision: # Correct detections / # Detections
- Combine all detections from all test images to draw a P-R curve for each class; AP is the area under the curve
- Compute the average precision (AP) separately for each class. Its mean over classes gives mAP
- 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)
- mAP is a number from 0 to 100; high is good


R-CNN Results
Bounding box regression
helps a bit
VOC 2007
VOC 2010
Mean Average Precision (mAP)
80
60
40
20
0
33.7
29.6
41.7
39.7
54.2
50.2
58.5
53.7
66
62.9
DPM (2011)
Regionlets
(2013)
R-CNN
(2014,
AlexNet)
R-CNN +
bbox reg
(AlexNet)
R-CNN
(VGG-16)

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 are not updated in response to SVMs and regressors
3. Complex multistage training pipeline

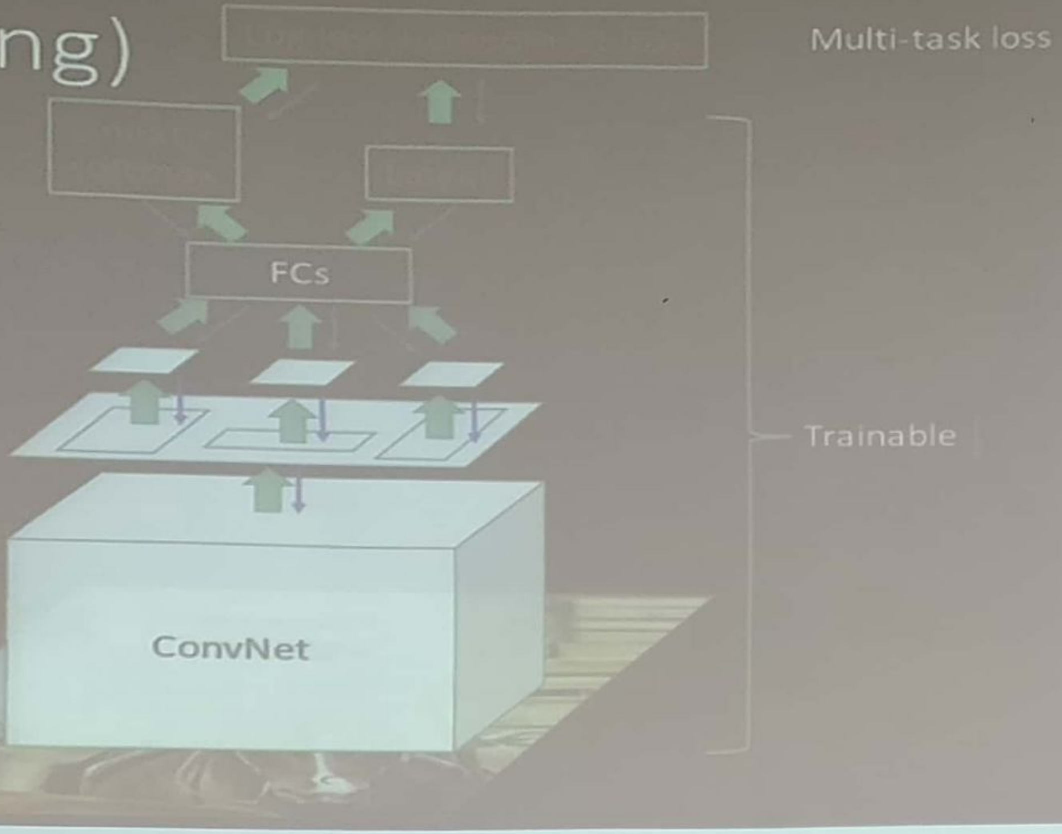
Fast R-CNN (test time)



Girschick, "Fast R-CNN", ICCV 2015

Slide credit: Ross Girschick

Fast R-CNN (training)



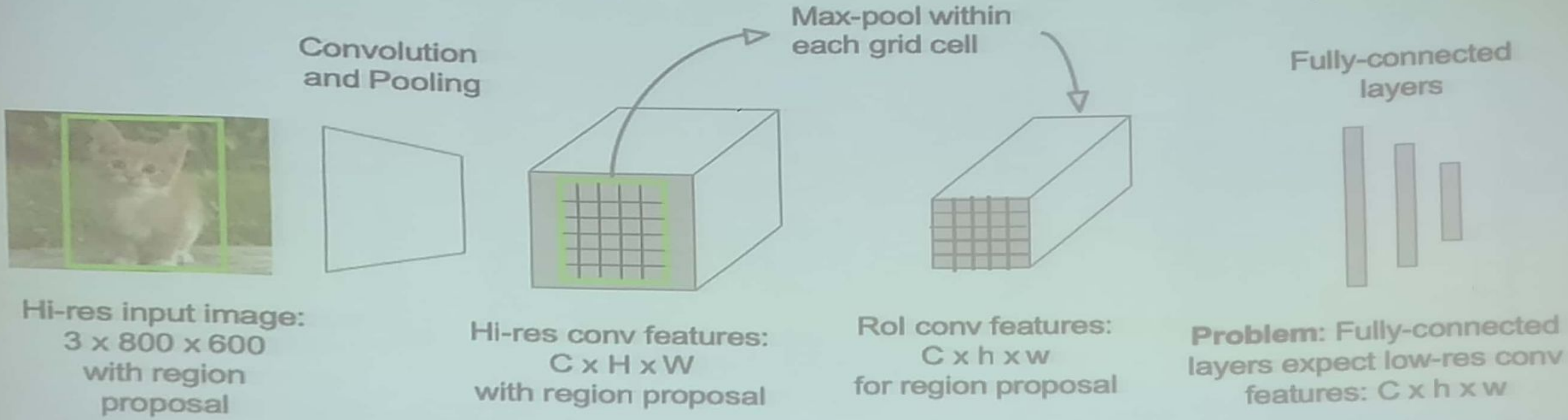
R-CNN Problem #2:
Post-hoc training: CNN not updated in response to final classifiers and regressors

R-CNN Problem #3:
Complex training pipeline

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

Slide credit: Ross Girschick

Fast R-CNN: Region of Interest Pooling

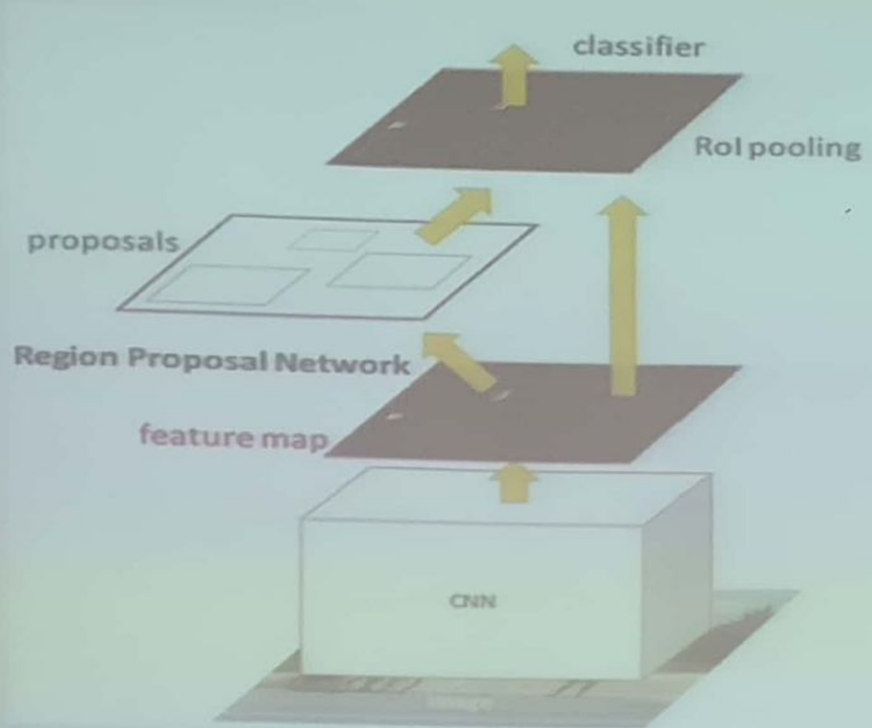


Fast R-CNN Results

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

Using VGG-16 CNN on Pascal VOC 2007 dataset

Faster R-CNN:



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 Girschick

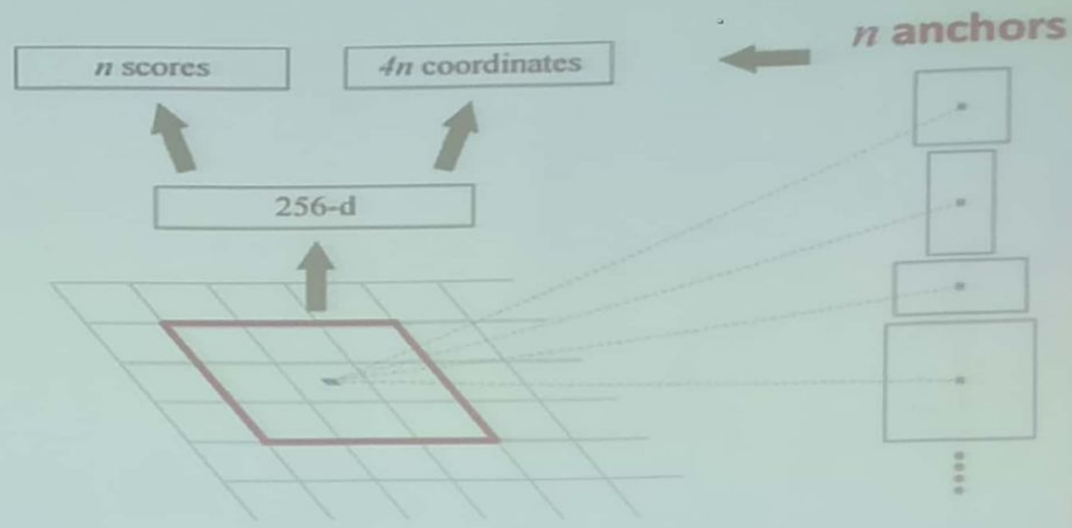
Faster R-CNN: Region Proposal Network

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



Scanned by CamScanner