CS 4803 / 7643: Deep Learning

Topics:

- (Finish) Modern CNNs
- Detection & Segmentation architectures

Zsolt Kira

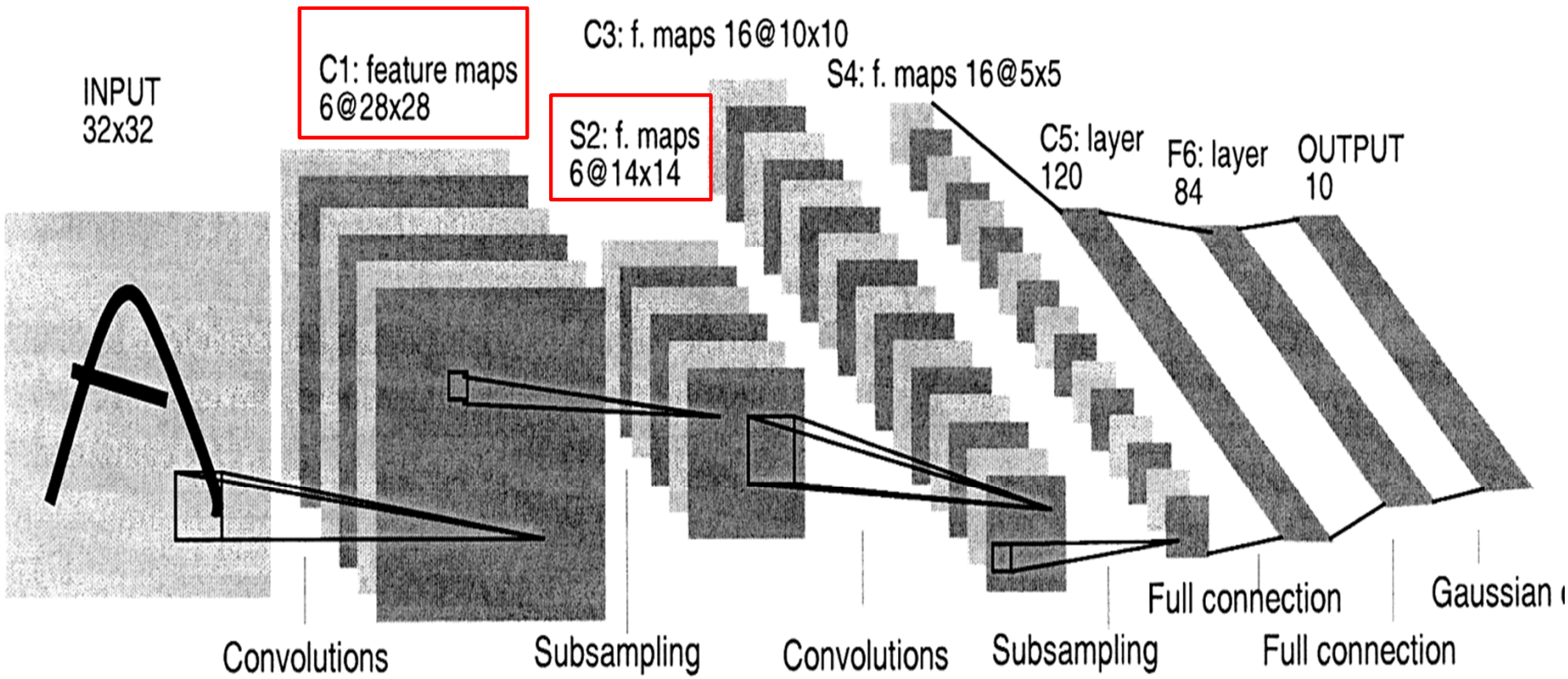
Georgia Tech

Administrivia

- PS2/HW2 out today!
  - Due: 03/03, 11:55pm
  - Less involved than PS1/HW1

- Please send project proposal!
  - Get feedback before next week

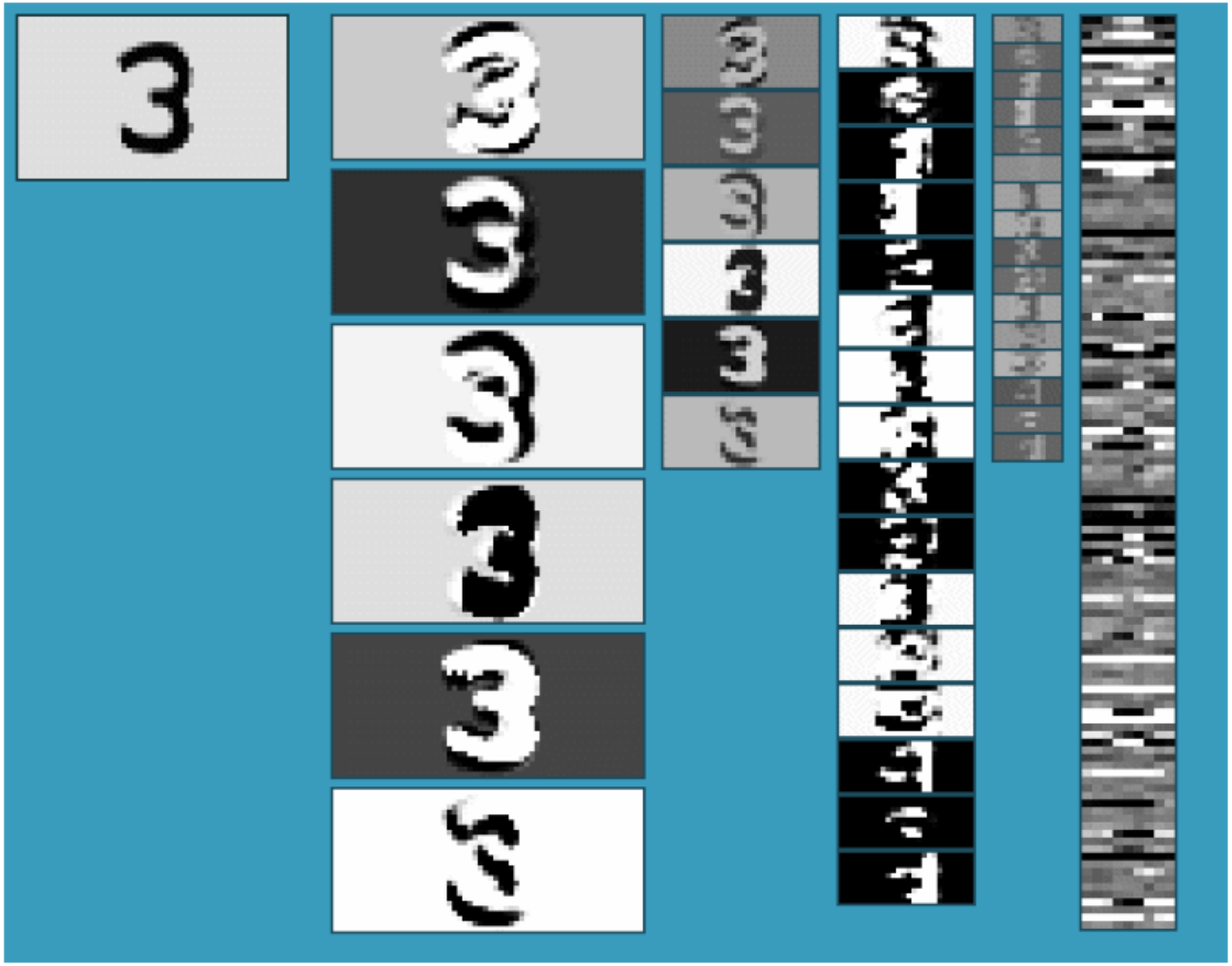
Last Time


INPUT
32x32
C1: feature maps
6@28x28
S2: f. maps
6@14x14
C3: f. maps 16@10x10
S4: f. maps 16@5x5
C5: layer
120
F6: layer
84
OUTPUT
10
Convolutions
Subsampling
Convolutions
Subsampling
Full connection
Full connection
Gaussian

Handwriting Recognition Example

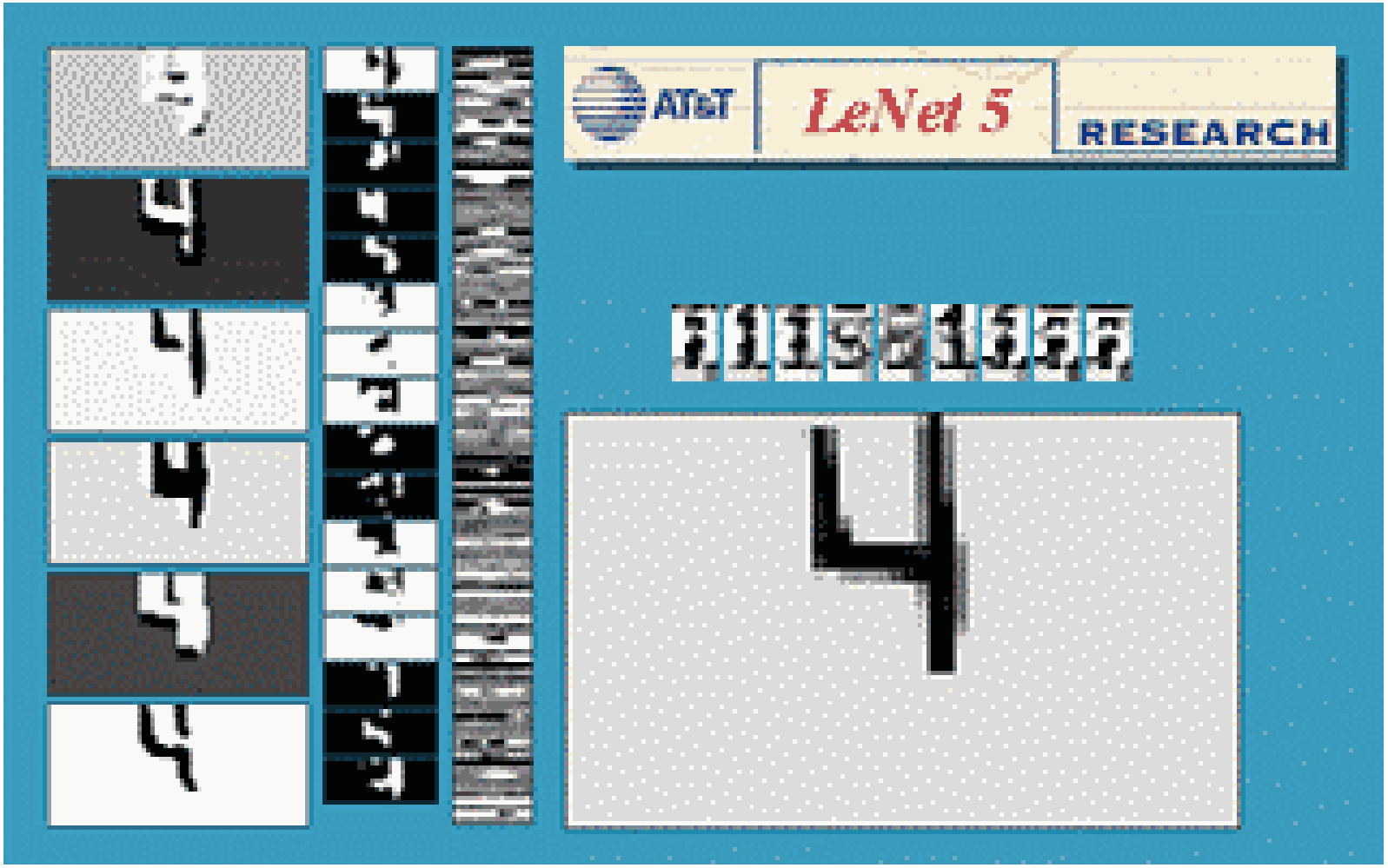
- filters → tanh → average-tanh → filters → tanh → average-tanh → filters → tanh

Curved manifold

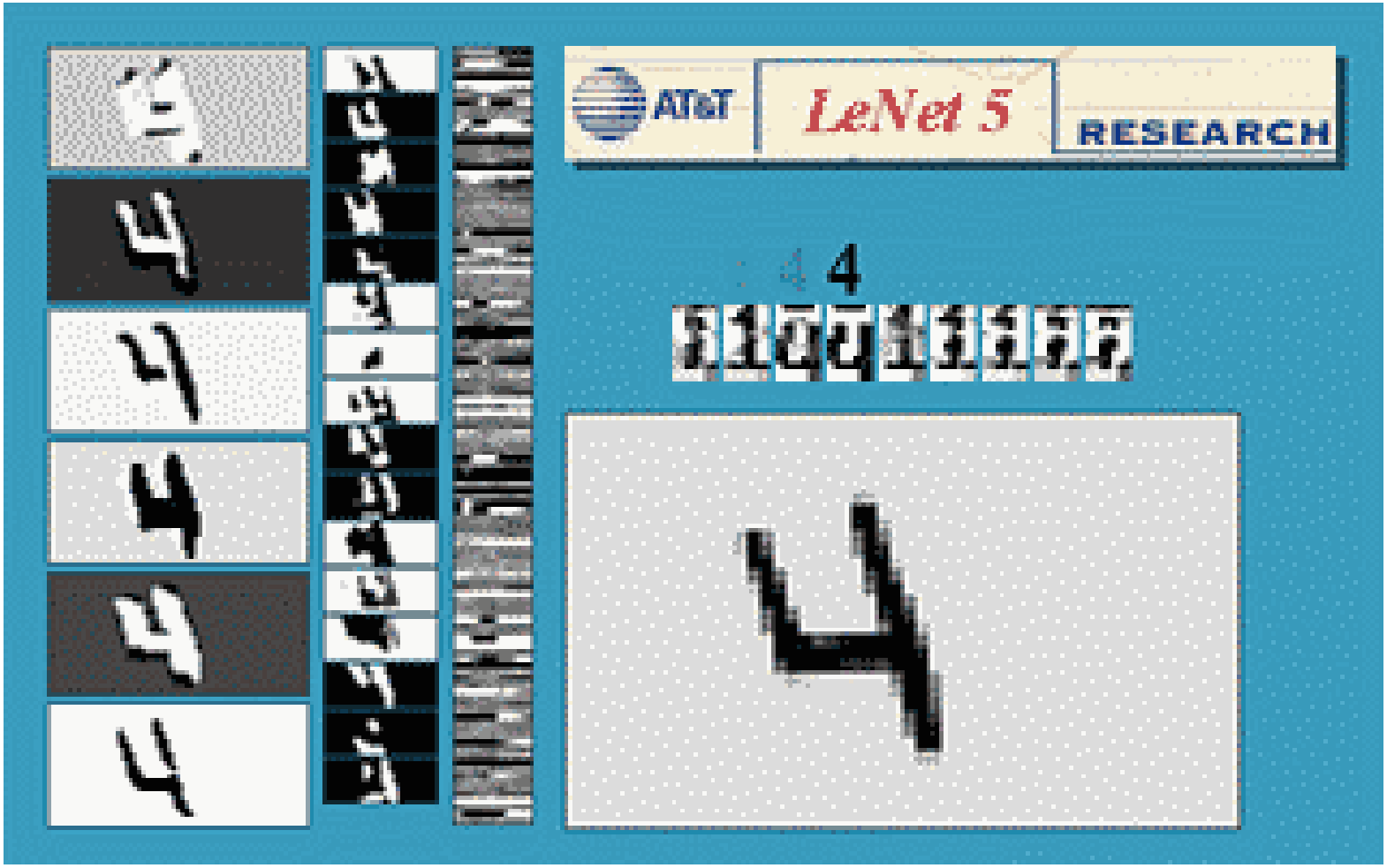


Flatter manifold

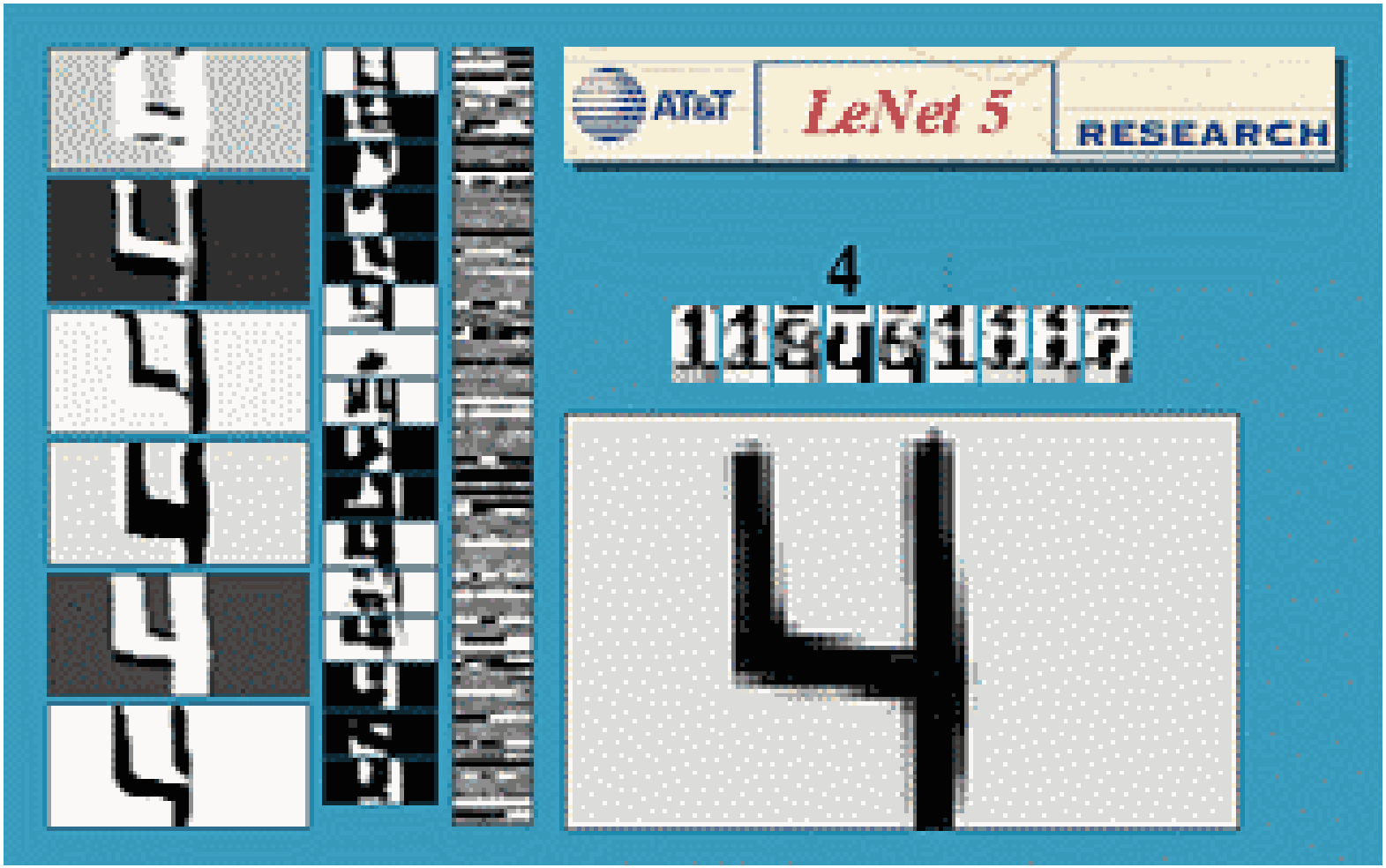
Translation Invariance



Some Rotation Invariance



Some Scale Invariance

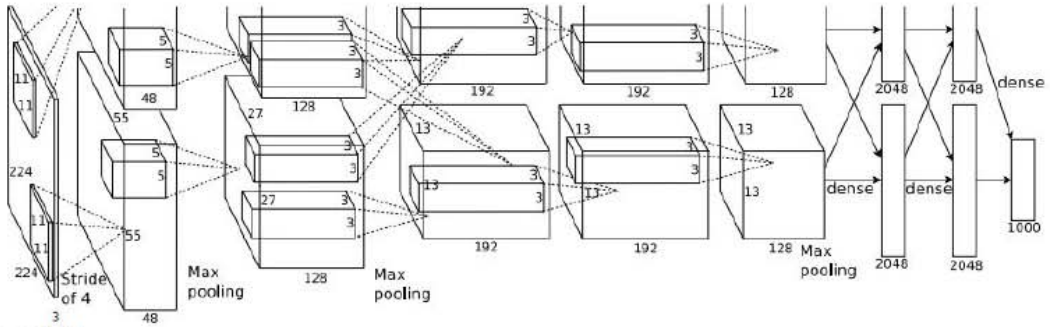


Case Studies

- There are several generations of ConvNets
  - 2012 – 2014: AlexNet, ZNet, VGGNet
    - Conv-Relu, Pooling, Fully connected, Softmax
    - Deeper ones (VGGNet) tend to do better
  - 2014
    - Fully-convolutional networks for **semantic segmentation**
    - Matrix outputs rather than just one probability distribution
  - 2014-2016
    - Fully-convolutional networks for **classification**
    - Less parameters, faster than comparable Gen1 networks
    - GoogleNet, ResNet
  - 2014-2016
    - Detection layers (proposals)
    - Caption generation (combine with RNNs for language)

Case Study: AlexNet

[Krizhevsky et al. 2012]



Full (simplified) AlexNet architecture:
[227x227x3] INPUT
[55x55x96] CONV1: 96 11x11 filters at stride 4, pad 0
[27x27x96] MAX POOL1: 3x3 filters at stride 2
[27x27x96] NORM1: Normalization layer
[27x27x256] CONV2: 256 5x5 filters at stride 1, pad 2
[13x13x256] MAX POOL2: 3x3 filters at stride 2
[13x13x256] NORM2: Normalization layer
[13x13x384] CONV3: 384 3x3 filters at stride 1, pad 1
[13x13x384] CONV4: 384 3x3 filters at stride 1, pad 1
[13x13x256] CONV5: 256 3x3 filters at stride 1, pad 1
[6x6x256] MAX POOL3: 3x3 filters at stride 2
[4096] FC6: 4096 neurons
[4096] FC7: 4096 neurons
[1000] FC8: 1000 neurons (class scores)

Details/Retrospectives:
- first use of ReLU
- used Norm layers (not common anymore)
- heavy data augmentation
- dropout 0.5
- batch size 128
- SGD Momentum 0.9
- Learning rate 1e-2, reduced by 10 manually when val accuracy plateaus
- L2 weight decay 5e-4
- 7 CNN ensemble: 18.2% -> 15.4%

(not counting biases)

INPUT: [224x224x3] memory: 224*224*3=150K params: 0
CONV3-64: [224x224x64] memory: 224*224*64=3.2M params: (3*3*3)*64 = 1,728
CONV3-64: [224x224x64] memory: 224*224*64=3.2M params: (3*3*64)*64 = 36,864
POOL2: [112x112x64] memory: 112*112*64=800K params: 0
CONV3-128: [112x112x128] memory: 112*112*128=1.6M params: (3*3*64)*128 = 73,728
CONV3-128: [112x112x128] memory: 112*112*128=1.6M params: (3*3*128)*128 = 147,456
POOL2: [56x56x128] memory: 56*56*128=400K params: 0
CONV3-256: [56x56x256] memory: 56*56*256=800K params: (3*3*128)*256 = 294,912
CONV3-256: [56x56x256] memory: 56*56*256=800K params: (3*3*256)*256 = 589,824
CONV3-256: [56x56x256] memory: 56*56*256=800K params: (3*3*256)*256 = 589,824
POOL2: [28x28x256] memory: 28*28*256=200K params: 0
CONV3-512: [28x28x512] memory: 28*28*512=400K params: (3*3*256)*512 = 1,179,648
CONV3-512: [28x28x512] memory: 28*28*512=400K params: (3*3*512)*512 = 2,359,296
CONV3-512: [28x28x512] memory: 28*28*512=400K params: (3*3*512)*512 = 2,359,296
POOL2: [14x14x512] memory: 14*14*512=100K params: 0
CONV3-512: [14x14x512] memory: 14*14*512=100K params: (3*3*512)*512 = 2,359,296
CONV3-512: [14x14x512] memory: 14*14*512=100K params: (3*3*512)*512 = 2,359,296
CONV3-512: [14x14x512] memory: 14*14*512=100K params: (3*3*512)*512 = 2,359,296
POOL2: [7x7x512] memory: 7*7*512=25K params: 0
FC: [1x1x4096] memory: 4096 params: 7*7*512*4096 = 102,760,448
FC: [1x1x4096] memory: 4096 params: 4096*4096 = 16,777,216
FC: [1x1x1000] memory: 1000 params: 4096*1000 = 4,096,000

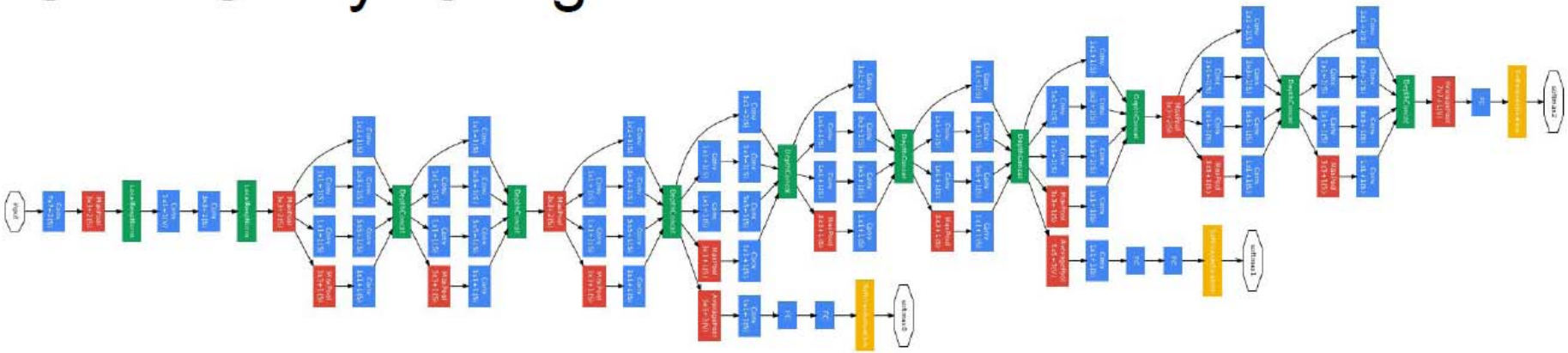
ConvNet Configuration			
B	C	D	
13 weight layers	16 weight layers	16 weight layers	19
put (224 × 224 RGB image)			
conv3-64 **conv3-64**	conv3-64 conv3-64	conv3-64 conv3-64	cc cc
maxpool			
conv3-128 **conv3-128**	conv3-128 conv3-128	conv3-128 conv3-128	co co
maxpool			
conv3-256 conv3-256	conv3-256 conv3-256 **conv1-256**	conv3-256 conv3-256 **conv3-256**	co co co **co**
maxpool			
conv3-512 conv3-512	conv3-512 conv3-512 **conv1-512**	conv3-512 conv3-512 **conv3-512**	co co co **co**
maxpool			
conv3-512 conv3-512	conv3-512 conv3-512 **conv1-512**	conv3-512 conv3-512 **conv3-512**	co co co **co**
maxpool			
FC-4096			
FC-4096			
FC-1000			
soft-max			

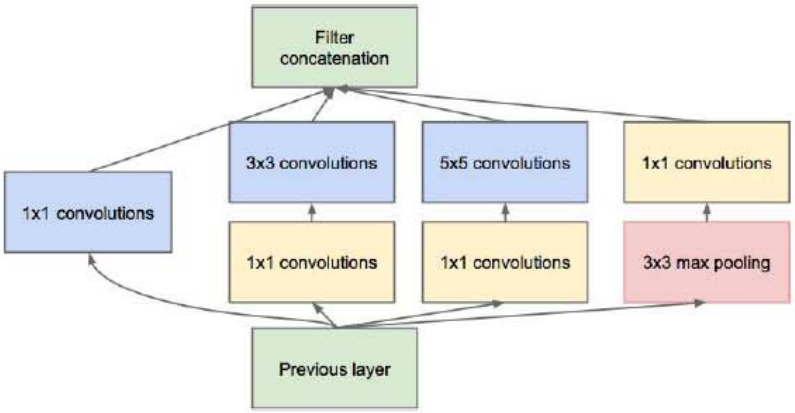
TOTAL memory: 24M * 4 bytes ~= 93MB / image (only forward! ~*2 for bwd)
TOTAL params: 138M parameters

Slide Credit: Fei-Fei Li, Justin Johnson, Serena Yeung, CS 231n

Case Study: GoogLeNet

[Szegedy et al., 2014]





Inception module

ILSVRC 2014 winner (6.7% top 5 error)

Slide Credit: Fei-Fei Li, Justin Johnson, Serena Yeung, CS 231n

Case Study: GoogLeNet

type	patch size/ stride	output size	depth	#1×1	#3×3 reduce	#3×3	#5×5 reduce	#5×5	pool proj	params	ops
convolution	7×7/2	112×112×64	1							2.7K	34M
max pool	3×3/2	56×56×64	0								
convolution	3×3/1	56×56×192	2		64	192				112K	360M
max pool	3×3/2	28×28×192	0								
inception (3a)		28×28×256	2	64	96	128	16	32	32	159K	128M
inception (3b)		28×28×480	2	128	128	192	32	96	64	380K	304M
max pool	3×3/2	14×14×480	0								
inception (4a)		14×14×512	2	192	96	208	16	48	64	364K	73M
inception (4b)		14×14×512	2	160	112	224	24	64	64	437K	88M
inception (4c)		14×14×512	2	128	128	256	24	64	64	463K	100M
inception (4d)		14×14×528	2	112	144	288	32	64	64	580K	119M
inception (4e)		14×14×832	2	256	160	320	32	128	128	840K	170M
max pool	3×3/2	7×7×832	0								
inception (5a)		7×7×832	2	256	160	320	32	128	128	1072K	54M
inception (5b)		7×7×1024	2	384	192	384	48	128	128	1388K	71M
avg pool	7×7/1	1×1×1024	0								
dropout (40%)		1×1×1024	0								
linear		1×1×1000	1							1000K	1M
softmax		1×1×1000	0								

Fun features:

- Only 5 million params! (Removes FC layers completely)

Compared to AlexNet:
- 12X less params
- 2x more compute
- 6.67% (vs. 16.4%)

AlexNet: 60M params

ZNet: 75M

VGG: 138M

GoogleNet: 5M

Slide Credit: Fei-Fei Li, Justin Johnson, Serena Yeung, CS 231n

Case Study: ResNet *[He et al., 2015]*

ILSVRC 2015 winner (3.6% top 5 error)

Microsoft Research

MSRA @ ILSVRC & COCO 2015 Competitions

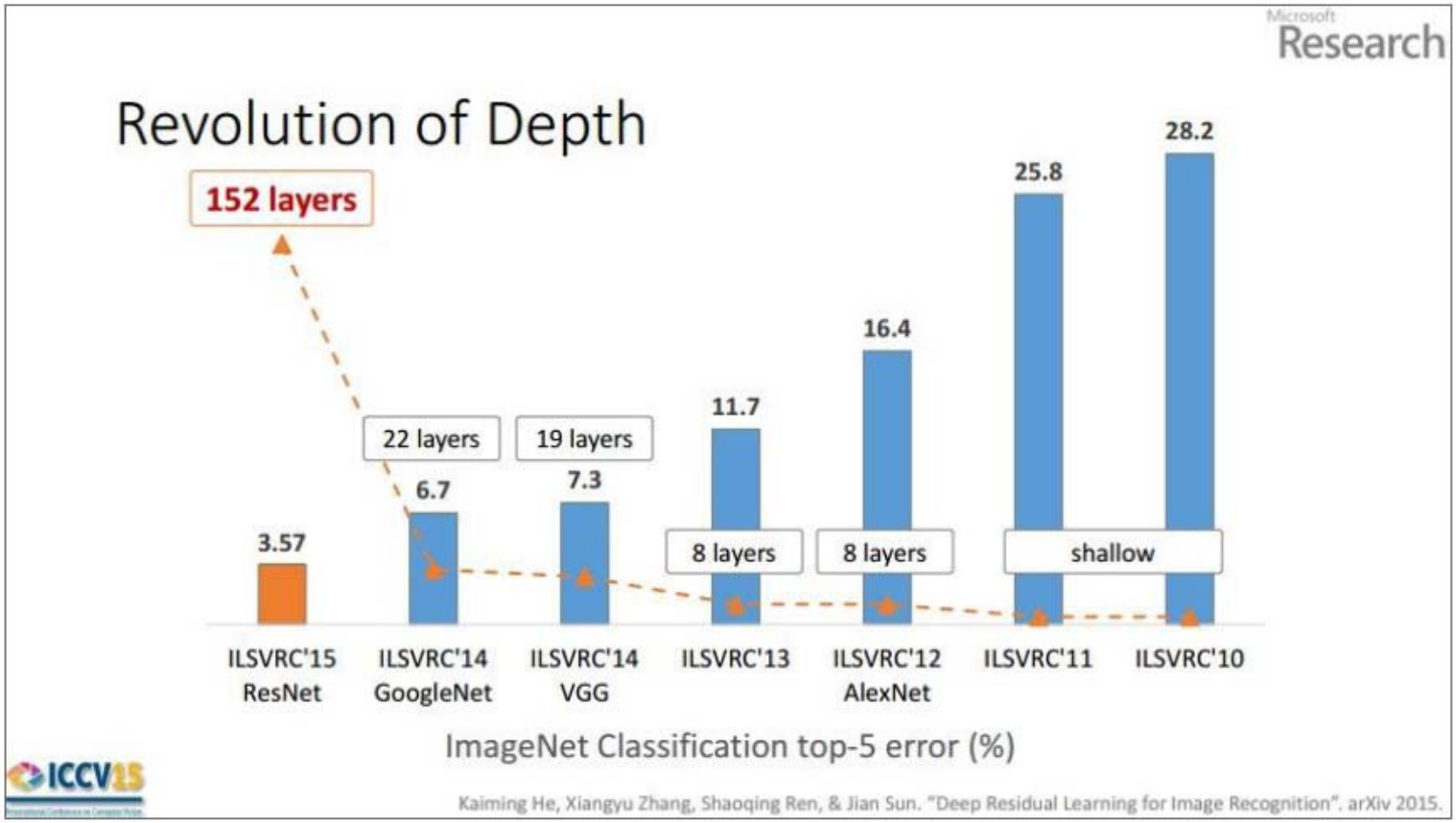
- **1st places in all five main tracks**
  - ImageNet Classification: "*Ultra-deep*" (quote Yann) 152-layer nets
  - ImageNet Detection: 16% better than 2nd
  - ImageNet Localization: 27% better than 2nd
  - COCO Detection: 11% better than 2nd
  - COCO Segmentation: 12% better than 2nd

*improvements are relative numbers

ICCV15

Kaiming He, Xiangyu Zhang, Shaoqing Ren, & Jian Sun. "Deep Residual Learning for Image Recognition". arXiv 2015.

Slide from Kaiming He's recent presentation https://www.youtube.com/watch?v=1PGLj-uKT1w

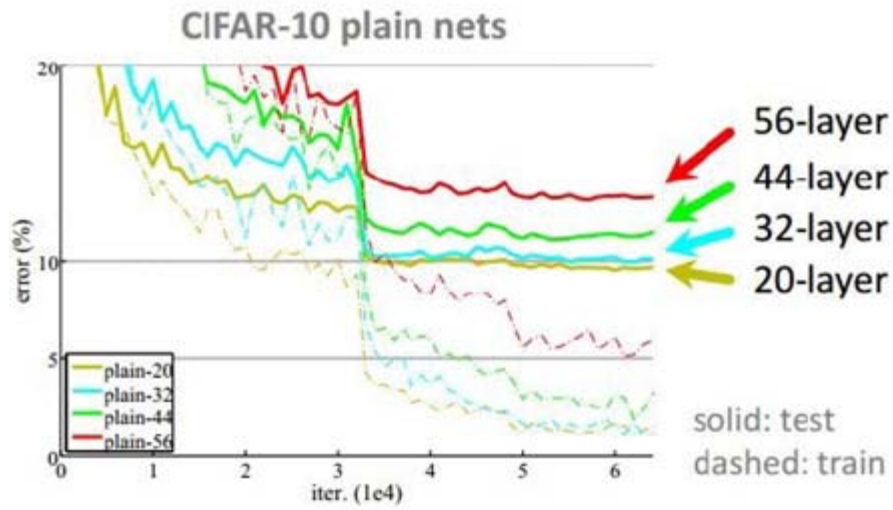


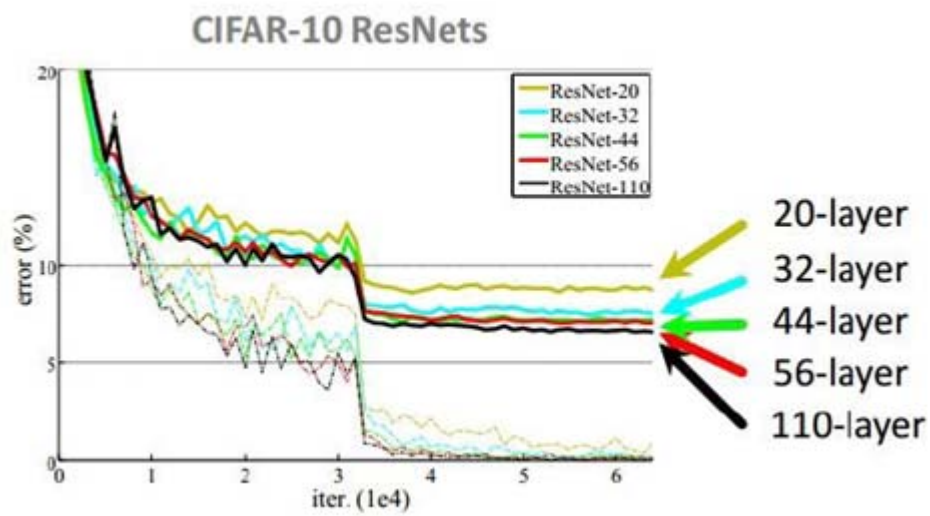
(slide from Kaiming He's recent presentation)

Slide Credit: Fei-Fei Li, Justin Johnson, Serena Yeung, CS 231n

Importance of Depth

- After a while, adding depth *decreases* performance
- At first, vanishing/exploding gradients
  - normalized initialization
  - Batch normalization
  - 2nd order methods
- Then, optimization limitation
  - Deeper network should be able to mimic shallow ones

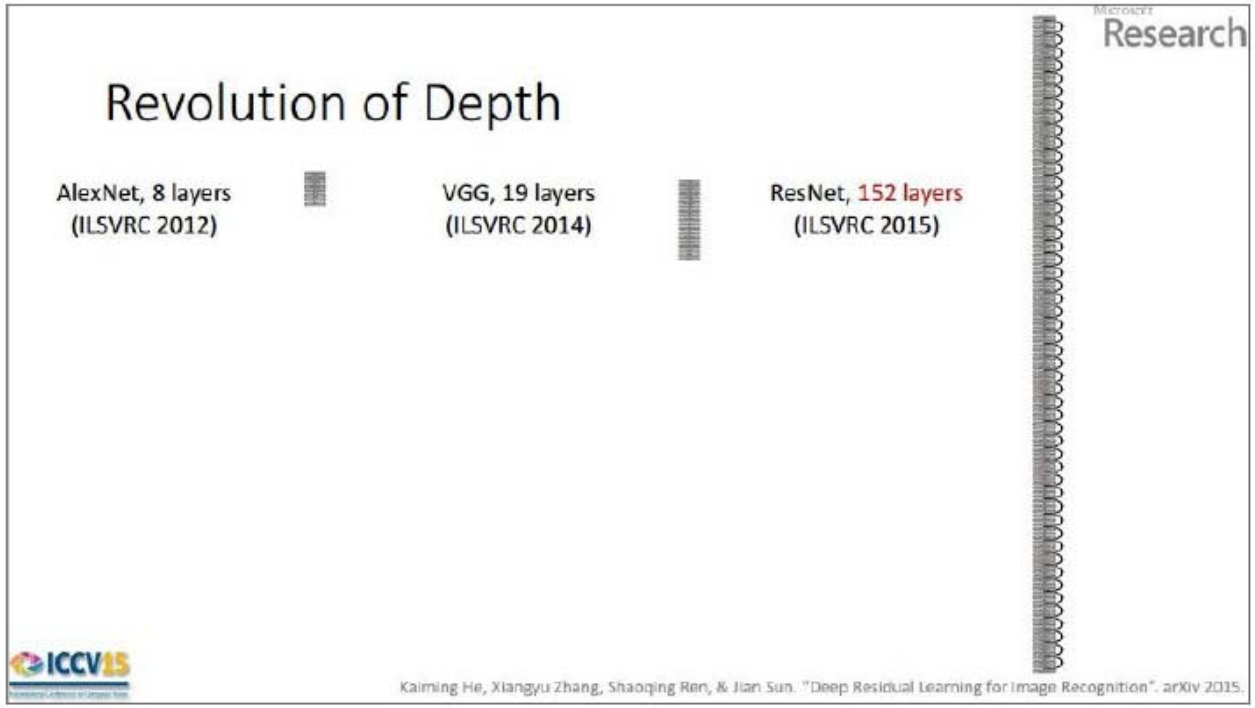




Slide Credit: Fei-Fei Li, Justin Johnson, Serena Yeung, CS 231n

Case Study: ResNet *[He et al., 2015]*

ILSVRC 2015 winner (3.6% top 5 error)



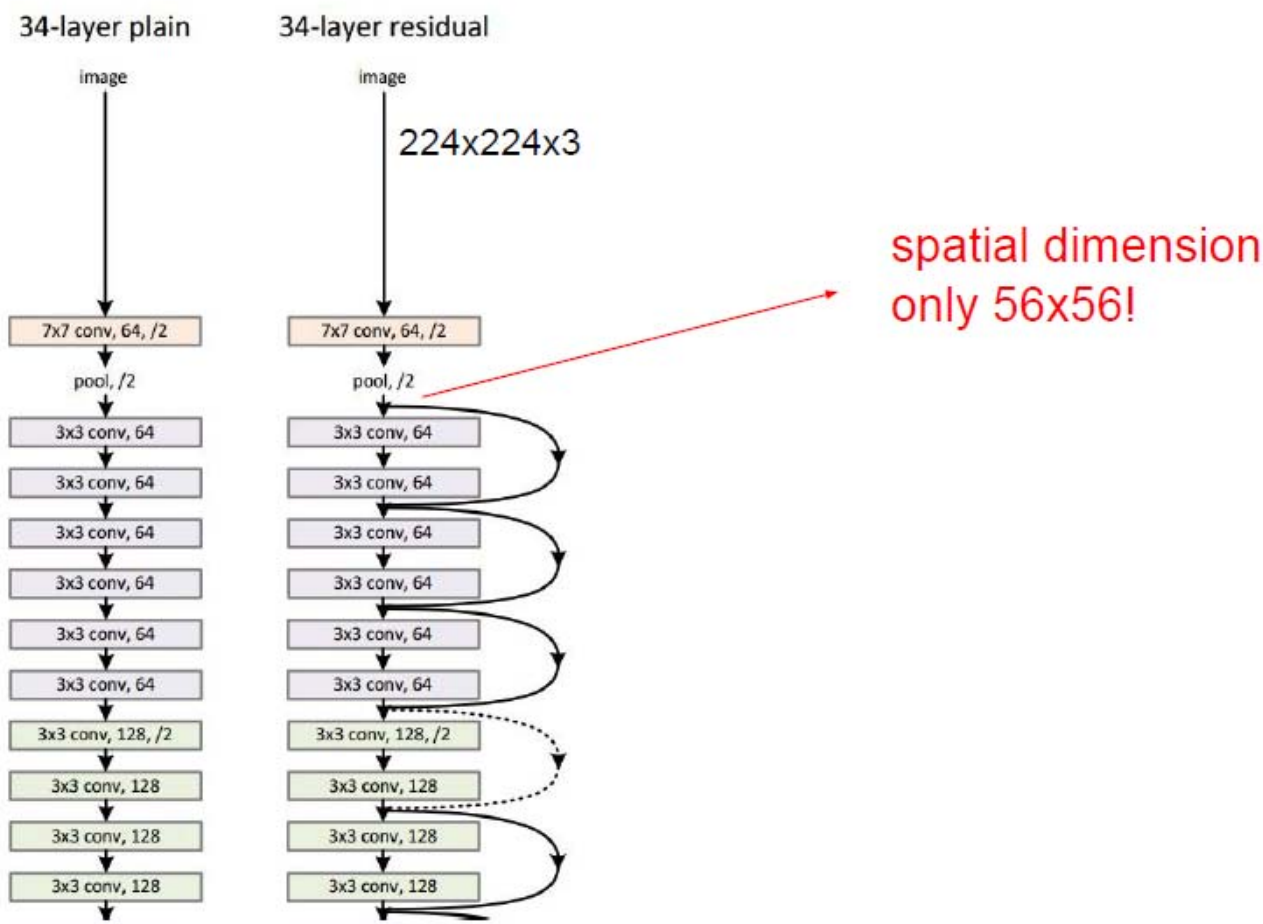
(slide from Kaiming He's recent presentation)

2-3 weeks of training on 8 GPU machine

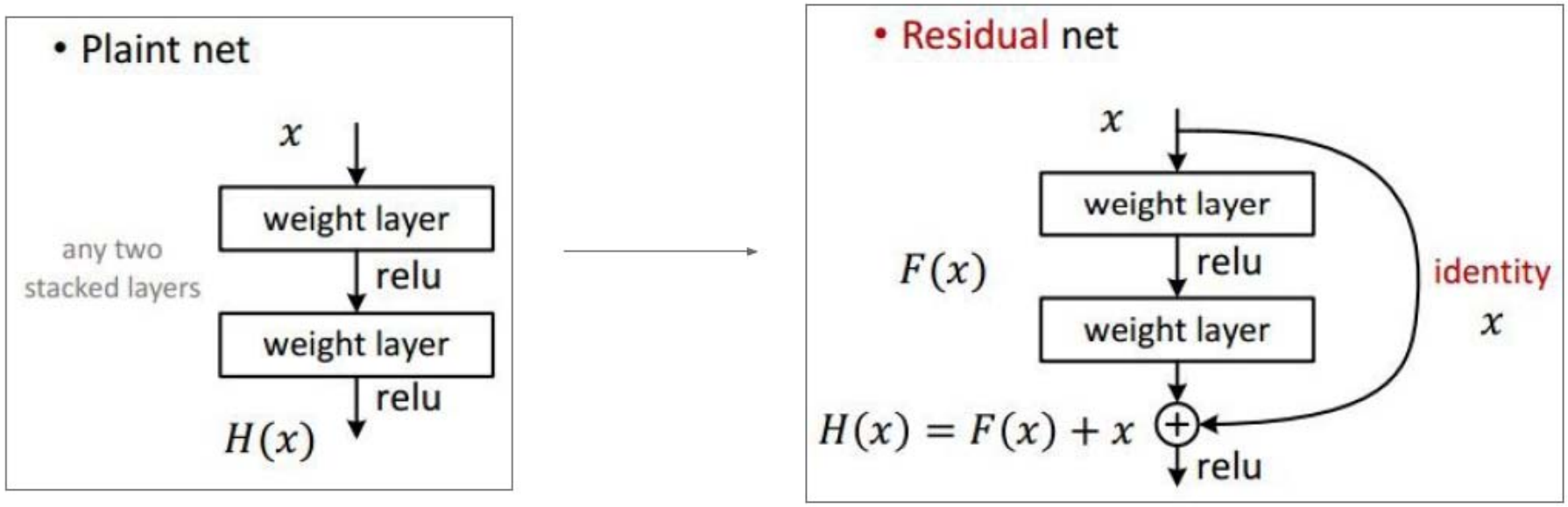
at runtime: faster than a VGGNet! (even though it has 8x more layers)

Case Study: ResNet

[He et al., 2015]



Case Study: ResNet *[He et al., 2015]*

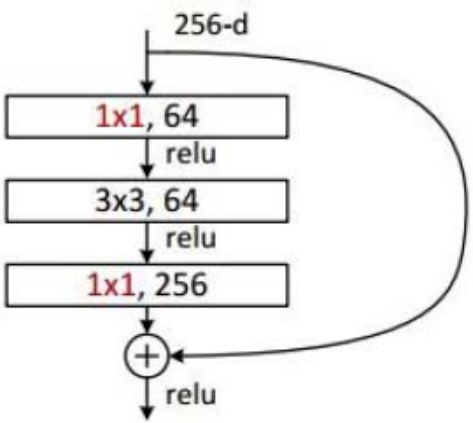


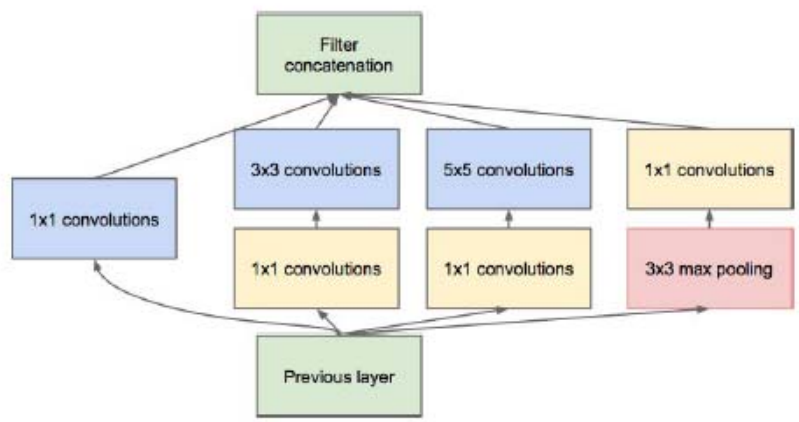
Slide Credit: Fei-Fei Li, Justin Johnson, Serena Yeung, CS 231n

Case Study: ResNet *[He et al., 2015]*

- Batch Normalization after every CONV layer
- Xavier/2 initialization from He et al.
- SGD + Momentum (0.9)
- Learning rate: 0.1, divided by 10 when validation error plateaus
- Mini-batch size 256
- Weight decay of 1e-5
- No dropout used

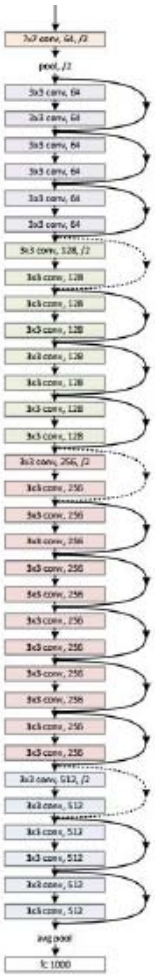
Case Study: ResNet *[He et al., 2015]*





(this trick is also used in GoogLeNet)

Slide Credit: Fei-Fei Li, Justin Johnson, Serena Yeung, CS 231n

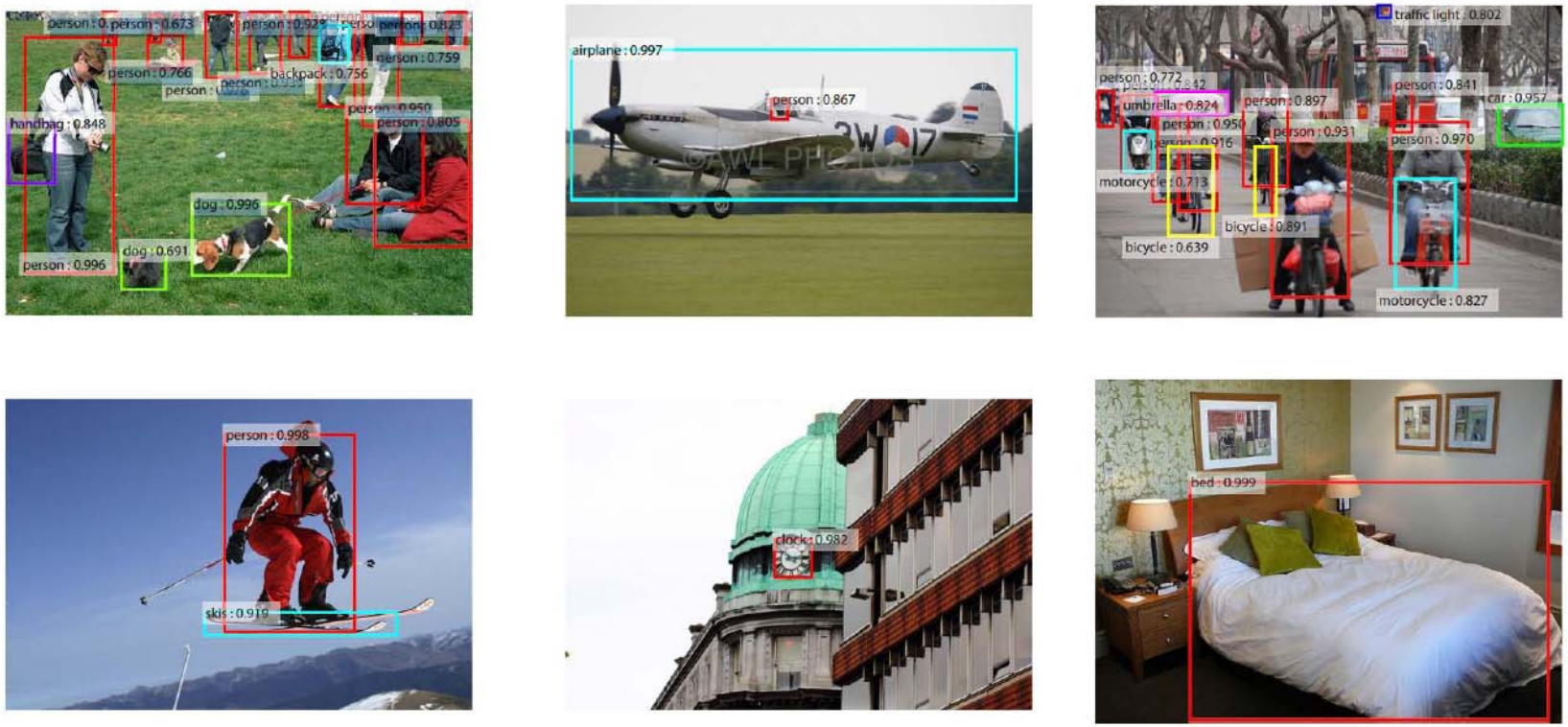


Case Study: ResNet *[He et al., 2015]*

layer name	output size	18-layer	34-layer	50-layer	101-layer	152-layer
conv1	112×112	7×7, 64, stride 2				
conv2_x	56×56	3×3 max pool, stride 2				
		$\begin{bmatrix} 3\times3, 64 \\ 3\times3, 64 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 64 \\ 3\times3, 64 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$
conv3_x	28×28	$\begin{bmatrix} 3\times3, 128 \\ 3\times3, 128 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 128 \\ 3\times3, 128 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times8$
conv4_x	14×14	$\begin{bmatrix} 3\times3, 256 \\ 3\times3, 256 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 256 \\ 3\times3, 256 \end{bmatrix}\times6$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times6$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times23$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times36$
conv5_x	7×7	$\begin{bmatrix} 3\times3, 512 \\ 3\times3, 512 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 512 \\ 3\times3, 512 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$
	1×1	average pool, 1000-d fc, softmax				
FLOPs		1.8×10^9	3.6×10^9	3.8×10^9	7.6×10^9	11.3×10^9

Slide Credit: Fei-Fei Li, Justin Johnson, Serena Yeung, CS 231n

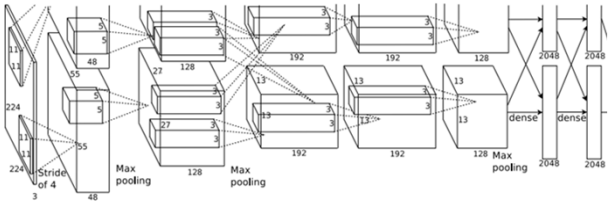
Localization and Detection



Results from Faster R-CNN, Ren et al 2015

Slide Credit: Fei-Fei Li, Justin Johnson, Serena Yeung, CS 231n

So far: Image Classification



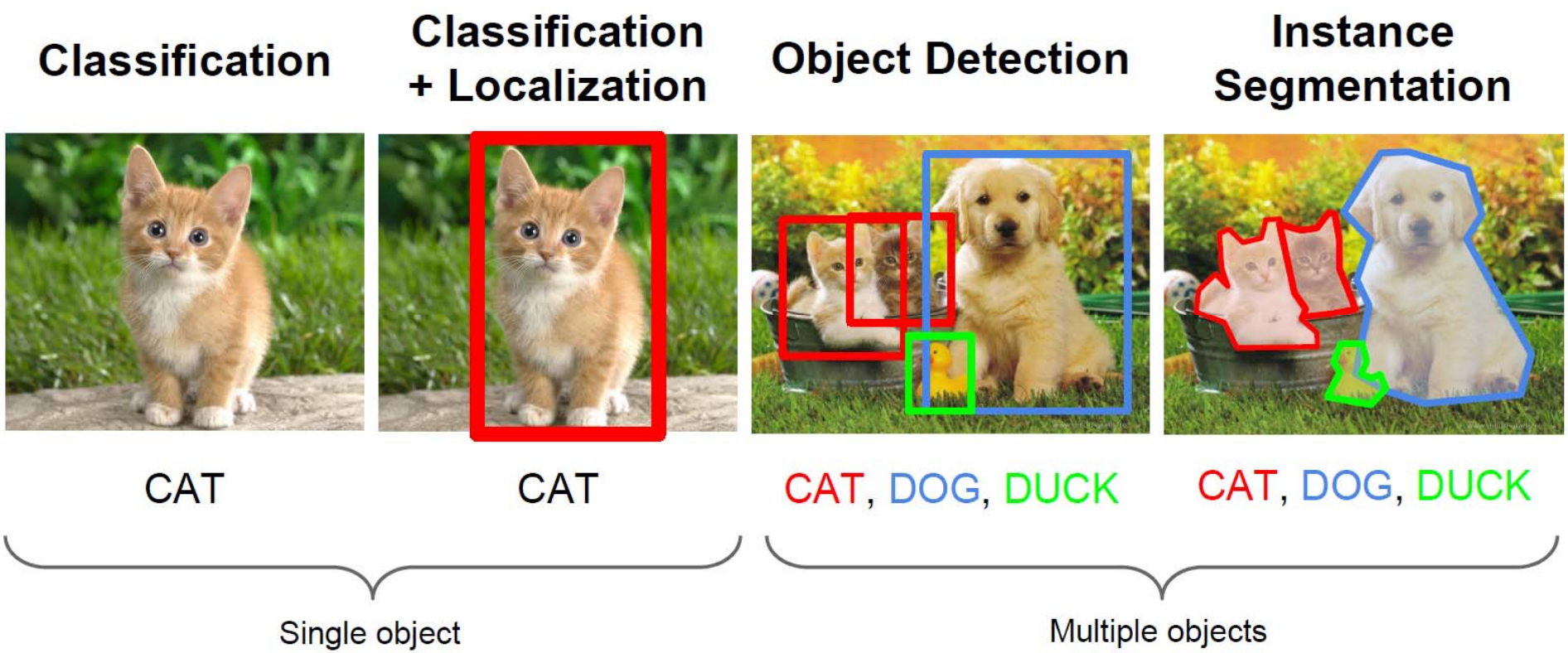
Vector:
4096

Fully-Connected:
4096 to 1000

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

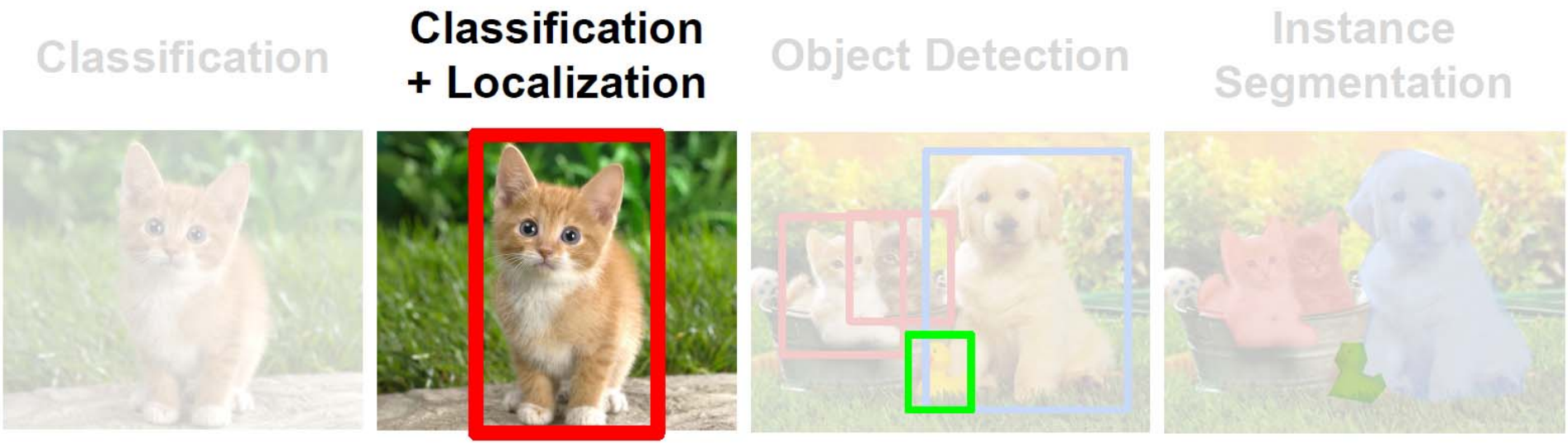
Slide Credit: Fei-Fei Li, Justin Johnson, Serena Yeung, CS 231n

Computer Vision Tasks



Slide Credit: Fei-Fei Li, Justin Johnson, Serena Yeung, CS 231n

Computer Vision Tasks



Slide Credit: Fei-Fei Li, Justin Johnson, Serena Yeung, CS 231n

Classification + Localization

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

Slide Credit: Fei-Fei Li, Justin Johnson, Serena Yeung, CS 231n

CLS - ImageNet

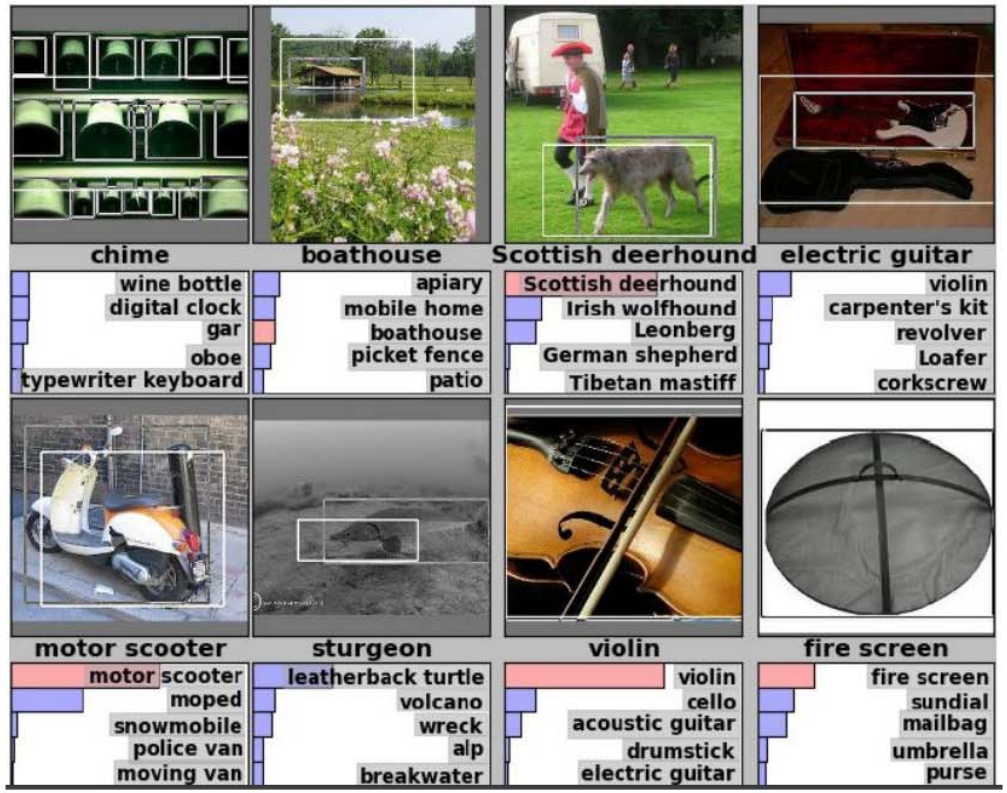
1000 classes (same as classification)

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

Idea 1: Localization as Regression

Input: image

Neural Net

Output:
Box coordinates
(4 numbers)

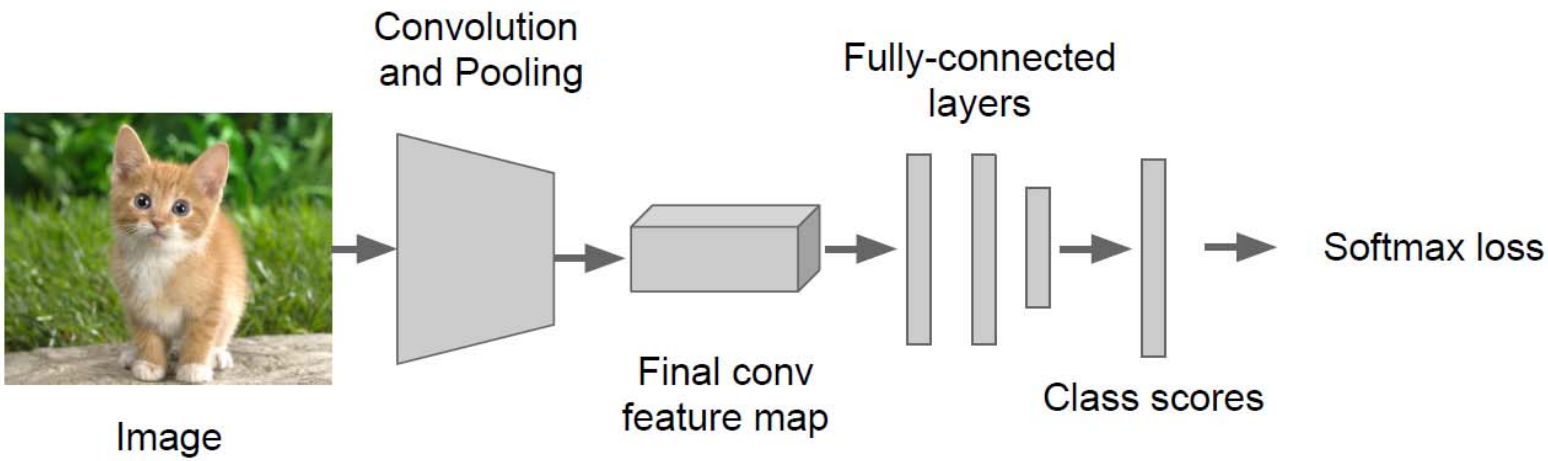
Correct output:
box coordinates
(4 numbers)

Loss:
L2 distance

Only one object,
simpler than detection

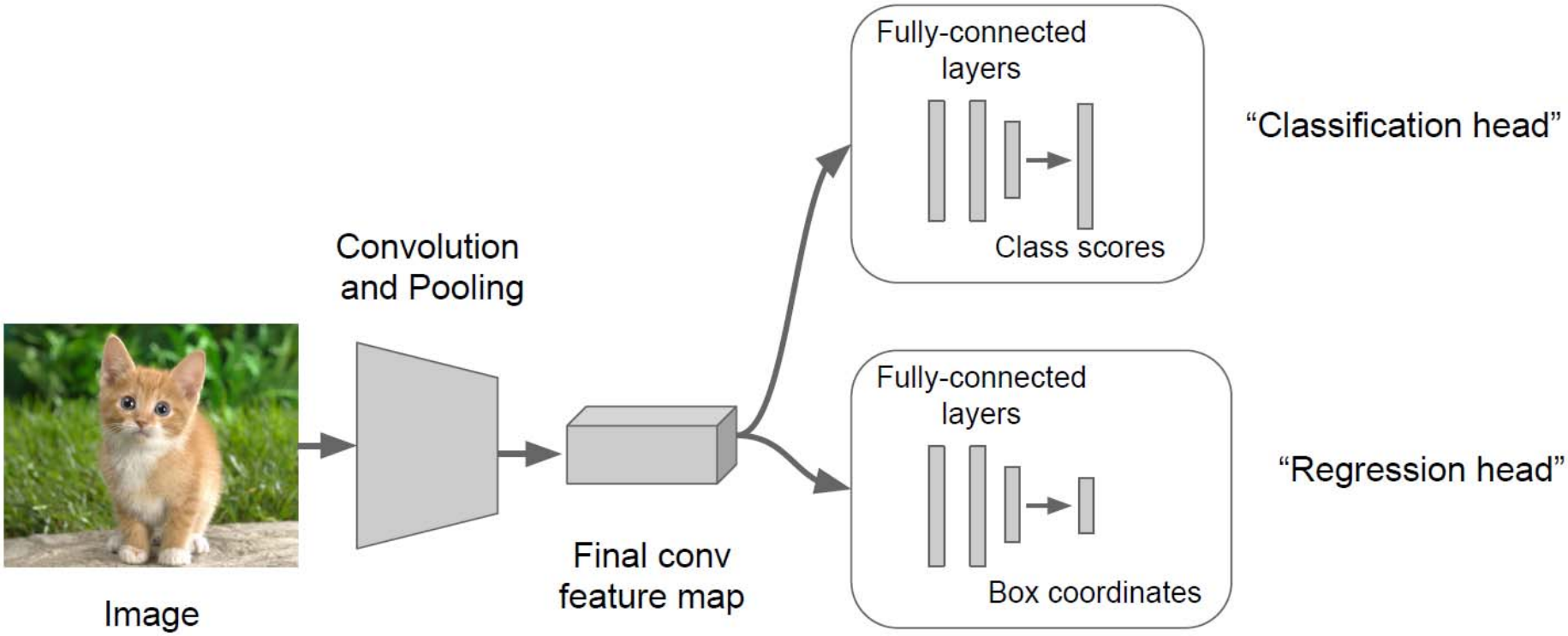
Slide Credit: Fei-Fei Li, Justin Johnson, Serena Yeung, CS 231n

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



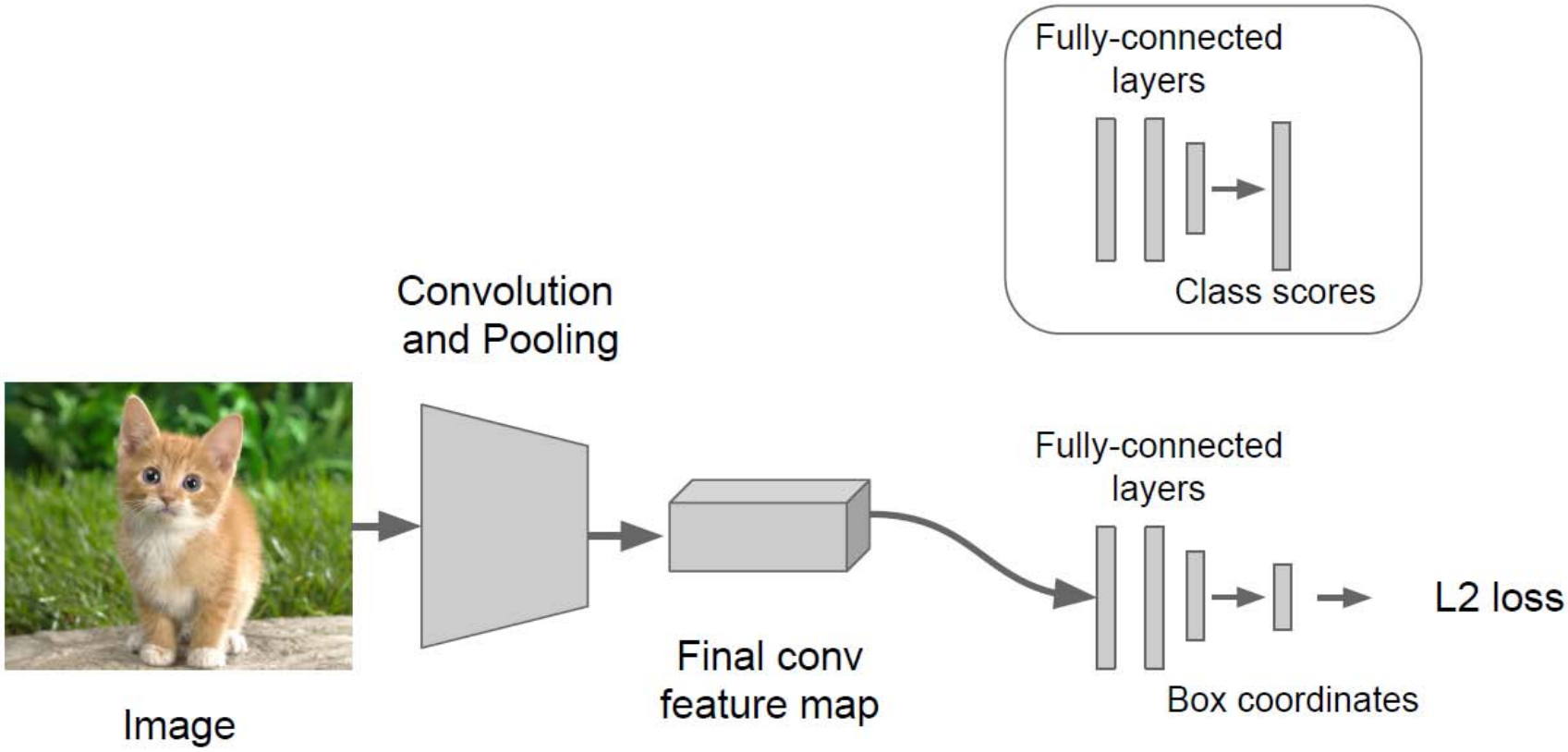
Slide Credit: Fei-Fei Li, Justin Johnson, Serena Yeung, CS 231n

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



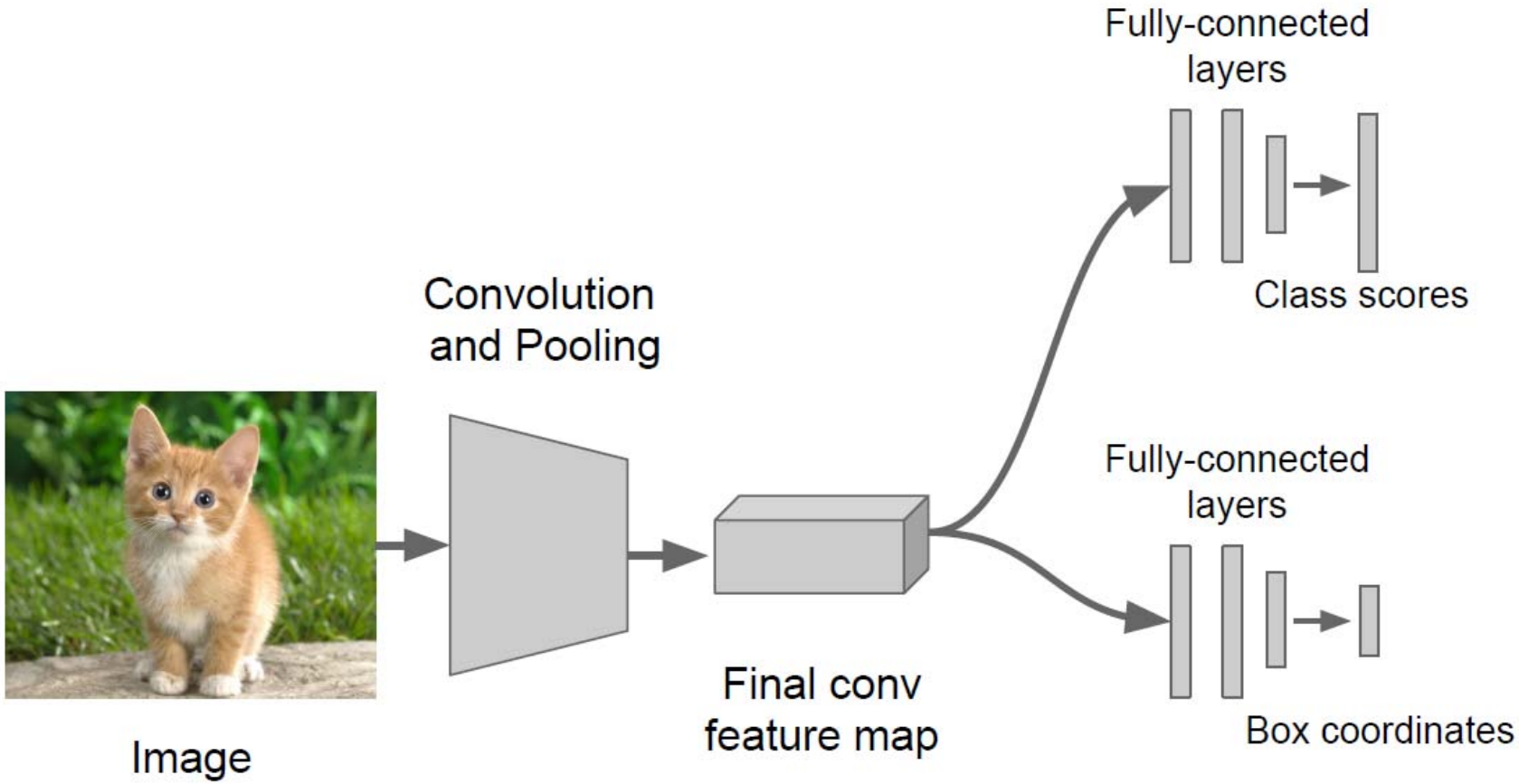
Slide Credit: Fei-Fei Li, Justin Johnson, Serena Yeung, CS 231n

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



Slide Credit: Fei-Fei Li, Justin Johnson, Serena Yeung, CS 231n

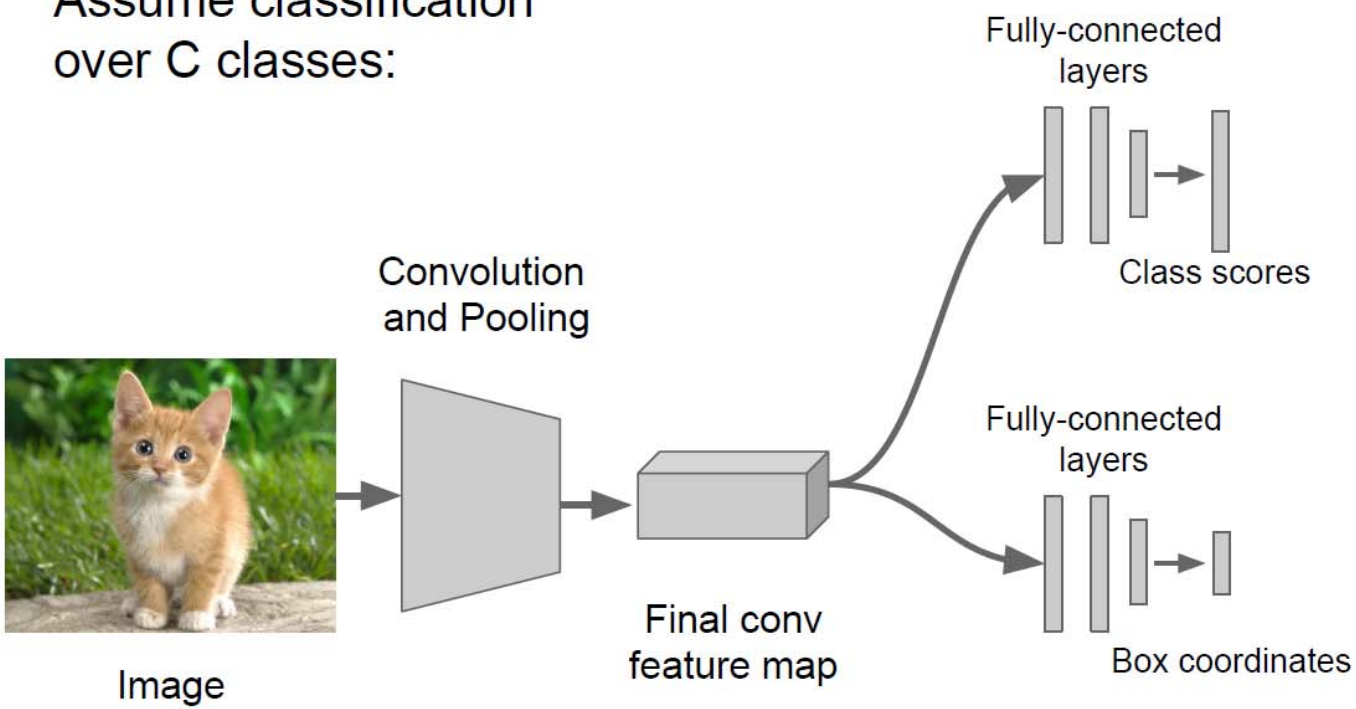
Step 4: At test time use both heads



Slide Credit: Fei-Fei Li, Justin Johnson, Serena Yeung, CS 231n

Per-Class vs. Class Agnostic

Assume classification over C classes:



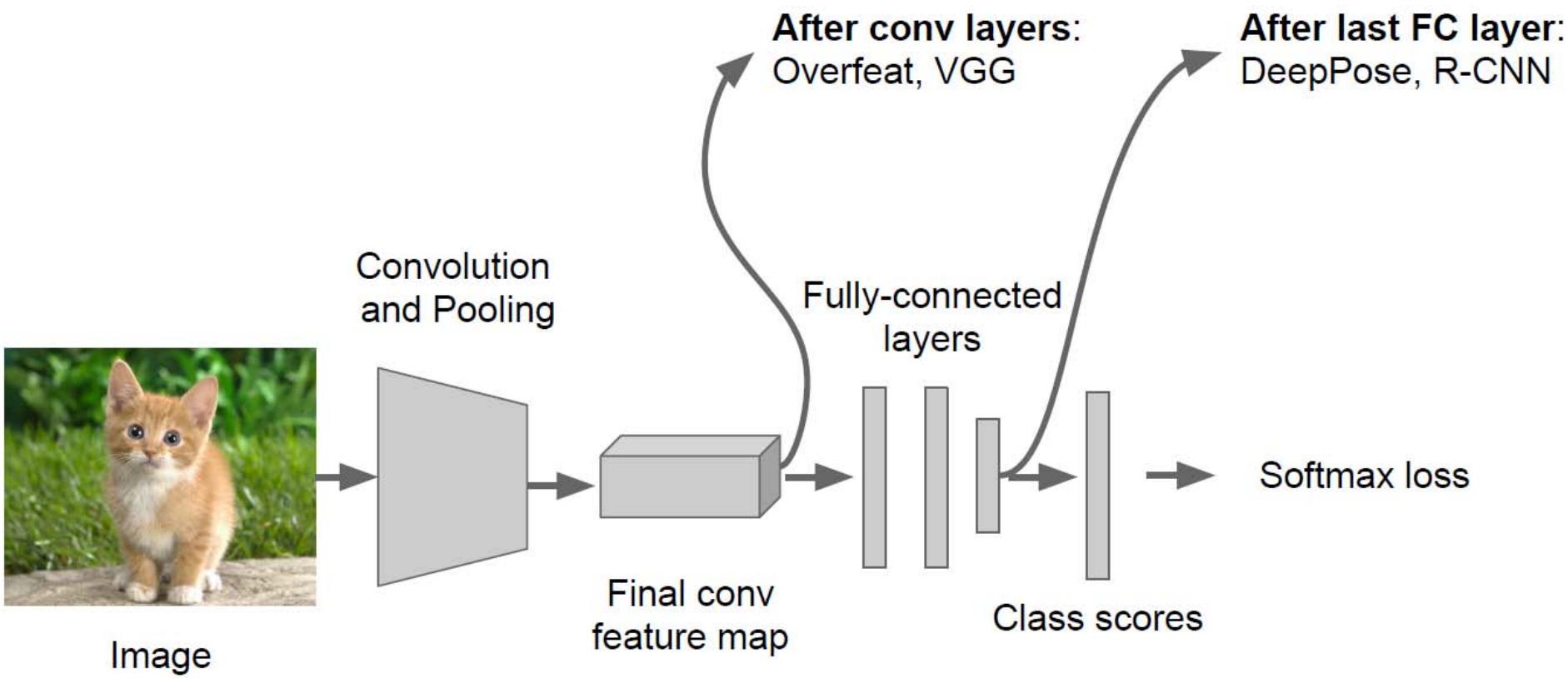
Classification head:
C numbers
(one per class)

Class agnostic:
4 numbers
(one box)

Class specific:
C x 4 numbers
(one box per class)

Slide Credit: Fei-Fei Li, Justin Johnson, Serena Yeung, CS 231n

Where to attach?

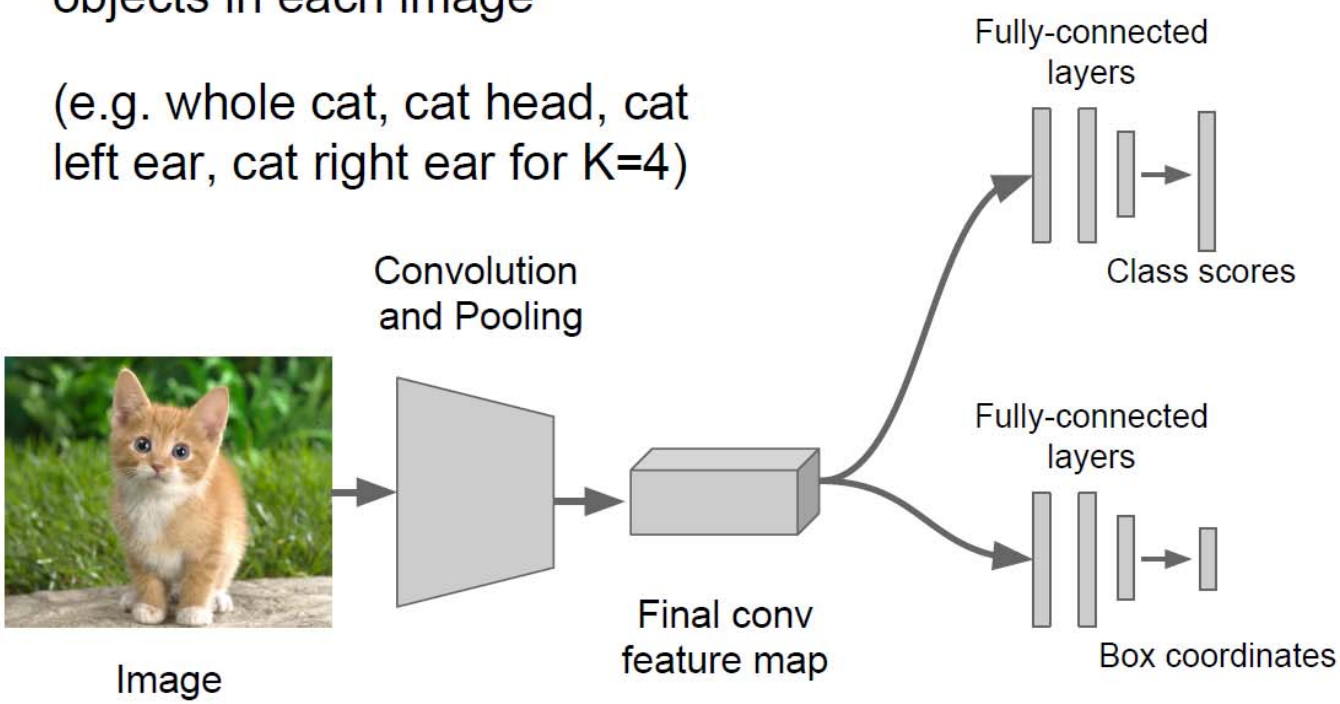


Slide Credit: Fei-Fei Li, Justin Johnson, Serena Yeung, CS 231n

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)



K x 4 numbers
(one box per object)

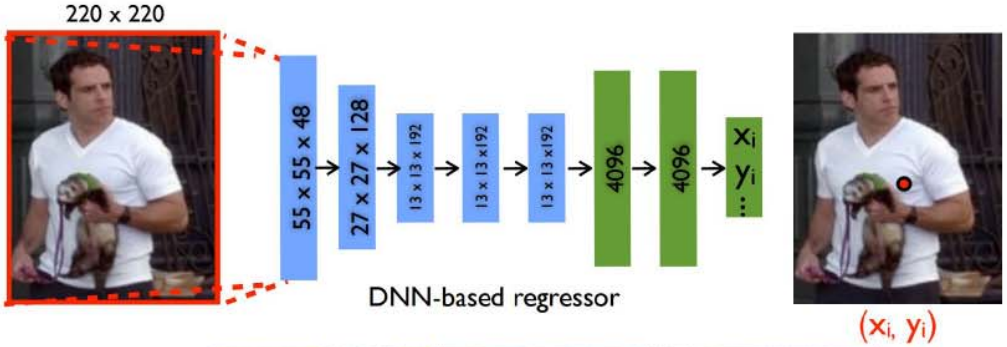
Slide Credit: Fei-Fei Li, Justin Johnson, Serena Yeung, CS 231n

Human Pose Estimation

Represent a person by K joints

Regress (x, y) for each joint from last fully-connected layer of AlexNet

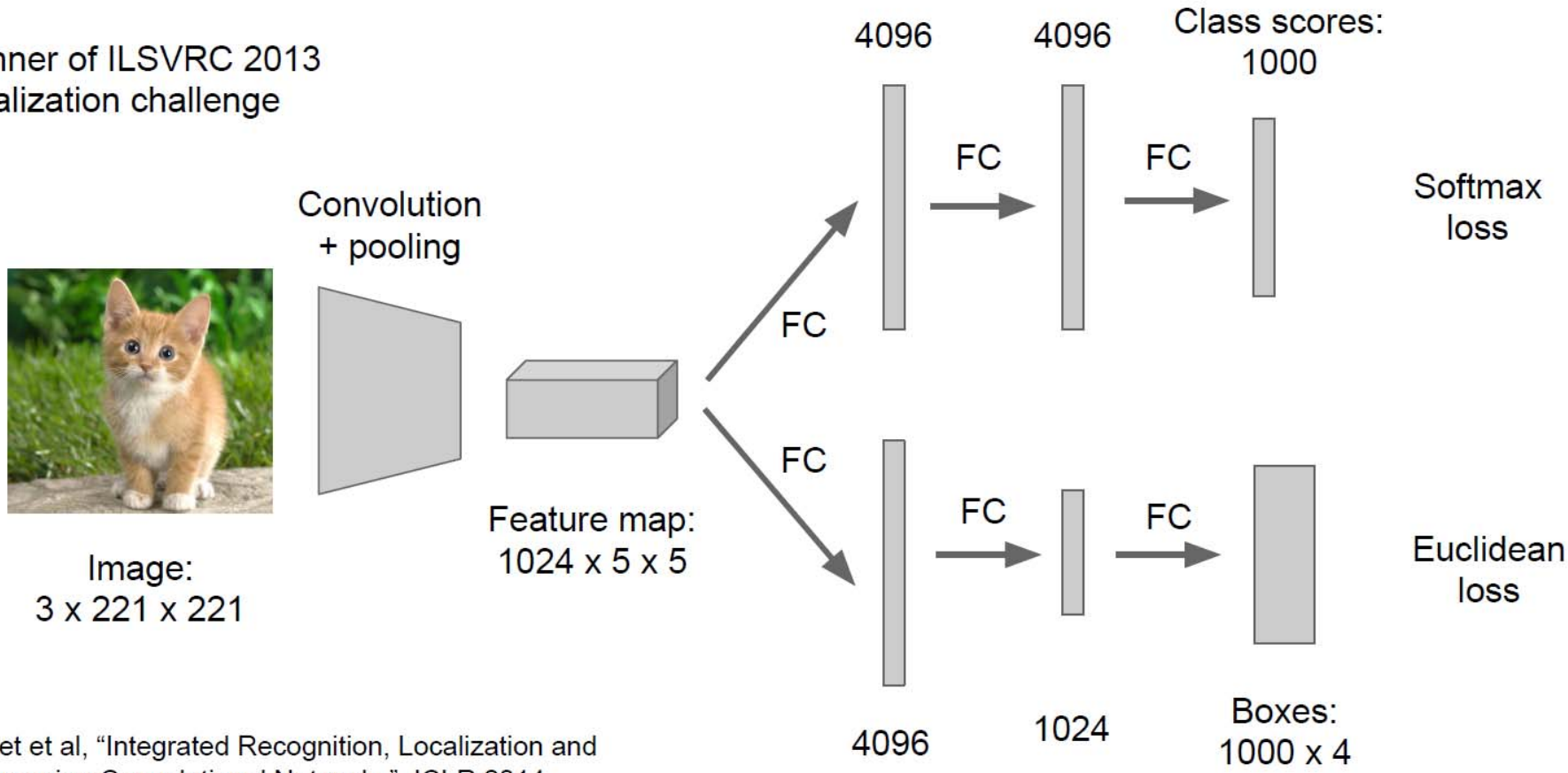
(Details: Normalized coordinates, iterative refinement)



Toshev and Szegedy, "DeepPose: Human Pose Estimation via Deep Neural Networks", CVPR 2014

Sliding Window: Overfeat

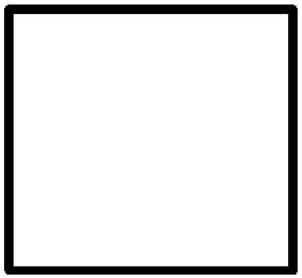
Winner of ILSVRC 2013
localization challenge



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

Sliding Window: Overfeat

Sliding Window: Overfeat



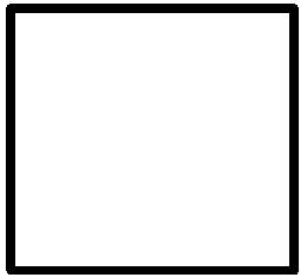
Network input:
3 x 221 x 221

Larger image:
3 x 257 x 257

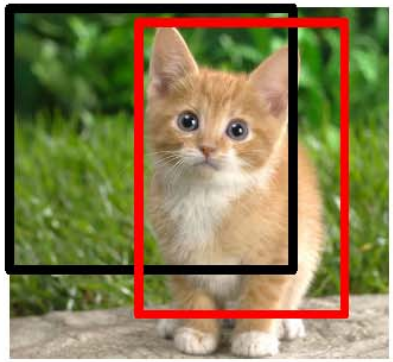
Slide Credit: Fei-Fei Li, Justin Johnson, Serena Yeung, CS 231n

Sliding Window: Overfeat

Sliding Window: Overfeat



Network input:
3 x 221 x 221



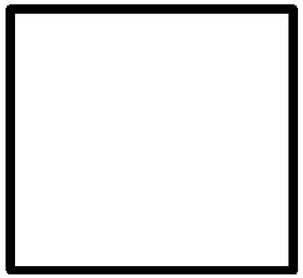
Larger image:
3 x 257 x 257

Classification scores:
P(cat)

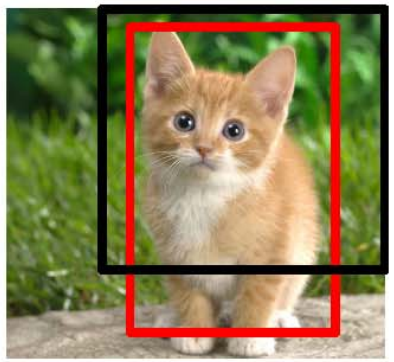
Slide Credit: Fei-Fei Li, Justin Johnson, Serena Yeung, CS 231n

Sliding Window: Overfeat

Sliding Window: Overfeat



Network input:
3 x 221 x 221



Larger image:
3 x 257 x 257

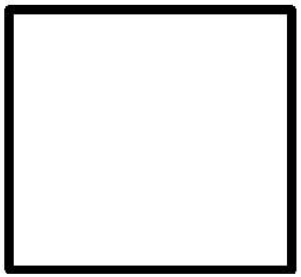
0.5	0.75

Classification scores:
P(cat)

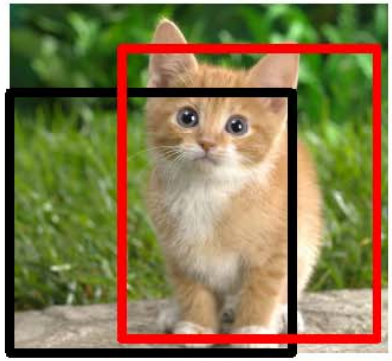
Slide Credit: Fei-Fei Li, Justin Johnson, Serena Yeung, CS 231n

Sliding Window: Overfeat

Sliding Window: Overfeat



Network input:
3 x 221 x 221



Larger image:
3 x 257 x 257

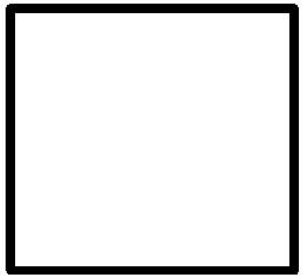
0.5	0.75
0.6	

Classification scores:
P(cat)

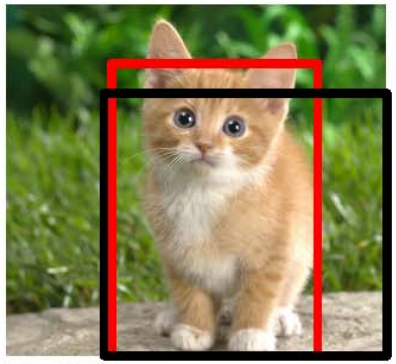
Slide Credit: Fei-Fei Li, Justin Johnson, Serena Yeung, CS 231n

Sliding Window: Overfeat

Sliding Window: Overfeat



Network input:
3 x 221 x 221



Larger image:
3 x 257 x 257

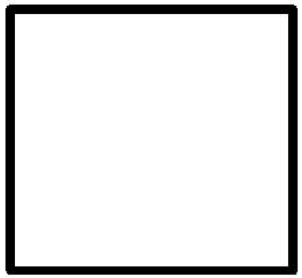
0.5	0.75
0.6	0.8

Classification scores:
P(cat)

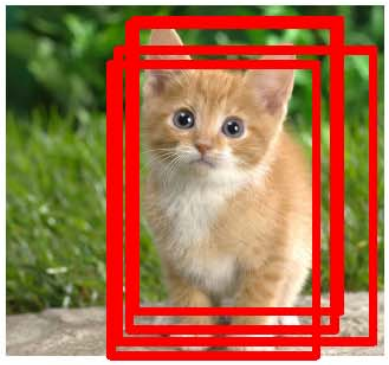
Slide Credit: Fei-Fei Li, Justin Johnson, Serena Yeung, CS 231n

Sliding Window: Overfeat

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)

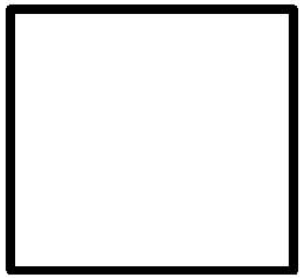
Why aren't boxes across grid?

Slide Credit: Fei-Fei Li, Justin Johnson, Serena Yeung, CS 231n

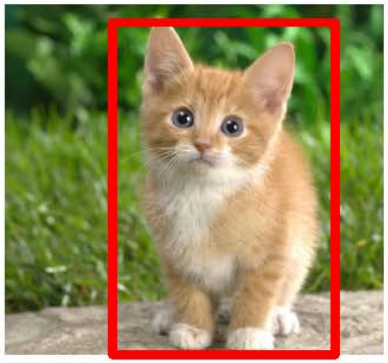
Sliding Window: Overfeat

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

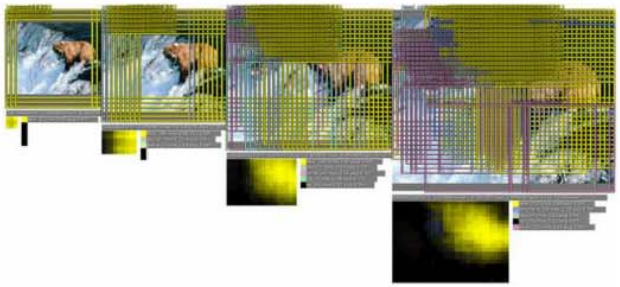
Classification score: P (cat)

Slide Credit: Fei-Fei Li, Justin Johnson, Serena Yeung, CS 231n

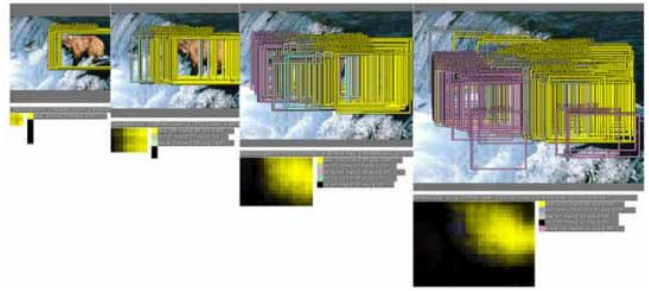
Sliding Window: Overfeat

In practice use many sliding window locations and multiple scales

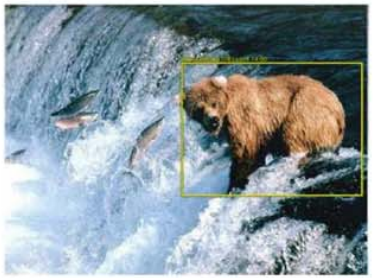
Window positions + score maps



Box regression outputs

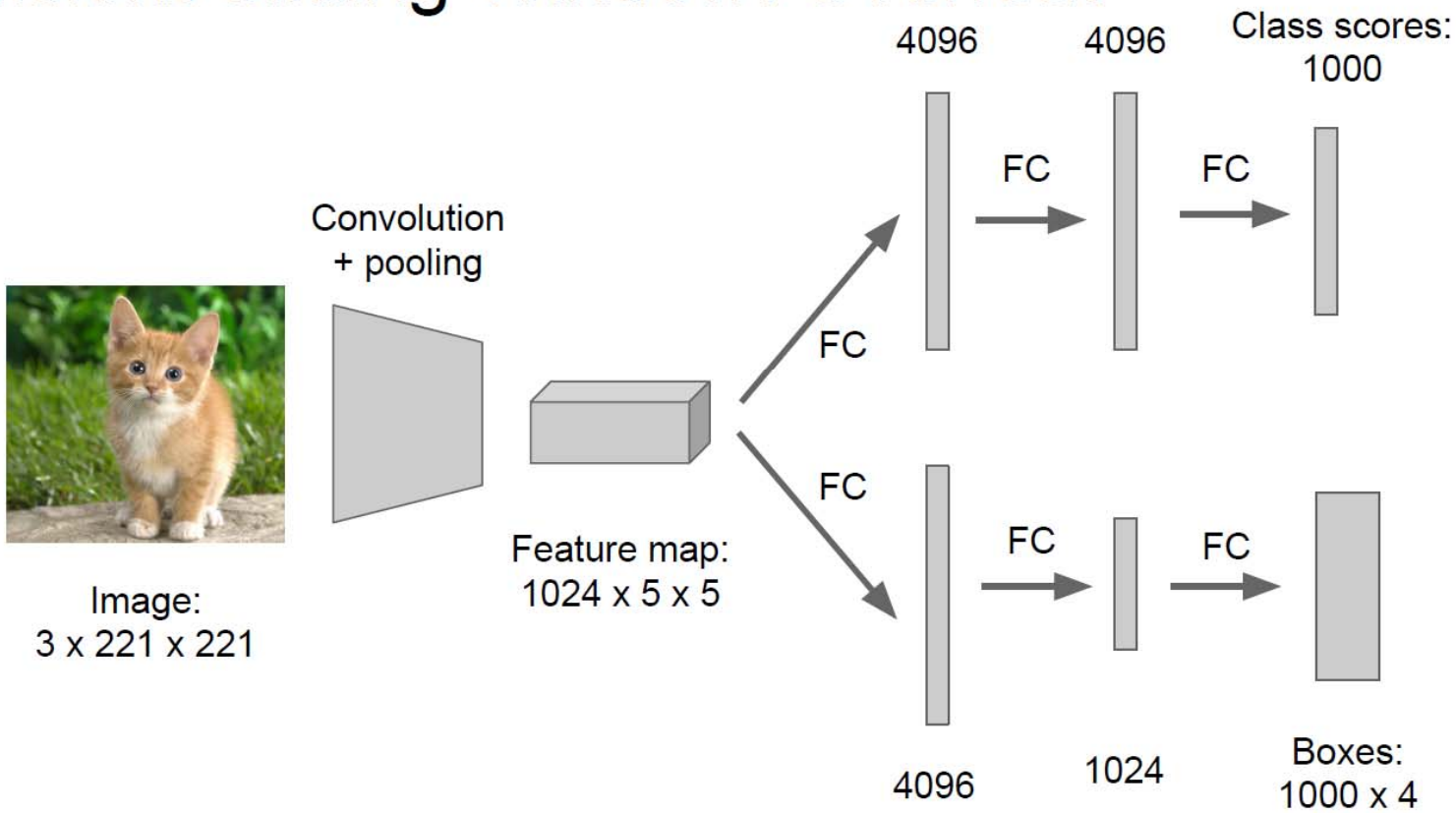


Final Predictions



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

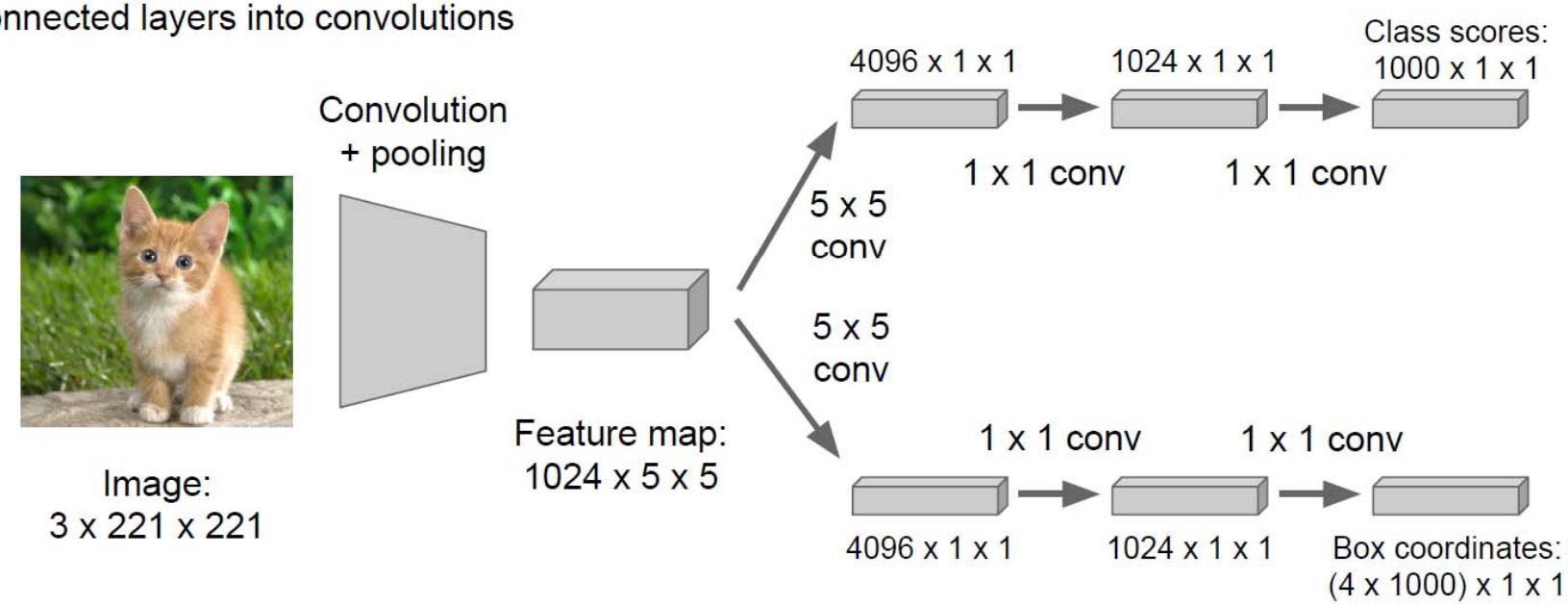
Efficient Sliding Window: Overfeat



Slide Credit: Fei-Fei Li, Justin Johnson, Serena Yeung, CS 231n

Efficient Sliding Window: Overfeat

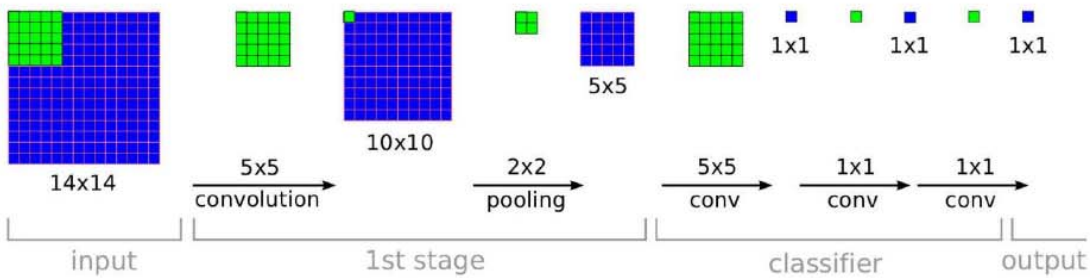
Efficient sliding window by converting fully-connected layers into convolutions



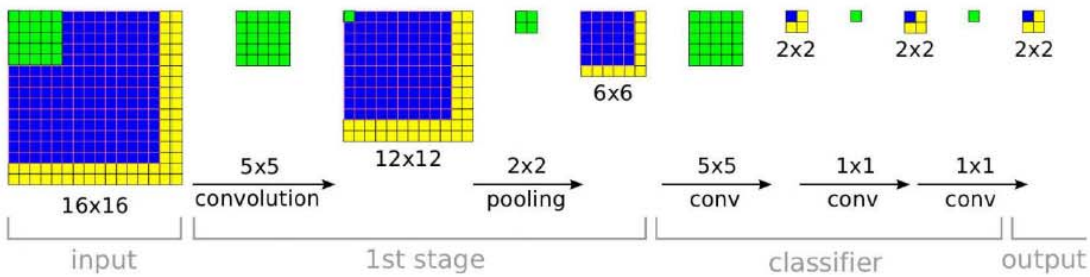
Slide Credit: Fei-Fei Li, Justin Johnson, Serena Yeung, CS 231n

Efficient Sliding Window: Overfeat

Training time: Small image, 1 x 1 classifier output



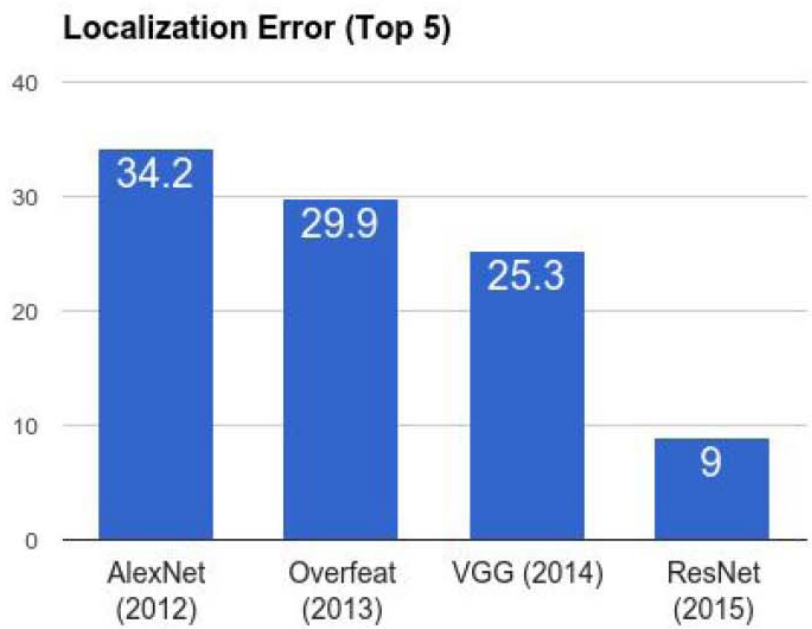
Test time: Larger image, 2 x 2 classifier output, only extra compute at yellow regions



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

Slide Credit: Fei-Fei Li, Justin Johnson, Serena Yeung, CS 231n

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

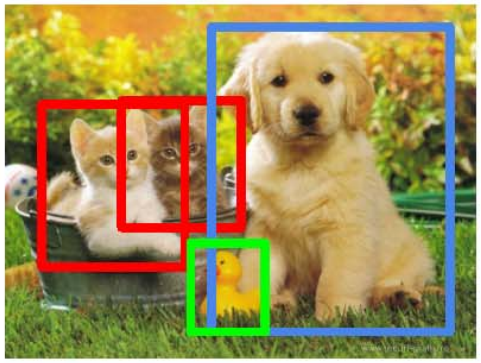
Slide Credit: Fei-Fei Li, Justin Johnson, Serena Yeung, CS 231n

Computer Vision Tasks

Classification

Classification + Localization

Object Detection



Instance Segmentation

Slide Credit: Fei-Fei Li, Justin Johnson, Serena Yeung, CS 231n

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

Slide Credit: Fei-Fei Li, Justin Johnson, Serena Yeung, CS 231n

Detection as Regression?

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

= 8 numbers

Slide Credit: Fei-Fei Li, Justin Johnson, Serena Yeung, CS 231n

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

= many numbers

Need variable sized outputs

Slide Credit: Fei-Fei Li, Justin Johnson, Serena Yeung, CS 231n

Detection as Classification

CAT? NO

DOG? NO

Slide Credit: Fei-Fei Li, Justin Johnson, Serena Yeung, CS 231n

Detection as Classification

CAT? YES!

DOG? NO

Slide Credit: Fei-Fei Li, Justin Johnson, Serena Yeung, CS 231n

Detection as Classification

CAT? NO

DOG? NO

Slide Credit: Fei-Fei Li, Justin Johnson, Serena Yeung, CS 231n

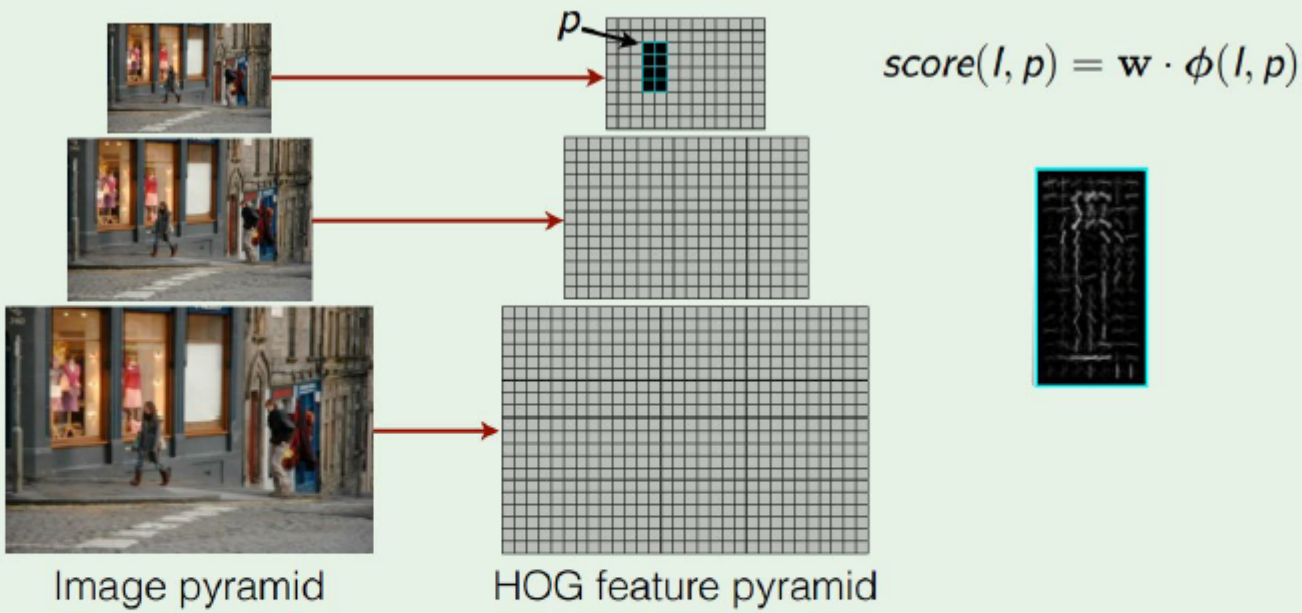
Detection as Classification

Problem: Need to test many positions and scales

Solution: If your classifier is fast enough, just do it

Slide Credit: Fei-Fei Li, Justin Johnson, Serena Yeung, CS 231n

Histogram of Oriented Gradients



- Compute HOG of the whole image at multiple resolutions
- Score every subwindow of the feature pyramid
- Apply non-maxima suppression

Dalal and Triggs, "Histograms of Oriented Gradients for Human Detection", CVPR 2005
Slide credit: Ross Girshick

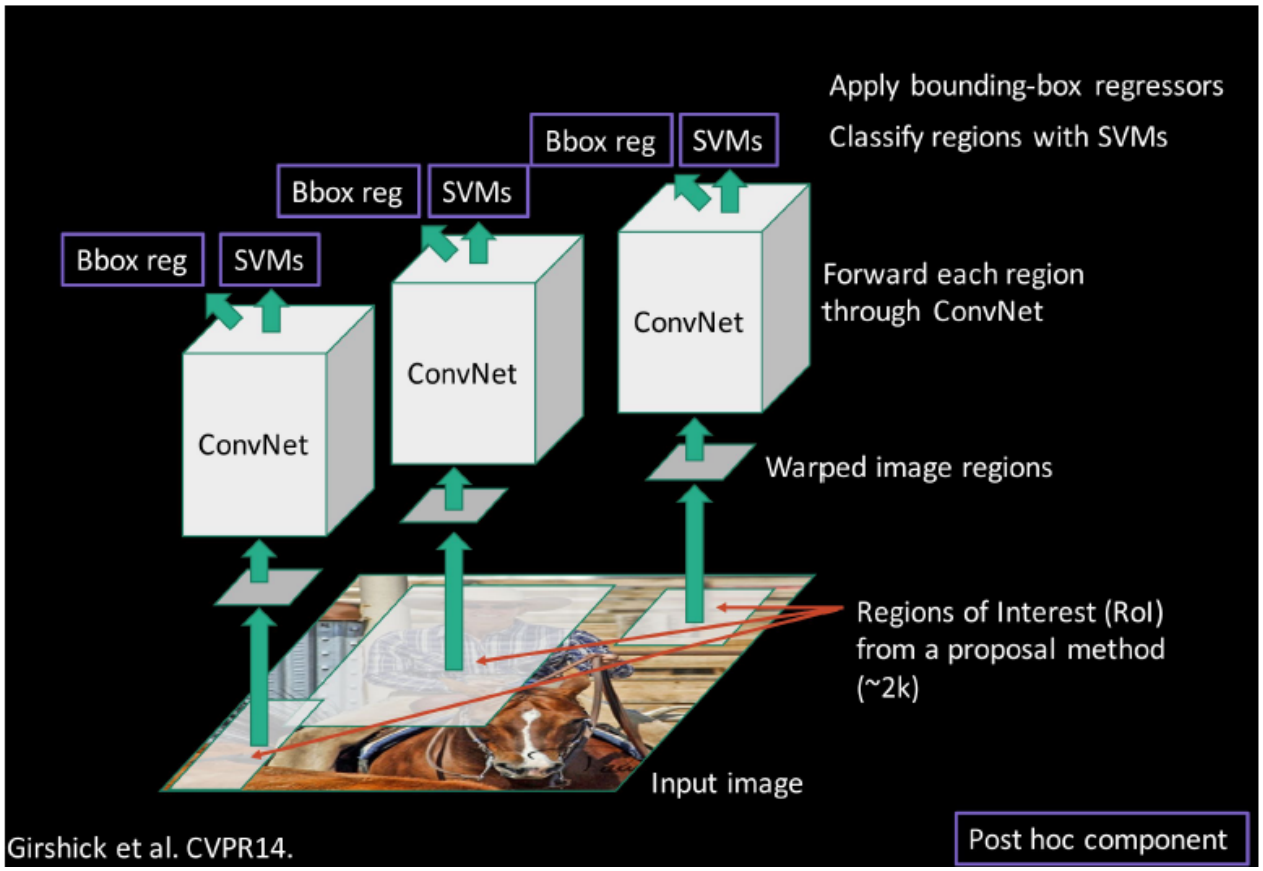
Detection as Classification

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

Slide Credit: Fei-Fei Li, Justin Johnson, Serena Yeung, CS 231n

R-CNN

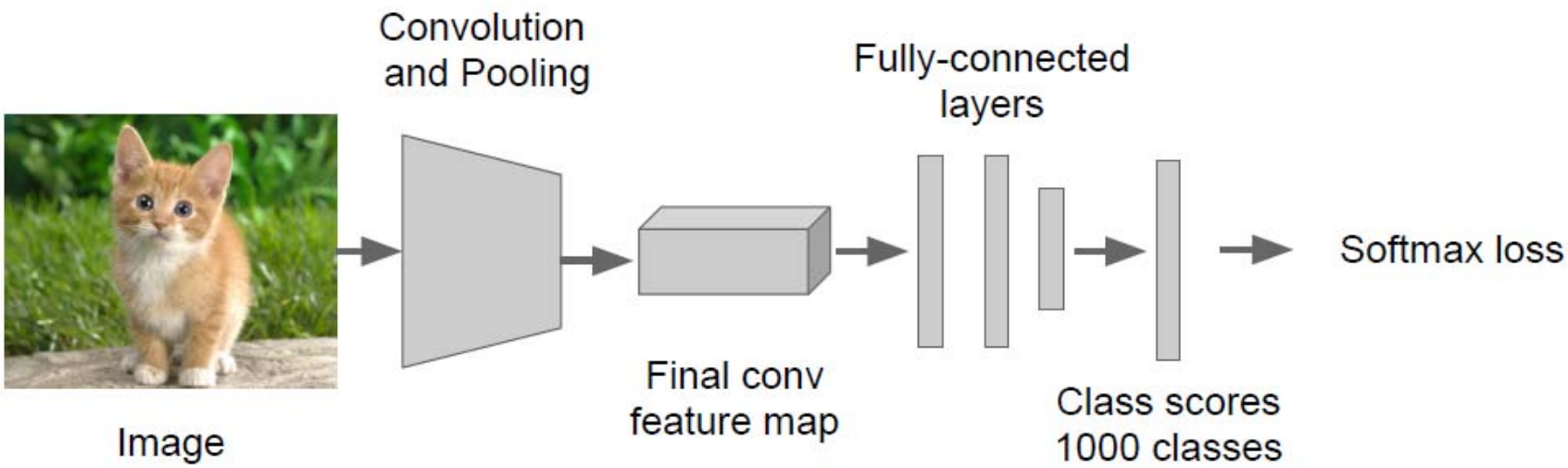


Girschick et al, "Rich feature hierarchies for accurate object detection and semantic segmentation", CVPR 2014

Slide credit: Ross Girschick

R-CNN Training

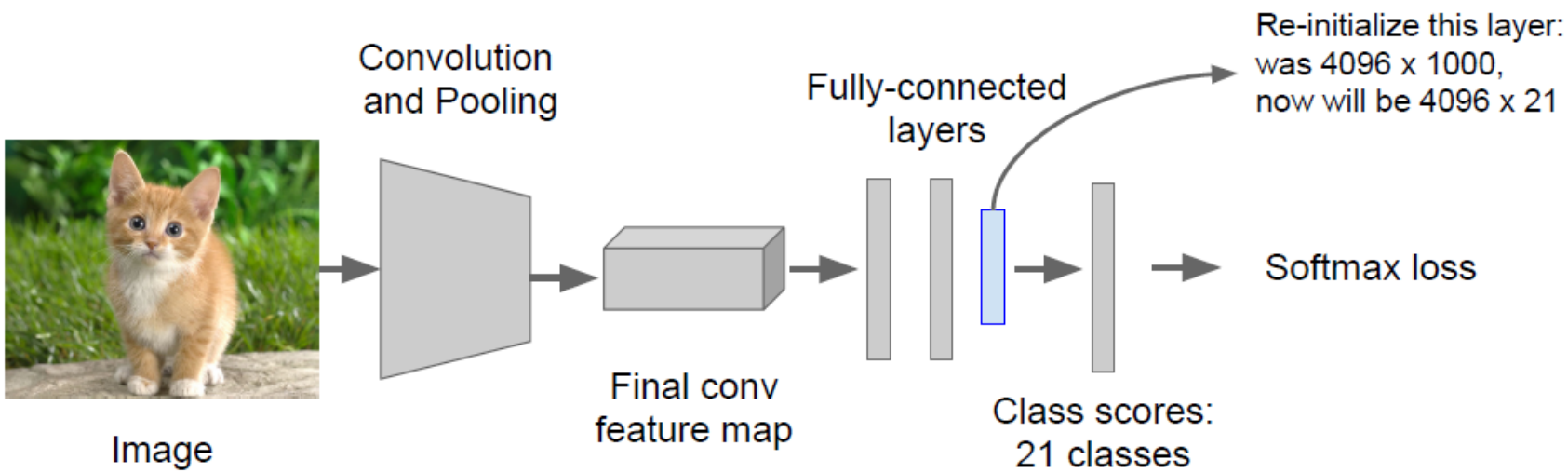
Step 1: Train (or download) a classification model for ImageNet (AlexNet)



R-CNN Training

Step 2: Fine-tune model for detection

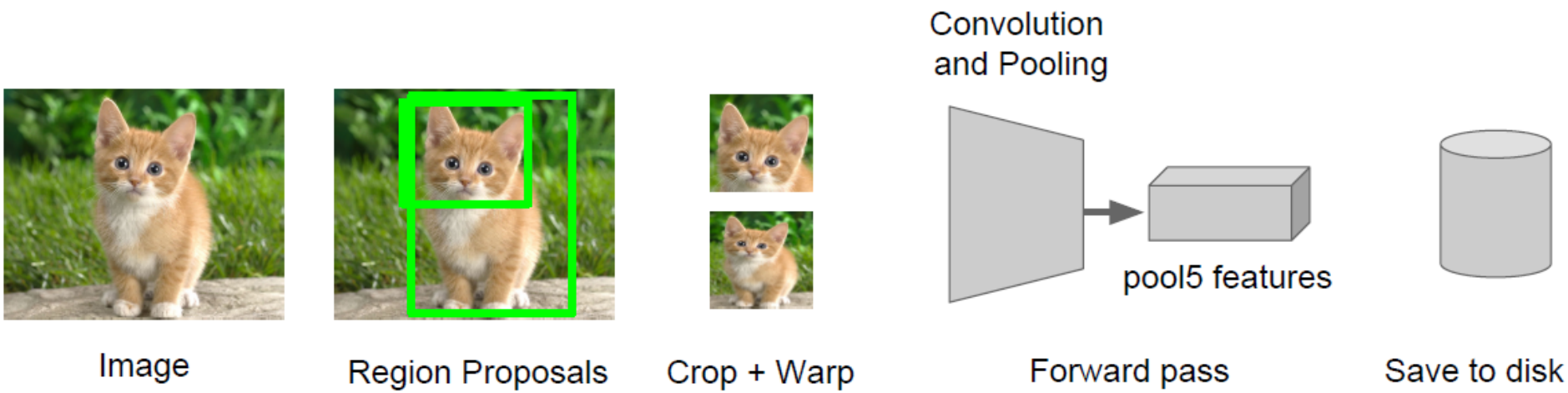
- Instead of 1000 ImageNet classes, want 20 object classes + background
- Throw away final fully-connected layer, reinitialize from scratch
- Keep training model using positive / negative regions from detection images



R-CNN Training

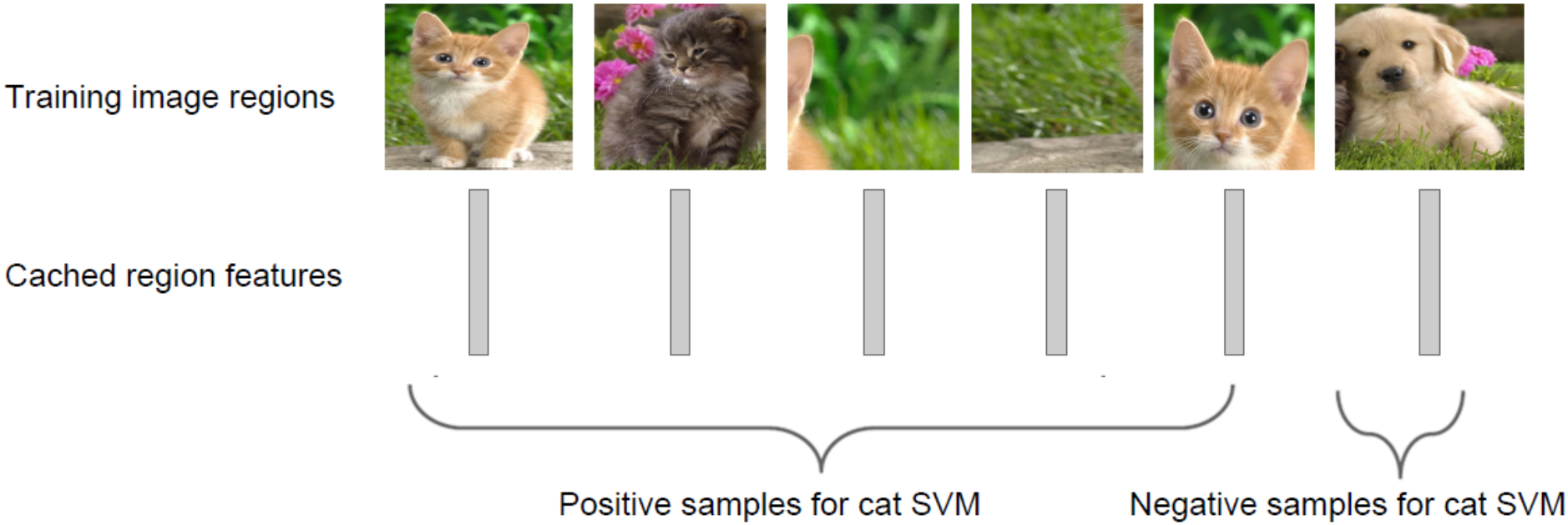
Step 3: Extract features

- 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!



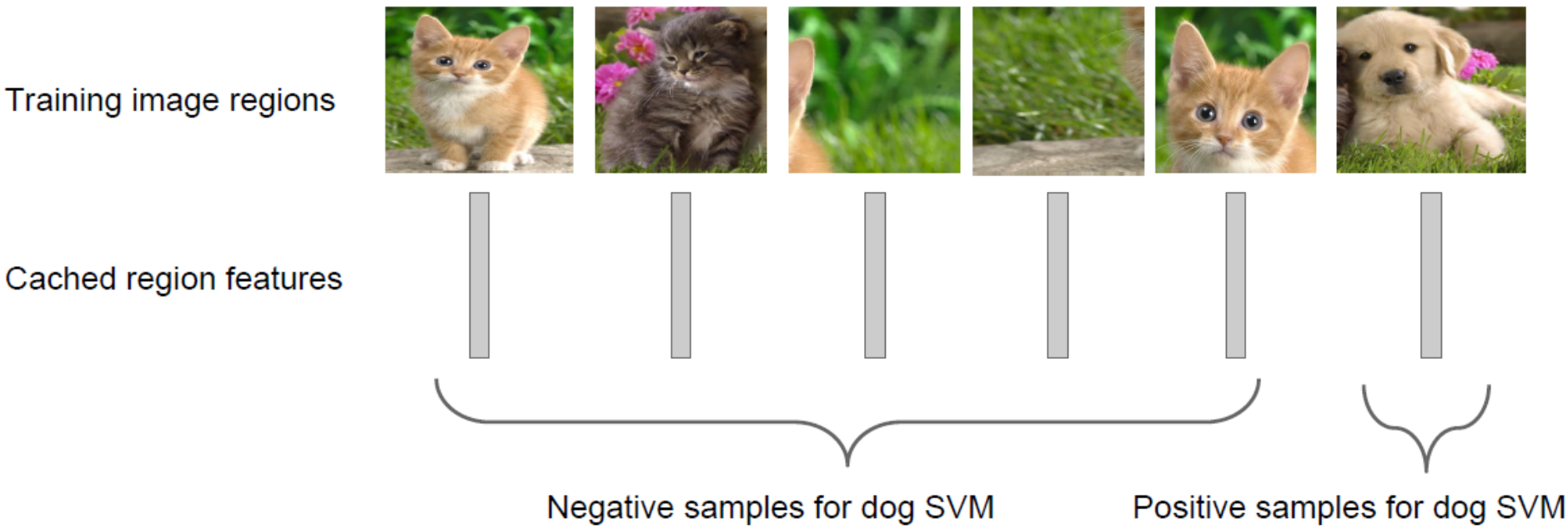
R-CNN Training

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

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

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**

Evaluation Metric

We use a metric called “mean average precision” (mAP)

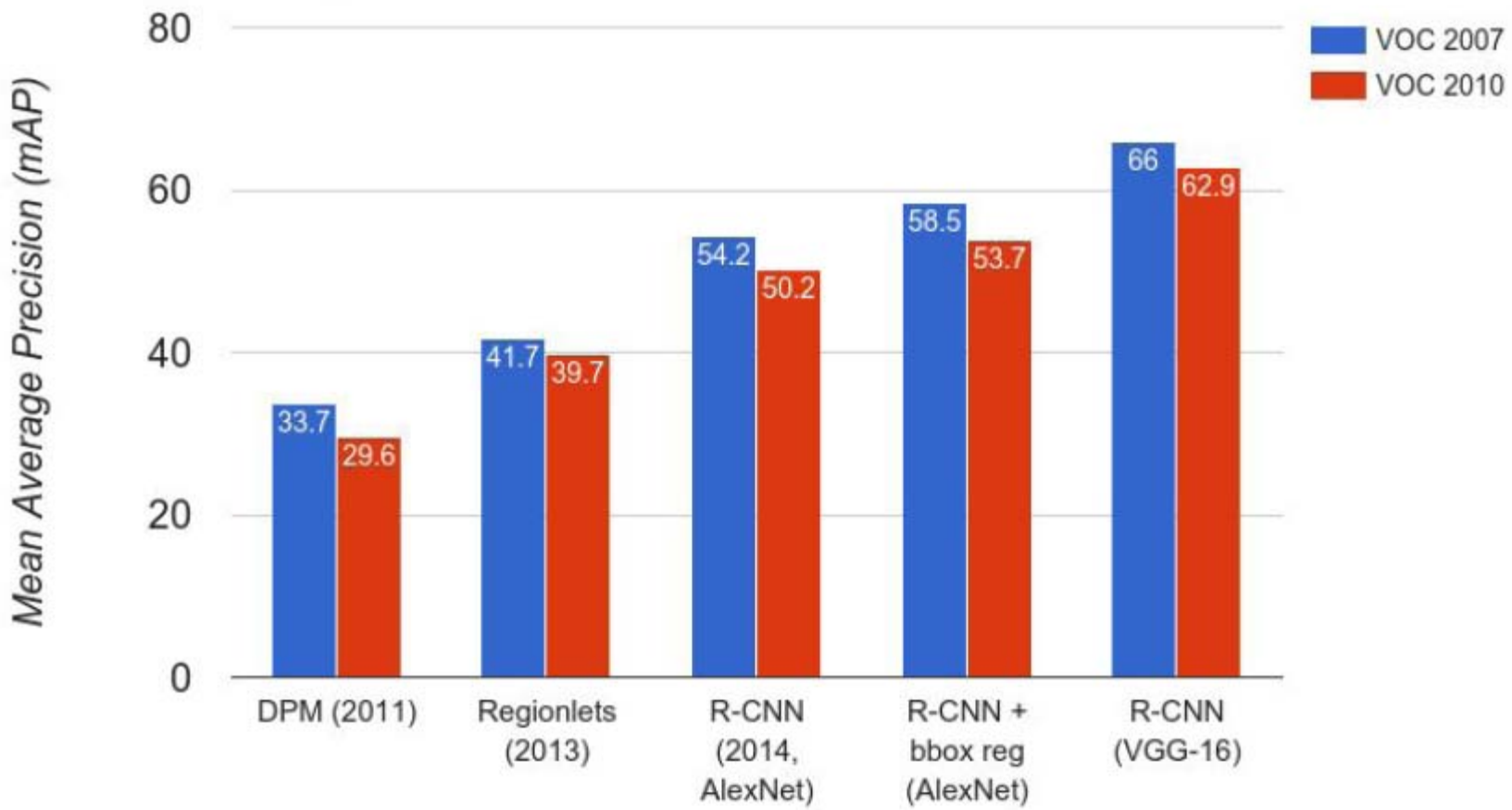
Compute average precision (AP) separately for each class, then average over classes

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)

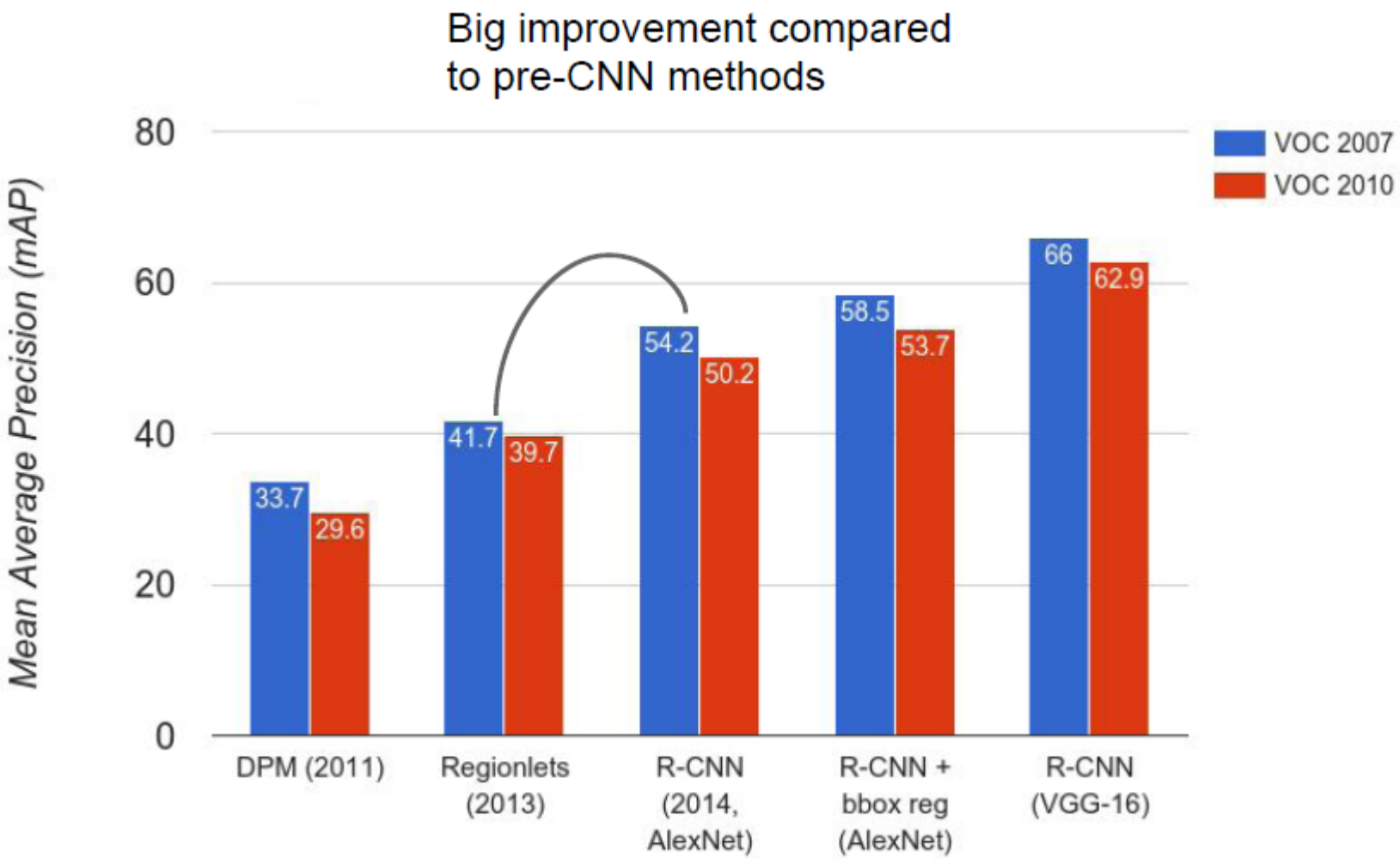
Combine all detections from all test images to draw a precision / recall curve for each class; AP is area under the curve

TL;DR mAP is a number from 0 to 100; high is good

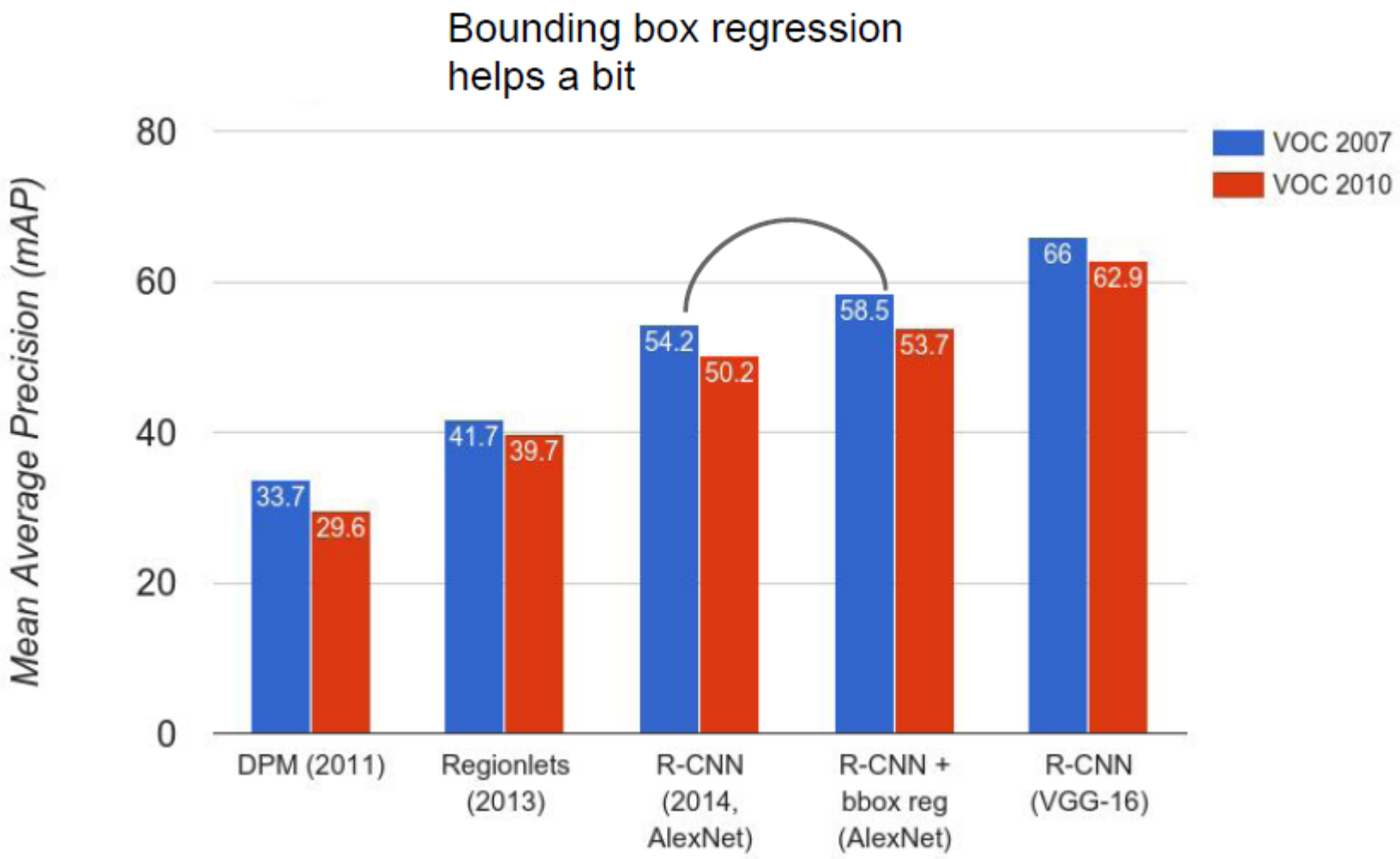
R-CNN Results



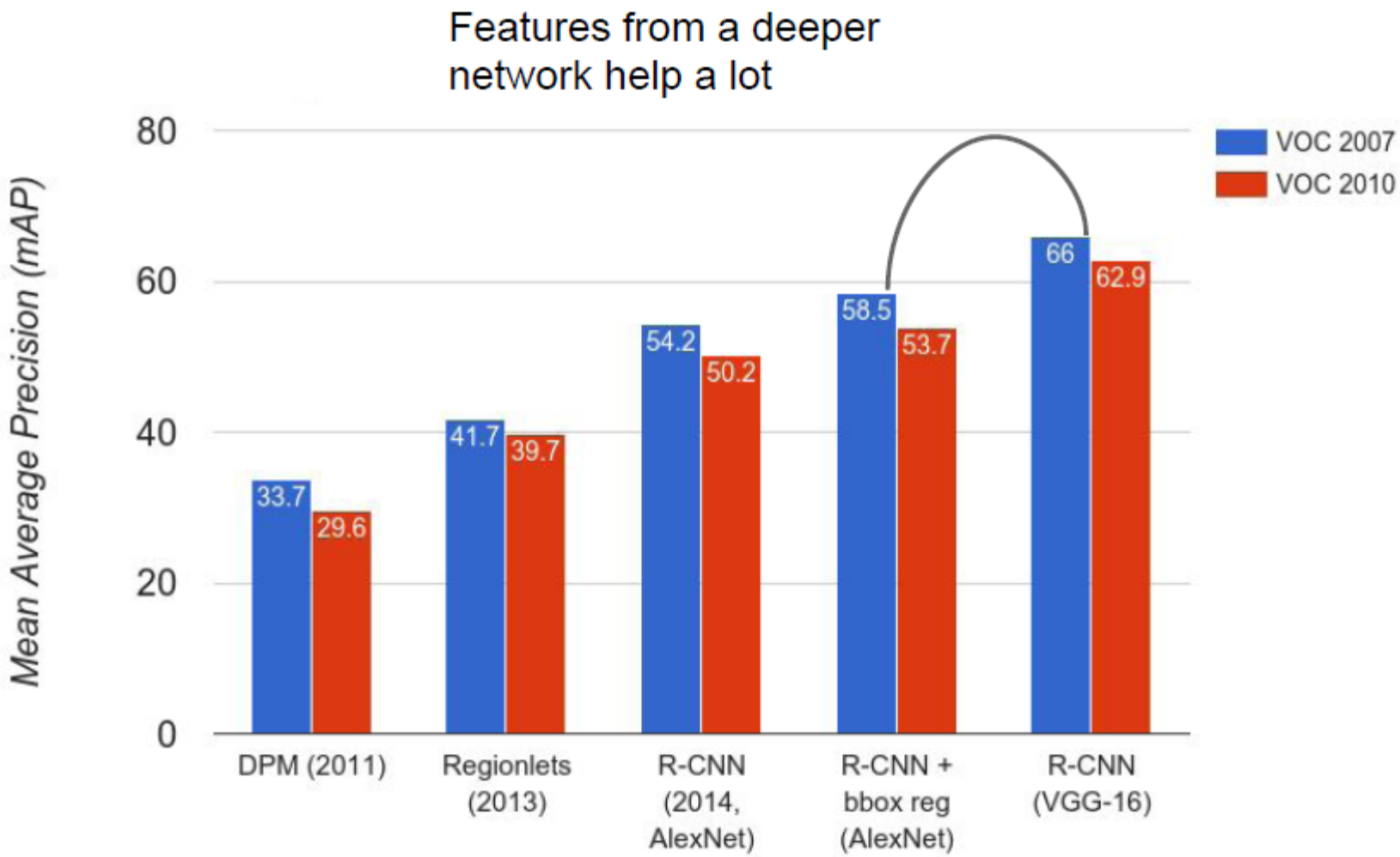
R-CNN Results



R-CNN Results



R-CNN Results

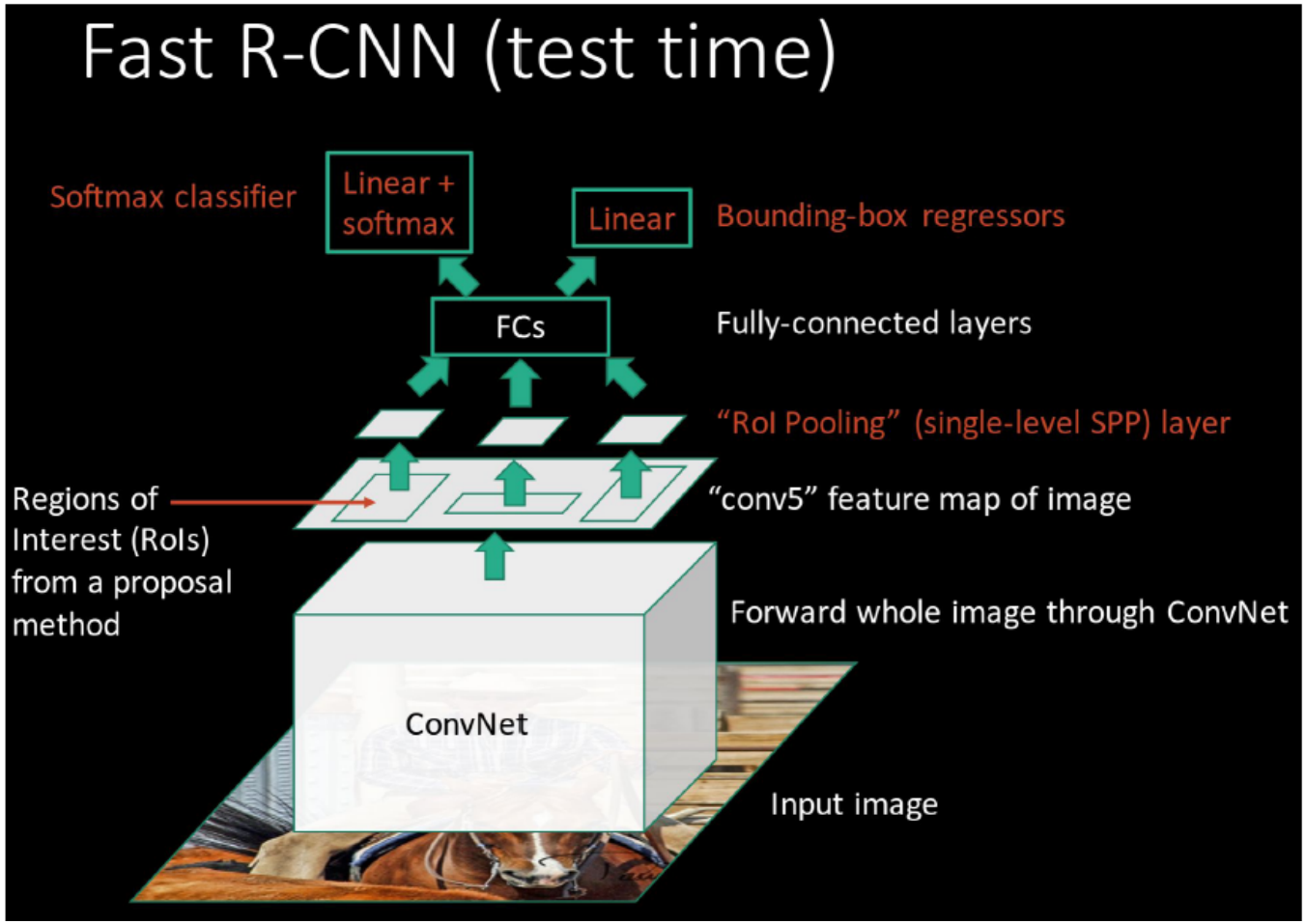


R-CNN Results

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

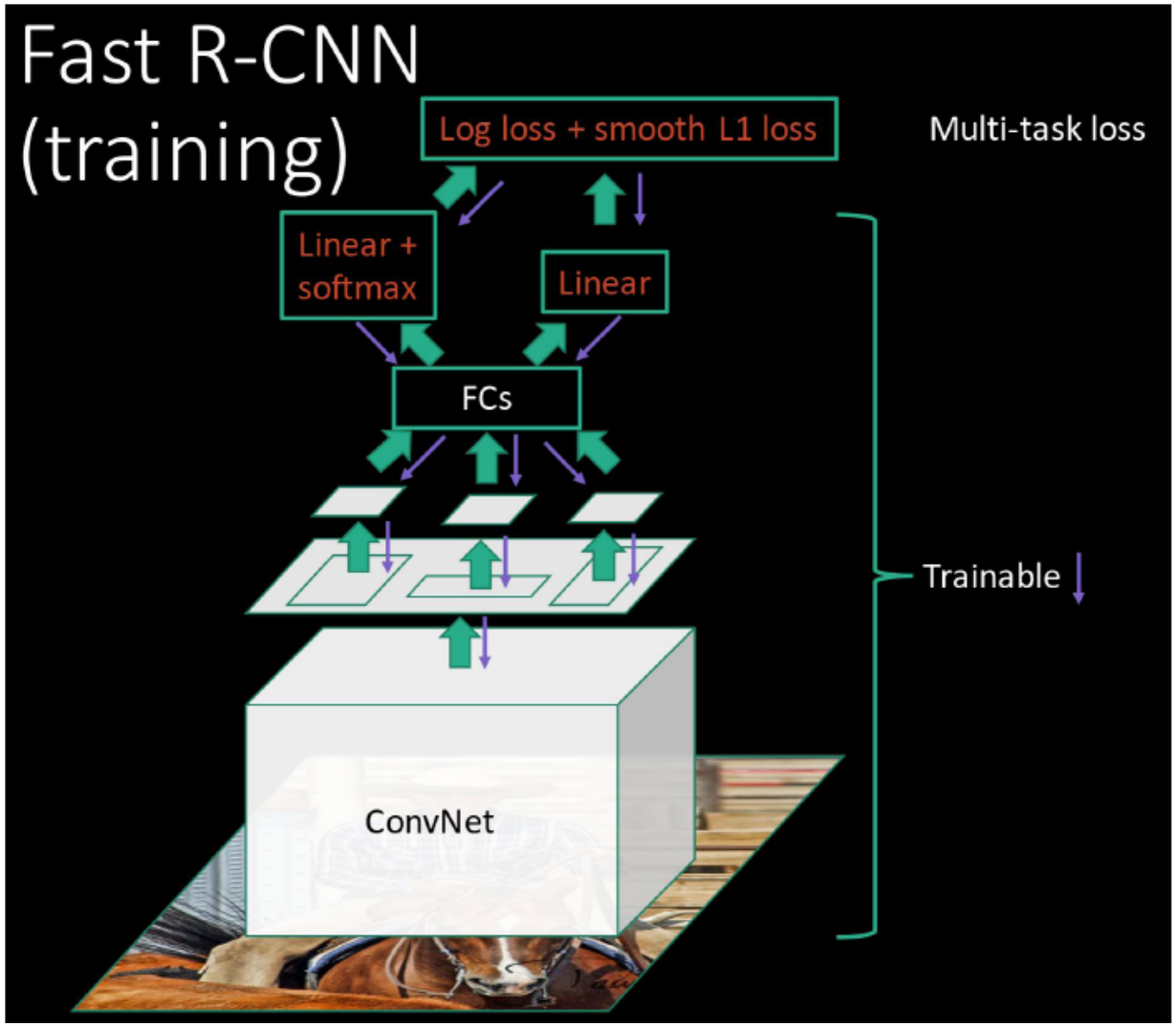
3. Complex multistage training pipeline



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

Slide Credit: Fei-Fei Li, Justin Johnson, Serena Yeung, CS 231n



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

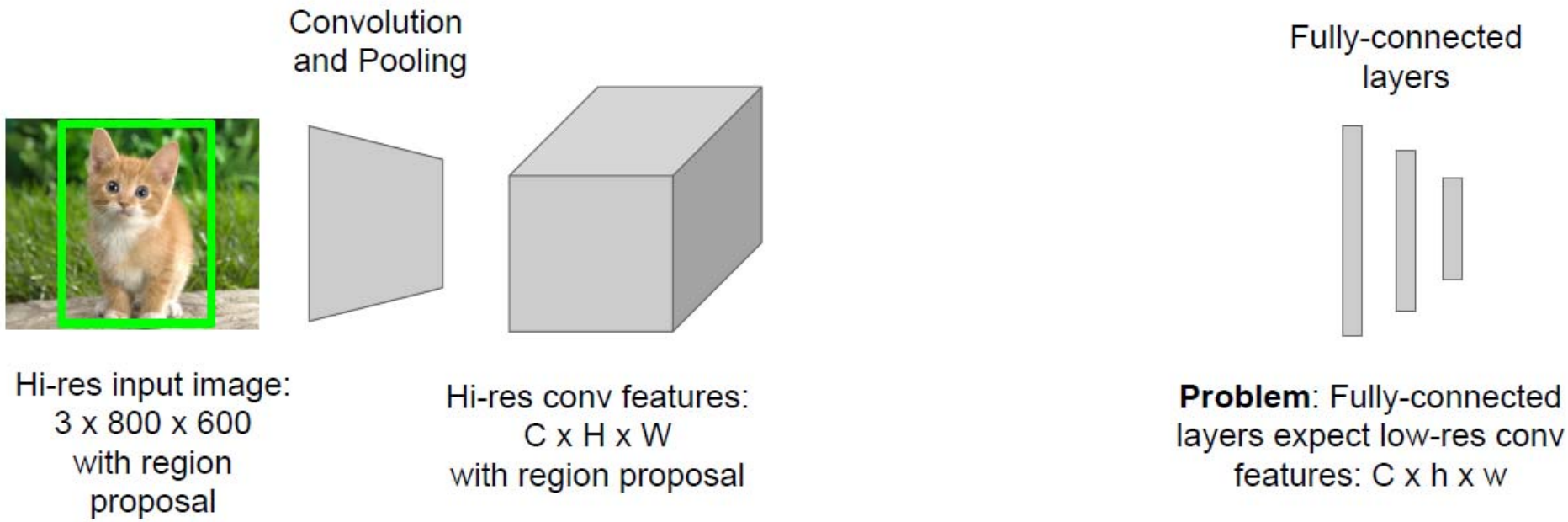
R-CNN Problem #3:
Complex training pipeline

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

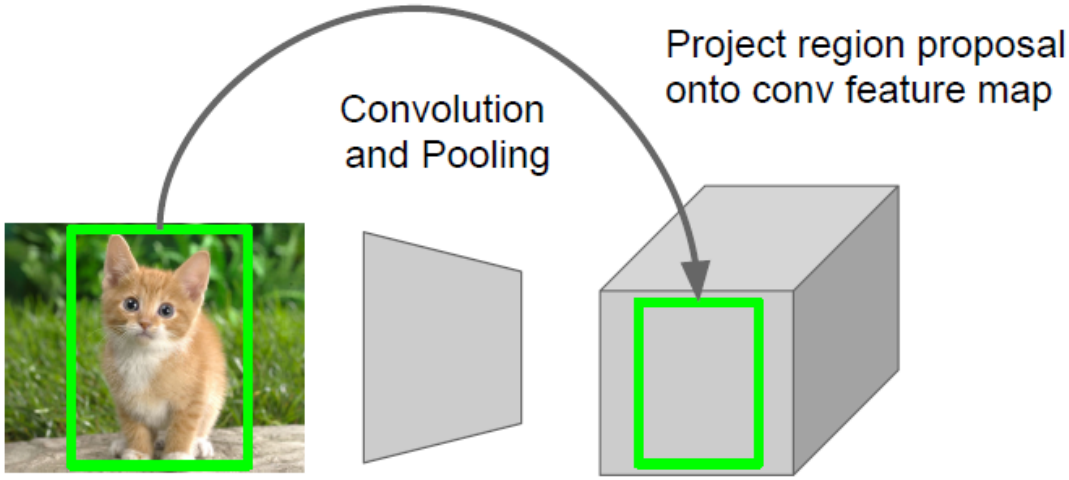
Slide credit: Ross Girschick

Slide Credit: Fei-Fei Li, Justin Johnson, Serena Yeung, CS 231n

Region of Interest (ROI) Pooling

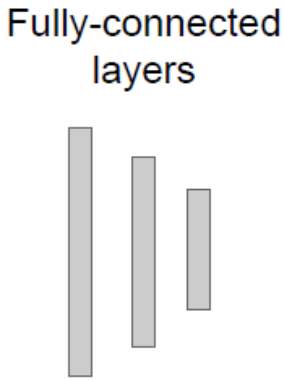


Region of Interest (ROI) Pooling



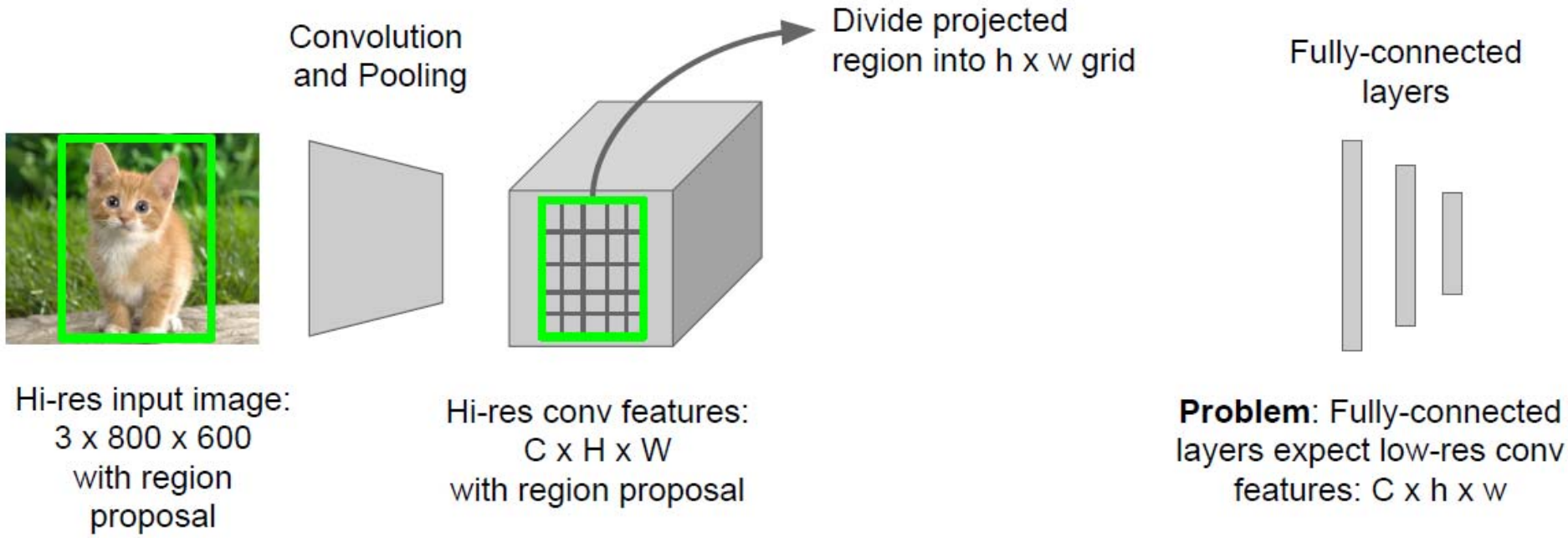
Hi-res input image:
3 x 800 x 600
with region
proposal

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

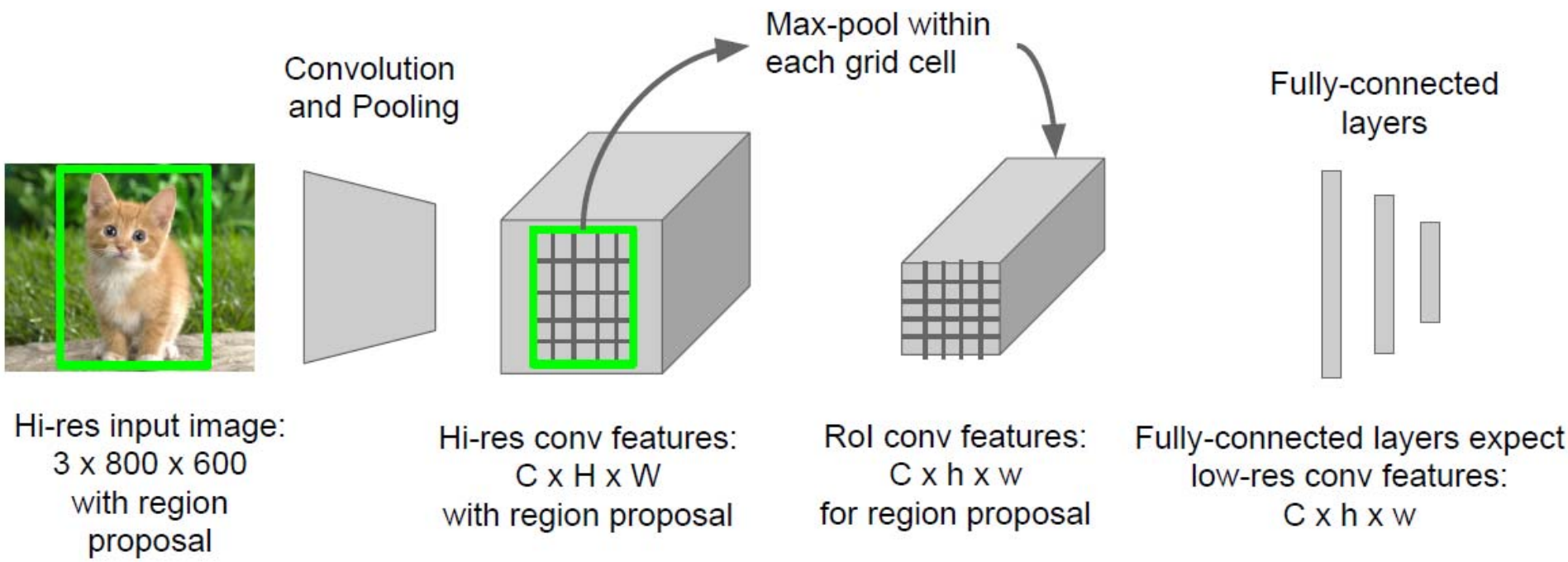


Problem: Fully-connected layers expect low-res conv features: C x h x w

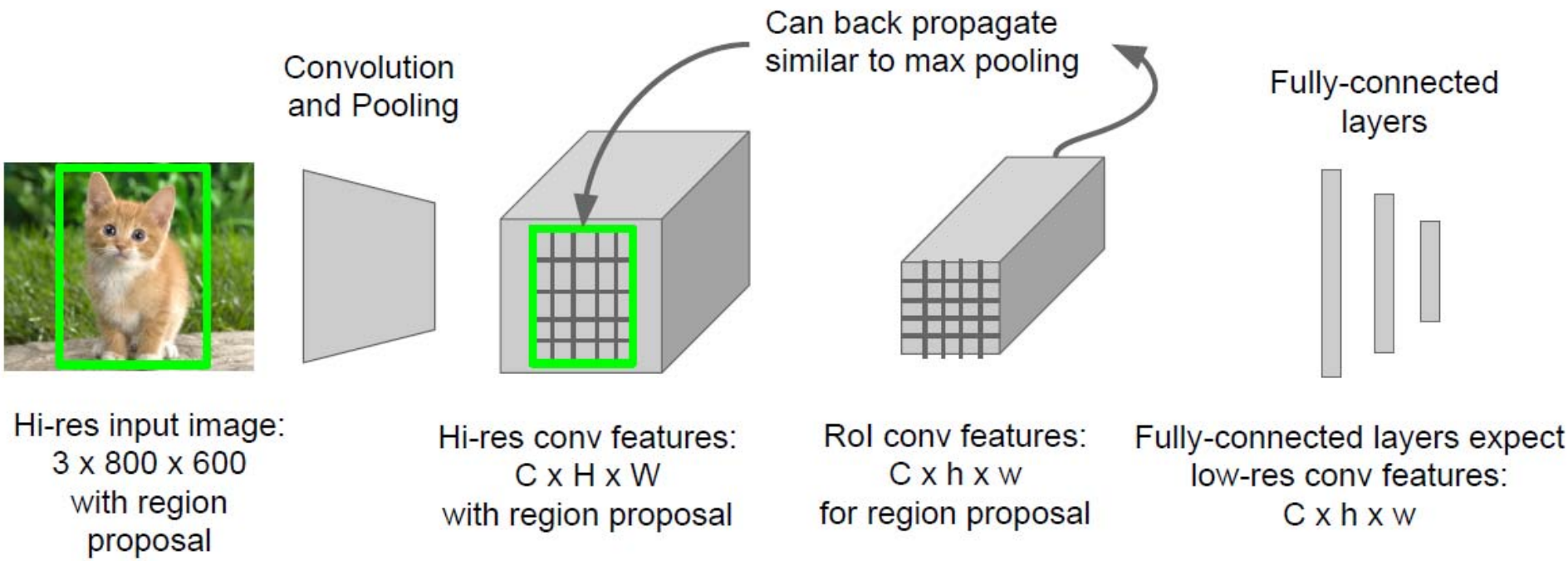
Region of Interest (ROI) Pooling



Region of Interest (ROI) Pooling



Region of Interest (ROI) Pooling



Fast R-CNN Results

Faster!

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

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

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**
Better!	mAP (VOC 2007)	66.0	**66.9**

Using VGG-16 CNN on Pascal VOC 2007 dataset

Fast R-CNN Results

Test-time speeds don't include region proposals

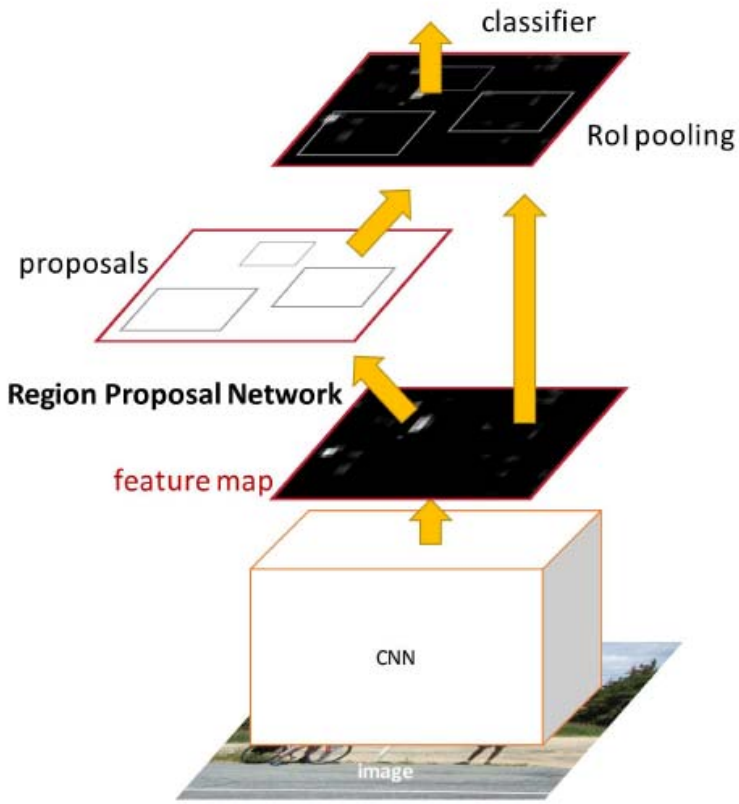
	R-CNN	**Fast R-CNN**
Test time per image	47 seconds	**0.32 seconds**
(Speedup)	1x	**146x**
Test time per image with Selective Search	50 seconds	**2 seconds**
(Speedup)	1x	**25x**

Fast R-CNN Results

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**

Faster R-CNN:



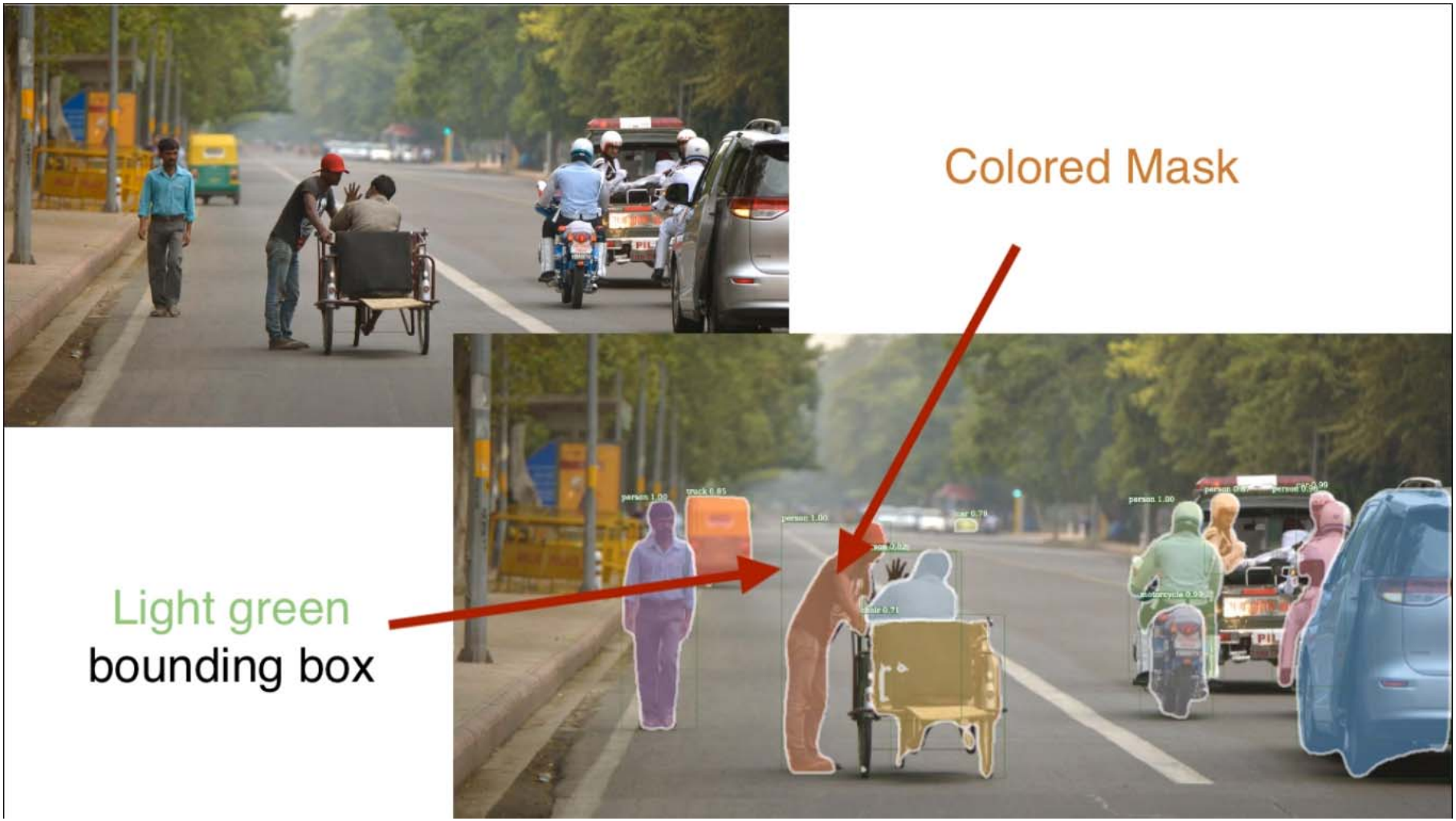
Insert a **Region Proposal Network (RPN)** after the last convolutional layer

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

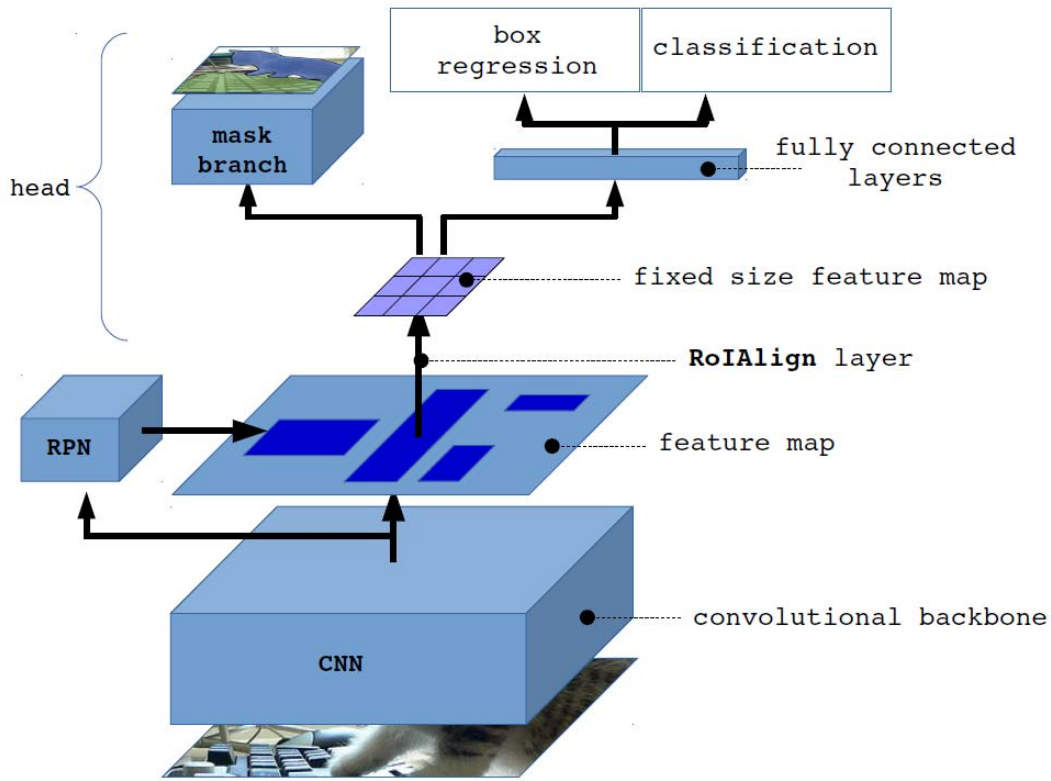
Ren et al, "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks", NIPS 2015

Slide credit: Ross Girschick



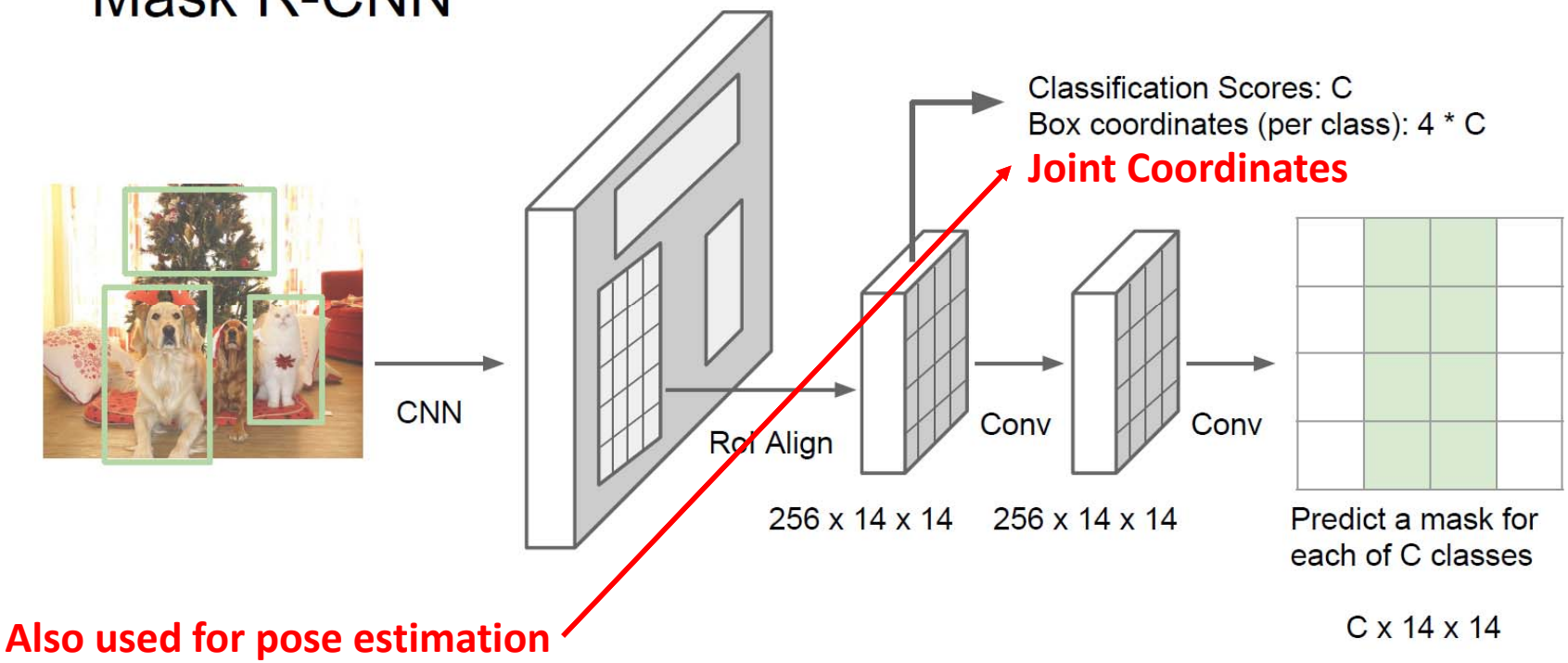
Slide Credit: Yacov Hel-Or

Mask R-CNN: overview



Slide Credit: Yacov Hel-Or

Mask R-CNN



https://www.youtube.com/watch?v=KYNDzIcQMWA

YOLO: You Only Look Once
Detection as Regression

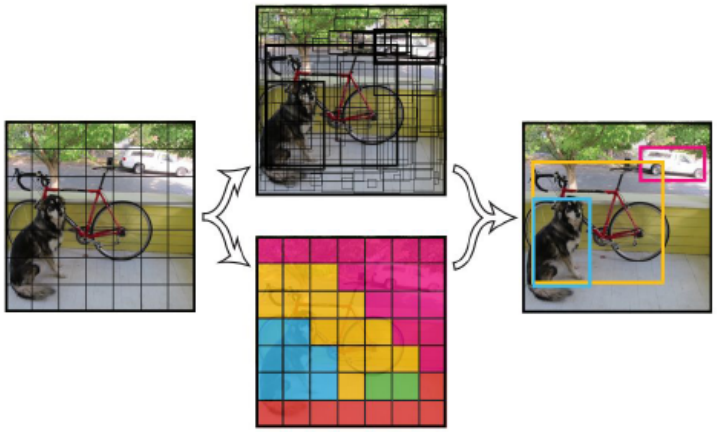
Divide image into S x S grid

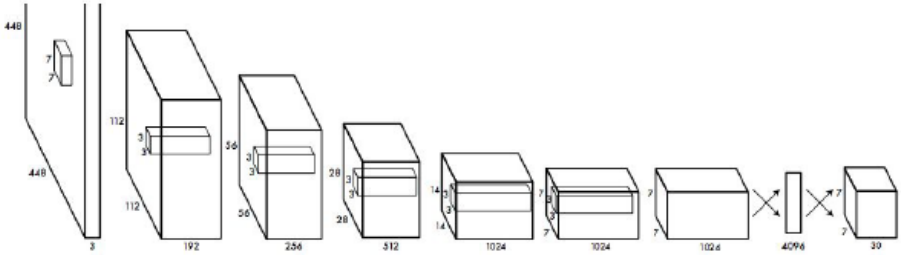
Within each grid cell predict:

- B Boxes: 4 coordinates + confidence
- Class scores: C numbers

Regression from image to
7 x 7 x (5 * B + C) tensor

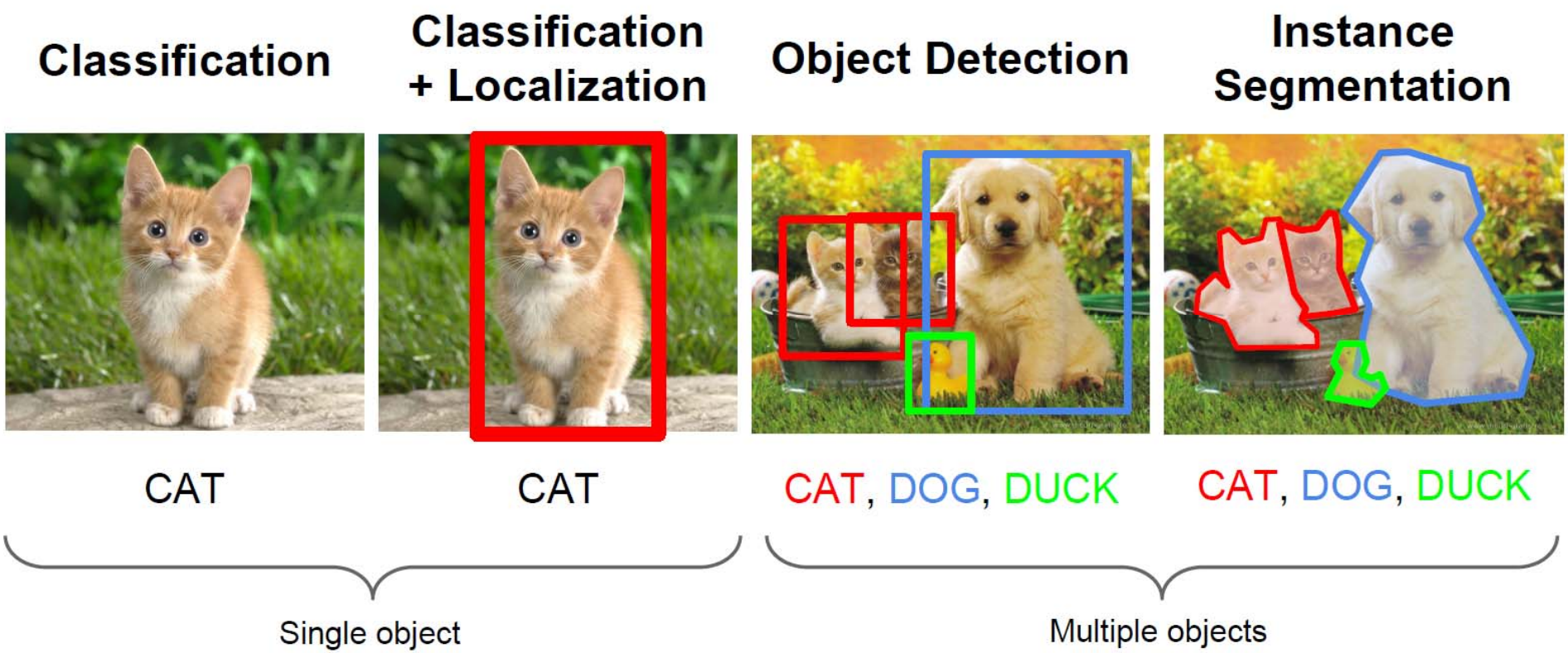
Direct prediction using a CNN





Redmon et al, "You Only Look Once:
Unified, Real-Time Object Detection", arXiv 2015

Computer Vision Tasks

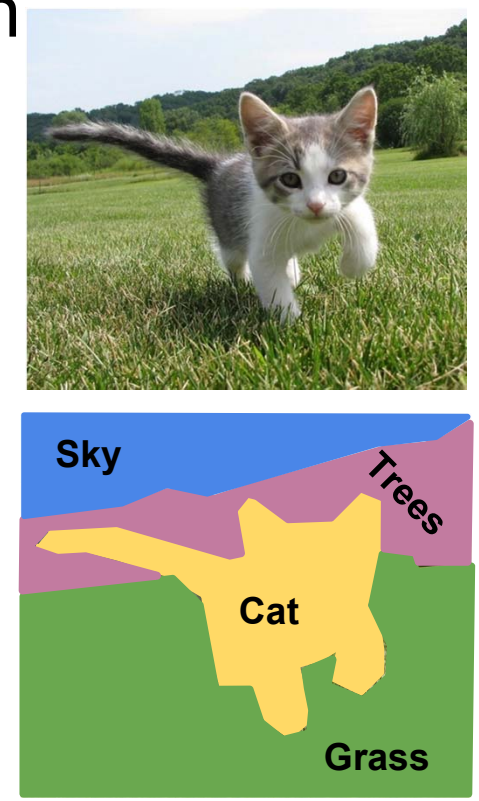


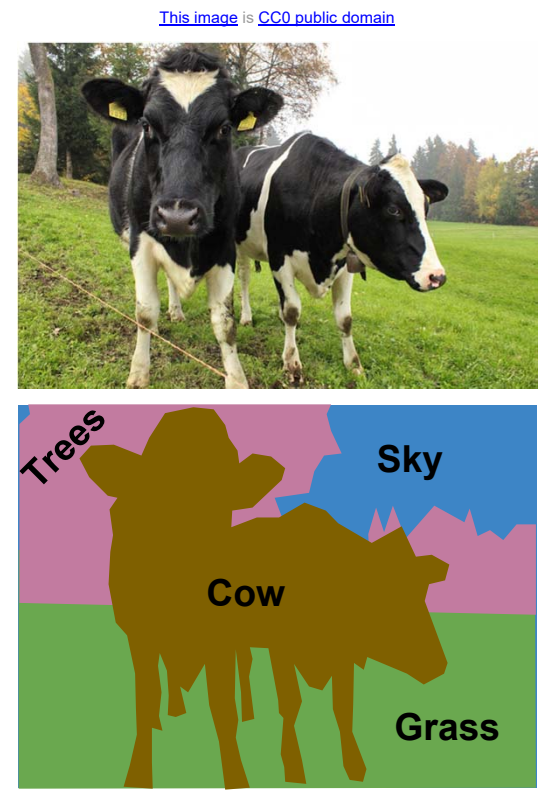
Slide Credit: Fei-Fei Li, Justin Johnson, Serena Yeung, CS 231n

Semantic Segmentation

Label each pixel in the image with a category label

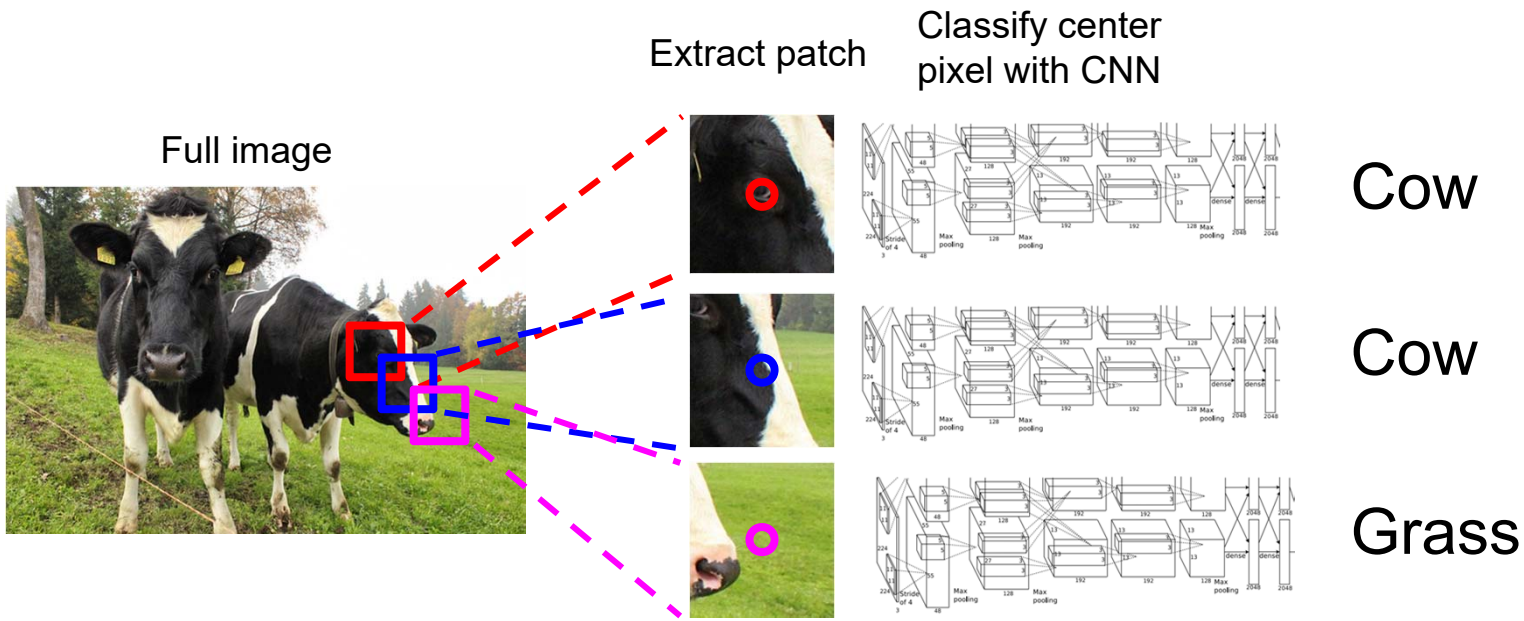
Don’t differentiate instances, only care about pixels





Slide Credit: Fei-Fei Li, Justin Johnson, Serena Yeung, CS 231n

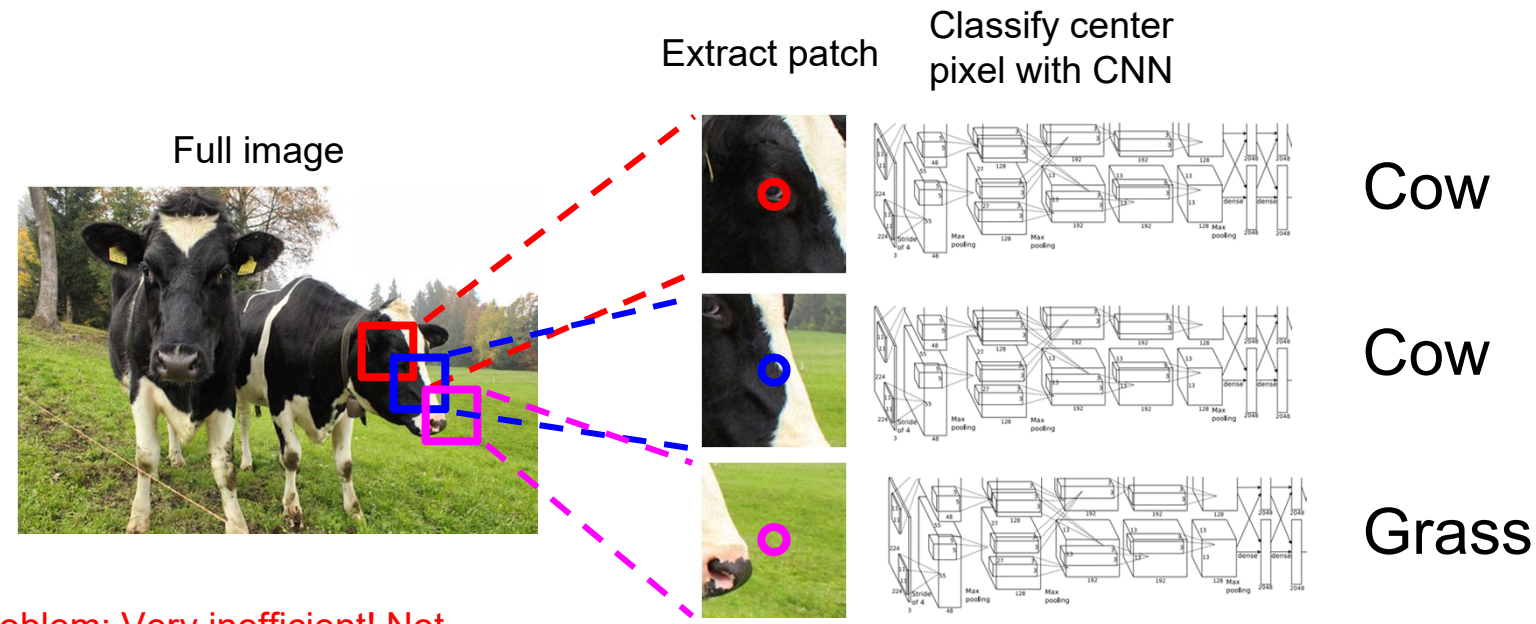
Semantic Segmentation Idea: Sliding Window



Farabet et al, “Learning Hierarchical Features for Scene Labeling,” TPAMI 2013
Pinheiro and Collobert, “Recurrent Convolutional Neural Networks for Scene Labeling”, ICML 2014

Slide Credit: Fei-Fei Li, Justin Johnson, Serena Yeung, CS 231n

Semantic Segmentation Idea: Sliding Window

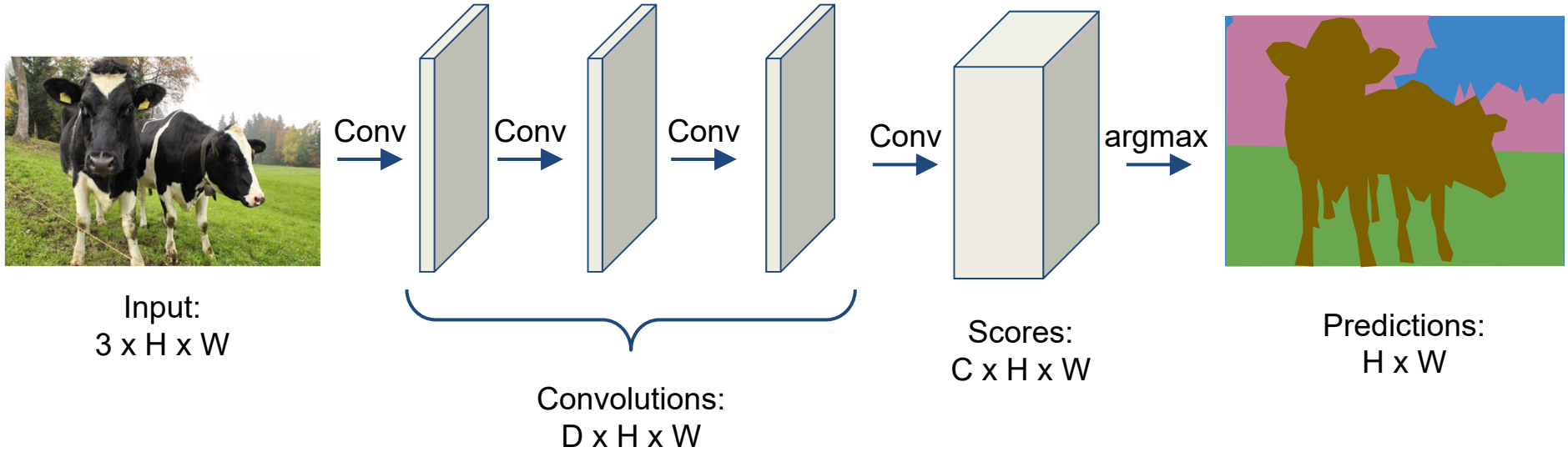


Problem: Very inefficient! Not reusing shared features between overlapping patches

Farabet et al, “Learning Hierarchical Features for Scene Labeling,” TPAMI 2013
Pinheiro and Collobert, “Recurrent Convolutional Neural Networks for Scene Labeling”, ICML 2014

Semantic Segmentation Idea: Fully Convolutional

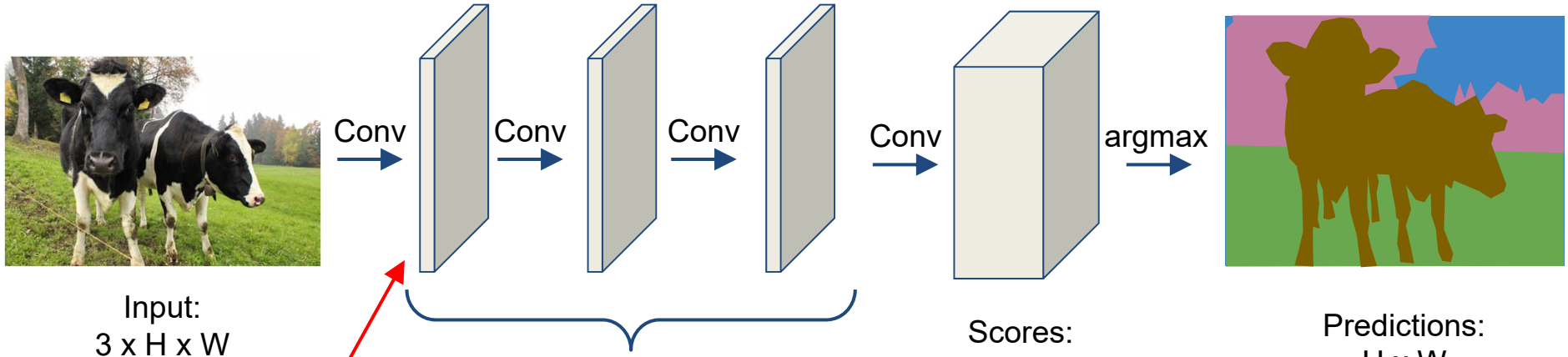
Design a network as a bunch of convolutional layers
to make predictions for pixels all at once!



Slide Credit: Fei-Fei Li, Justin Johnson, Serena Yeung, CS 231n

Semantic Segmentation Idea: Fully Convolutional

Design a network as a bunch of convolutional layers to make predictions for pixels all at once!



Problem: convolutions at original image resolution will be very expensive ...

Slide Credit: Fei-Fei Li, Justin Johnson, Serena Yeung, CS 231n

Dilated Convolutions

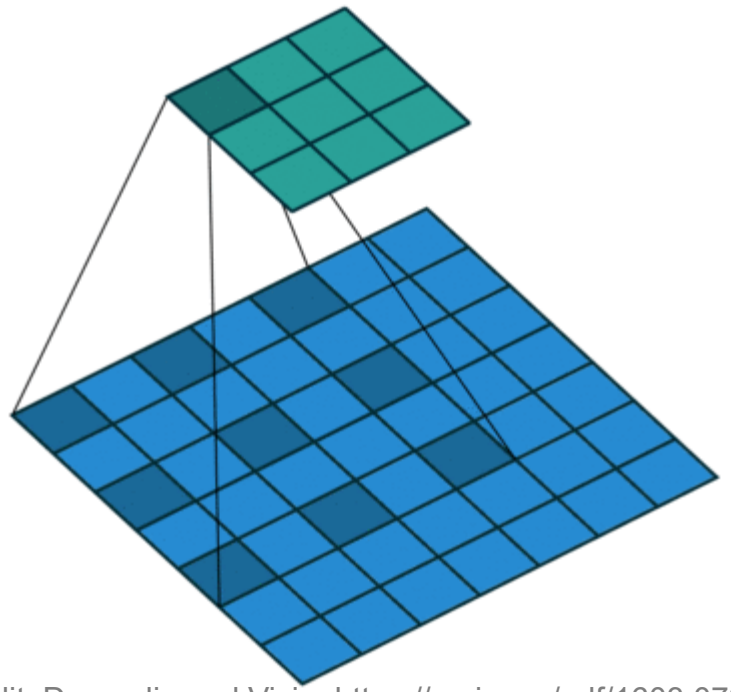
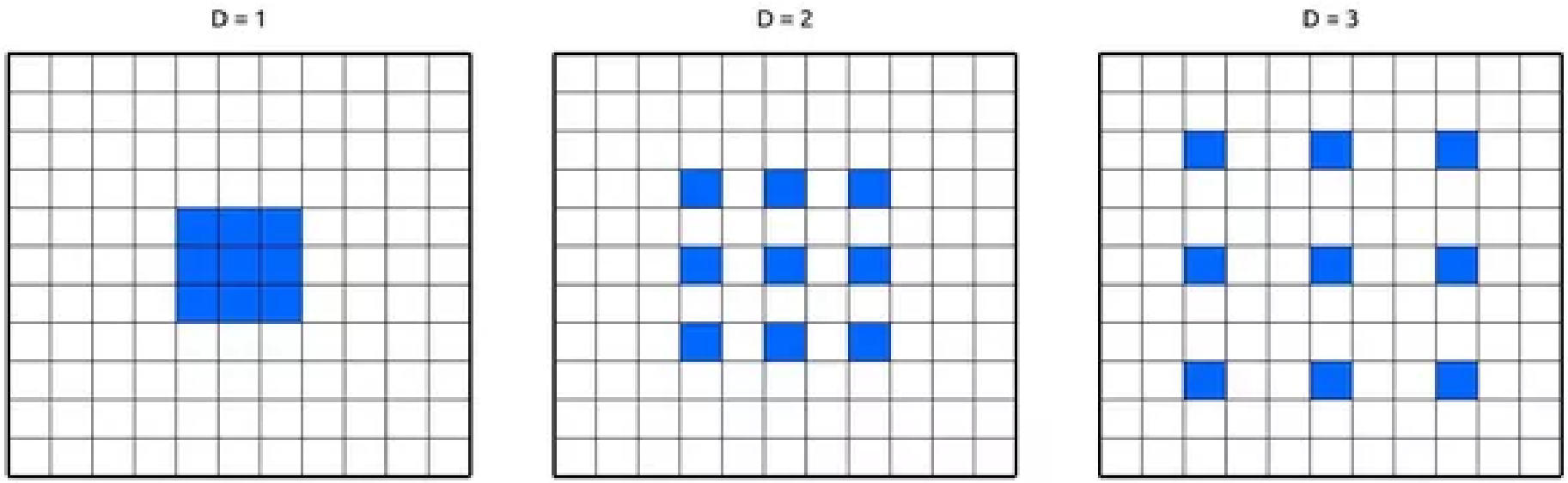


Figure Credit: Dumoulin and Visin, https://arxiv.org/pdf/1603.07285.pdf

Dilated Convolutions (Examples)



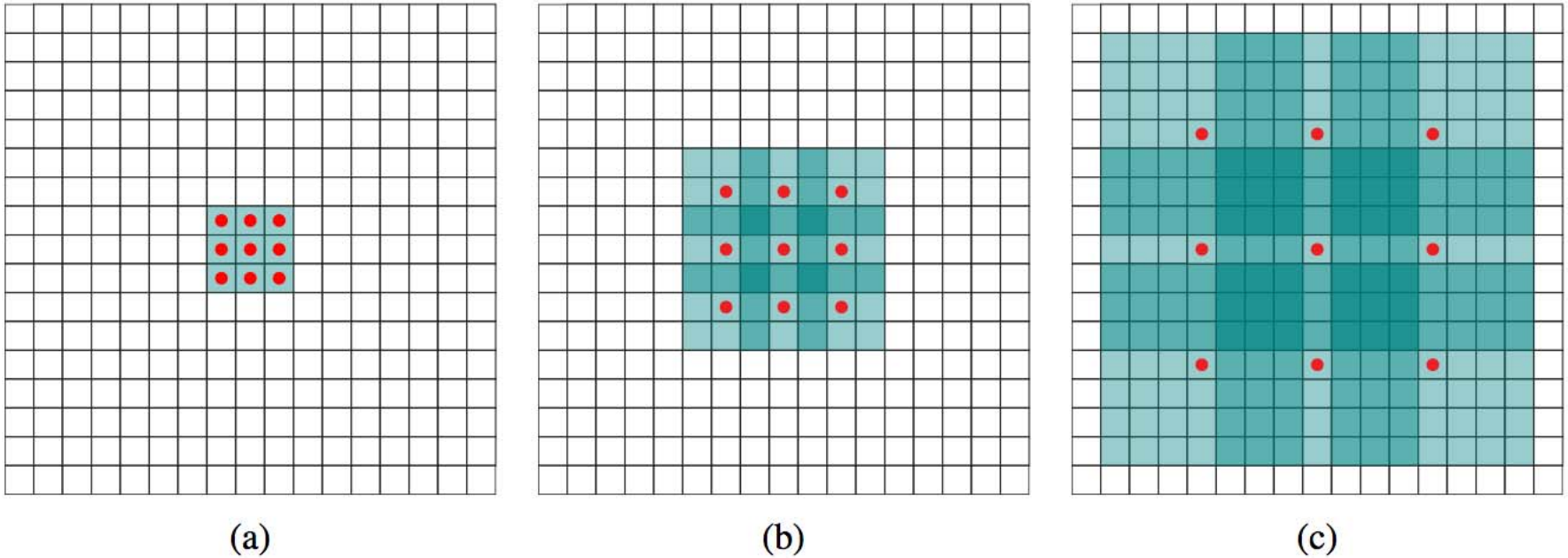


Figure 1: Systematic dilation supports exponential expansion of the receptive field without loss of resolution or coverage. (a) F_1 is produced from F_0 by a 1-dilated convolution; each element in F_1 has a receptive field of 3×3. (b) F_2 is produced from F_1 by a 2-dilated convolution; each element in F_2 has a receptive field of 7×7. (c) F_3 is produced from F_2 by a 4-dilated convolution; each element in F_3 has a receptive field of 15×15. The number of parameters associated with each layer is identical. The receptive field grows exponentially while the number of parameters grows linearly.

Figure Credit: Yu and Koltun, ICLR16

Different Idea: Still Classify, Change Test Time

- Can run on an image of any size!

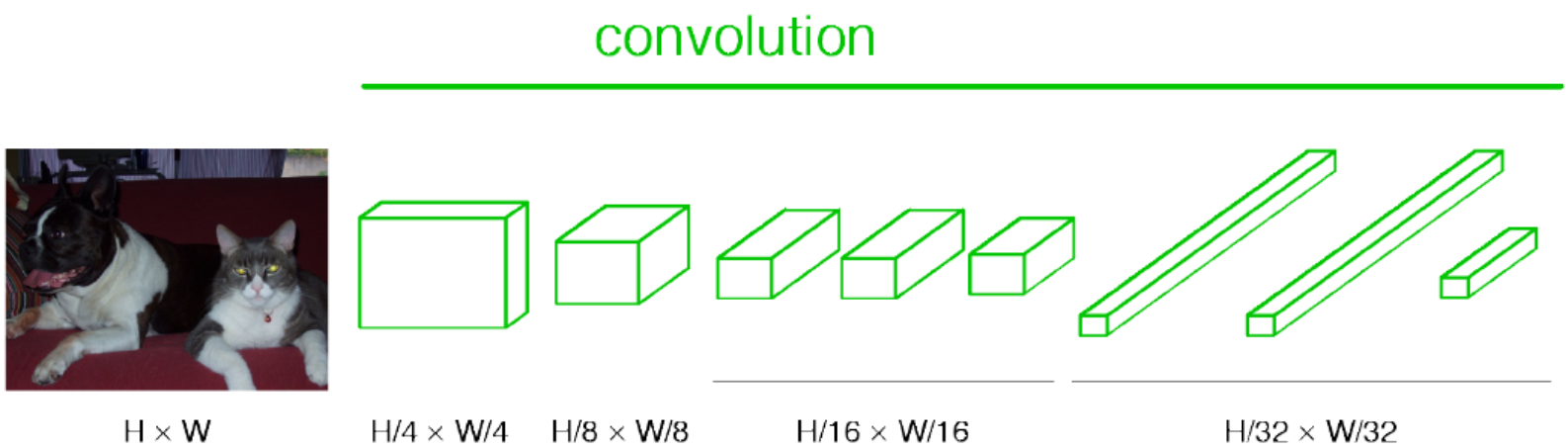
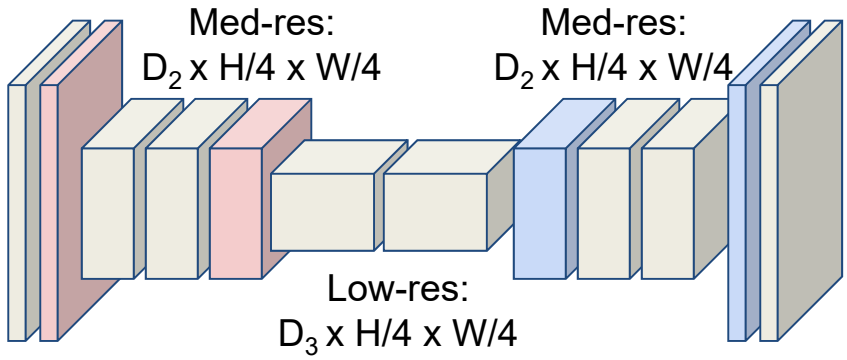


Figure Credit: [Long, Shelhamer, Darrell CVPR15]

Better Semantic Segmentation Idea: Fully Convolutional with Bottleneck

Design network as a bunch of convolutional layers, with **downsampling** and **upsampling** inside the network!

Input:
3 x H x W



High-res:
D_1 x H/2 x W/2

High-res:
D_1 x H/2 x W/2

Predictions:
H x W

Long, Shelhamer, and Darrell, "Fully Convolutional Networks for Semantic Segmentation", CVPR 2015
Noh et al, "Learning Deconvolution Network for Semantic Segmentation", ICCV 2015

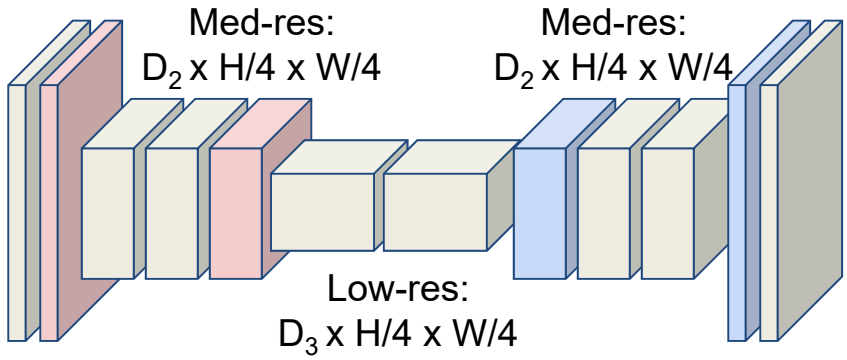
Semantic Segmentation Idea: Fully Convolutional

Downsampling:
Pooling, strided convolution

Design network as a bunch of convolutional layers, with **downsampling** and **upsampling** inside the network!

Upsampling:
???

Input:
3 x H x W



High-res:
D_1 x H/2 x W/2

High-res:
D_1 x H/2 x W/2

Predictions:
H x W

Long, Shelhamer, and Darrell, "Fully Convolutional Networks for Semantic Segmentation", CVPR 2015
Noh et al, "Learning Deconvolution Network for Semantic Segmentation", ICCV 2015

In-Network upsampling: “Unpooling”

Nearest Neighbor

1	2
3	4

Input: 2 x 2

1	1	2	2
1	1	2	2
3	3	4	4
3	3	4	4

Output: 4 x 4

“Bed of Nails”

1	2
3	4

Input: 2 x 2

1	0	2	0
0	0	0	0
3	0	4	0
0	0	0	0

Output: 4 x 4

Slide Credit: Fei-Fei Li, Justin Johnson, Serena Yeung, CS 231n

In-Network upsampling: “Max Unpooling”

Max Pooling
Remember which element was max!

1	2	6	3
3	5	2	1
1	2	2	1
7	3	4	8

Input: 4 x 4

5	6
7	8

Output: 2 x 2

. . .

Rest of the network

Max Unpooling
Use positions from pooling layer

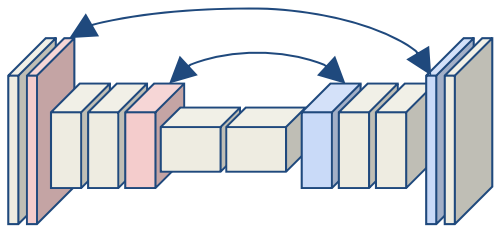
1	2
3	4

Input: 2 x 2

0	0	2	0
0	1	0	0
0	0	0	0
3	0	0	4

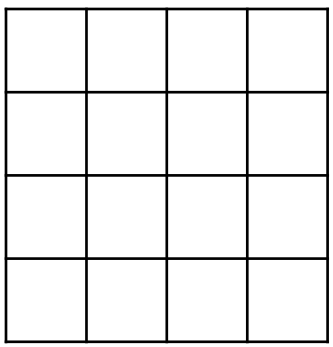
Output: 4 x 4

Corresponding pairs of downsampling and upsampling layers

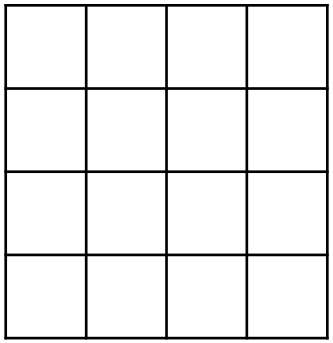


Learnable Upsampling: Transpose Convolution

Recall: Typical 3 x 3 convolution, stride 1 pad 1



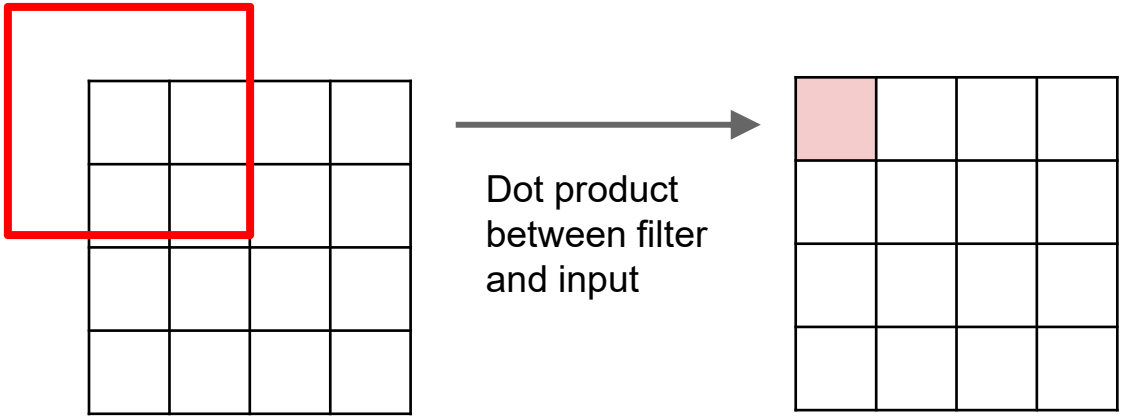
Input: 4 x 4



Output: 4 x 4

Learnable Upsampling: Transpose Convolution

Recall: Normal 3 x 3 convolution, stride 1 pad 1



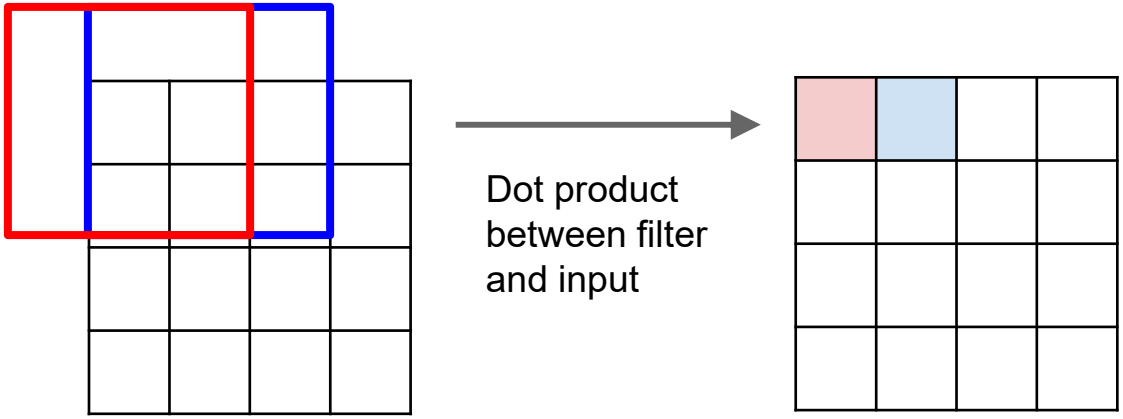
Input: 4 x 4

Output: 4 x 4

Slide Credit: Fei-Fei Li, Justin Johnson, Serena Yeung, CS 231n

Learnable Upsampling: Transpose Convolution

Recall: Normal 3 x 3 convolution, stride 1 pad 1



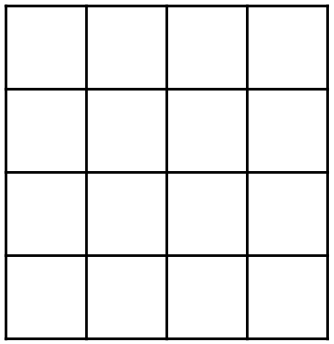
Input: 4 x 4

Output: 4 x 4

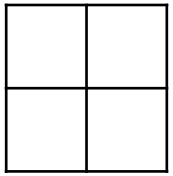
Slide Credit: Fei-Fei Li, Justin Johnson, Serena Yeung, CS 231n

Learnable Upsampling: Transpose Convolution

Recall: Normal 3 x 3 convolution, <u>stride 2</u> pad 1



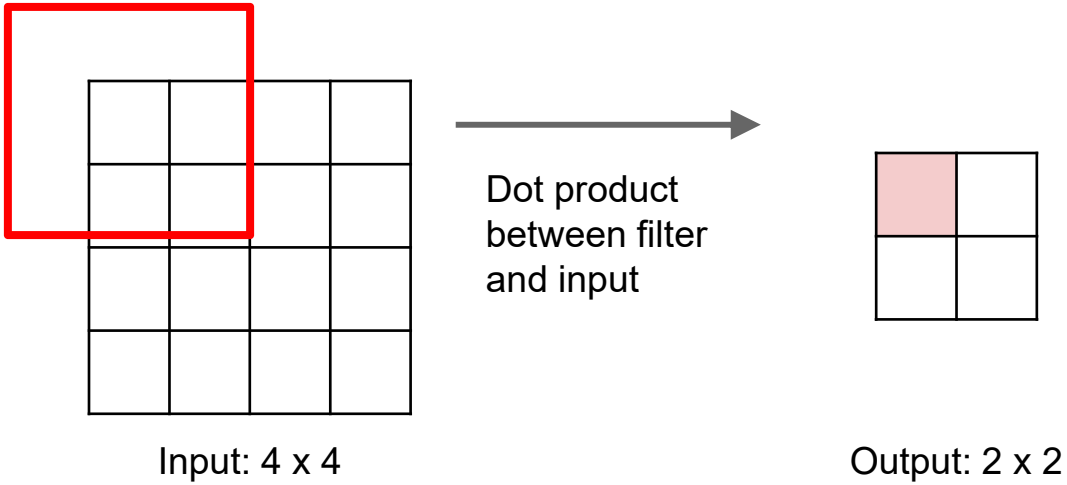
Input: 4 x 4



Output: 2 x 2

Learnable Upsampling: Transpose Convolution

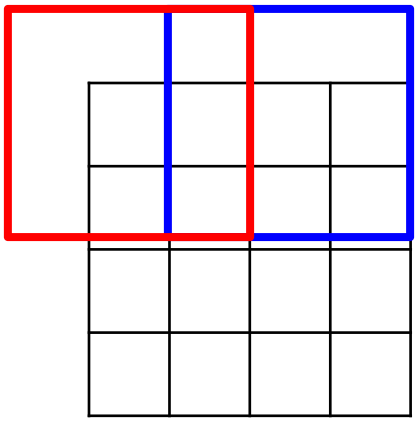
Recall: Normal 3 x 3 convolution, stride 2 pad 1



Slide Credit: Fei-Fei Li, Justin Johnson, Serena Yeung, CS 231n

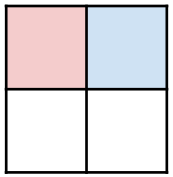
Learnable Upsampling: Transpose Convolution

Recall: Normal 3 x 3 convolution, <u>stride 2</u> pad 1



Input: 4 x 4

Dot product between filter and input



Output: 2 x 2

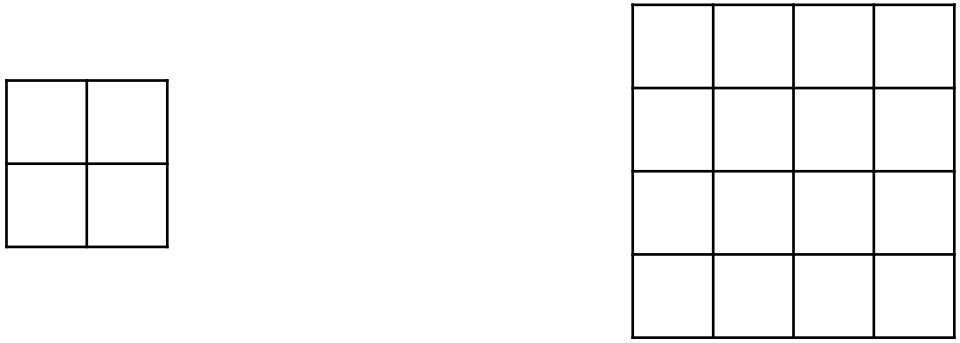
Filter moves 2 pixels in the input for every one pixel in the output

Stride gives ratio between movement in input and output

Slide Credit: Fei-Fei Li, Justin Johnson, Serena Yeung, CS 231n

Learnable Upsampling: Transpose Convolution

3 x 3 **transpose** convolution, stride 2 pad 1



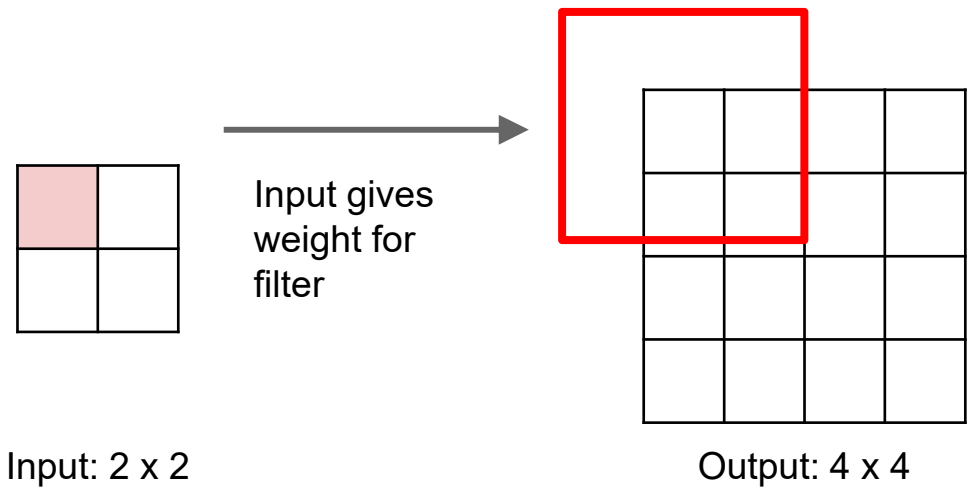
Input: 2 x 2

Output: 4 x 4

Slide Credit: Fei-Fei Li, Justin Johnson, Serena Yeung, CS 231n

Learnable Upsampling: Transpose Convolution

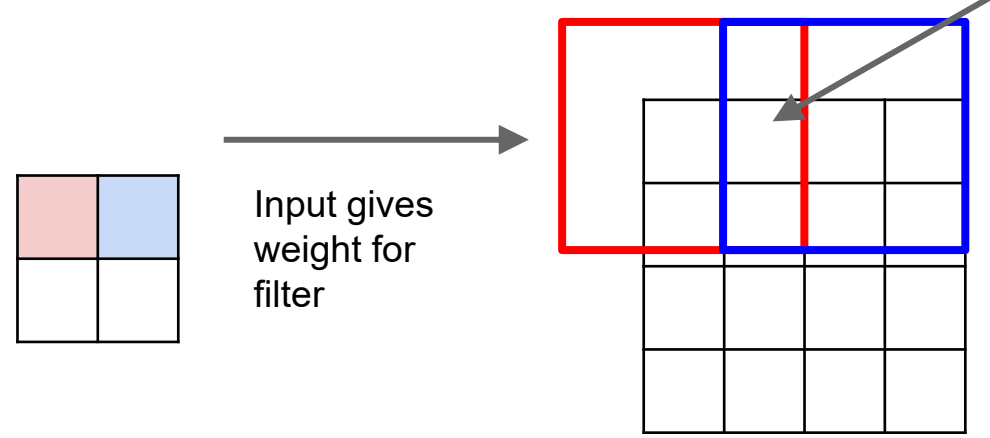
3 x 3 **transpose** convolution, stride 2 pad 1



Slide Credit: Fei-Fei Li, Justin Johnson, Serena Yeung, CS 231n

Learnable Upsampling: Transpose Convolution

3 x 3 **transpose** convolution, stride 2 pad 1



Sum where output overlaps

Filter moves 2 pixels in the <u>output</u> for every one pixel in the <u>input</u>

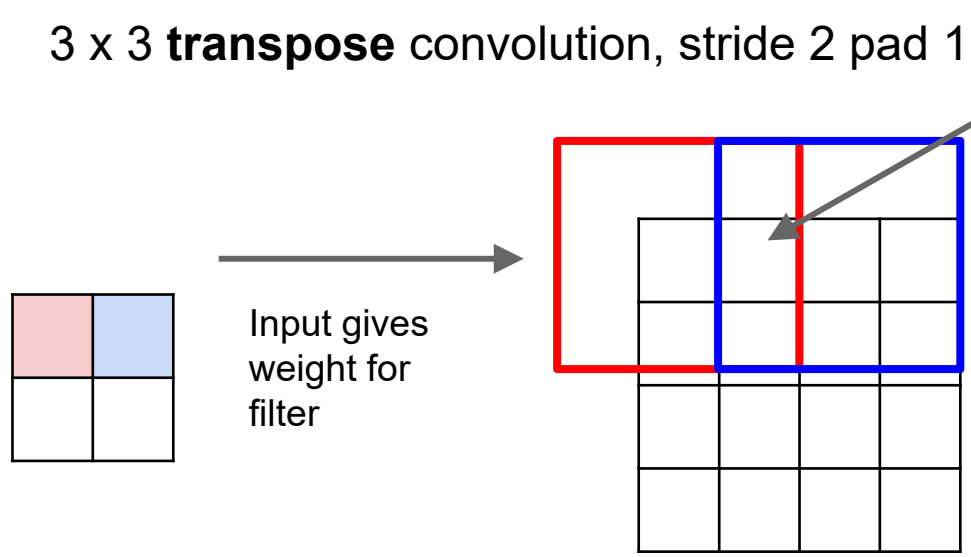
Stride gives ratio between movement in output and input

Slide Credit: Fei-Fei Li, Justin Johnson, Serena Yeung, CS 231n

Learnable Upsampling: Transpose Convolution

Other names:
-Deconvolution (bad)
-Upconvolution
-Fractionally strided convolution
-Backward strided convolution

3 x 3 **transpose** convolution, stride 2 pad 1



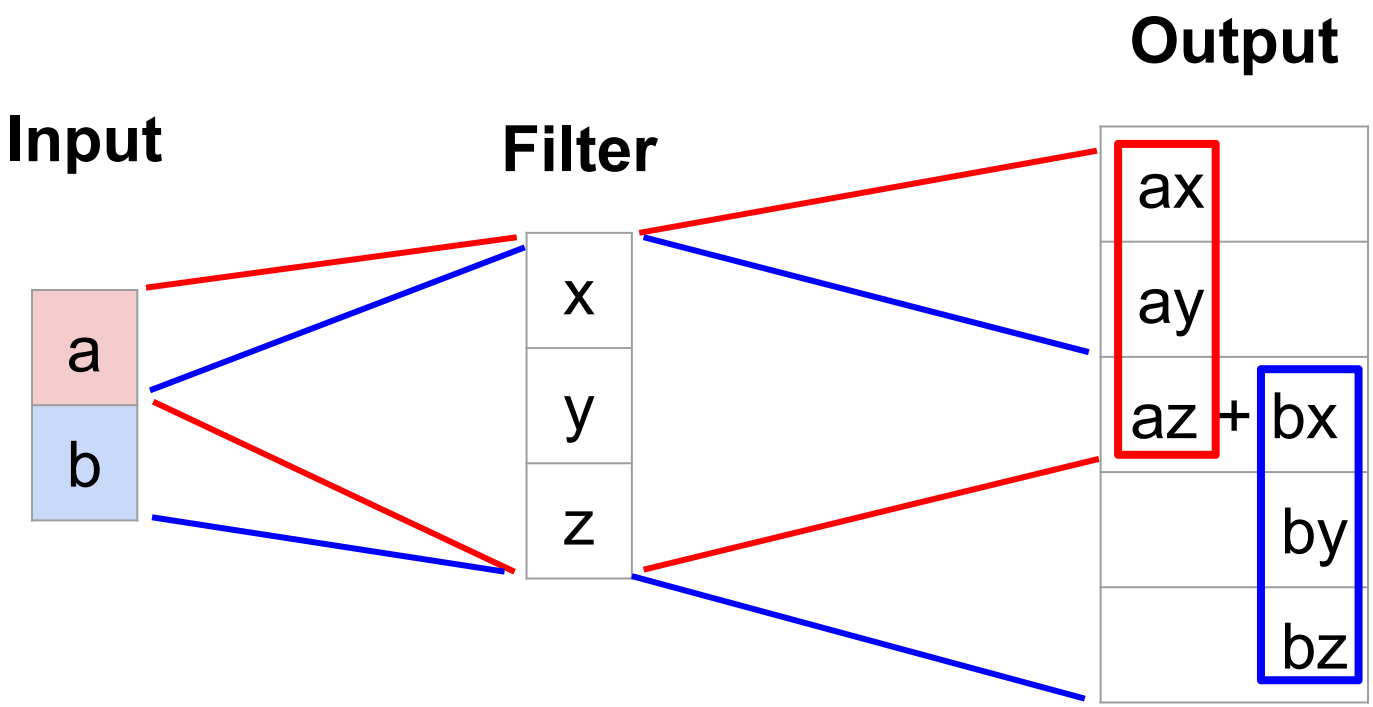
Sum where output overlaps

Filter moves 2 pixels in the output for every one pixel in the input

Stride gives ratio between movement in output and input

Slide Credit: Fei-Fei Li, Justin Johnson, Serena Yeung, CS 231n

Transpose Convolution: 1D Example



Output contains copies of the filter weighted by the input, summing at where at overlaps in the output

Need to crop one pixel from output to make output exactly 2x input

Slide Credit: Fei-Fei Li, Justin Johnson, Serena Yeung, CS 231n

Transposed Convolution

- https://distill.pub/2016/deconv-checkerboard/

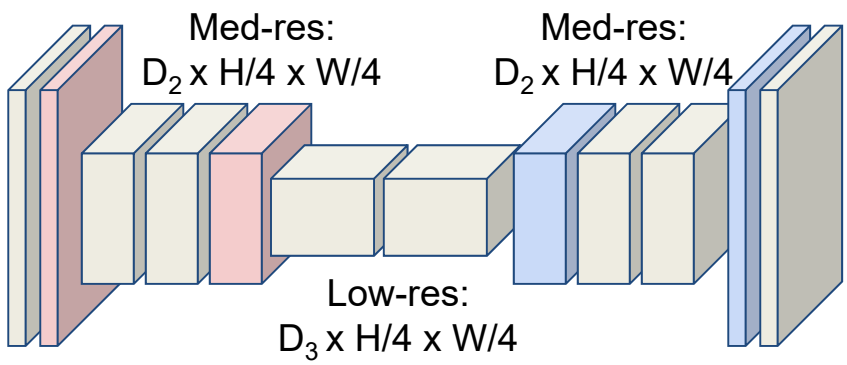
Semantic Segmentation Idea: Fully Convolutional

Downsampling:
Pooling, strided convolution

Design network as a bunch of convolutional layers, with **downsampling** and **upsampling** inside the network!

Upsampling:
Unpooling or strided transpose convolution

Input:
3 x H x W



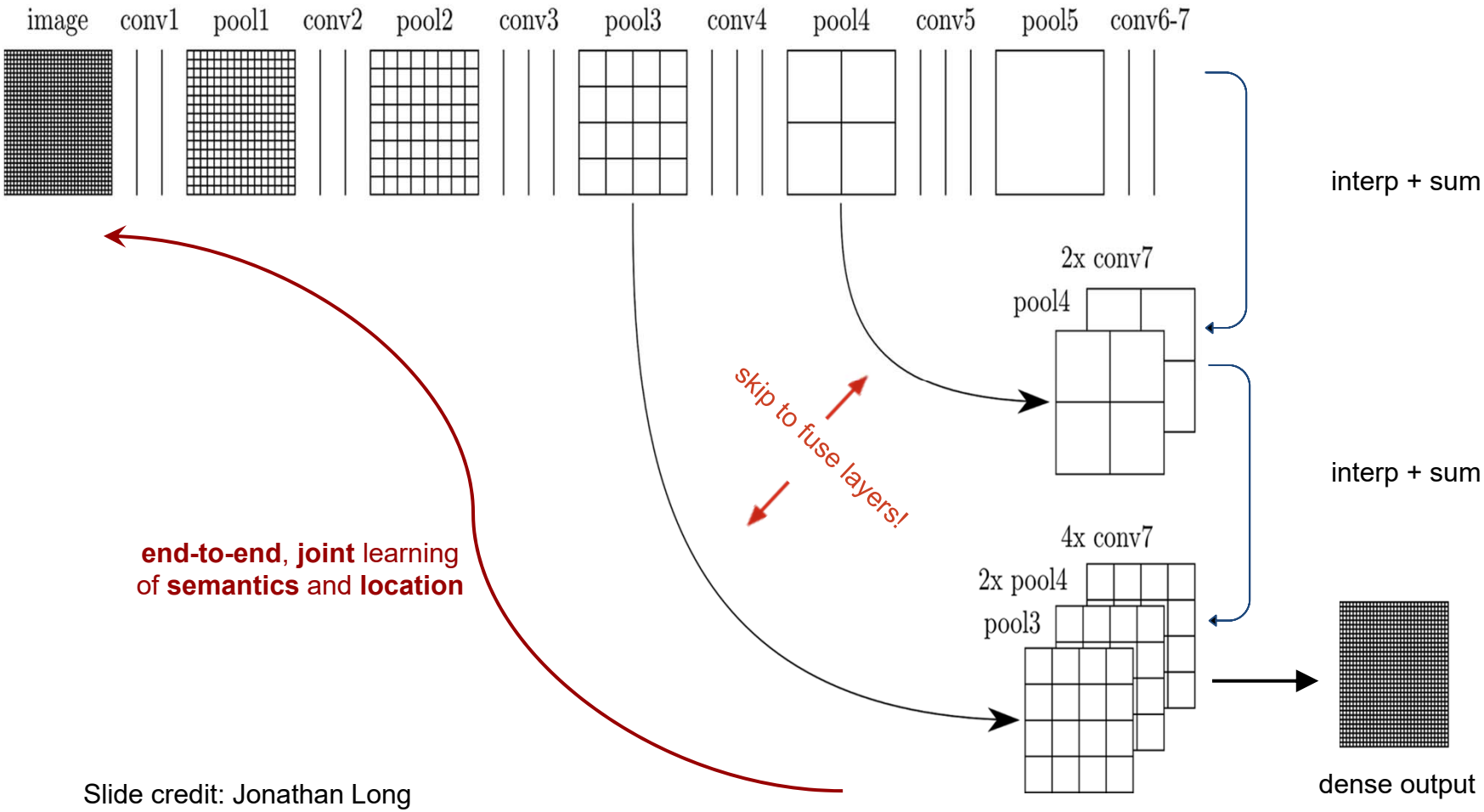
High-res:
D_1 x H/2 x W/2

High-res:
D_1 x H/2 x W/2

Predictions:
H x W

Long, Shelhamer, and Darrell, "Fully Convolutional Networks for Semantic Segmentation", CVPR 2015
Noh et al, "Learning Deconvolution Network for Semantic Segmentation", ICCV 2015

skip layers



end-to-end, **joint** learning of **semantics** and **location**

Slide credit: Jonathan Long

Next Time

- Guest lecture by Zhile Ren (Dhruv Batra post-doc) on **3D**