CSW182（2021）· 课程资料包 @ShowMeAI

视频
中英双语字幕

课件
一键打包下载

笔记
官方笔记翻译

代码
作业项目解析

视频·B 站 [扫码或点击链接]
https://www.bilibili.com/video/BV1Ff4y1n7ar

课件 & 代码·博客 [扫码或点击链接]
http://blog.showmeai.tech/berkeley-csw182

Berkeley

循环神经网络

可视化

梯度策略

Q-Learning

风格迁移

模仿学习

元学习

计算机视觉

机器学习基础

生成模型

卷积网络

Awesome AI Courses Notes Cheatsheets 是 **ShowMeAI** 资料库的分支系列，覆盖最具知名度的 **TOP50+** 门 AI 课程，旨在为读者和学习者提供一整套高品质中文学习笔记和速查表。

点击课程名称，跳转至课程**资料包**页面，**一键下载**课程全部资料！

机器学习	深度学习	自然语言处理	计算机视觉
Stanford · CS229	Stanford · CS230	Stanford · CS224n	Stanford · CS231n

Awesome AI Courses Notes Cheatsheets· 持续更新中

知识图谱	图机器学习	深度强化学习	自动驾驶
Stanford · CS520	Stanford · CS224W	UCBerkeley · CS285	MIT · 6.S094

微信公众号

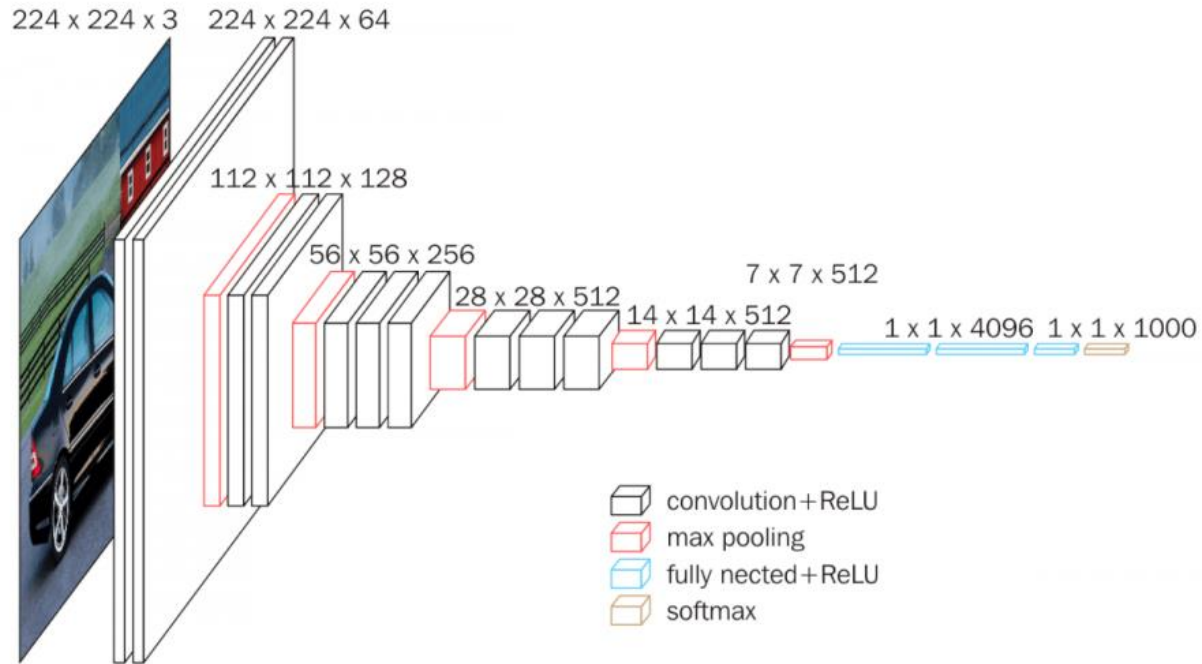资料下载方式 2：扫码点击底部菜单栏

称为 **AI 内容创作者？** 回复 [添砖加瓦]

Computer Vision

Designing, Visualizing and Understanding Deep Neural Networks

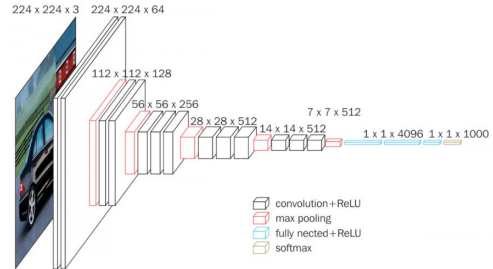# CS W182/282A

Instructor: Sergey Levine
UC Berkeley

So far…

224 x 224 x 3   224 x 224 x 64

112 x 112 x 128

56 x 56 x 256

28 x 28 x 512

14 x 14 x 512

7 x 7 x 512

1 x 1 x 4096   1 x 1 x 1000

convolution+ReLU
max pooling
fully nected+ReLU
softmax

convolutional networks: map **image** to **output value**
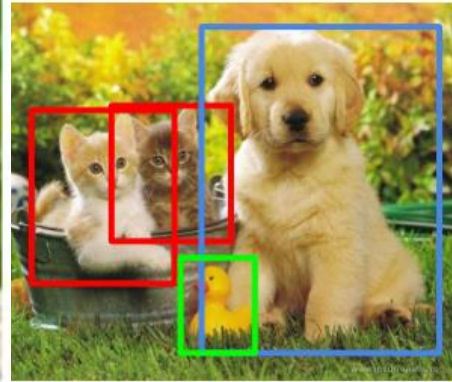
↑

e.g., semantic category ("bicycle")

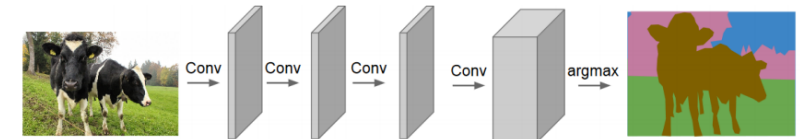# Standard computer vision problems

object classification

object localization

object detection

semantic segmentation
a.k.a. scene understanding

Object localization setup

Before: $\mathcal{D} = \{(x_i, y_i)\}$
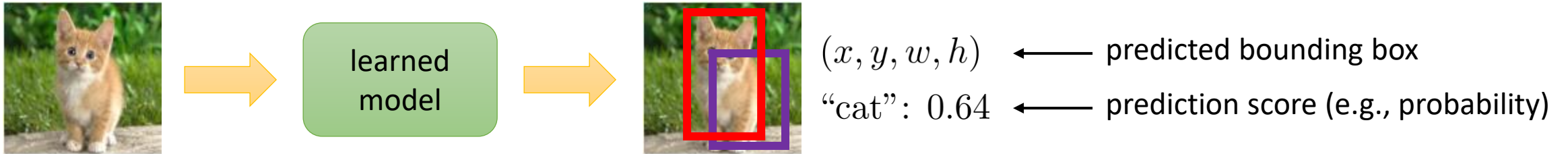
image   class label (categorical)

(x_i, y_i)

Now: $\mathcal{D} = \{(x_i, y_i)\}$
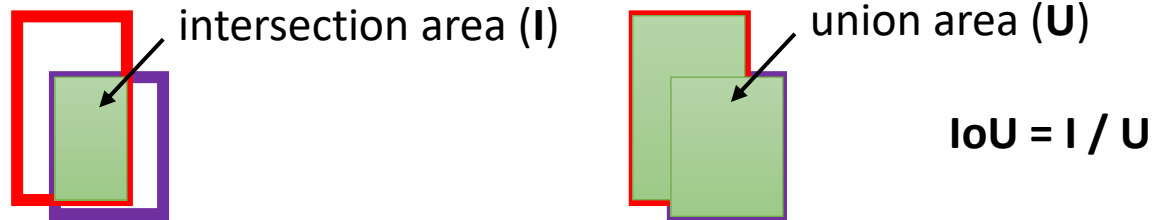
image $y_i = (\ell_i, x_i, y_i, w_i, h_i)$

h_i

w_i

Measuring localization accuracy

learned model

(x, y, w, h) ← predicted bounding box

"cat": 0.64 ← prediction score (e.g., probability)

Did we get it right?

Intersection over Union (IoU)

intersection area (**I**)

union area (**U**)

IoU = I / U

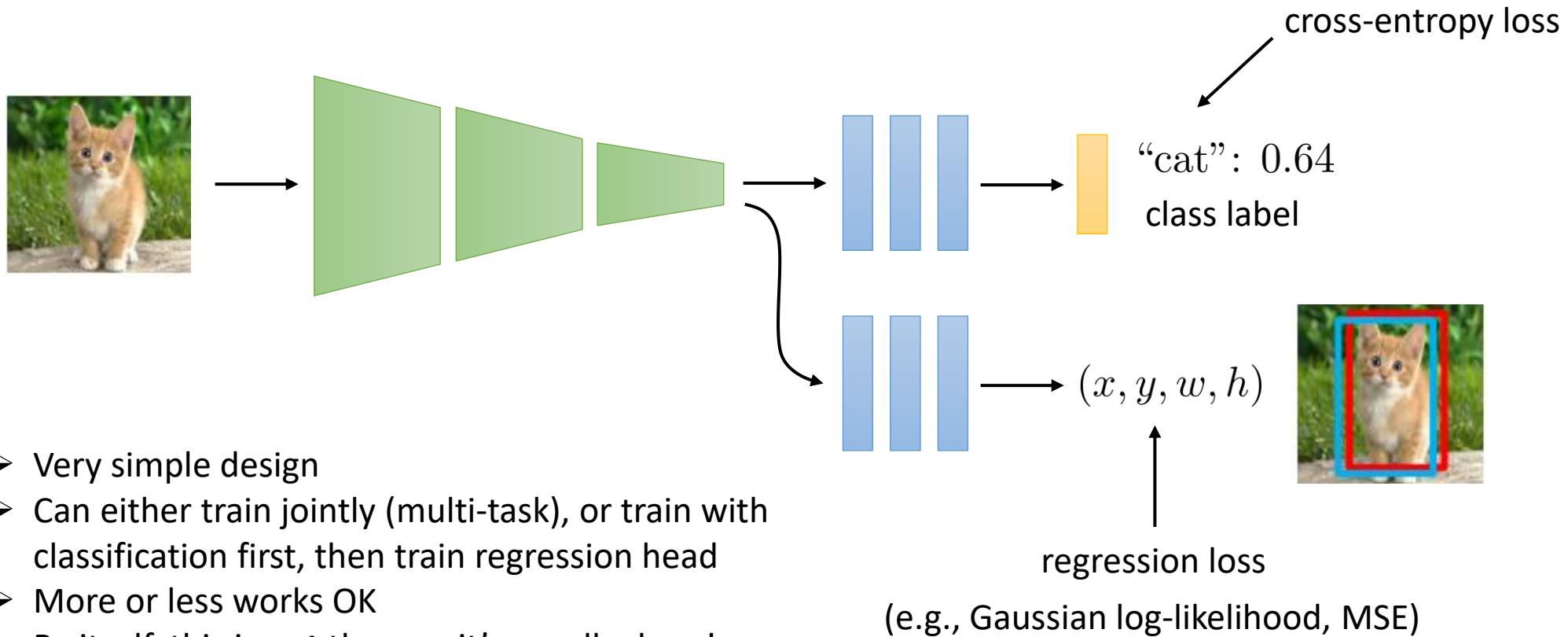Different datasets have different protocols, but one reasonable one is: **correct if IoU > 0.5**

If also outputting class label (usually the case): **correct if IoU > 0.5 and class is correct**

This is **not** a loss function! Just an evaluation standard

Object localization as regression

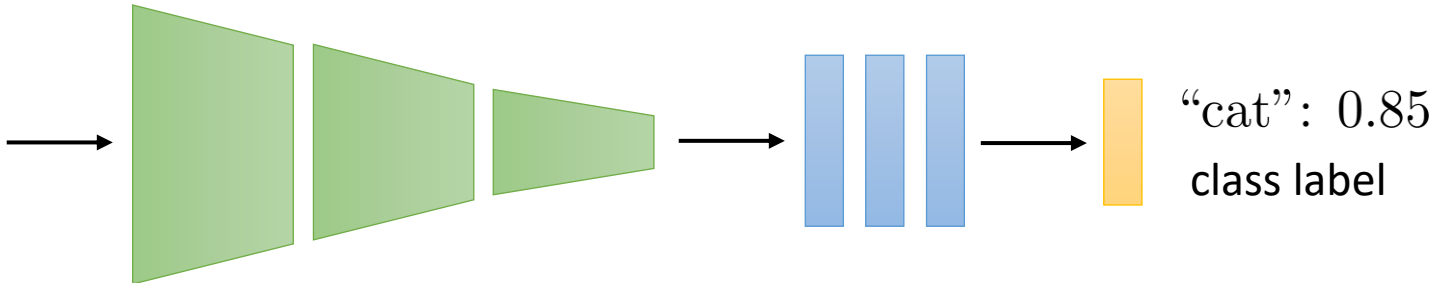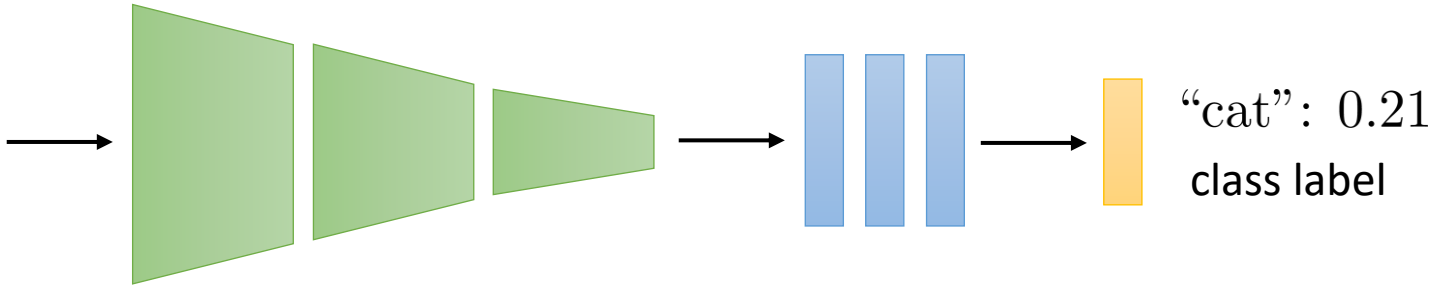$$\mathcal{D} = \{(x_i, y_i)\} \qquad y_i = (\ell_i, x_i, y_i, w_i, h_i)$$

cross-entropy loss

"cat": 0.64

class label

(x, y, w, h)

regression loss

(e.g., Gaussian log-likelihood, MSE)

➢ Very simple design
➢ Can either train jointly (multi-task), or train with classification first, then train regression head
➢ More or less works OK
➢ By itself, this is **not** the way it's usually done!
  ▪ We'll see why shortly

Sliding windows

$$\mathcal{D} = \{(x_i, y_i)\} \qquad y_i = (\ell_i, x_i, y_i, w_i, h_i)$$
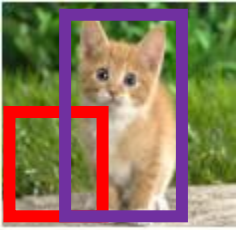
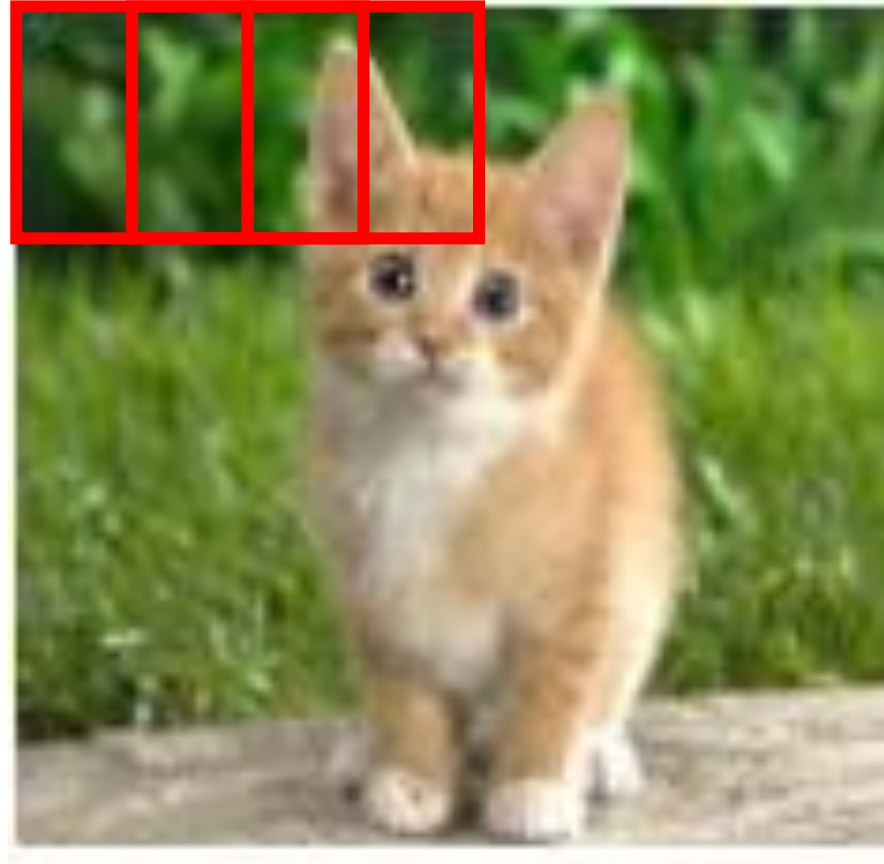What if we classify **every** patch in the image?

"cat": 0.21
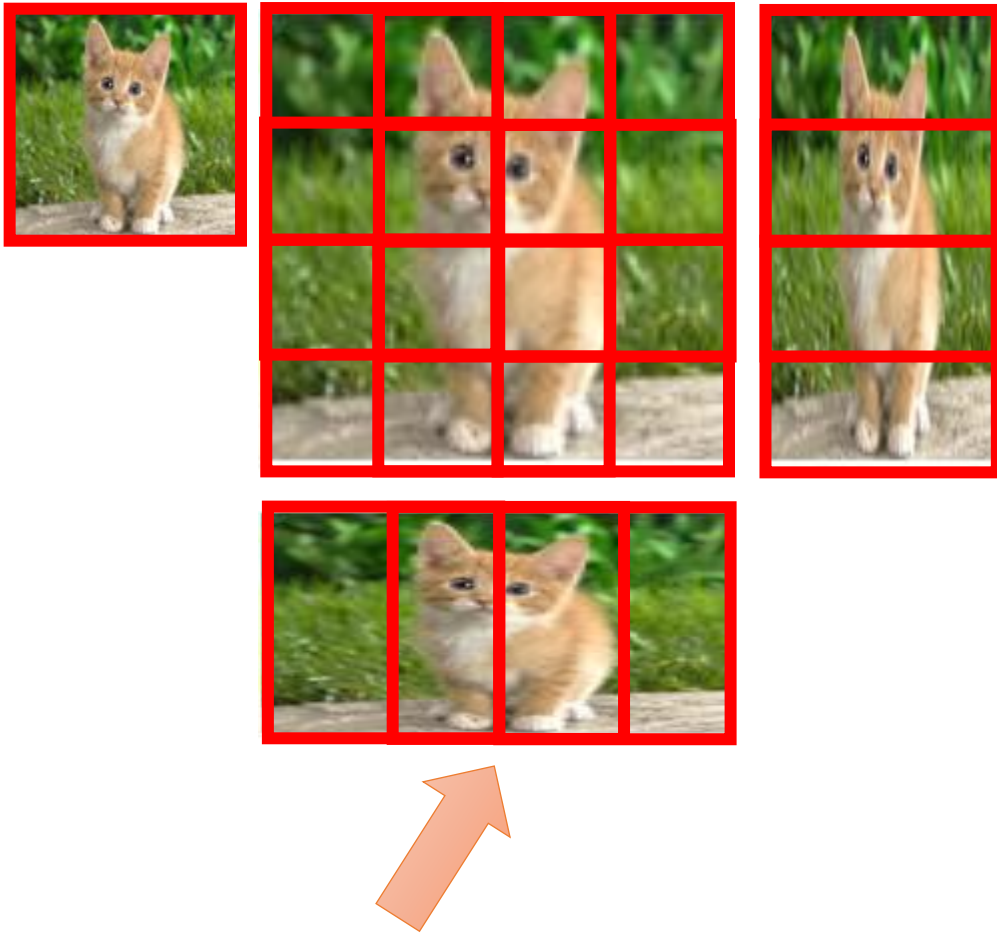class label

"cat": 0.85
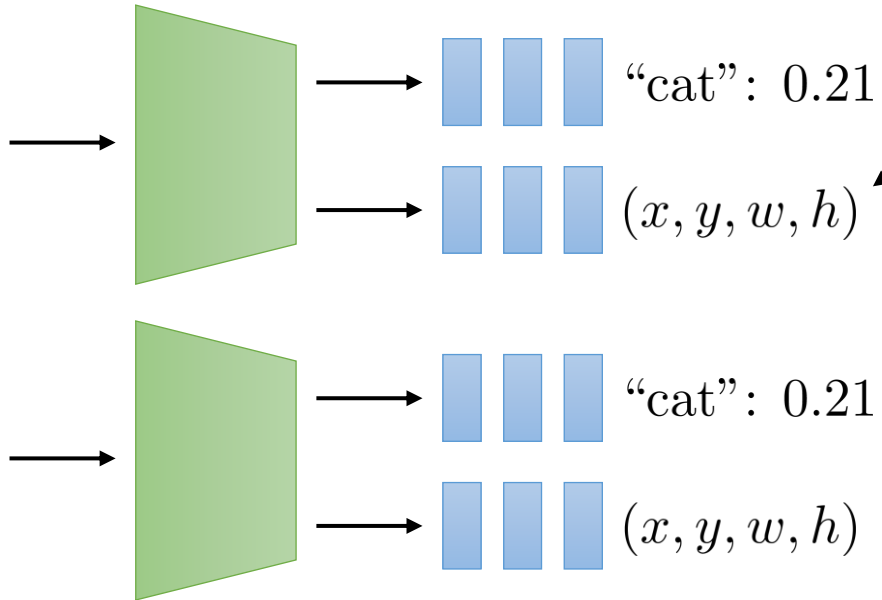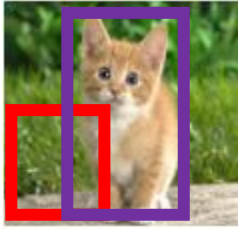class label

Sliding windows

$$\mathcal{D} = \{(x_i, y_i)\} \qquad y_i = (\ell_i, x_i, y_i, w_i, h_i)$$

could just take the box with the **highest** class probability

more generally: **non-maximal suppression**
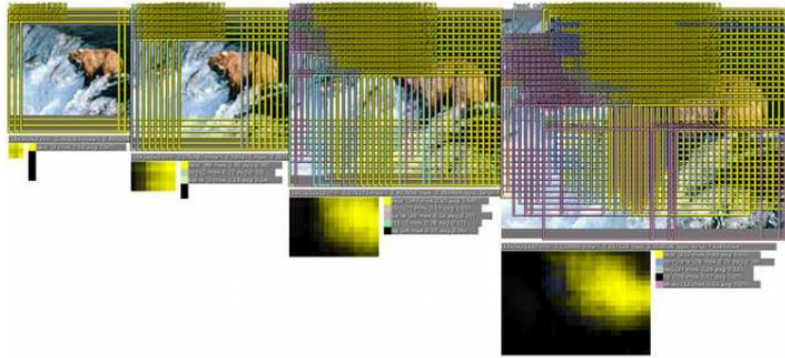
A practical approach: OverFeat

$$\mathcal{D} = \{(x_i, y_i)\} \qquad y_i = (\ell_i, x_i, y_i, w_i, h_i)$$

"cat": 0.21

(x, y, w, h)

provides a little "correction" to sliding window

"cat": 0.21

(x, y, w, h)

- ➤ Pretrain on **just** classification
- ➤ Train regression head on top of classification features
- ➤ Pass over different regions at different scales
- ➤ "Average" together the boxes to get a single answer

Sermanet et al. "**OverFeat: Integrated Recognition, Localization and Detection using Convolutional Networks.**" 2013

A practical approach: OverFeat

Sliding window **classification** outputs at each scale/position (yellow = bear)
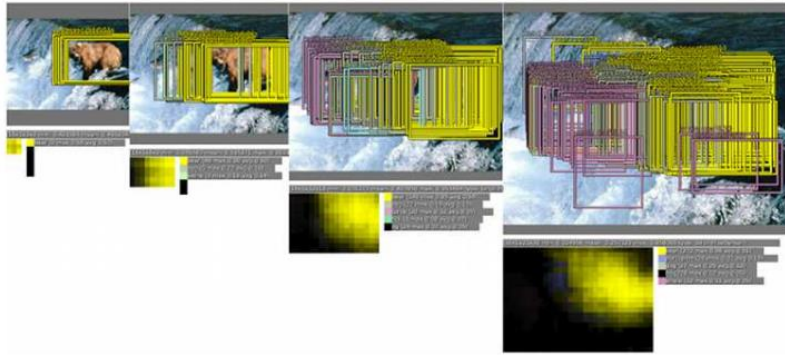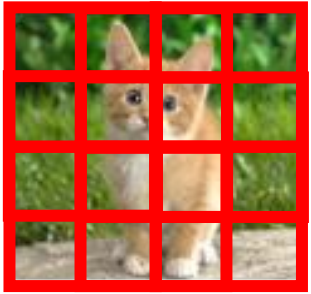
Predicted box x, y, w, h at each scale/position (yellow = bear)

Final combined bounding box prediction (yellow = bear)

Sermanet et al. "**OverFeat: Integrated Recognition, Localization and Detection using Convolutional Networks.**" 2013
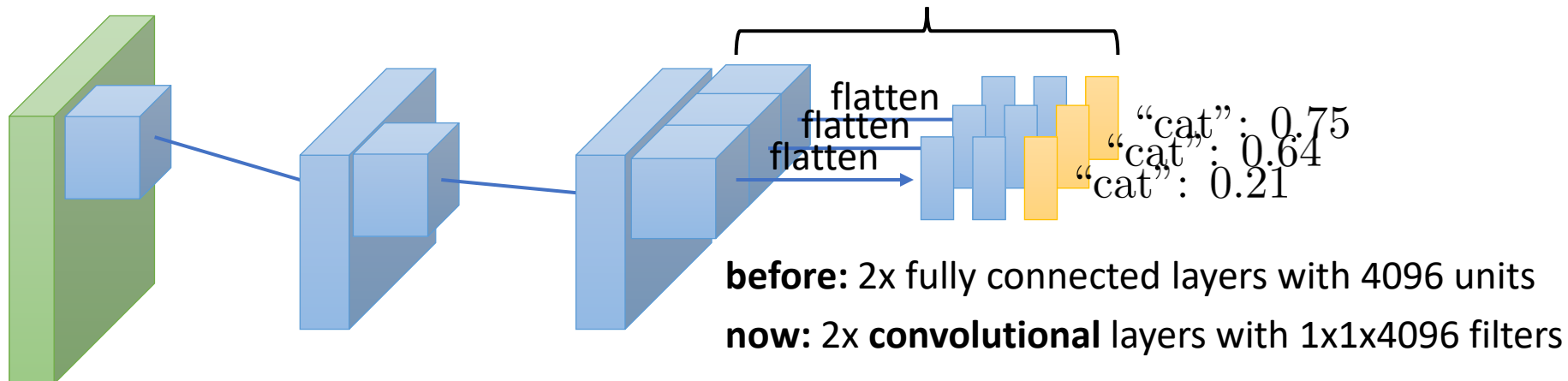
Sliding windows & reusing calculations

Problem: sliding window is very expensive! (36 windows = 36x the compute cost)
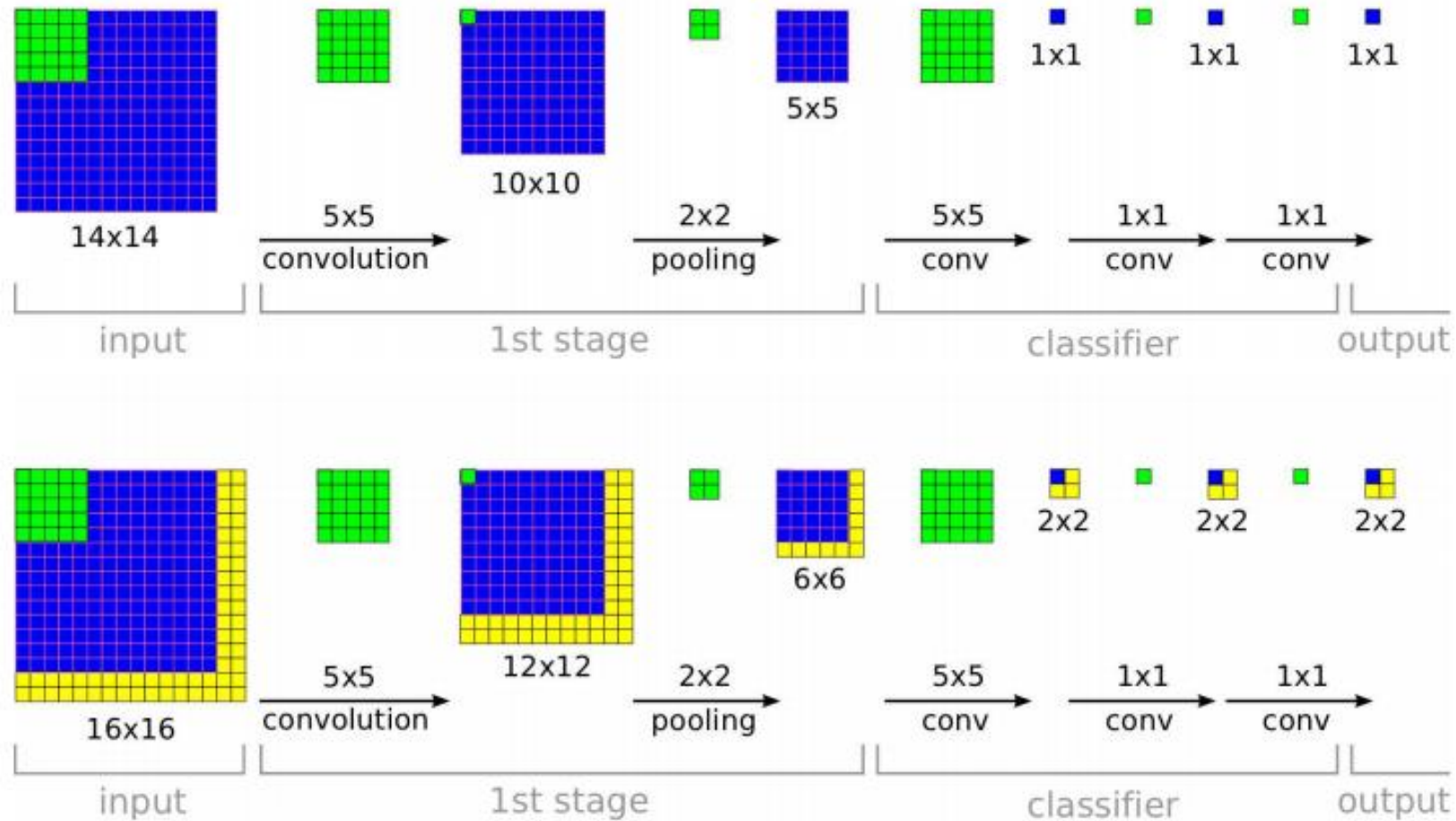
This looks **a lot** like convolution…

Can we just **reuse** calculations across windows?

"Convolutional classification"

these are just convolutional layers!

flatten
flatten
flatten

"cat": 0.75
"cat": 0.64
"cat": 0.21

before: 2x fully connected layers with 4096 units

now: 2x **convolutional** layers with 1x1x4096 filters
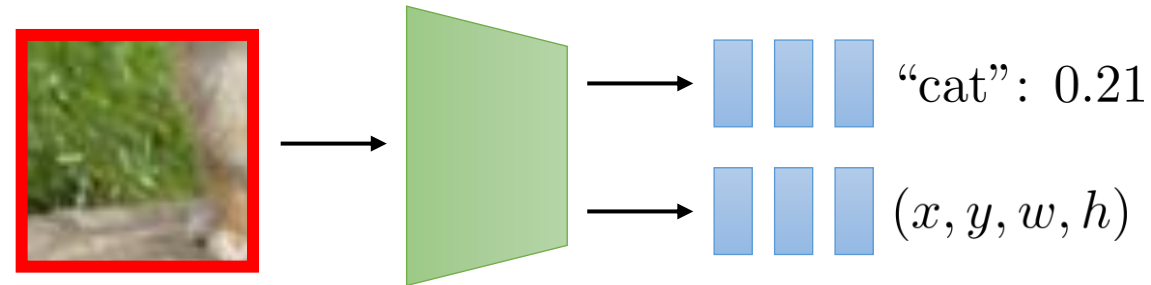
Sliding windows & reusing calculations

This kind of calculation reuse is extremely powerful for localization problems with conv nets

We'll see variants of this idea in every method we'll cover today!

Sermanet et al. "**OverFeat: Integrated Recognition, Localization and Detection using Convolutional Networks.**" 2013
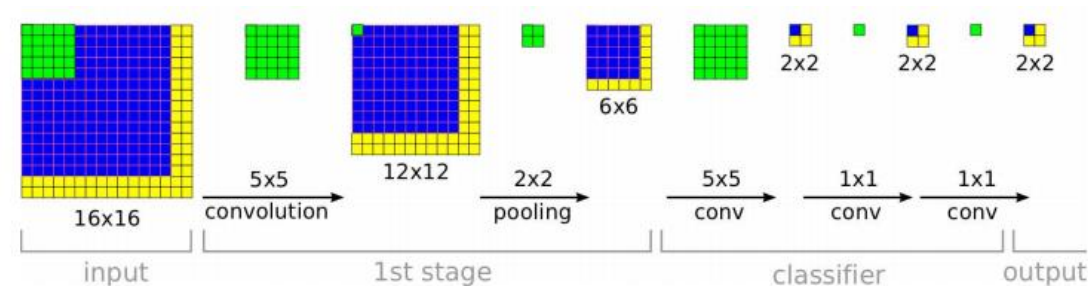
Summary

Building block: conv net that outputs class and bounding box coordinates

Evaluate this network at multiple scales and for many different crops, each one producing a probability and bounding box

Implement the sliding window as just another convolution, with 1x1 convolutions for the classifier/regressor at the end, to save on computation
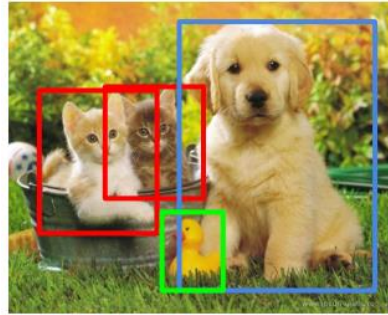
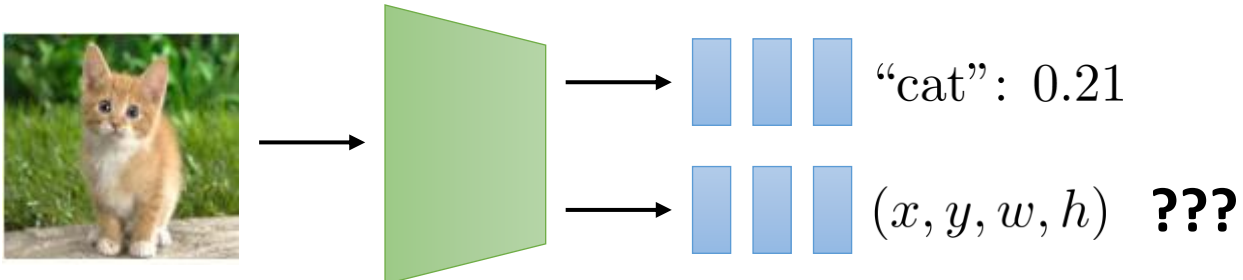# Object detection architectures

The problem setup

Before

Now

$(x_i,$

number of objects n_i different for each image x_i!

"cat": 0.21

(x, y, w, h) **???**

How do we get multiple outputs?

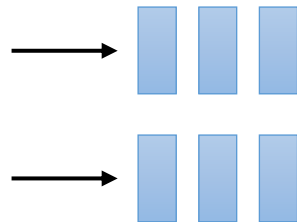**Sliding window:** each window can be a different object

Instead of selecting the window with the highest probability (or merging windows), just output an object in each window above some threshold

Big problem: a high-scoring window probably has **other** high-scoring windows nearby

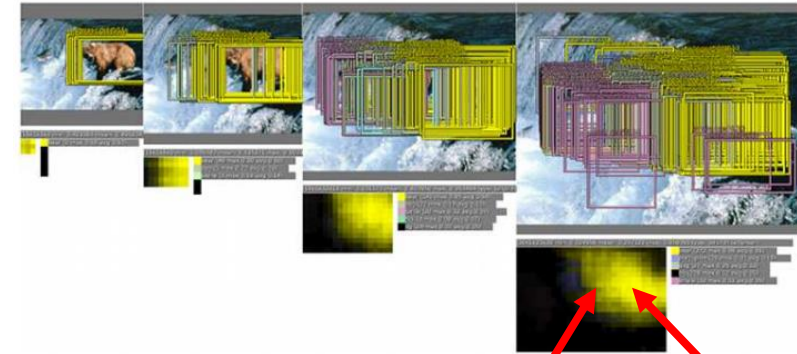**Non-maximal suppression:** (informally) kill off any detections that have other higher-scoring detections of the same class nearby

Actually output multiple things: output is a list of bounding boxes

Obvious problem: need to pick number, usually pretty small

non-maximal   maximal

works great if combined

not good by itself

$\{$ "cat": 0.21, "dog": 0.54 $\}$

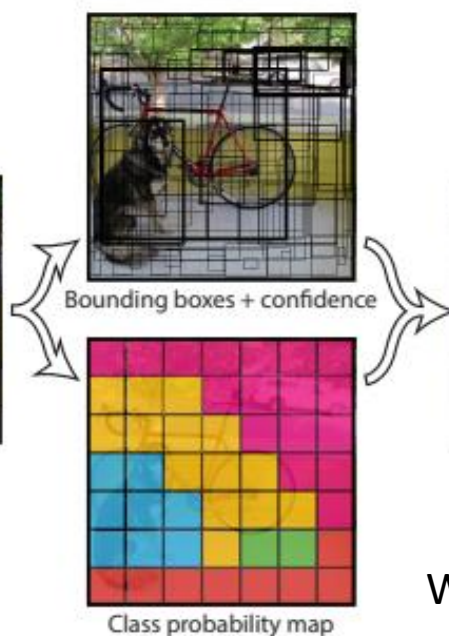$(x_1, y_1, w_1, h_1, x_2, y_2, w_2, h_2)$

Case study: you only ~~live~~ once (YOLO)
look

Actually, you look a few times (49 times to be exact...)

different output for each
of the 7x7 (49) grid cells
(a bit like sliding window)

for each cell, output:

zero if no object

(x, y, w, h)

IoU (confidence)

ℓ (class label)

output B of these

Bounding boxes + confidence

$S \times S$ grid on input

Class probability map

Final detections

some training details:

need to assign which output is "responsible"
for each true object during training

just use the "best-fit" object in that cell
(i.e., the one with highest IoU)
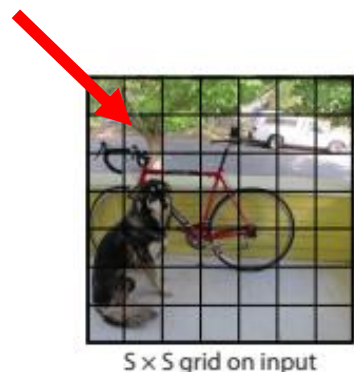
use the same trick as
before to reuse
computation (cost is
not 49x higher!)
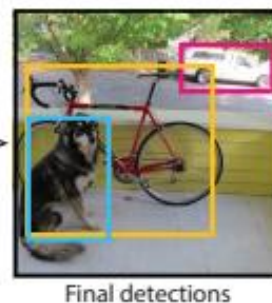
What if we have too many objects?

Well, nothing... we just miss them

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

CNNs + Region proposals

A smarter "sliding window": region of interest proposals

R-CNN: Regions with CNN features
1. Input image
2. Extract region proposals (~2k)
3. Compute CNN features
4. Classify regions
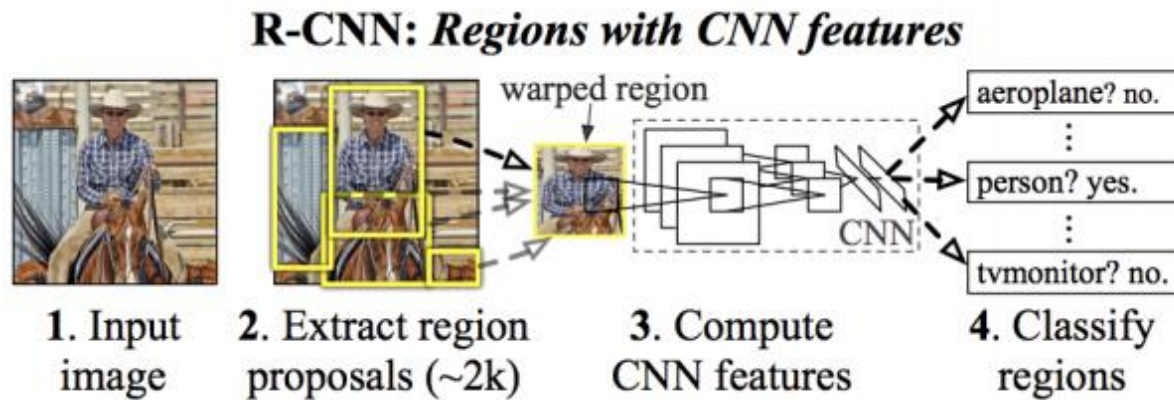
This is really slow

But we already know how to fix this!

Slow R-CNN

Girschick et al. "**Fast R-CNN.**" 2015
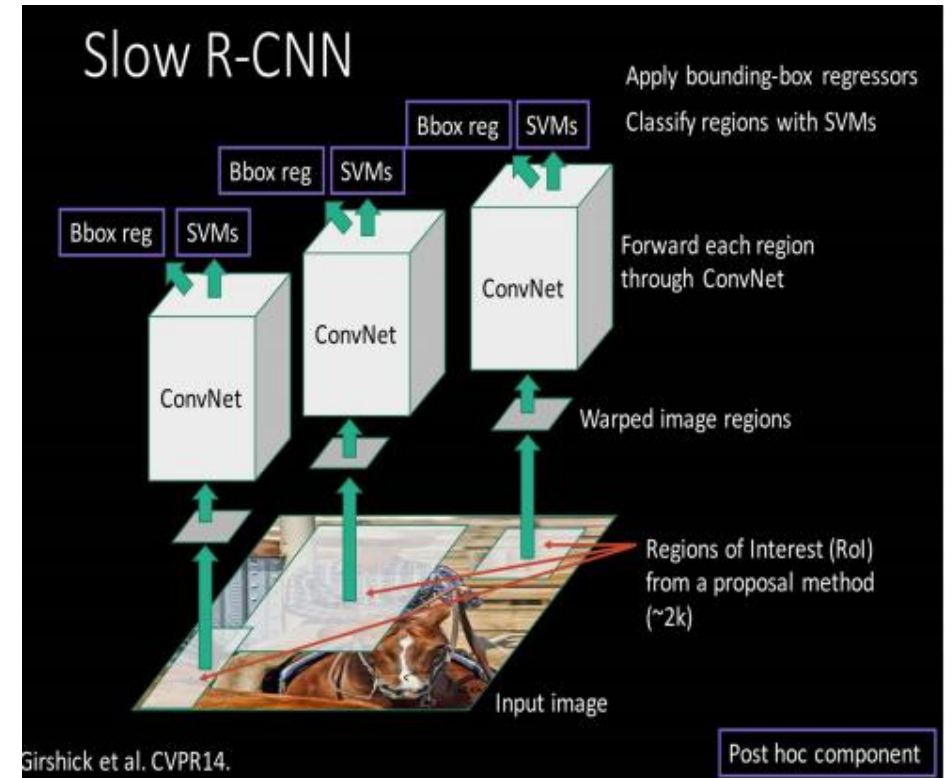
CNNs + Region proposals

A smarter "sliding window": region of interest proposals
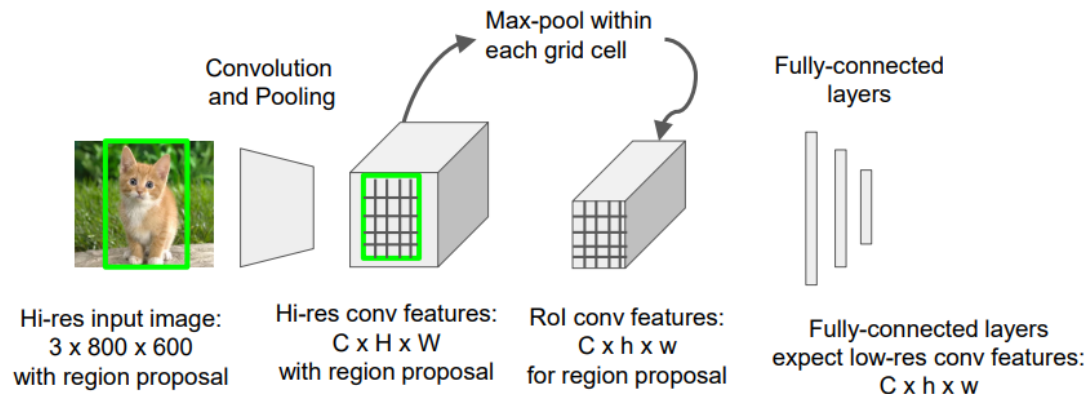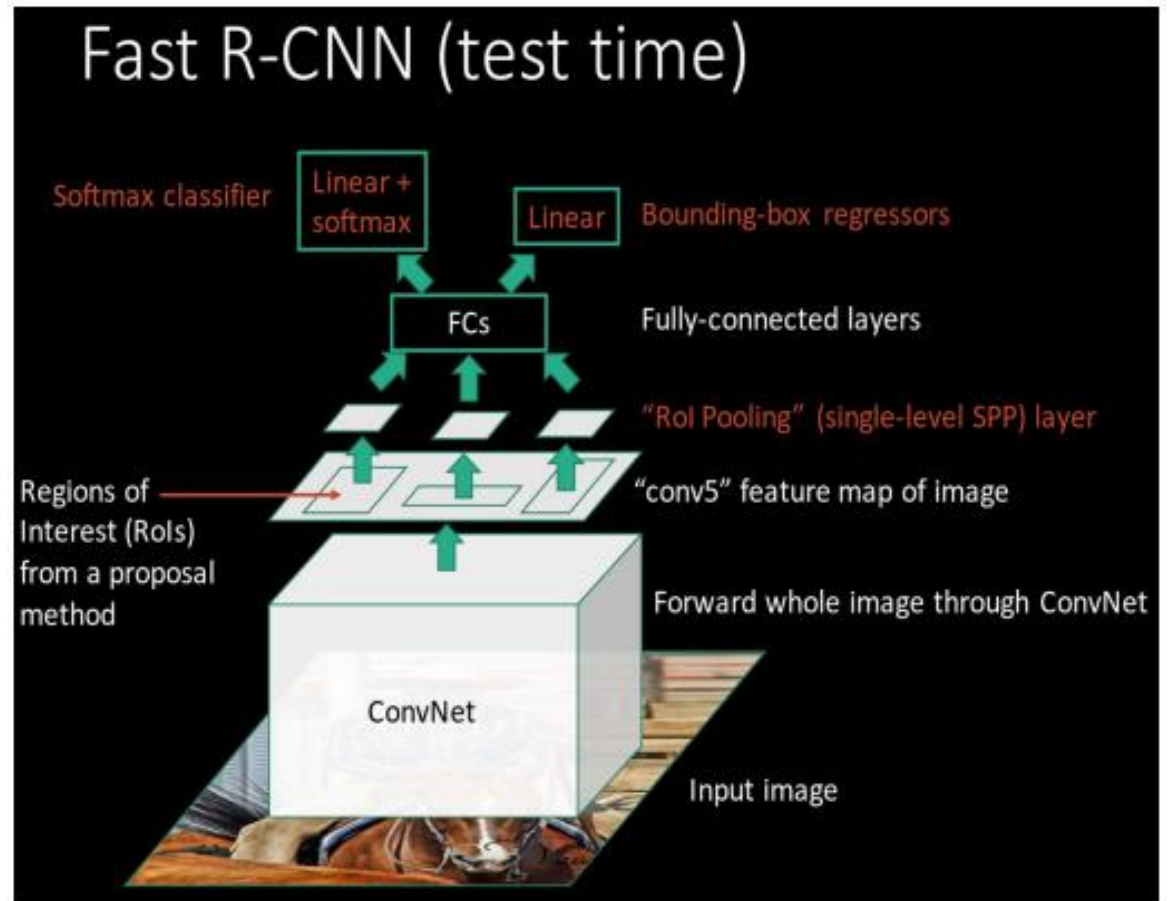
Compare this to evaluating every location:

Girschick et al. "**Fast R-CNN.**" 2015
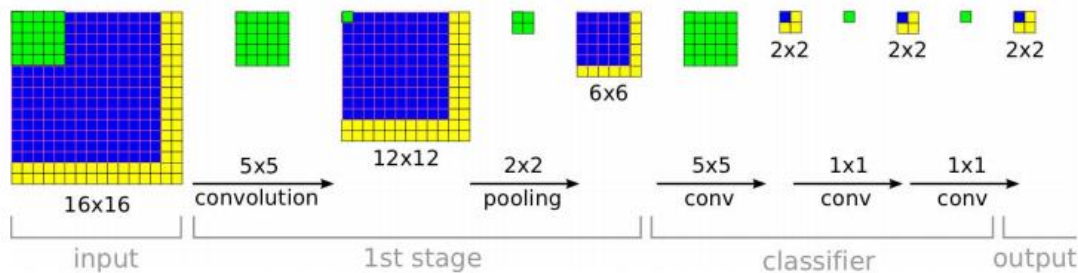
CNNs + Region proposals

How to train region of interest proposals?

Very similar design to what we saw before (e.g., OverFeat, YOLO), but now for predicting if **any** object is present around that location

classify obj./not-obj.   regress box locations

scores   coordinates

256-d

sliding window

convolutional feature map

classifier   facebo

RoI pooling

proposals

Region Proposal Network

feature map

CNN

image

Ren et al. "**Faster R-CNN.**" 2015
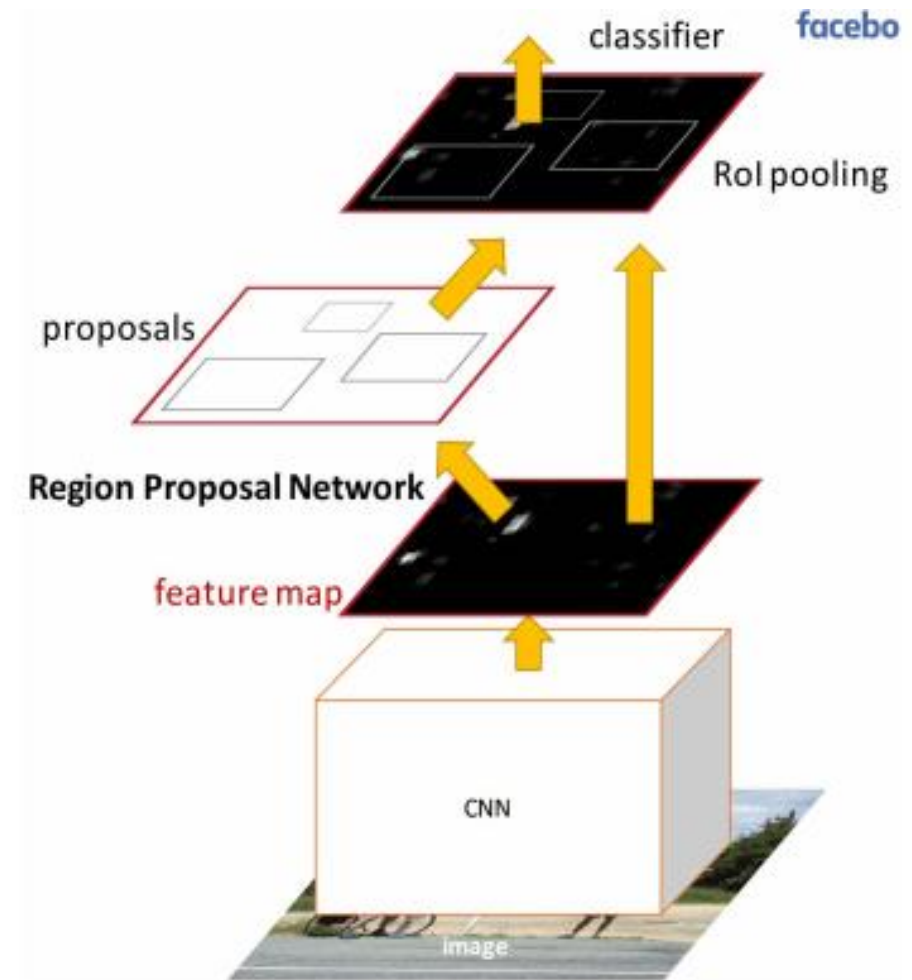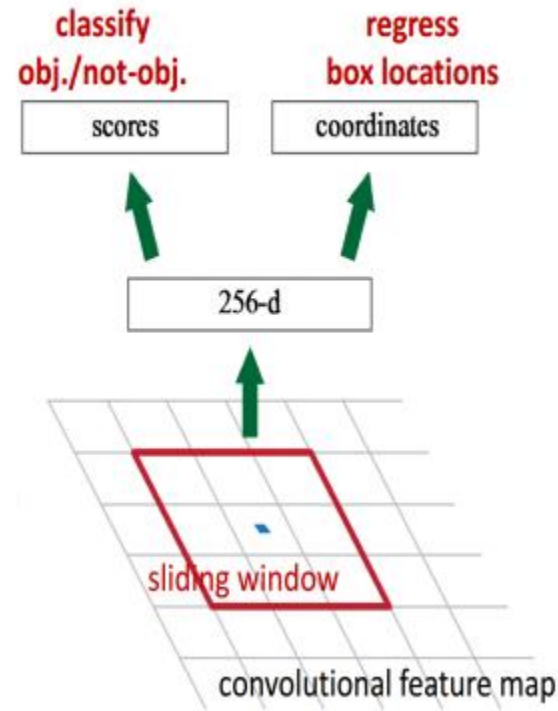
Suggested readings

- Redmon et al. **"You Only Look Once: Unified, Real-Time Object Detection."** 2015
  - Just regress to different bounding boxes in each cell
  - A few follow-ups (e.g., YOLO v5) that work better
- Girschick et al. **"Fast R-CNN."** 2015
  - Uses region of interest proposals instead of sliding window/convolution
- Ren et al. **"Faster R-CNN."** 2015
  - Same as above with a few improvements, like region of interest proposal learning
- Liu et al. **SSD: Single Shot MultiBox Detector**. 2015
  - Directly "classifies" locations with class and bounding box shape

Segmentation architectures

The problem setup

Before

Now

Problem:

We want the output to have the same resolution as the input!

Label **every single** pixel with its class

Actually simpler in some sense:

- No longer variable # of outputs

- Every pixel has a label

Not hard if we never downsample (i.e., zero padding, stride 1, no pooling), but that is very expensive
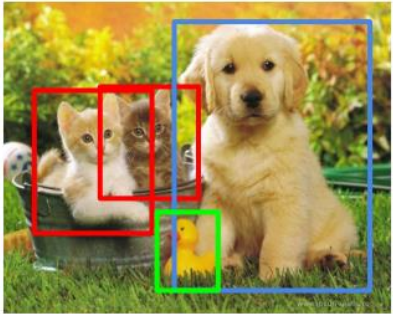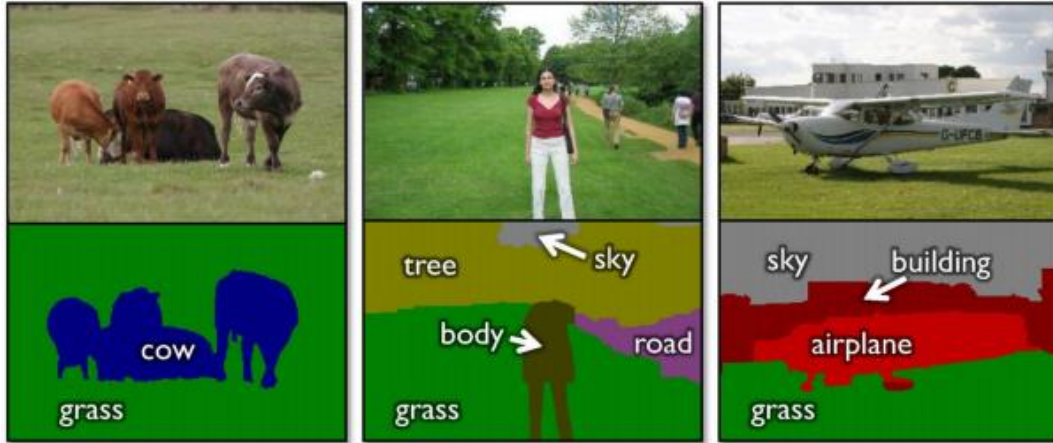
Simple solution:

"per pixel" classifier

flatten
flatten
flatten

"cat": 0.75
"cat": 0.64
"cat": 0.21

The problem setup

Classify every point with a class

Don't worry for now about instances
(e.g., two adjacent cows are just one "cow blob,"
and that's OK for some reason)

The challenge: design a network architecture
that makes this "per-pixel classification" problem
computationally tractable

Fully convolutional networks

Med-res:
$D_2 \times H/4 \times W/4$

Med-res:
$D_2 \times H/4 \times W/4$

Low-res:
$D_3 \times H/4 \times W/4$

Input:
$3 \times H \times W$
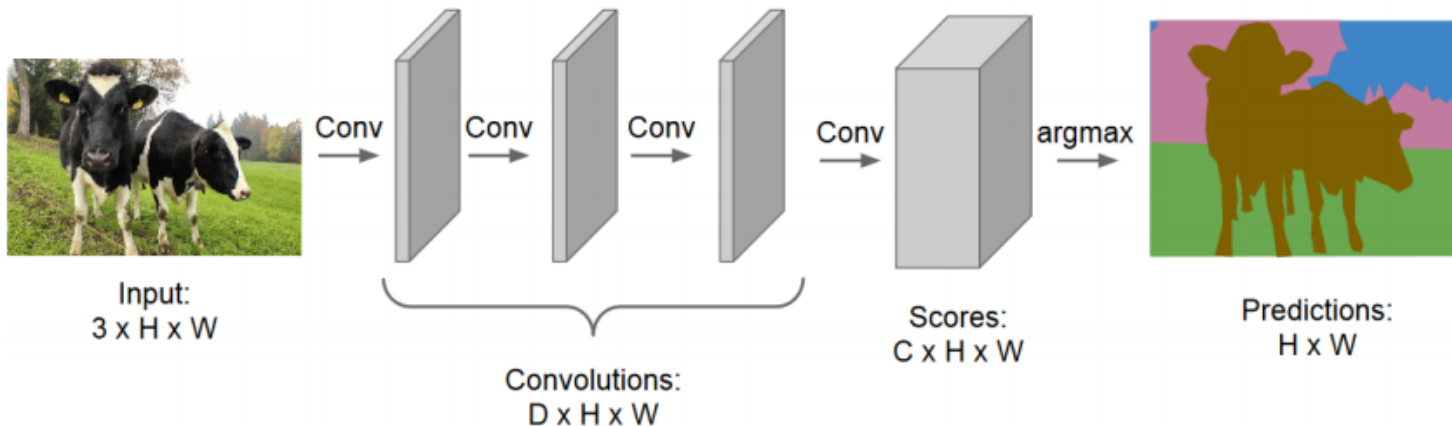
High-res:
$D_1 \times H/2 \times W/2$

High-res:
$D_1 \times H/2 \times W/2$
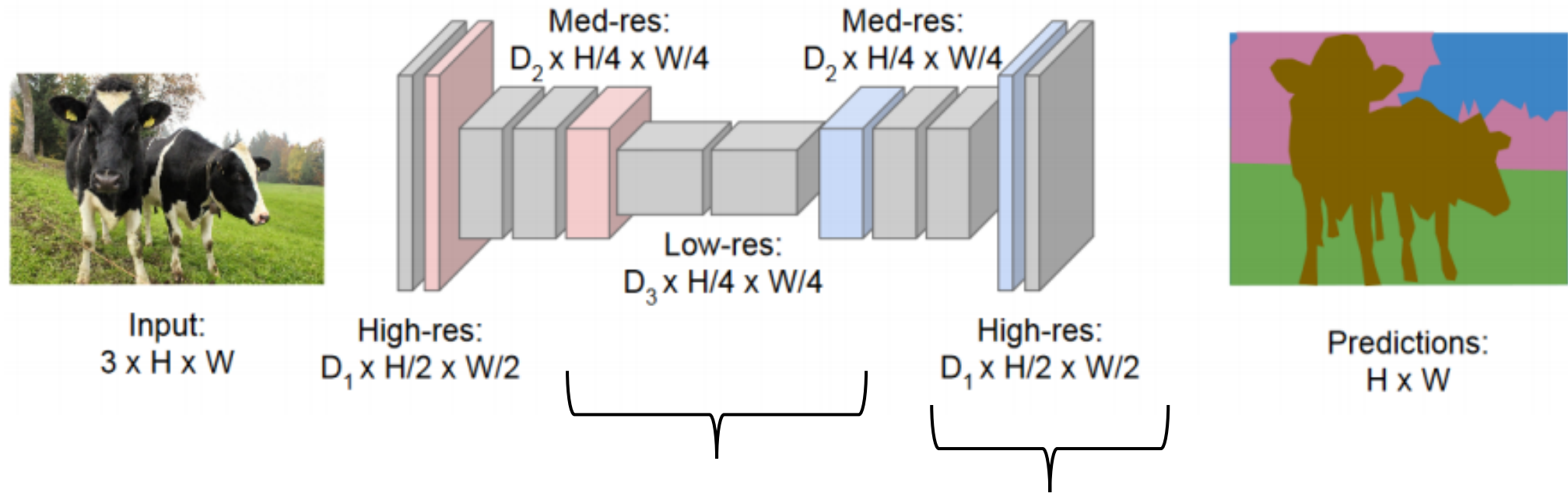
Predictions:
$H \times W$

Low-res (but high-depth) processing in the middle integrates context from the entire image

Up-sampling at the end turns these low-res feature vectors into high-res per-pixel predictions

Up-sampling/transpose convolution

Normal convolutions: reduce resolution with **stride**

Stride = 2

input: $H_f \times W_f \times C_{\text{in}}$
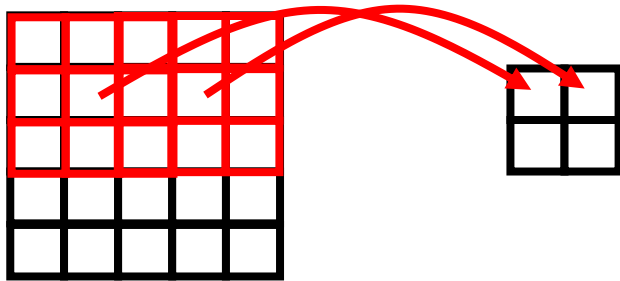
output: $1 \times 1 \times C_{\text{out}}$
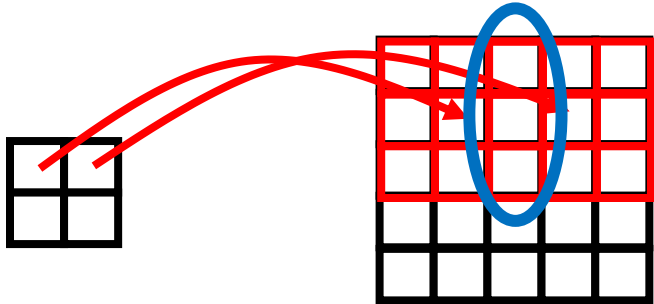
filter: $H_f \times W_f \times C_{\text{in}} \times C_{\text{out}}$

Transpose convolutions: increase resolution with **fractional "stride"**

Stride = 1/2   we have two sets of values here!   **just average them**

input: $1 \times 1 \times C_{\text{in}}$

output: $H_f \times W_f \times C_{\text{out}}$

filter: $C_{\text{in}} \times H_f \times W_f \times C_{\text{out}}$

Un-pooling

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

Bottleneck architecture

Input:
3 x H x W

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

Med-res:
D_2 x H/4 x W/4

Low-res:
D_3 x H/4 x W/4

Med-res:
D_2 x H/4 x W/4

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

Predictions:
H x W

"tabby cat"

convolutionalization

tabby cat heatmap

image   pool1   pool2   pool3   pool4   pool5

32x upsampled prediction (FCN-32s)

2x upsampled prediction

pool4 prediction

16x upsampled prediction (FCN-16s)

2x upsampled prediction

pool3 prediction

8x upsampled prediction (FCN-8s)

Long et al. "**Fully Convolutional Networks for Semantic Segmentation.**" 2014

U-Net architecture

concatenate activations from conv layers to upsampling layers

- conv 3x3, ReLU
- copy and crop
- max pool 2x2
- up-conv 2x2
- conv 1x1

Ronneberger et al. **U-net: Convolutional networks for biomedical image segmentation**. 2015
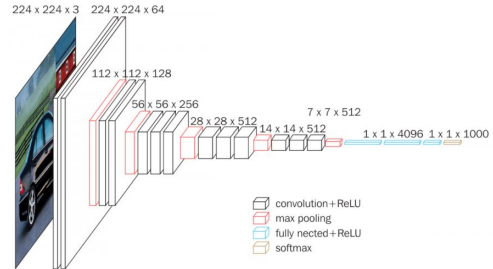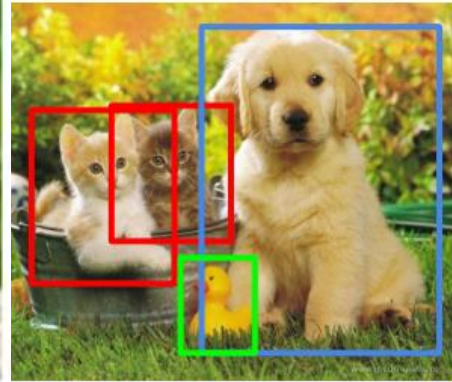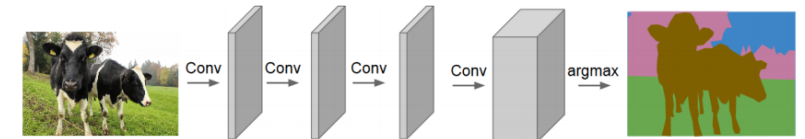
Standard computer vision problems

object classification

object localization

object detection

semantic segmentation
a.k.a. scene understanding

CSW182 (2021)· 课程资料包 @ShowMeAI

视频
中英双语字幕

课件
一键打包下载

笔记
官方笔记翻译

代码
作业项目解析

视频·B 站 [扫码或点击链接]
https://www.bilibili.com/video/BV1Ff4y1n7ar

课件 & 代码·博客 [扫码或点击链接]
http://blog.showmeai.tech/berkeley-csw182

Berkeley   循环神经网络   可视化   梯度策略

Q-Learning   风格迁移   模仿学习   元学习

计算机视觉   机器学习基础   生成模型   卷积网络

Awesome AI Courses Notes Cheatsheets 是 **ShowMeAI** 资料库的分支系列，覆盖最具知名度的 **TOP50+** 门 AI 课程，旨在为读者和学习者提供一整套高品质中文学习笔记和速查表。

点击课程名称，跳转至课程**资料包**页面，**一键下载**课程全部资料！

机器学习	深度学习	自然语言处理	计算机视觉
Stanford · CS229	Stanford · CS230	Stanford · CS224n	Stanford · CS231n

Awesome AI Courses Notes Cheatsheets· 持续更新中

知识图谱	图机器学习	深度强化学习	自动驾驶
Stanford · CS520	Stanford · CS224W	UCBerkeley · CS285	MIT · 6.S094

微信公众号

资料下载方式 2：扫码点击底部菜单栏

称为 **AI 内容创作者？** 回复 [添砖加瓦]