

CS 1674: Object Recognition and Image Segmentation

PhD. Nils Murrugarra-Llerena
nem177@pitt.edu

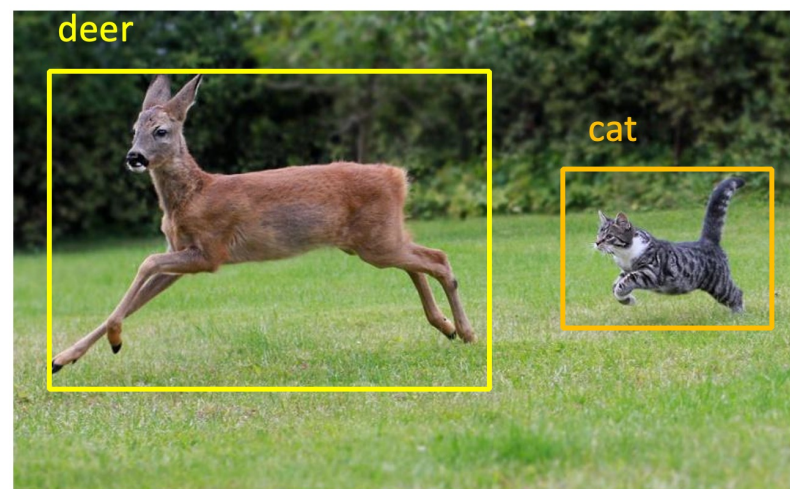


Lab 11a: Object Recognition and Image Segmentation

Duration: 10 min

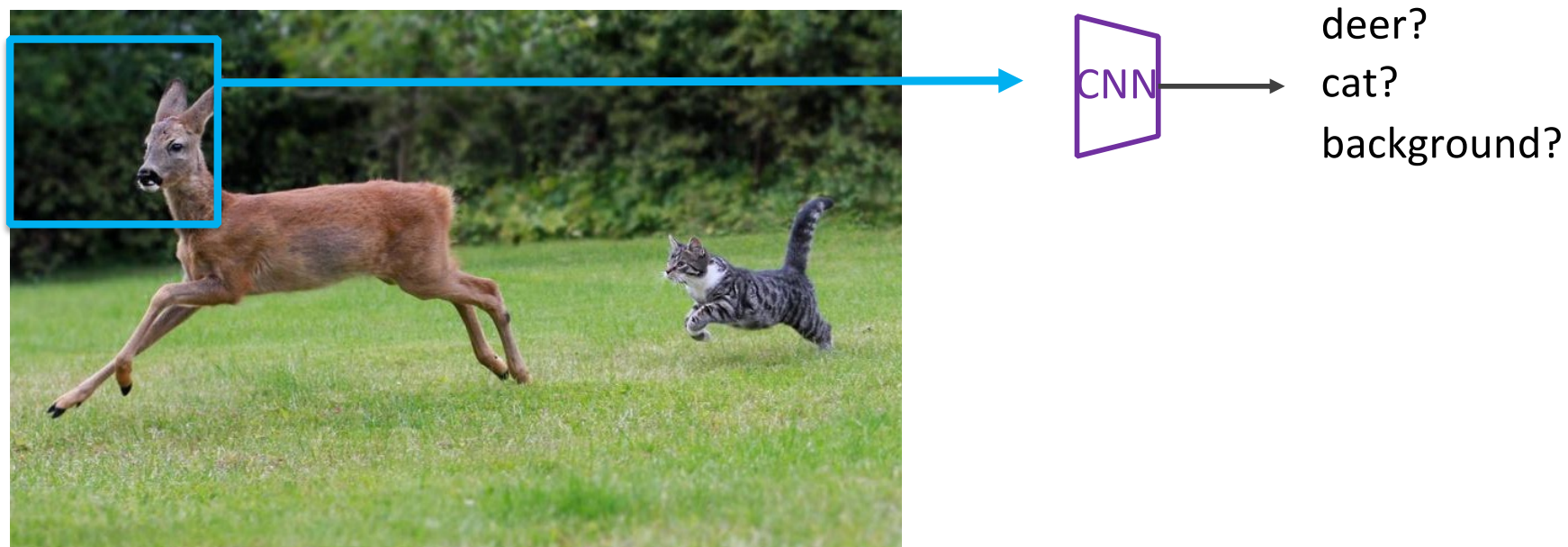


What will be the output of an Object Detector?



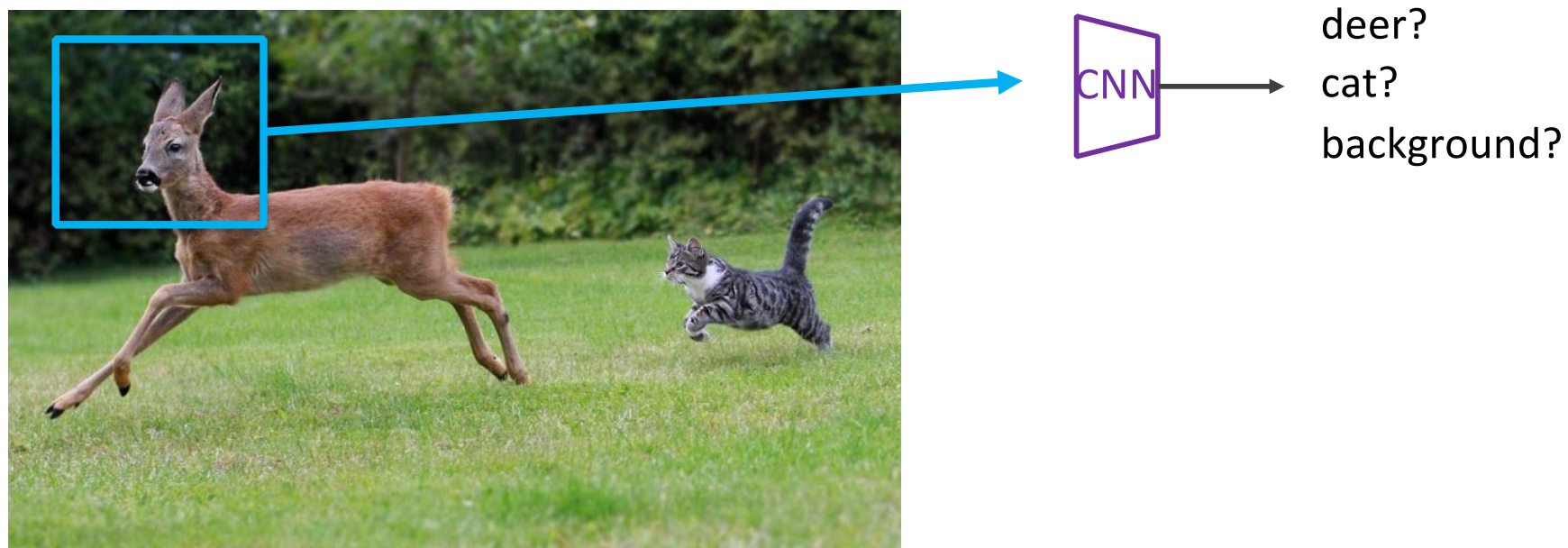
Adapted from Vicente Ordoñez

Object Detection as Classification



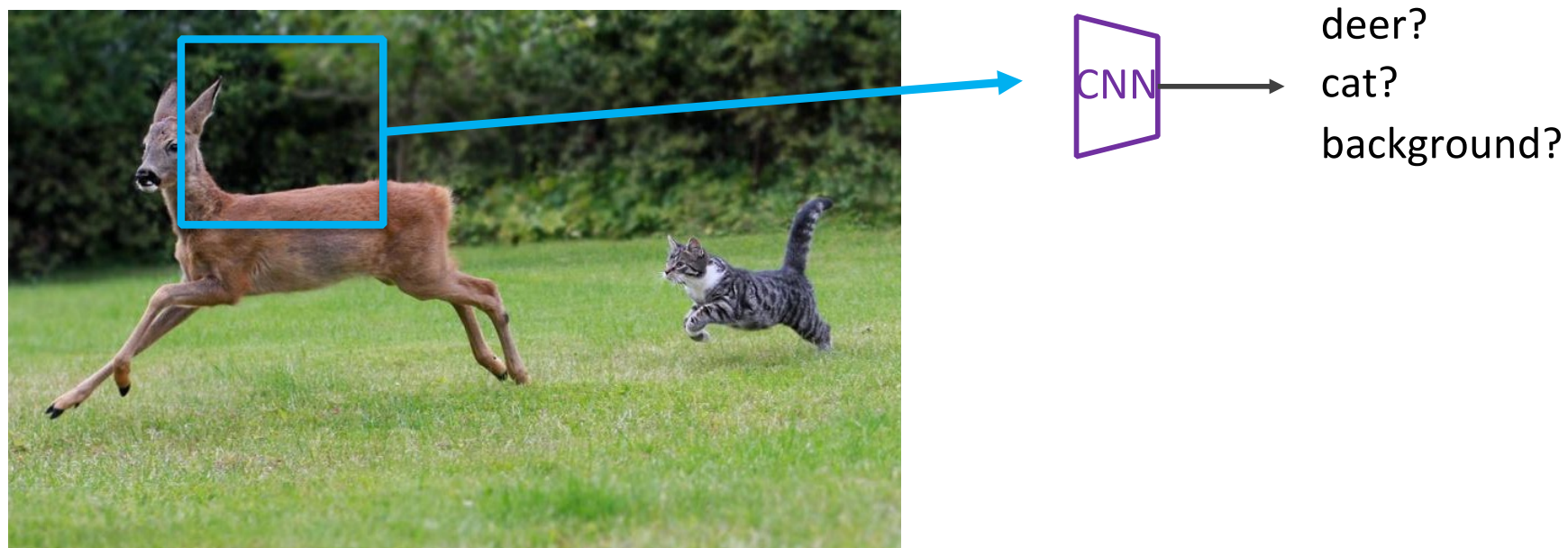
Adapted from Vicente Ordoñez

Object Detection as Classification



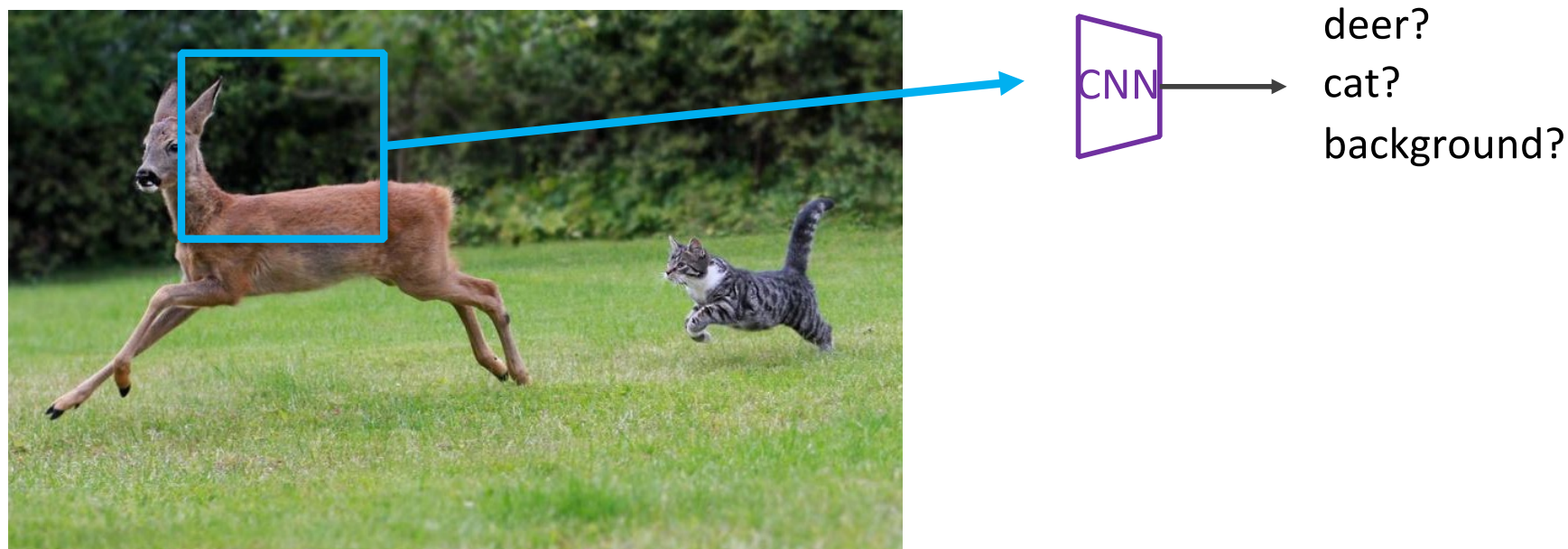
Adapted from Vicente Ordoñez

Object Detection as Classification



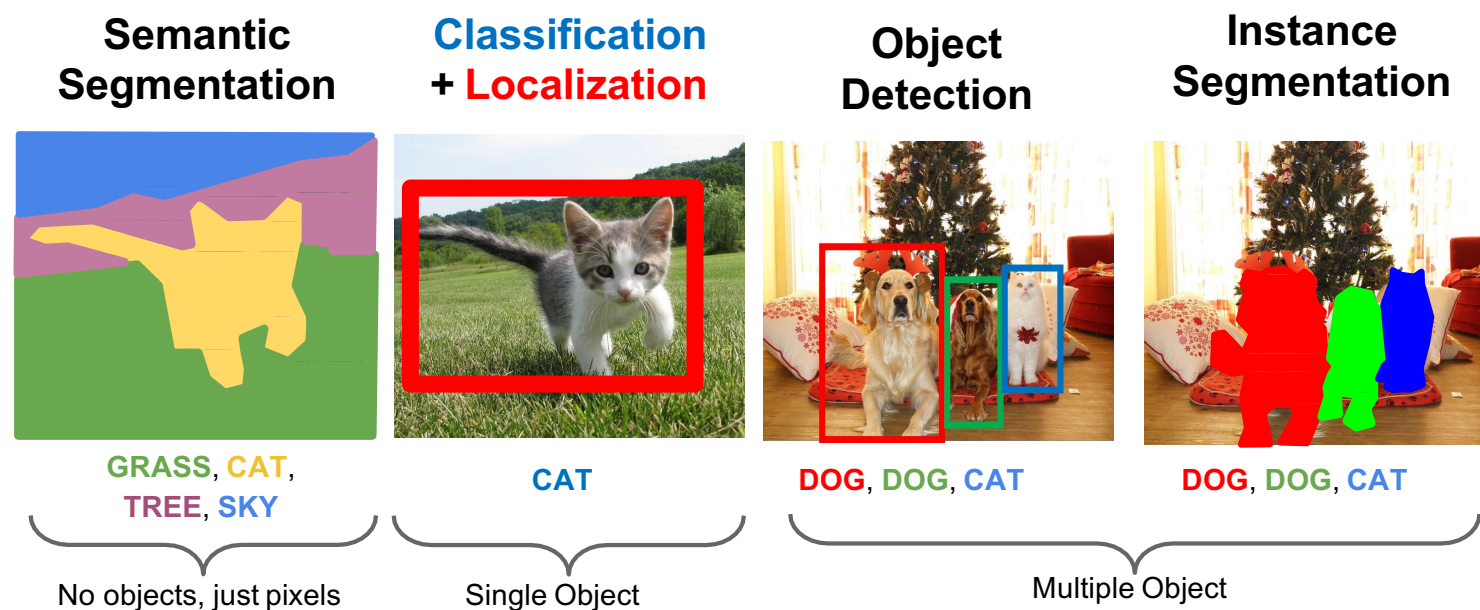
Adapted from Vicente Ordoñez

Object Detection as Classification with Sliding Window



Adapted from Vicente Ordoñez

Different Flavors of Object Recognition

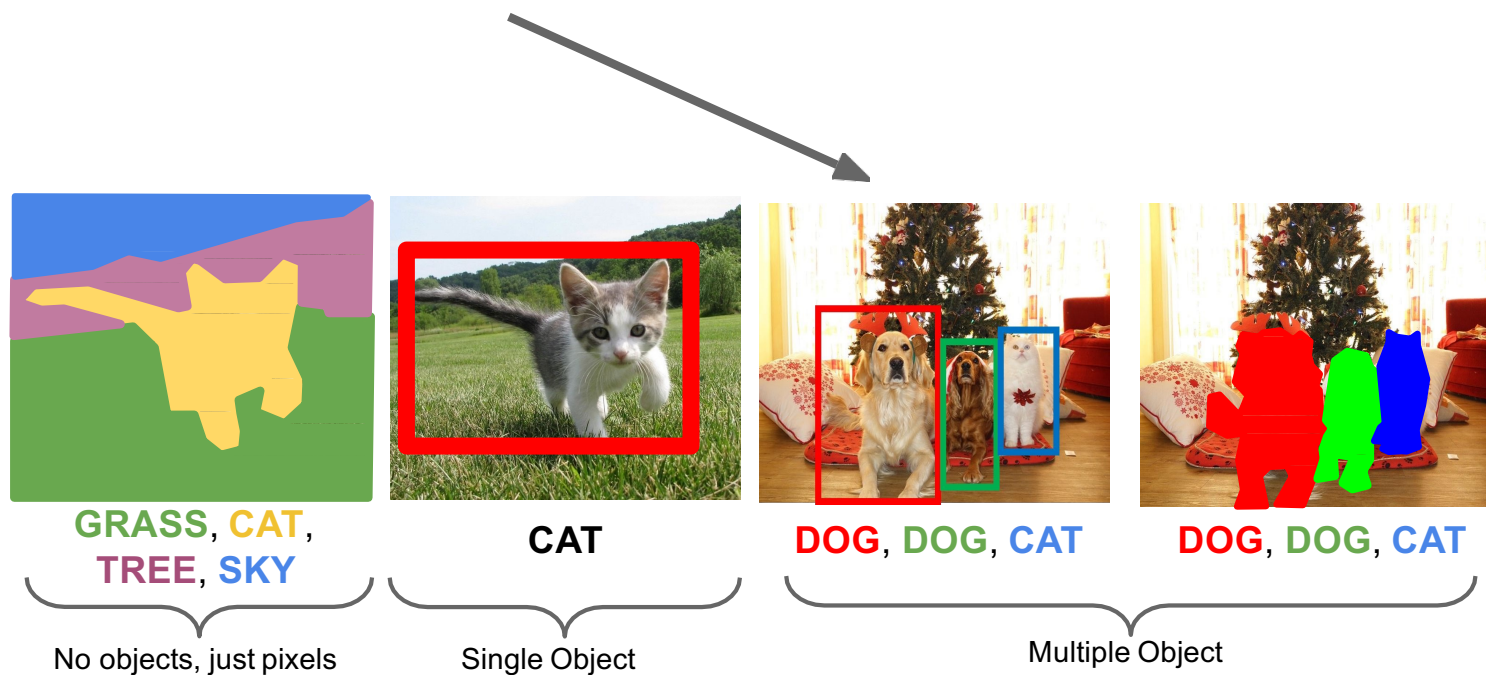


Adapted from Justin Johnson

Plan for Today

- Detection approaches
 - Pre-CNNs
 - Detection with whole windows: Pedestrian detection
 - Part-based detection: Deformable Part Models
 - Post-CNNs
 - Detection with region proposals: R-CNN, Fast R-CNN, Faster-R-CNN
 - Detection without region proposals: YOLO, SSD
- Segmentation approaches
 - Semantic segmentation: FCN
 - Instance segmentation: Mask R-CNN

Object Detection

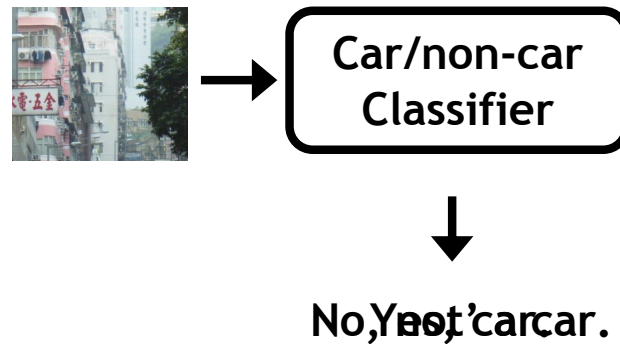


Object detection: basic framework

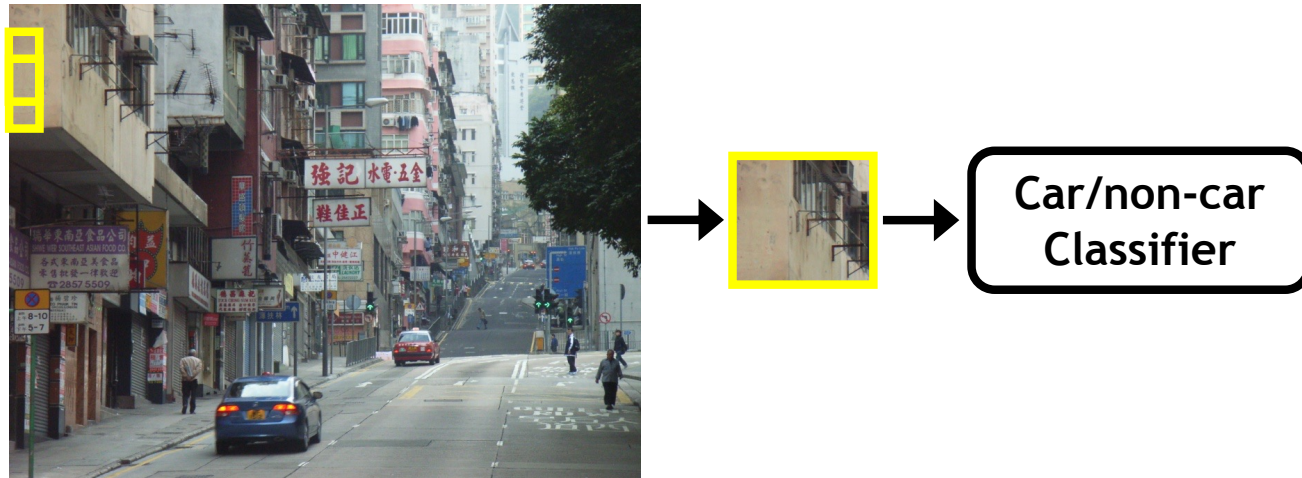
- Build/train object model
- Generate candidate regions in new image
- Score the candidates

Window-template-based models Building an object model

Given the representation, train a binary classifier



Window-template-based models Generating and scoring candidates



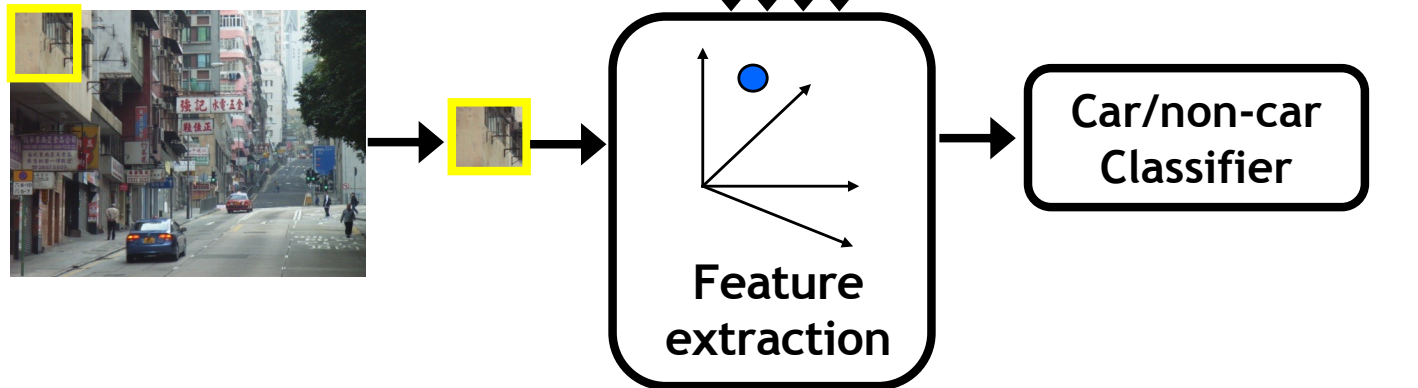
Window-template-based object detection: recap

Training:

1. Obtain training data
2. Define features
3. Define classifier

Given new image:

1. Slide window
2. Score by classifier



Evaluating detection methods

$$mAP = \frac{1}{|classes|} \sum_{c \in classes} \frac{\#TP(c)}{\#TP(c) + \#FP(c)}$$

- True Positive - TP(c): a predicted bounding box (pred_bb) was made for class c, there is a ground truth bounding box (gt_bb) of class c, and $IoU(pred_bb, gt_bb) \geq 0.5$.
- False Positive - FP(c): a pred_bb was made for class c, and there is no gt_bb of class c. Or there is a gt_bb of class c, but $IoU(pred_bb, gt_bb) < 0.5$.



Dalal-Triggs pedestrian detector

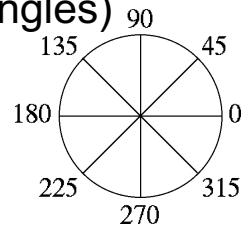


1. Extract fixed-sized (64x128 pixel) window at multiple positions and scales
2. Compute HOG (histogram of gradient) features within each window
3. Score the window with a linear SVM classifier
4. Perform non-maxima suppression to remove overlapping detections with lower scores

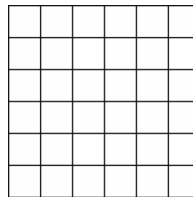
Histograms of oriented gradients (HOG)

Divide image into 8x8 regions

Orientation: 9 bins
(for unsigned
angles)



Histograms in
8x8 pixel cells



Votes weighted by magnitude



Train SVM for pedestrian detection using HoG



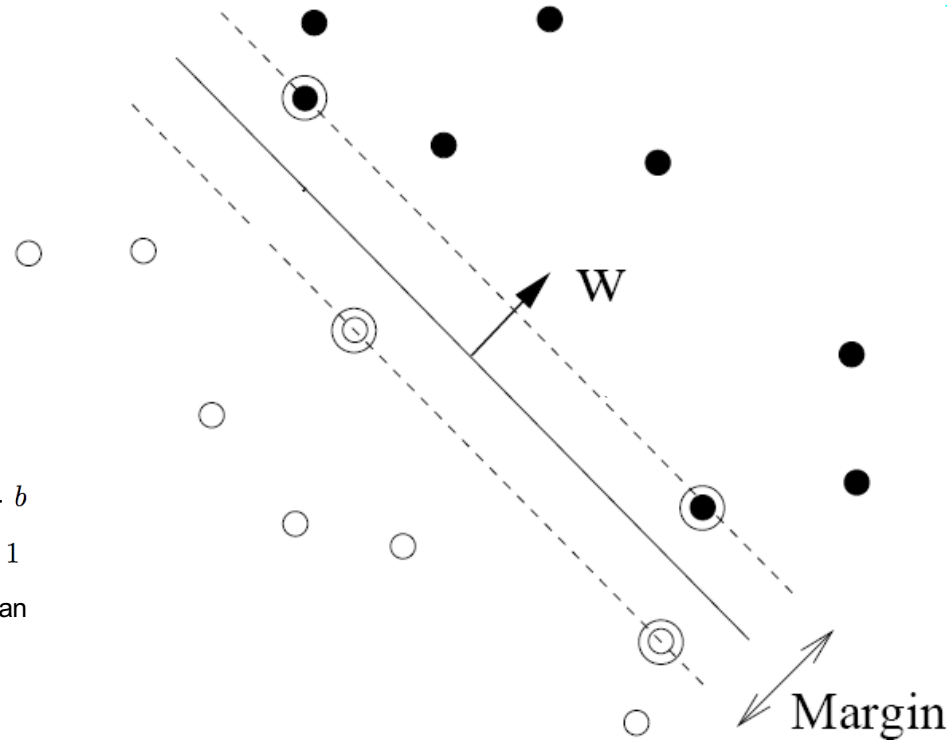
pos w



$$0.16 = w^T x + b$$

$$\text{sign}(0.16) = 1$$

$\Rightarrow$ pedestrian

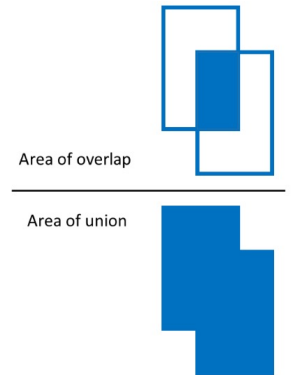
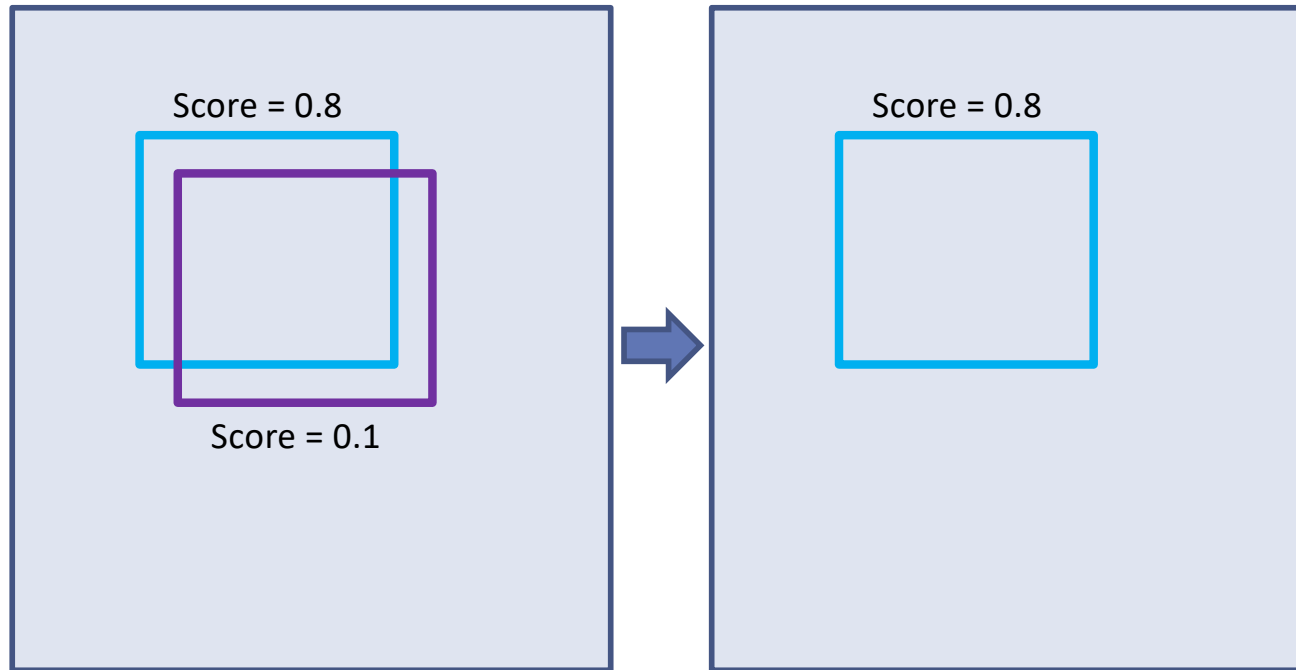


Adapted from Pete Barnum

Navneet Dalal and Bill Triggs, Histograms of Oriented Gradients for Human Detection, CVPR05

Remove overlapping detections

Non-max suppression



Are window templates enough?

- Many objects are articulated, or have parts that can vary in configuration

Images from Caltech-256, D. Ramanan



- Many object categories look very different from different viewpoints, or from instance to instance

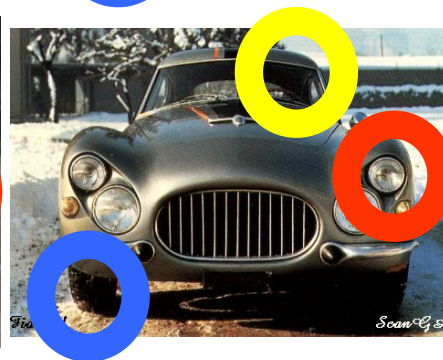


Adapted from N. Snavely, D. Tran

Parts-based Models

Define object by collection of parts modeled by

1. Appearance
2. Spatial configuration



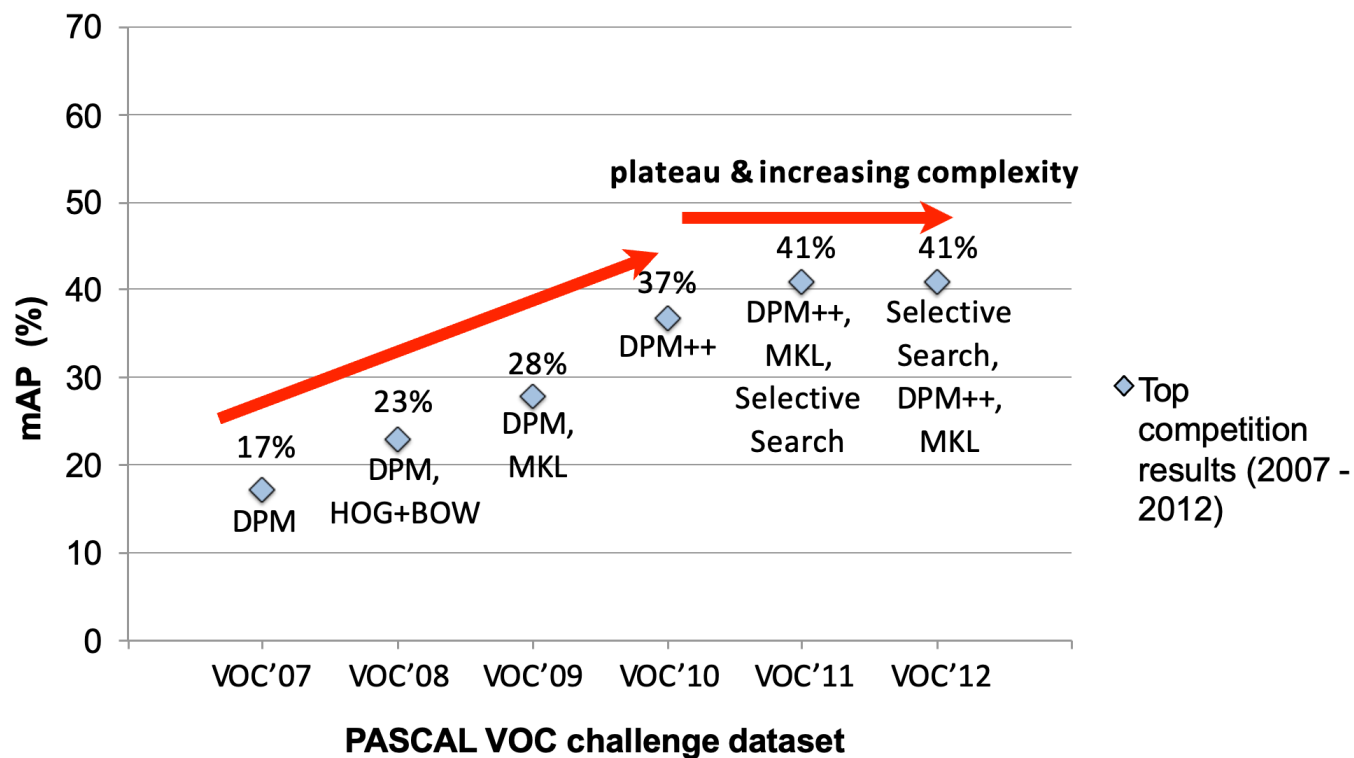
Slide credit: Rob Fergus

Plan for the next three lectures

- Detection approaches
 - Pre-CNNs
 - Detection with whole windows: Pedestrian detection
 - Part-based detection: Deformable Part Models
 - Post-CNNs
 - Detection with region proposals: R-CNN, Fast R-CNN, Faster-R-CNN
 - Detection without region proposals: YOLO, SSD
- Segmentation approaches
 - Semantic segmentation: FCN
 - Instance segmentation: Mask R-CNN

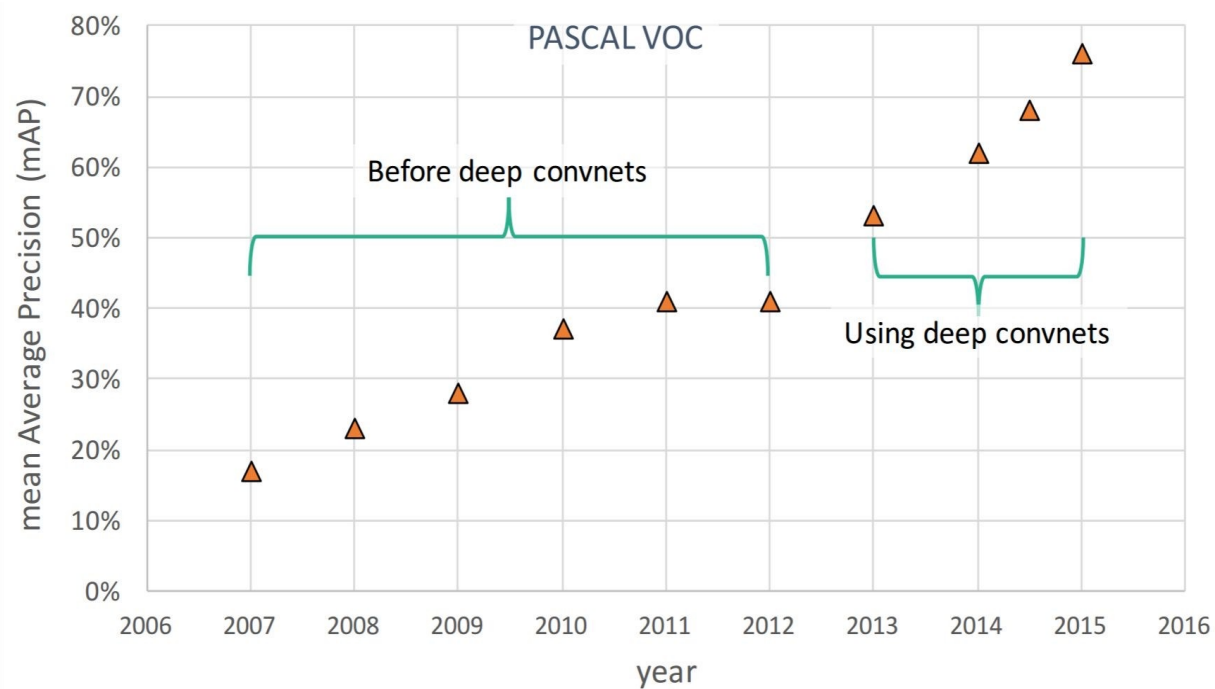
Complexity and the plateau

[Source: <http://pascallin.ecs.soton.ac.uk/challenges/VOC/voc20{07,08,09,10,11,12}/results/index.html>]



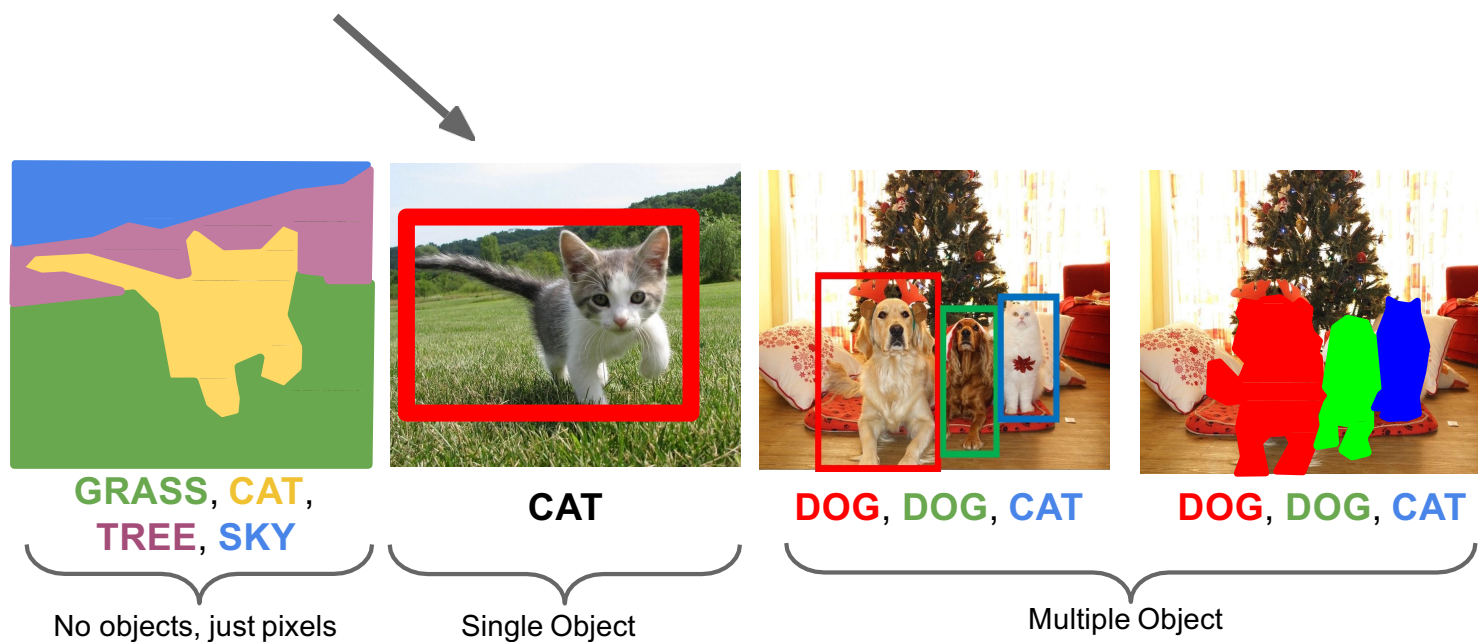
Girshick et al., "Rich Feature Hierarchies for Accurate Object Detection and Semantic Segmentation", CVPR 2014

Impact of Deep Learning

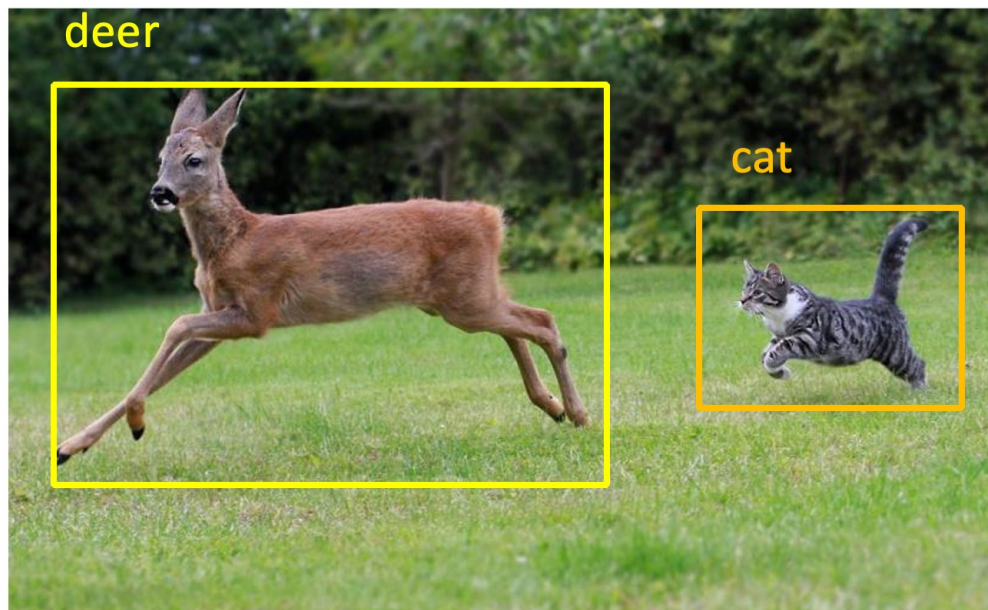


Slide by: Justin Johnson

Classification + Localization

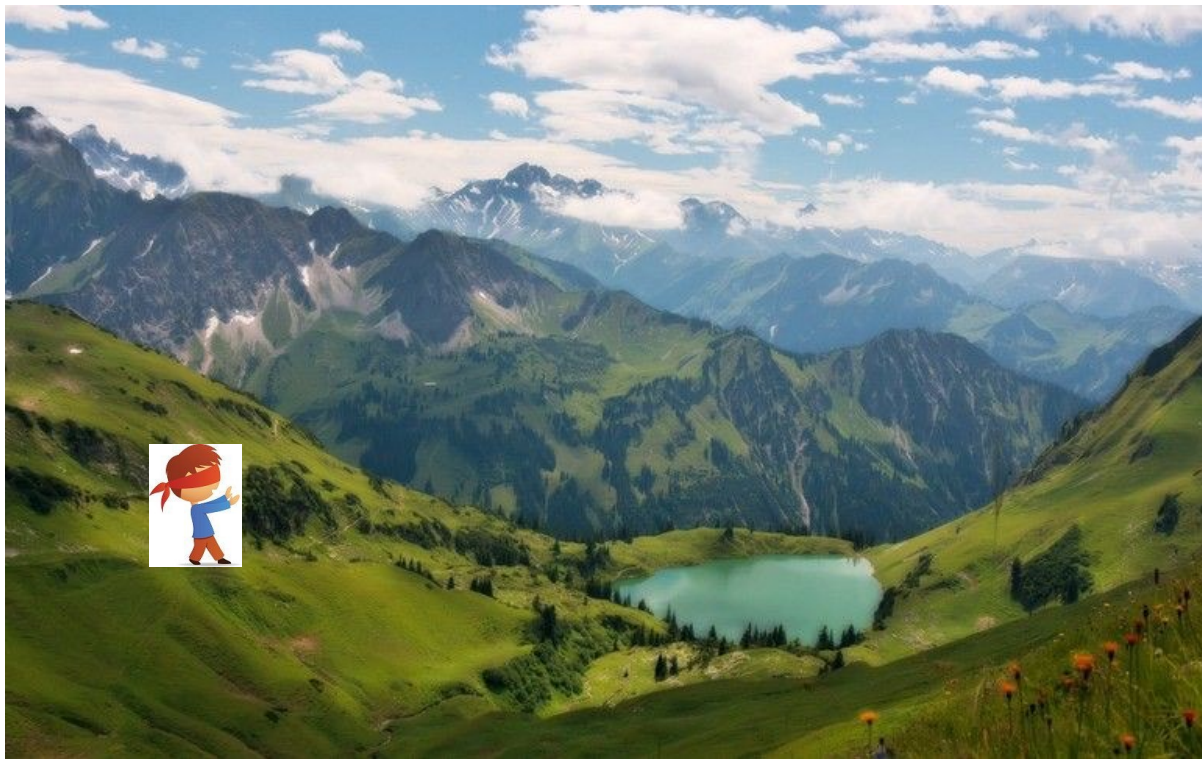


What will be the output of a neural network for object detection?

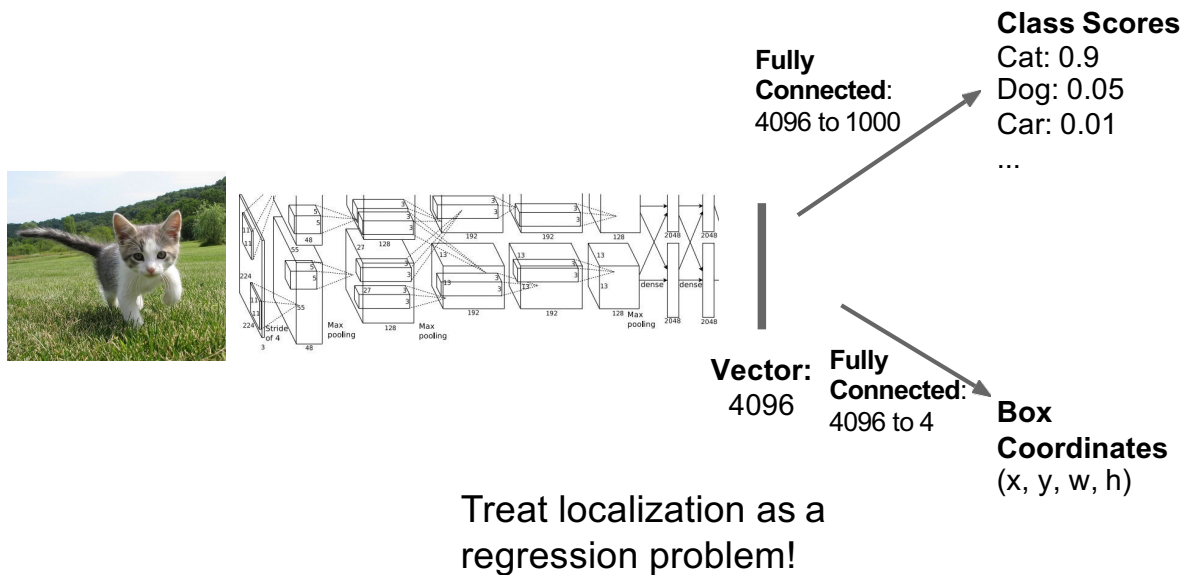


Adapted from Vicente Ordoñez

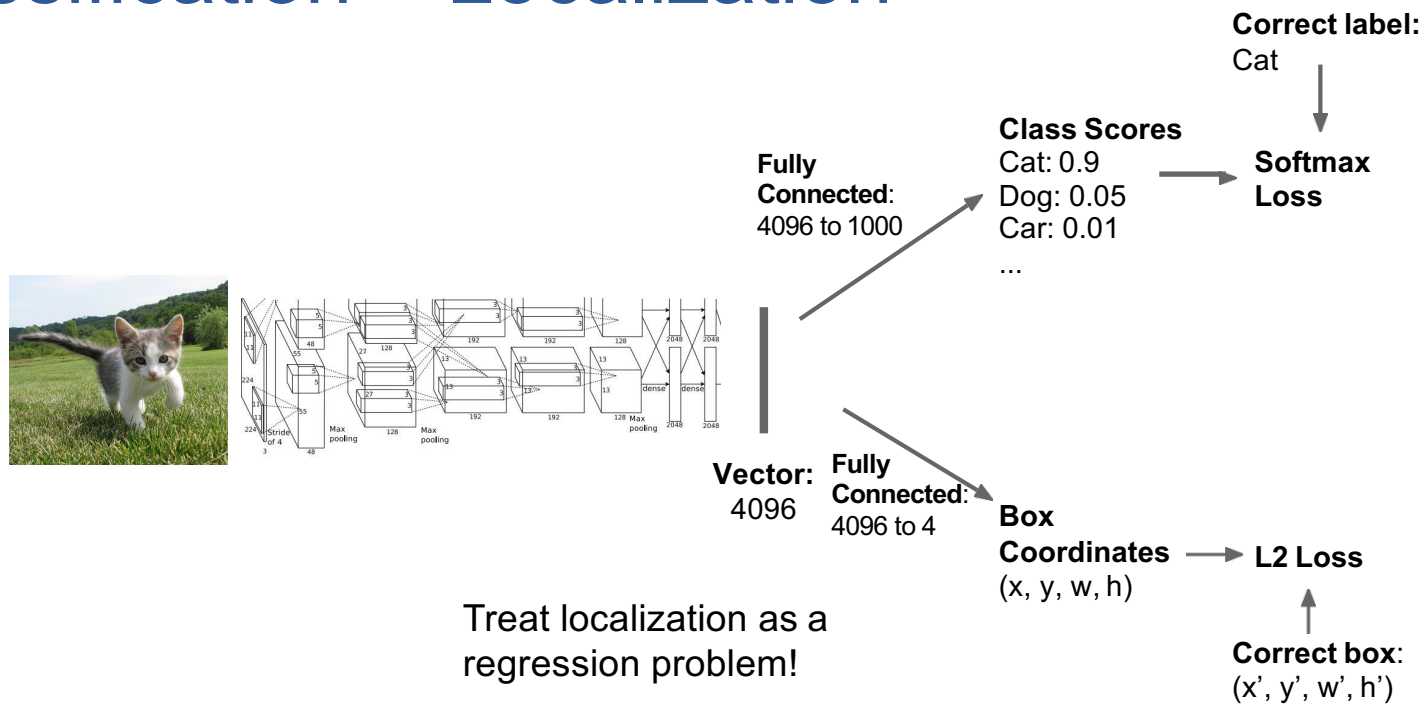
What guides the learning of a neural network?



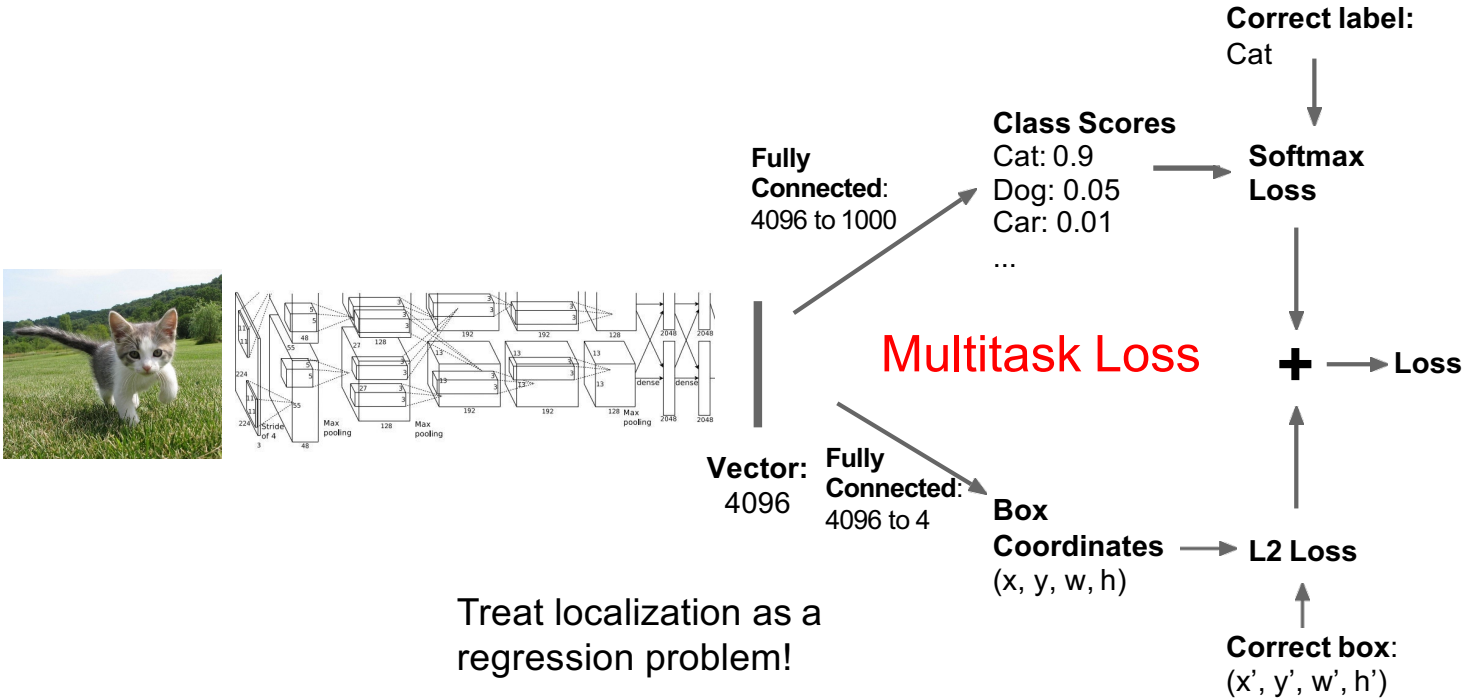
Classification + Localization



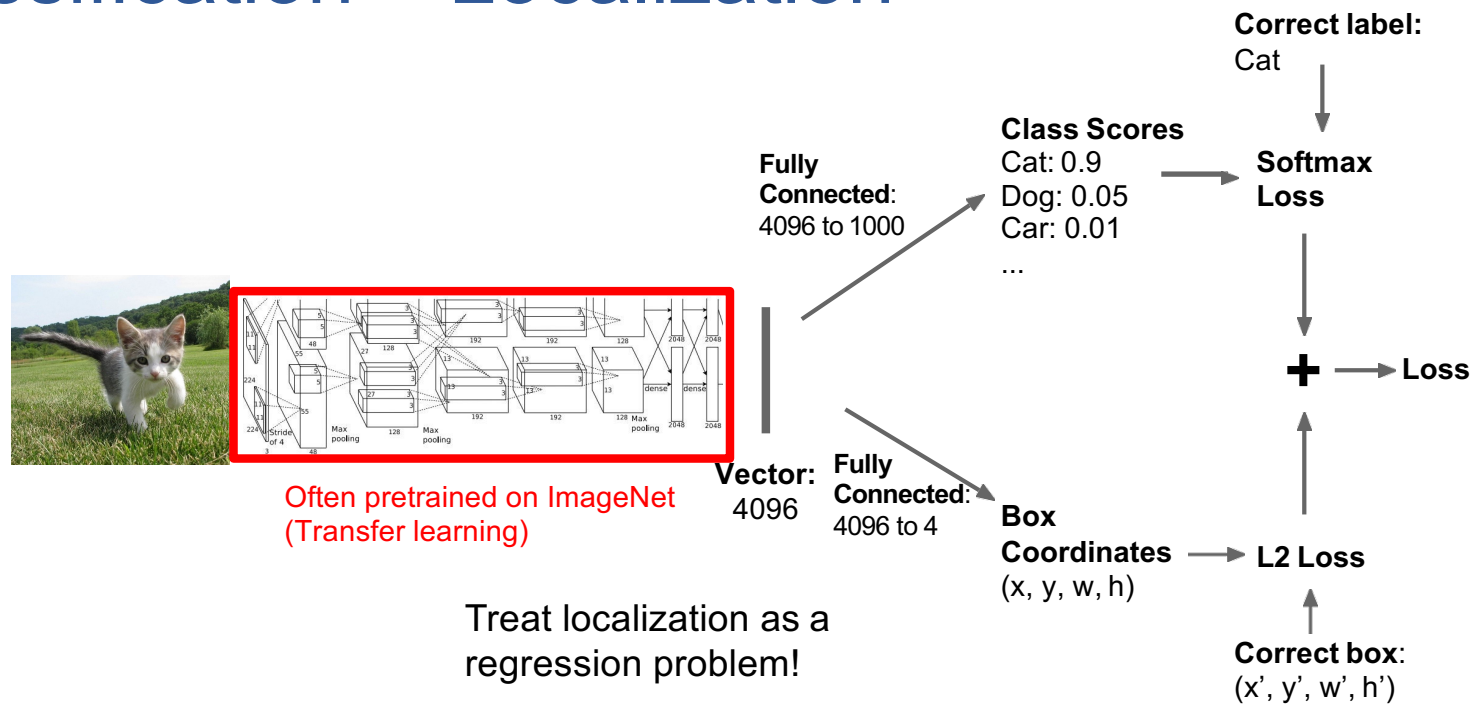
Classification + Localization



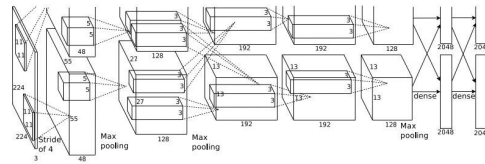
Classification + Localization



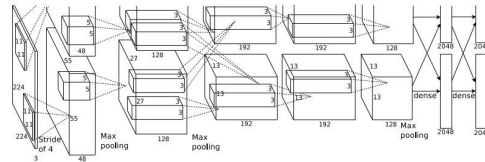
Classification + Localization



Object Detection as Regression?



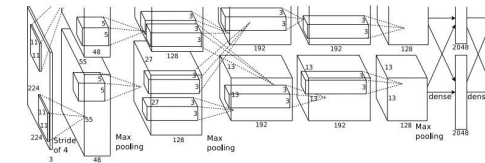
CAT: (x, y, w, h)



DOG: (x, y, w, h)

DOG: (x, y, w, h)

CAT: (x, y, w, h)



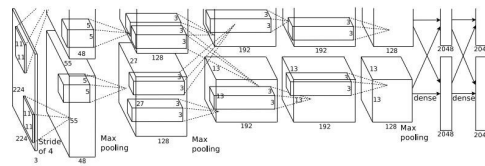
DUCK: (x, y, w, h)

DUCK: (x, y, w, h)

....

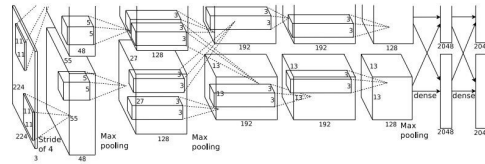
Slide by: Justin Johnson

Object Detection as Regression?



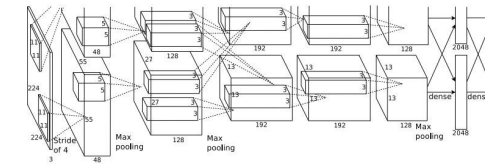
CAT: (x, y, w, h)

4 numbers



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

16 numbers



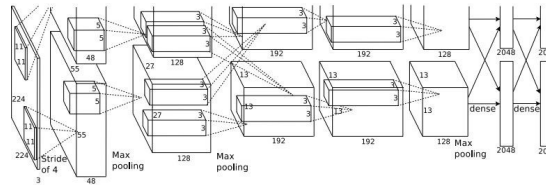
DUCK: (x, y, w, h)
DUCK: (x, y, w, h)
....

Many
numbers!

Each image needs a
different number of outputs!

Object Detection as Classification: Sliding Window

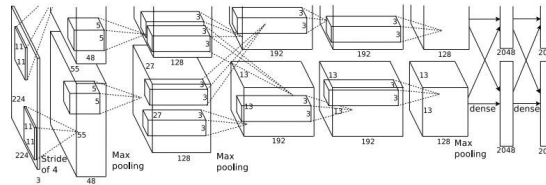
Apply a CNN to many different crops of the image, CNN classifies each crop as object or background



Dog? NO
Cat? NO
Background? YES

Object Detection as Classification: Sliding Window

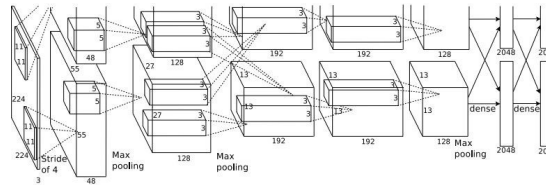
Apply a CNN to many different crops of the image, CNN classifies each crop as object or background



Dog? YES
Cat? NO
Background? NO

Object Detection as Classification: Sliding Window

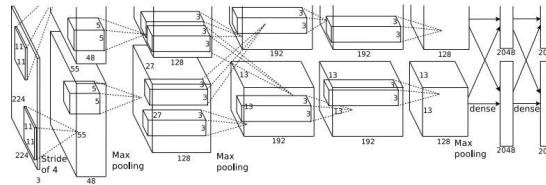
Apply a CNN to many different crops of the image, CNN classifies each crop as object or background



Dog? YES
Cat? NO
Background? NO

Object Detection as Classification: Sliding Window

Apply a CNN to many different crops of the image, CNN classifies each crop as object or background

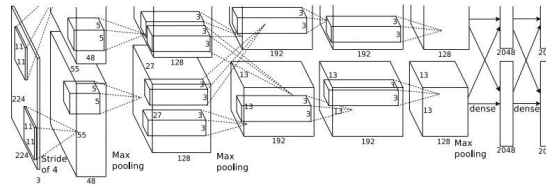


Dog? NO
Cat? YES
Background? NO

Slide by: Justin Johnson

Object Detection as Classification: Sliding Window

Apply a CNN to many different crops of the image, CNN classifies each crop as object or background



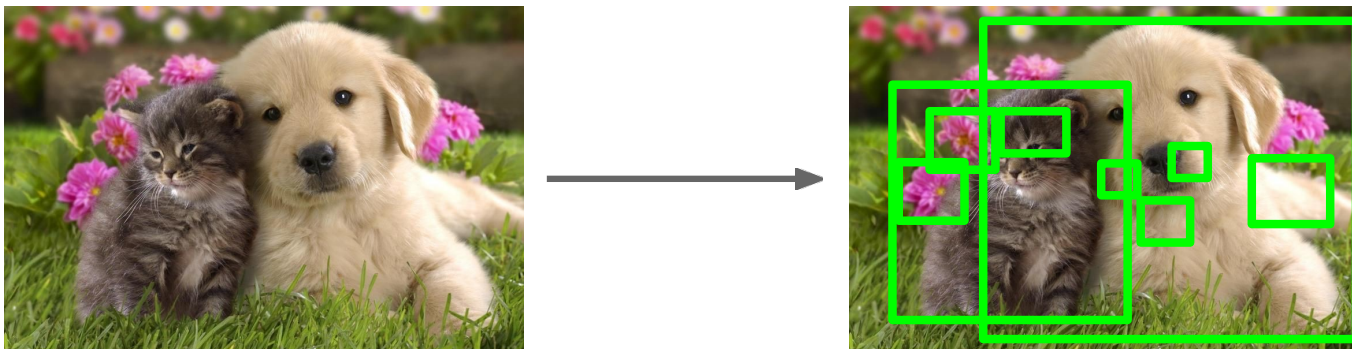
Dog? NO
Cat? YES
Background? NO

Problem: Need to apply CNN to huge number of locations and scales, very computationally expensive!

What can we do?

Region Proposals

- Find “blobby” image regions that are likely to contain objects
- Relatively fast to run; e.g. Selective Search gives 1000 region proposals in a few seconds on CPU



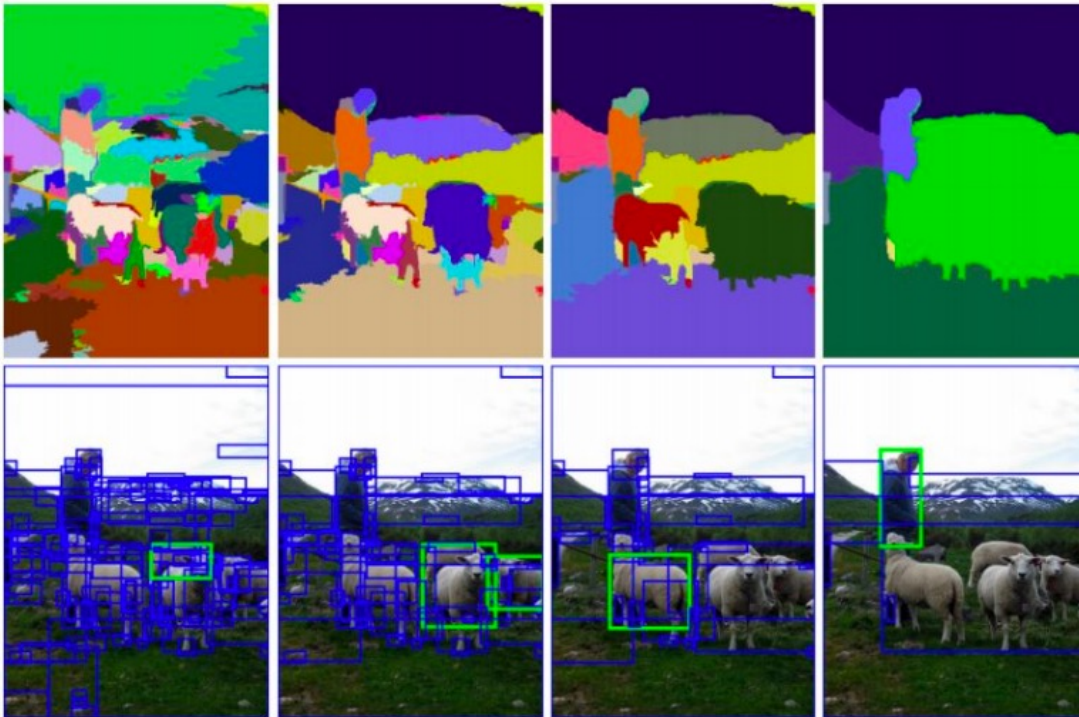
Alexe et al, “Measuring the objectness of image windows”, TPAMI 2012

Uijlings et al, “Selective Search for Object Recognition”, IJCV 2013

Cheng et al, “BING: Binarized normed gradients for objectness estimation at 300fps”, CVPR 2014

Zitnick and Dollar, “Edge boxes: Locating object proposals from edges”, ECCV 2014

Box Proposal Method – SS: Selective Search



Segmentation As
Selective Search for
Object Recognition.
van de Sande et al.
ICCV 2011

Adapted from Vicente Ordoñez

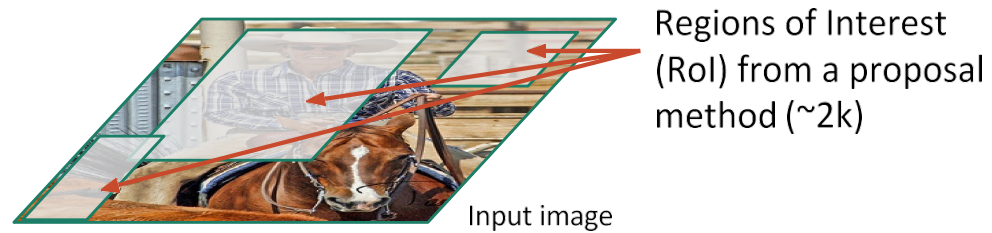
R-CNN



Input image

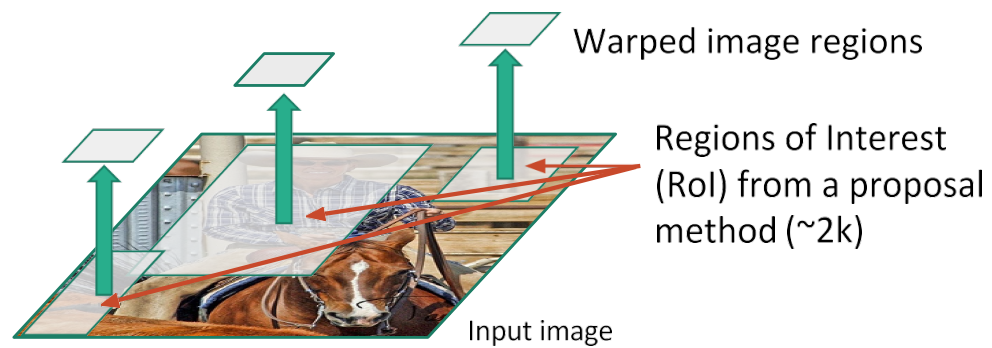
*Girshick et al., "Rich Feature Hierarchies for Accurate Object Detection and Semantic Segmentation",
CVPR 2014*

R-CNN



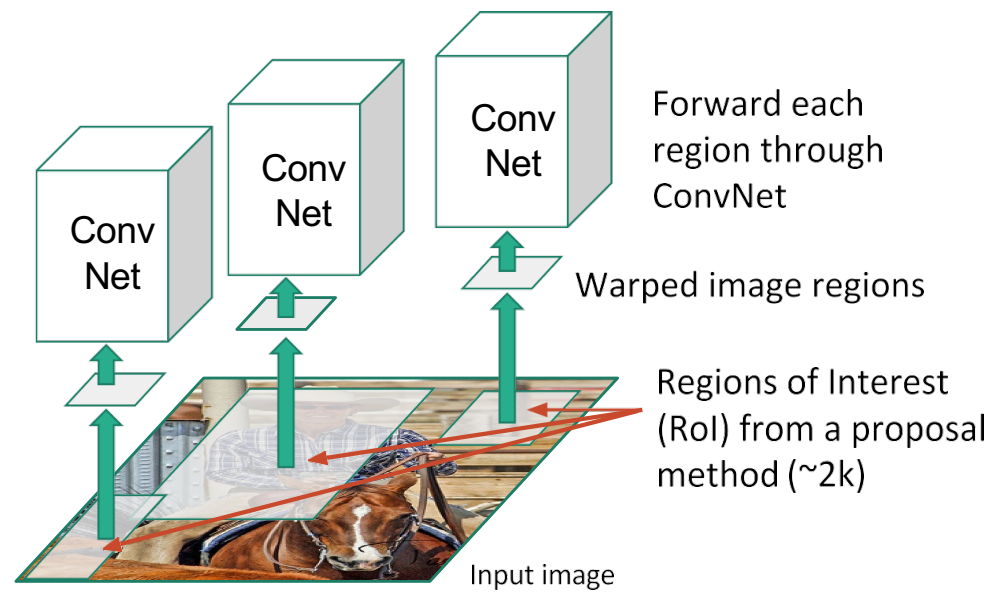
Girshick et al., "Rich Feature Hierarchies for Accurate Object Detection and Semantic Segmentation", CVPR 2014

R-CNN



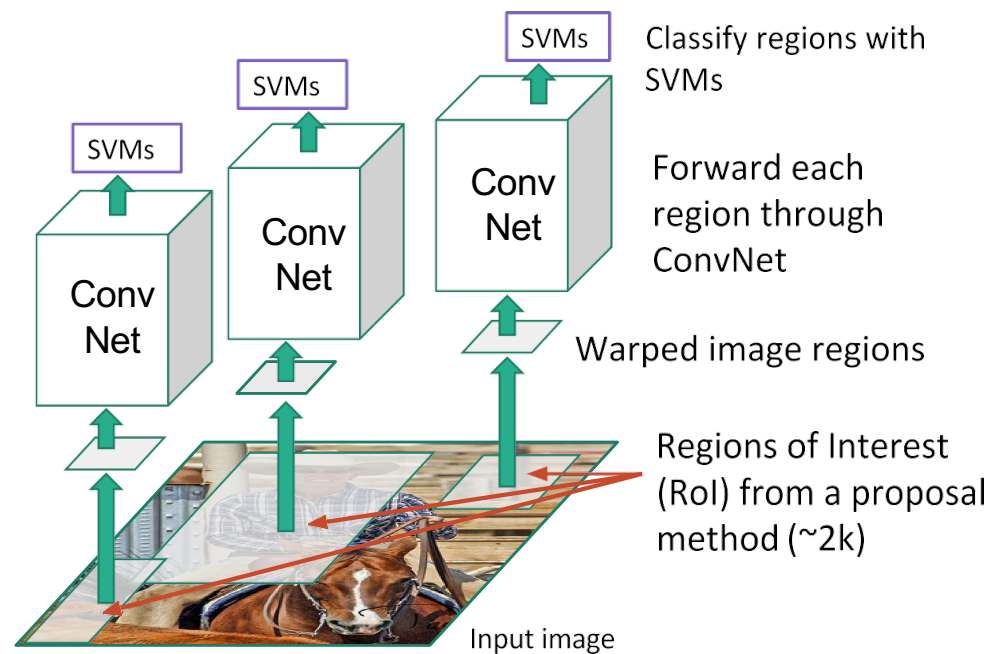
*Girshick et al., "Rich Feature Hierarchies for Accurate Object Detection and Semantic Segmentation",
CVPR 2014*

R-CNN



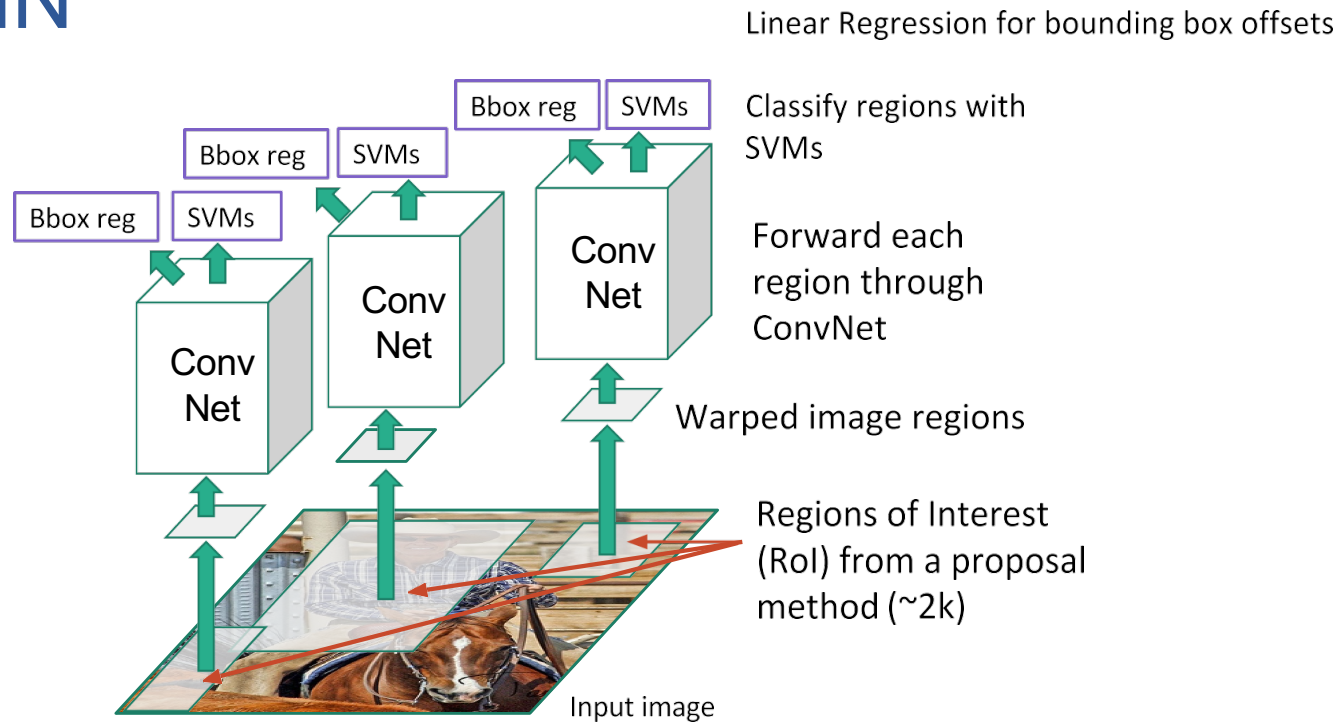
Girshick et al., "Rich Feature Hierarchies for Accurate Object Detection and Semantic Segmentation", CVPR 2014

R-CNN



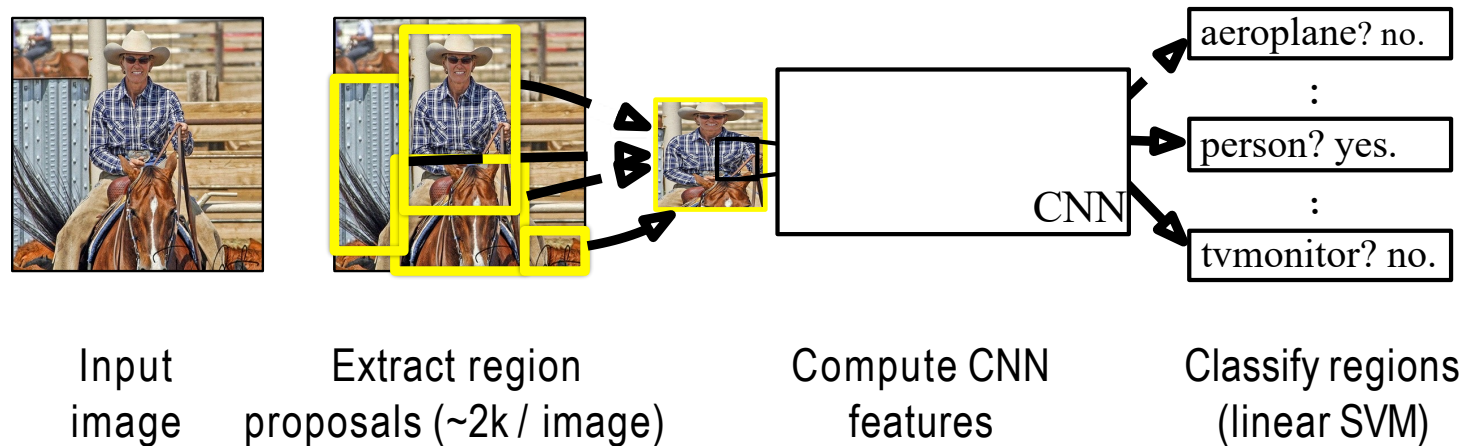
Girshick et al., "Rich Feature Hierarchies for Accurate Object Detection and Semantic Segmentation", CVPR 2014

R-CNN



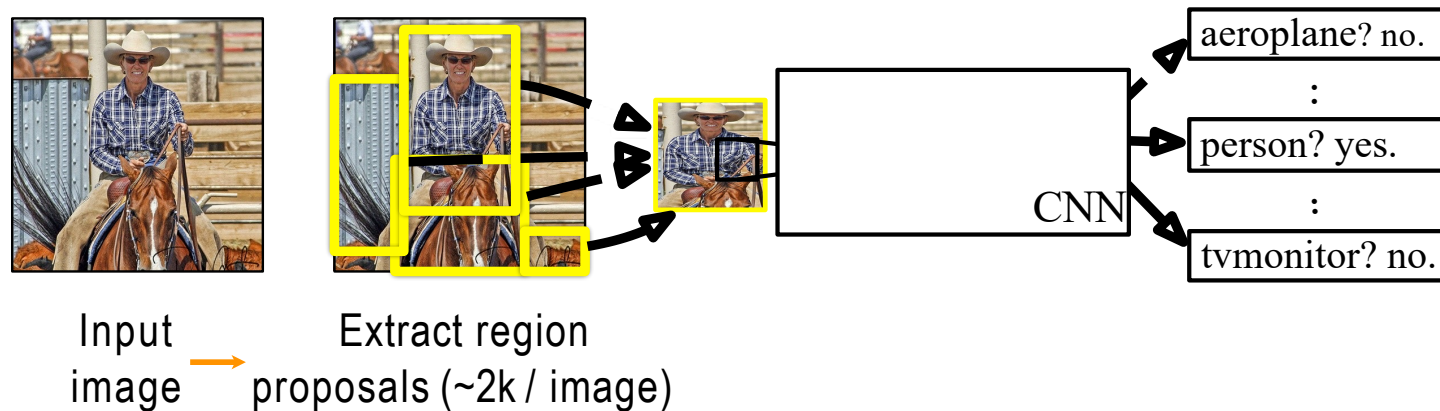
Girshick et al., "Rich Feature Hierarchies for Accurate Object Detection and Semantic Segmentation", CVPR 2014

R-CNN: Regions with CNN features



Girshick et al., "Rich Feature Hierarchies for Accurate Object Detection and Semantic Segmentation", CVPR 2014

R-CNN at test time: Step 1

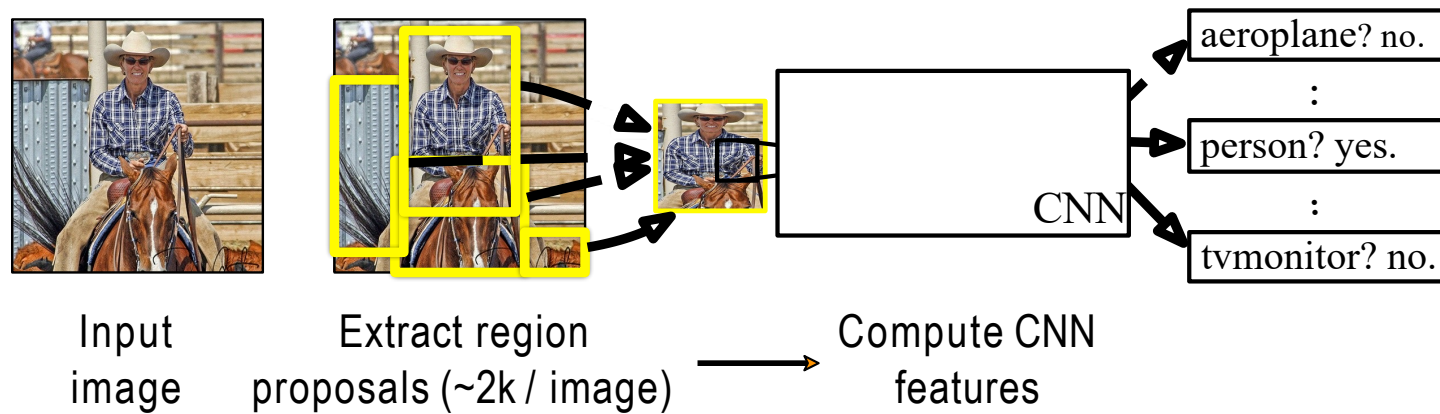


Proposal-method agnostic, many choices

- Selective Search [van de Sande, Uijlings et al.] (Used in this work)
- Objectness [Alexe et al.]
- Category independent object proposals [Endres & Hoiem]
- CPMC [Carreira & Sminchisescu]

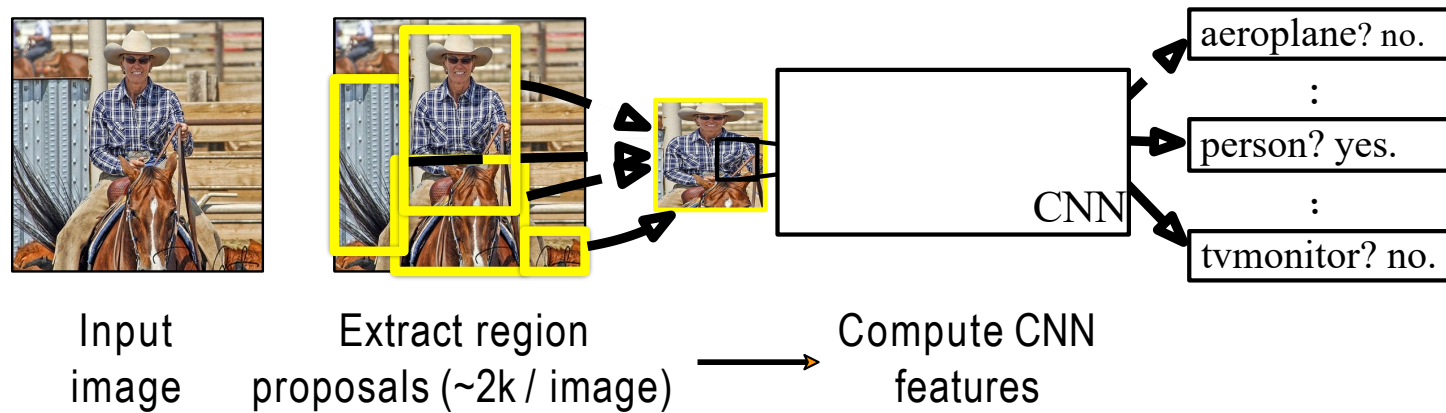
Girshick et al., "Rich Feature Hierarchies for Accurate Object Detection and Semantic Segmentation", CVPR 2014

R-CNN at test time: Step 2



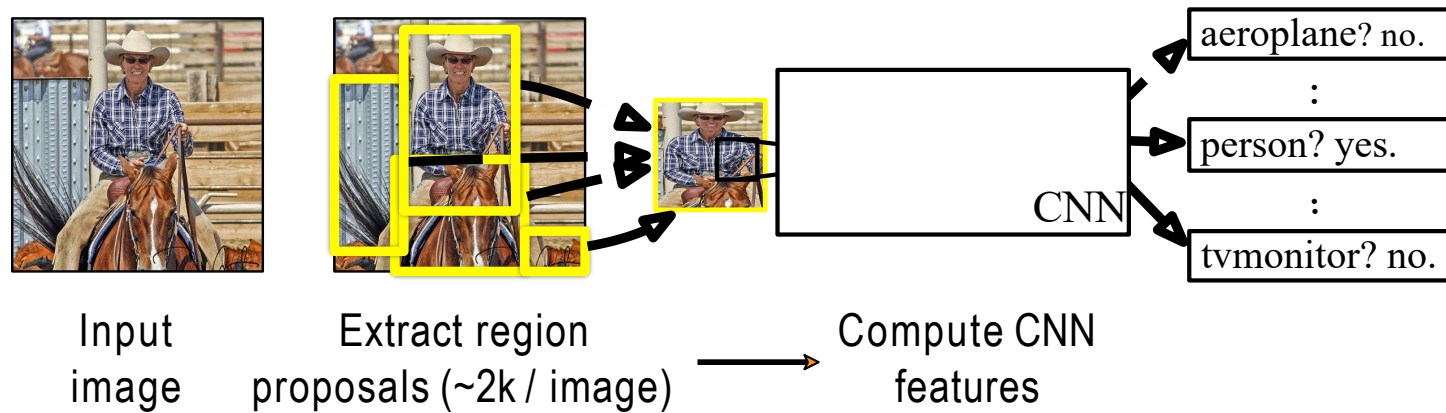
Girshick et al., "Rich Feature Hierarchies for Accurate Object Detection and Semantic Segmentation", CVPR 2014

R-CNN at test time: Step 2



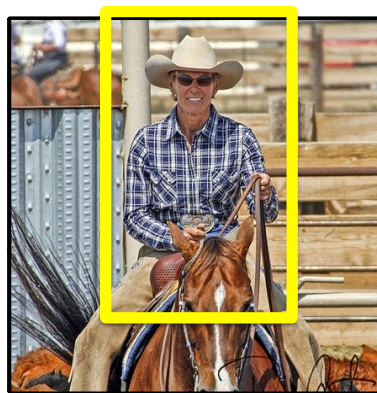
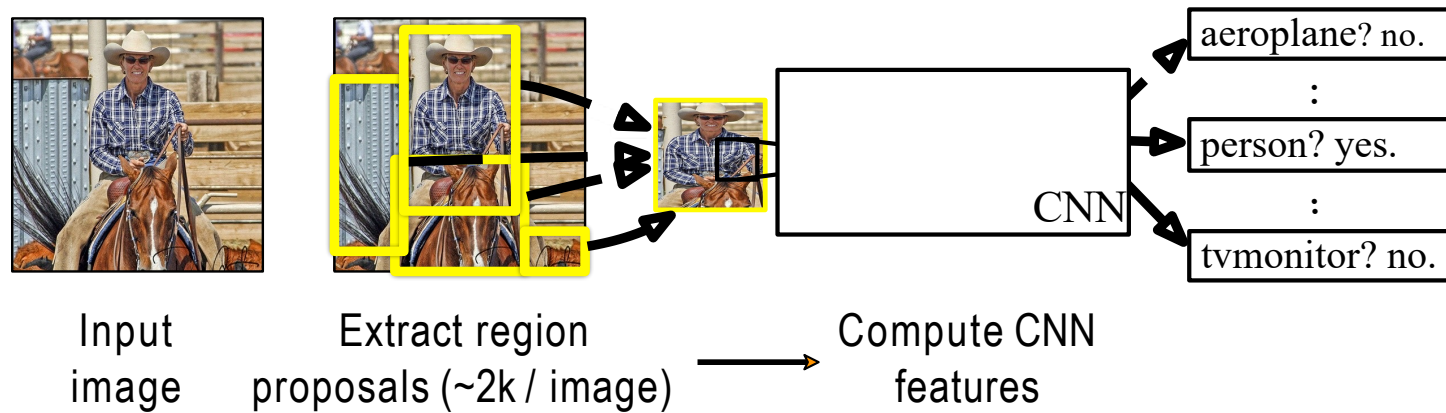
Dilate proposal

R-CNN at test time: Step 2



Girshick et al., "Rich Feature Hierarchies for Accurate Object Detection and Semantic Segmentation", CVPR 2014

R-CNN at test time: Step 2



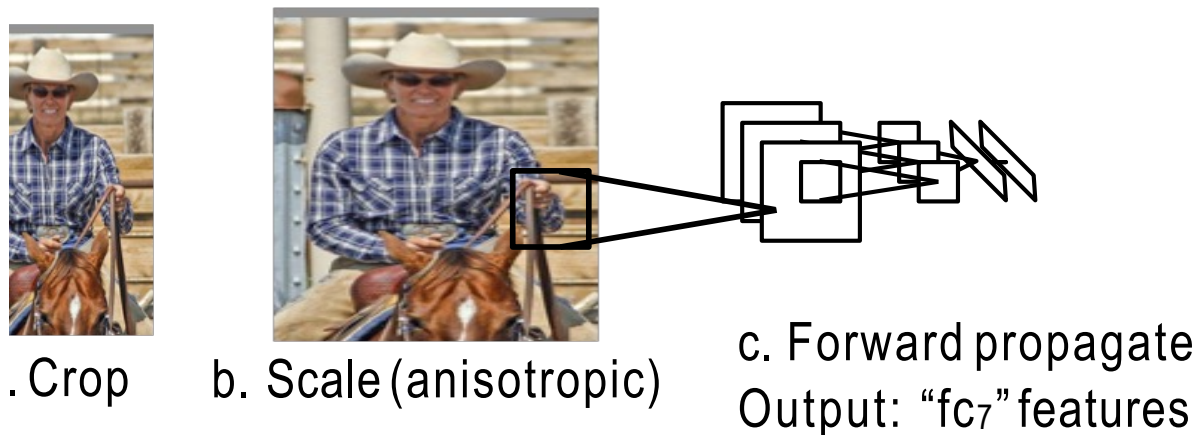
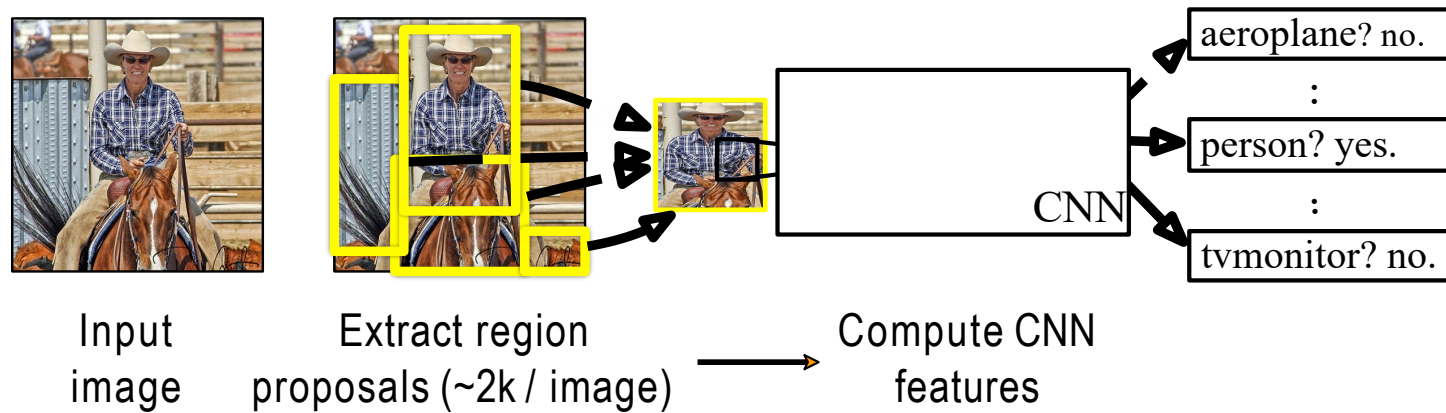
a. Crop



b. Scale (anisotropic)

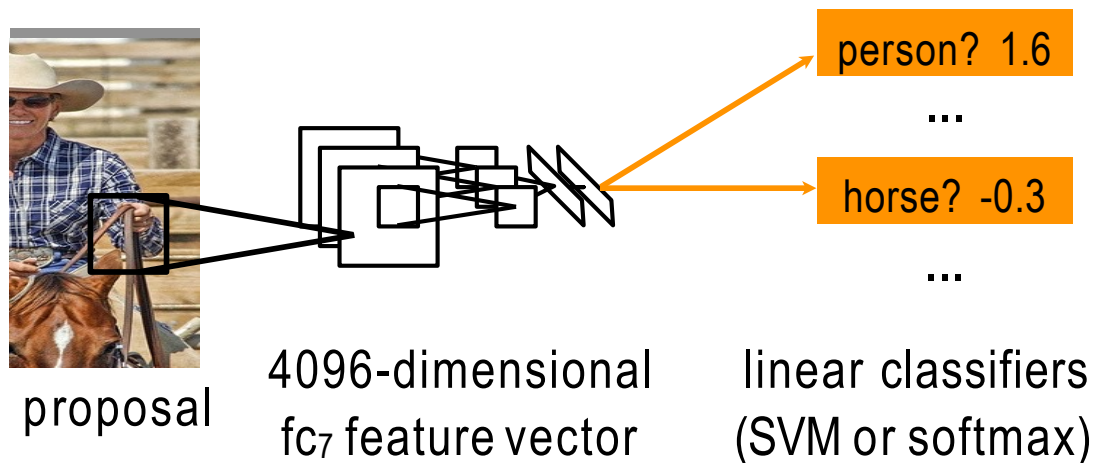
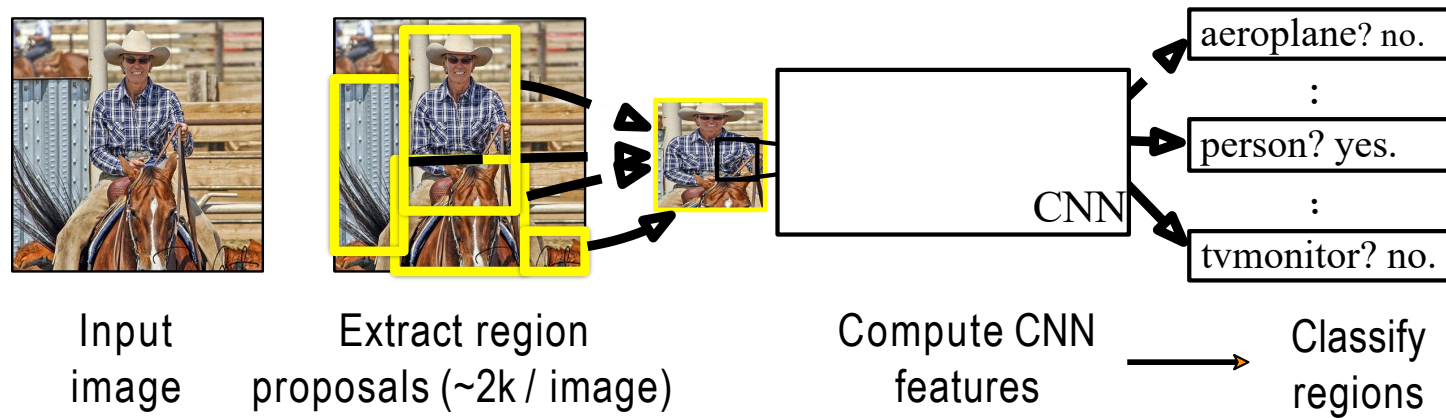
227 x 227

R-CNN at test time: Step 2



Girshick et al., "Rich Feature Hierarchies for Accurate Object Detection and Semantic Segmentation", CVPR 2014

R-CNN at test time: Step 3



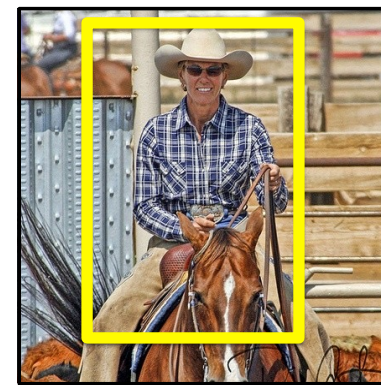
Girshick et al., "Rich Feature Hierarchies for Accurate Object Detection and Semantic Segmentation", CVPR 2014

Step 4: Object proposal refinement



Original
proposal

Linear regression
→
on CNN features



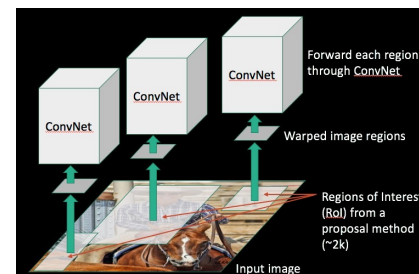
Predicted
object bounding box

Bounding-box regression

Girshick et al., "Rich Feature Hierarchies for Accurate Object Detection and Semantic Segmentation", CVPR 2014

What's wrong with slow R-CNN?

- Ad-hoc training objectives
 - Train post-hoc linear SVMs (hinge loss)
 - Train post-hoc bounding-box regressions (L2 loss)
- Training is slow (84h), takes a lot of disk space
 - Need to store all region crops
- Inference (detection) is slow
 - 47s / image with VGG16 [Simonyan & Zisserman, ICLR15]



Adapted from Girshick, "Fast R-CNN", ICCV 2015

~2000 ConvNet forward passes per image

Fast R-CNN

- One network, applied one time, not 2000 times
- Trained end-to-end (in one stage)
- Fast test time
- Higher mean average precision

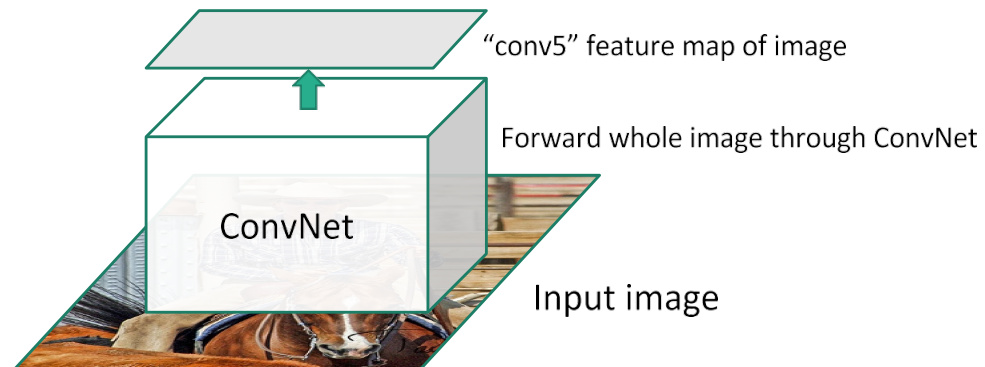
Fast R-CNN



Input image

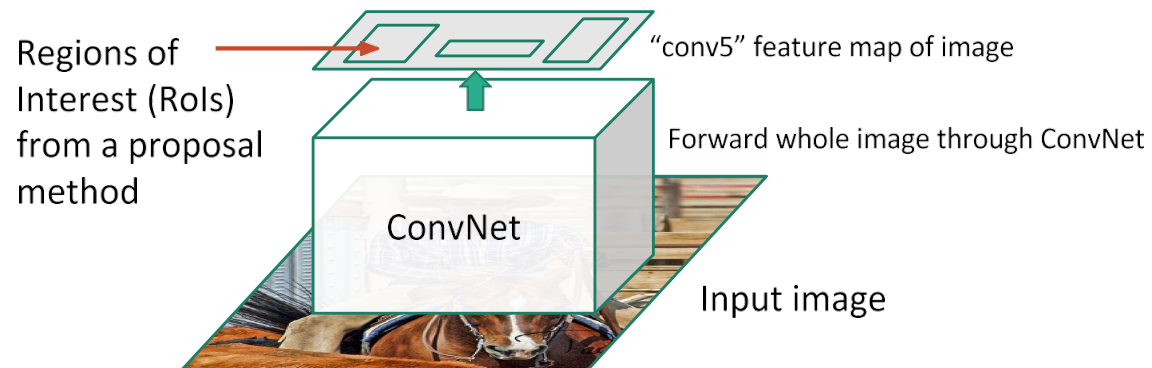
Adapted from Girshick, "Fast R-CNN", ICCV 2015

Fast R-CNN



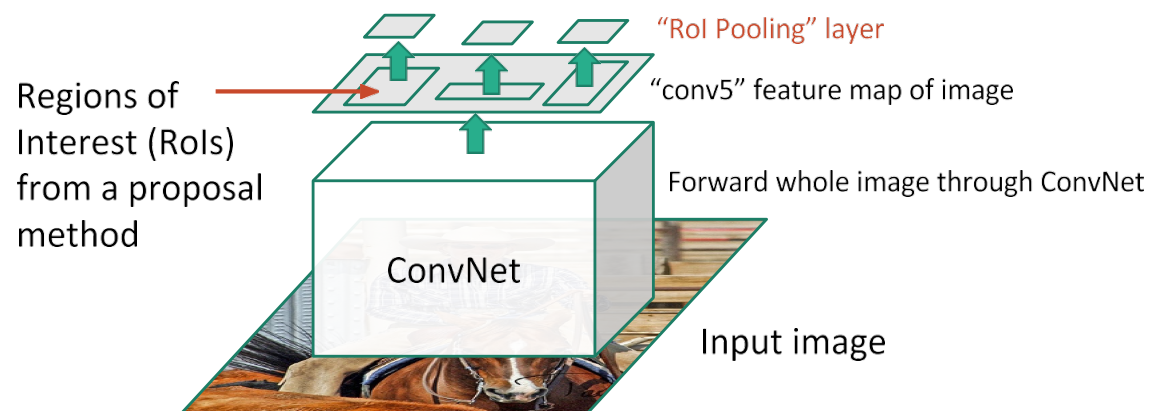
Adapted from Girshick, "Fast R-CNN", ICCV 2015

Fast R-CNN



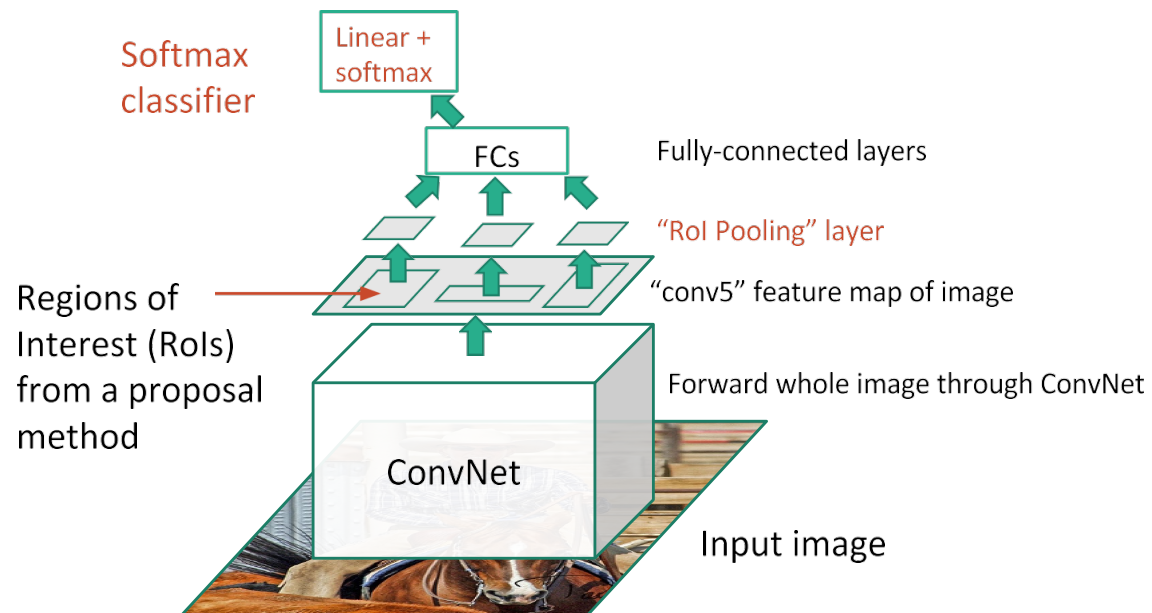
Adapted from Girshick, "Fast R-CNN", ICCV 2015

Fast R-CNN



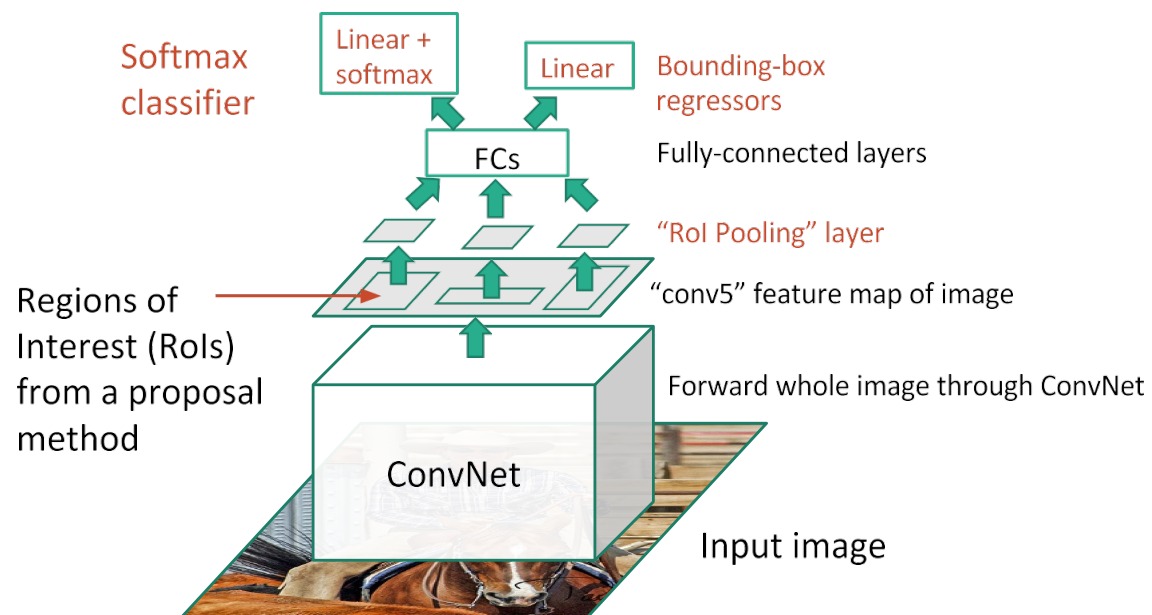
Adapted from Girshick, "Fast R-CNN", ICCV 2015

Fast R-CNN



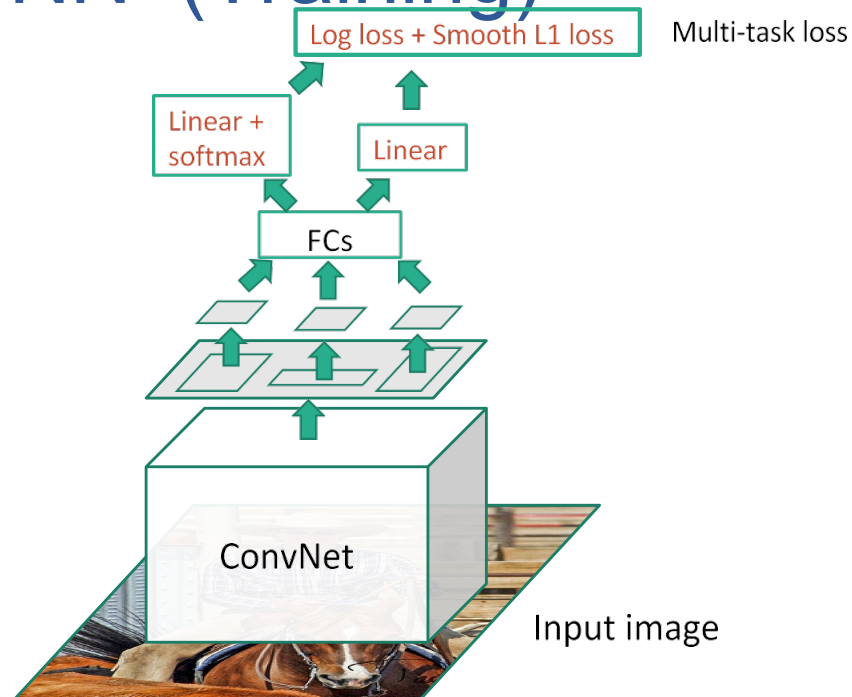
Adapted from Girshick, "Fast R-CNN", ICCV 2015

Fast R-CNN



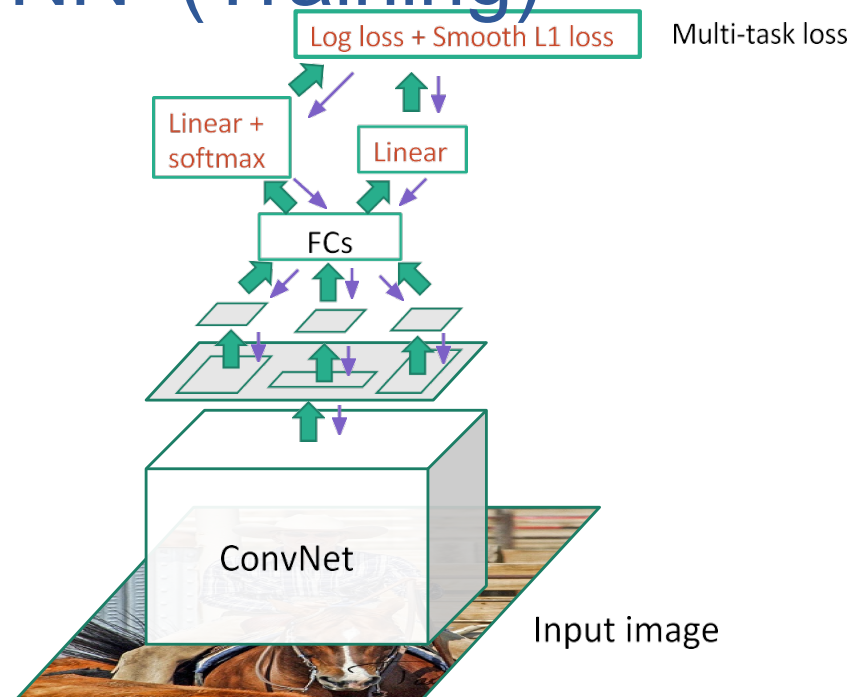
Adapted from Girshick, "Fast R-CNN", ICCV 2015

Fast R-CNN (Training)



Adapted from Girshick, "Fast R-CNN", ICCV 2015

Fast R-CNN (Training)



Adapted from Girshick, "Fast R-CNN", ICCV 2015

Fast R-CNN vs R-CNN

	Fast R-CNN	R-CNN
Train time (h)	9.5	84
Speedup	8.8x	1x
Test time / image	0.32s	47.0s
Test speedup	146x	1x
mAP	66.9%	66.0%

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

Adapted from Girshick, "Fast R-CNN", ICCV 2015

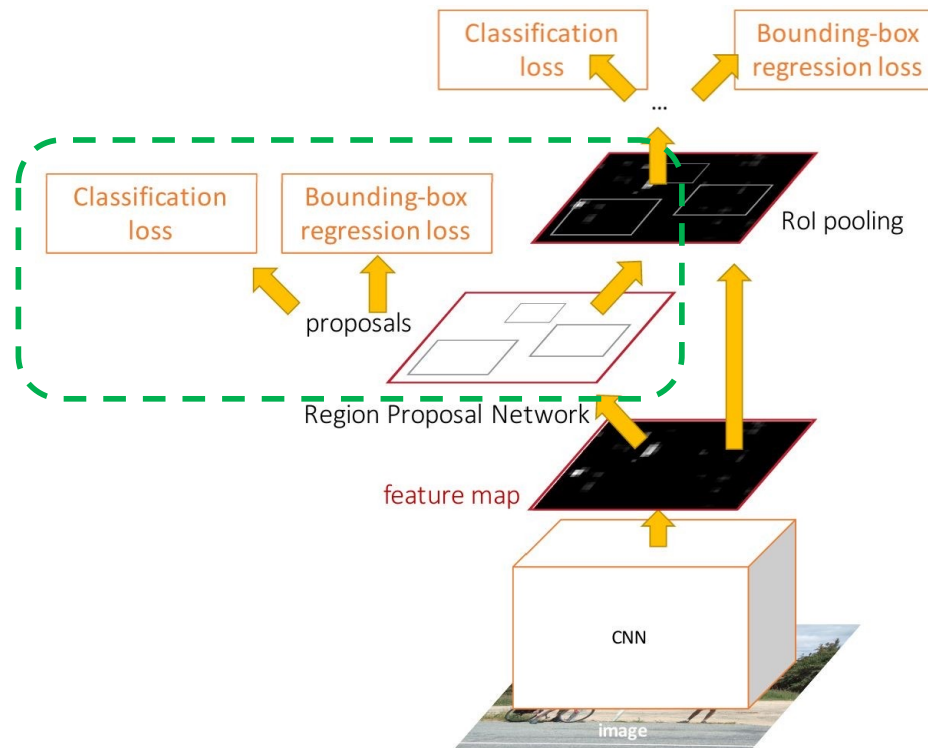
Faster R-CNN

Make CNN do proposals!

Insert **Region Proposal Network (RPN)** to predict proposals from features

Jointly train with 4 losses:

1. RPN classify object / not object
2. RPN regress box coordinates
3. Final classification score (object classes)
4. Final box coordinates



Accurate object detection is slow!

	Pascal 2007 mAP	Speed	
DPM v5	33.7	.07 FPS	14 s/img
R-CNN	66.0	.05 FPS	20 s/img



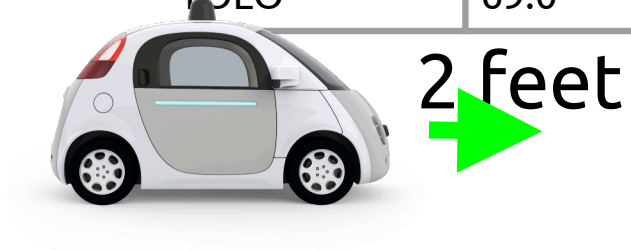
1/3 Mile, 1760 feet



Redmon et al., "You Only Look Once: Unified, Real-Time Object Detection", CVPR 2016

Accurate object detection is slow!

	Pascal 2007 mAP	Speed	
DPM v5	33.7	.07 FPS	14 s/img
R-CNN	66.0	.05 FPS	20 s/img
Fast R-CNN	70.0	.5 FPS	2 s/img
Faster R-CNN	73.2	7 FPS	140 ms/img
YOLO	69.0	45 FPS	22 ms/img



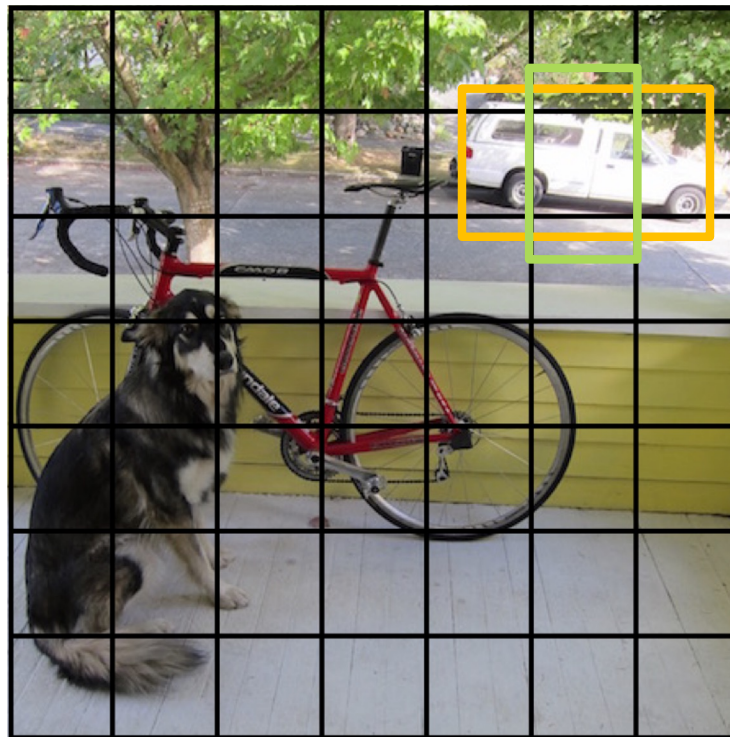
Redmon et al., "You Only Look Once: Unified, Real-Time Object Detection", CVPR 2016

Detection without Proposals: YOLO



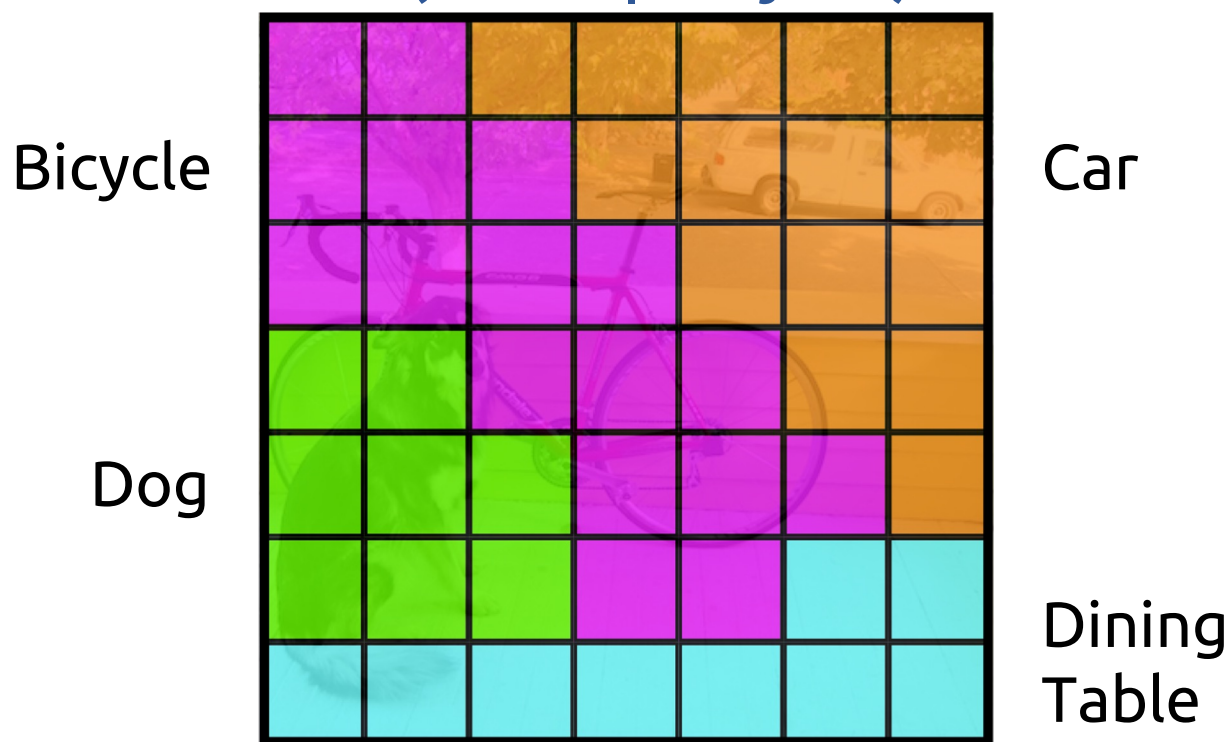
Redmon et al., "You Only Look Once: Unified, Real-Time Object Detection", CVPR 2016

Each cell predicts boxes and confidences:
 $P(\text{Object})$



Redmon et al., "You Only Look Once: Unified, Real-Time Object Detection", CVPR 2016

Each cell also predicts a probability
 $P(\text{Class} \mid \text{Object})$

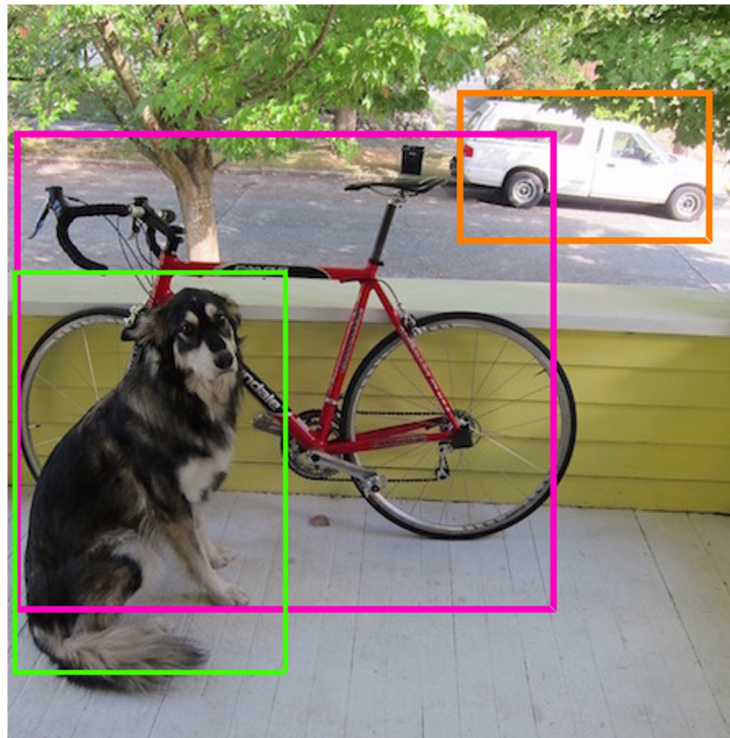


Combine the box and class predictions



Redmon et al., "You Only Look Once: Unified, Real-Time Object Detection", CVPR 2016

Finally do NMS and threshold detections

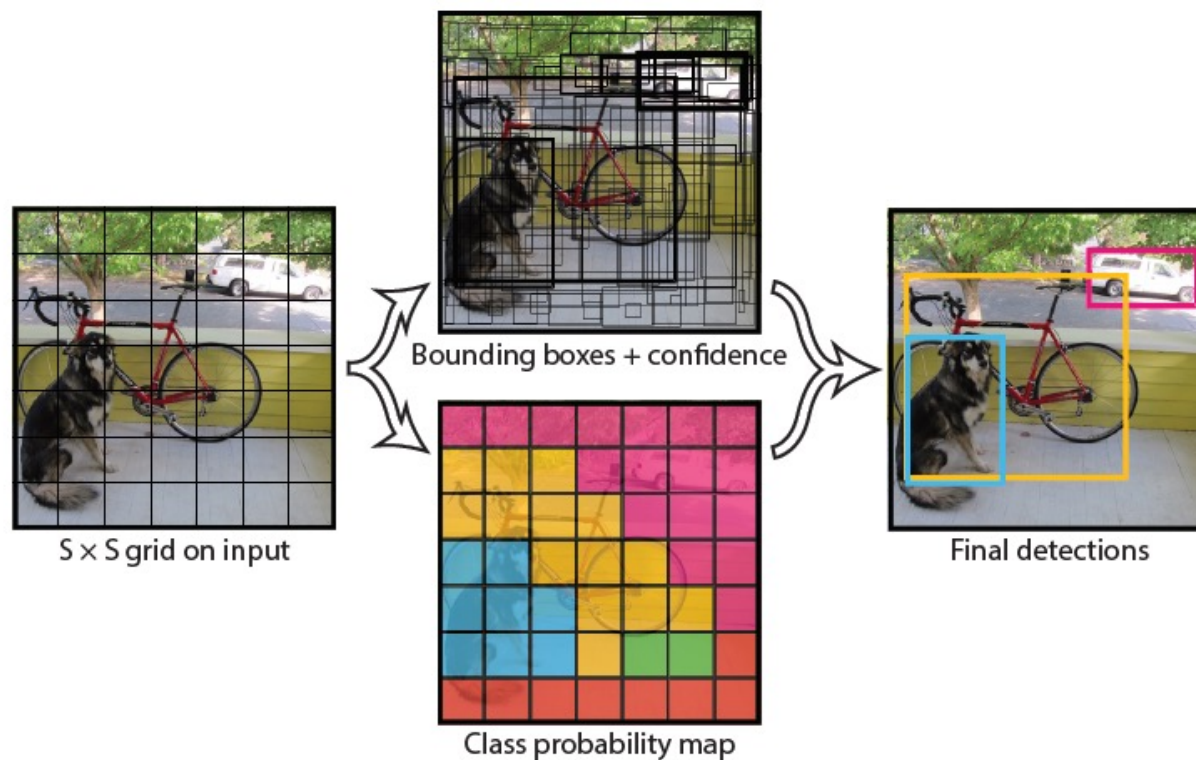


Redmon et al., "You Only Look Once: Unified, Real-Time Object Detection", CVPR 2016

YOLO- You Only Look Once

Idea: No bounding box proposals.

Predict a class and a box for every location in a grid.



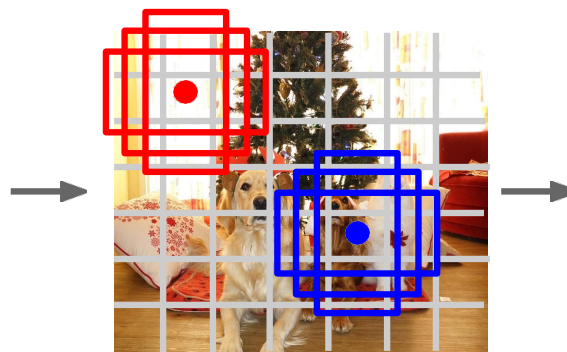
<https://arxiv.org/abs/1506.02640>

Redmon et al. CVPR 2016.

Detection without Proposals: YOLO



Input image
 $3 \times H \times W$



Divide image into grid
 7×7

Image a set of **base boxes**
centered at each grid cell
Here $B = 3$

Within each grid cell:

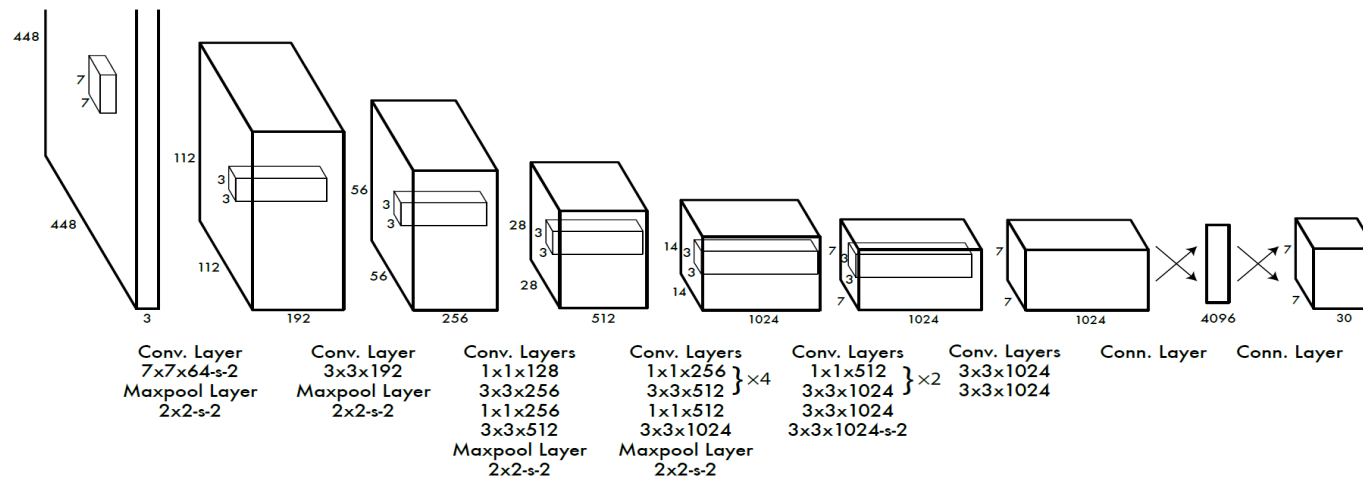
- Regress from each of the B base boxes to a final box with 5 numbers: $(x, y, w, h, \text{confidence})$
- Predict scores for each of C classes (including background as a class)

Output:
 $7 \times 7 \times (5 * B + C)$

Redmon et al, "You Only Look Once: Unified, Real-Time Object Detection", CVPR 2016
Liu et al, "SSD: Single-Shot MultiBox Detector", ECCV 2016

Slide by: Justin Johnson

YOLO- You Only Look Once



Divide the image into 7x7 cells.

Each cell trains a detector.

The detector needs to predict the object's class distributions.

The detector has 2 bounding-box predictors to predict bounding-boxes and confidence scores.

<https://arxiv.org/abs/1506.02640>

Redmon et al. CVPR 2016.

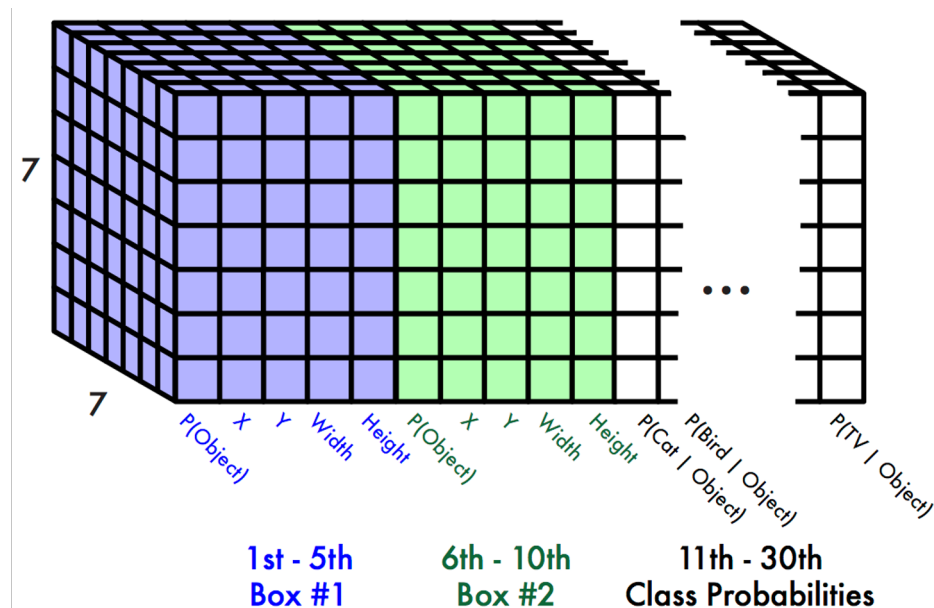
This parameterization fixes the output size

Each cell predicts:

- For each bounding box:
 - 4 coordinates (x, y, w, h)
 - 1 confidence value
- Some number of class probabilities

For Pascal VOC:

- 7x7 grid
- 2 bounding boxes / cell
- 20 classes



$$7 \times 7 \times (5 \times 2 + 20) = 7 \times 7 \times 30 \text{ tensor} = \mathbf{1470 \text{ outputs}}$$

YOLO - Loss Function

$$\lambda_{\text{coord}} \sum_{i=0}^{S^2} \sum_{j=0}^B \mathbb{1}_{ij}^{\text{obj}} \left[(x_i - \hat{x}_i)^2 + (y_i - \hat{y}_i)^2 \right]$$

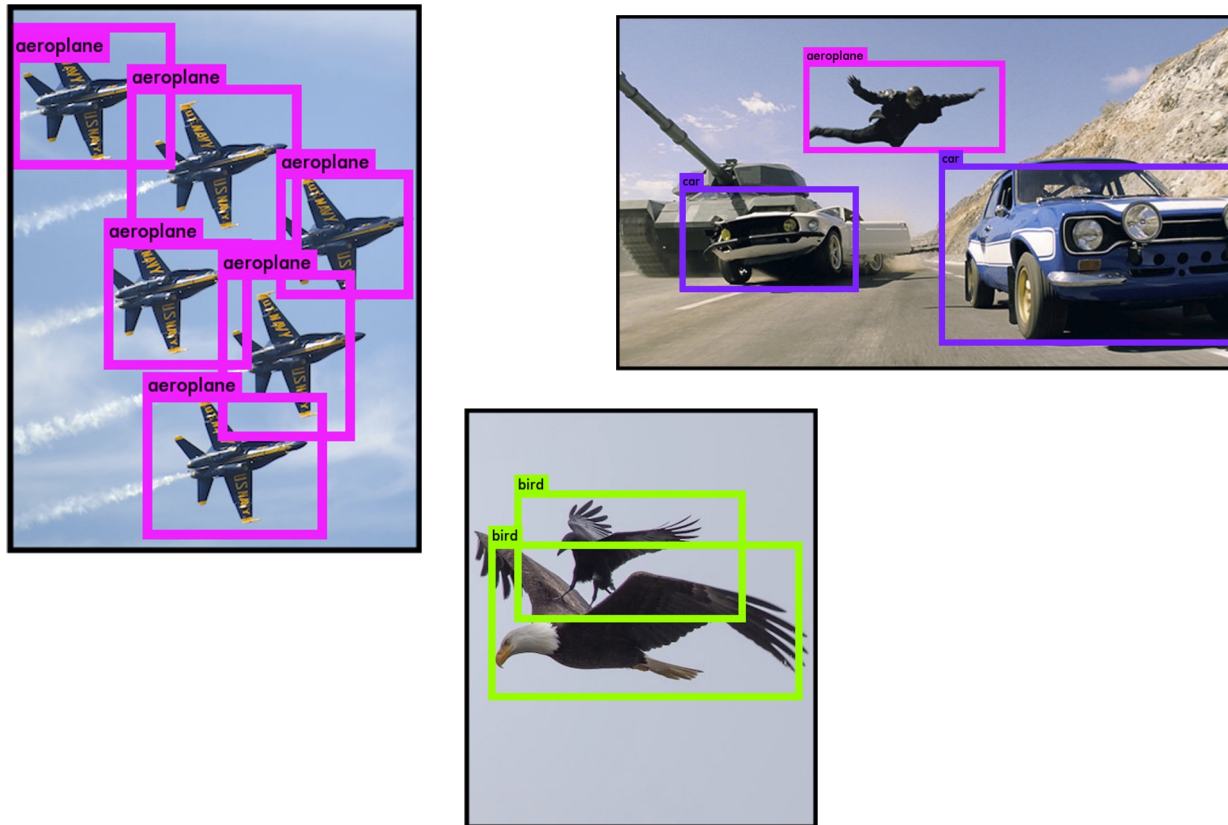
$$+ \lambda_{\text{coord}} \sum_{i=0}^{S^2} \sum_{j=0}^B \mathbb{1}_{ij}^{\text{obj}} \left[\left(\sqrt{w_i} - \sqrt{\hat{w}_i} \right)^2 + \left(\sqrt{h_i} - \sqrt{\hat{h}_i} \right)^2 \right]$$

$$+ \sum_{i=0}^{S^2} \sum_{j=0}^B \mathbb{1}_{ij}^{\text{obj}} (C_i - \hat{C}_i)^2$$

$$+ \lambda_{\text{noobj}} \sum_{i=0}^{S^2} \sum_{j=0}^B \mathbb{1}_{ij}^{\text{noobj}} (C_i - \hat{C}_i)^2$$

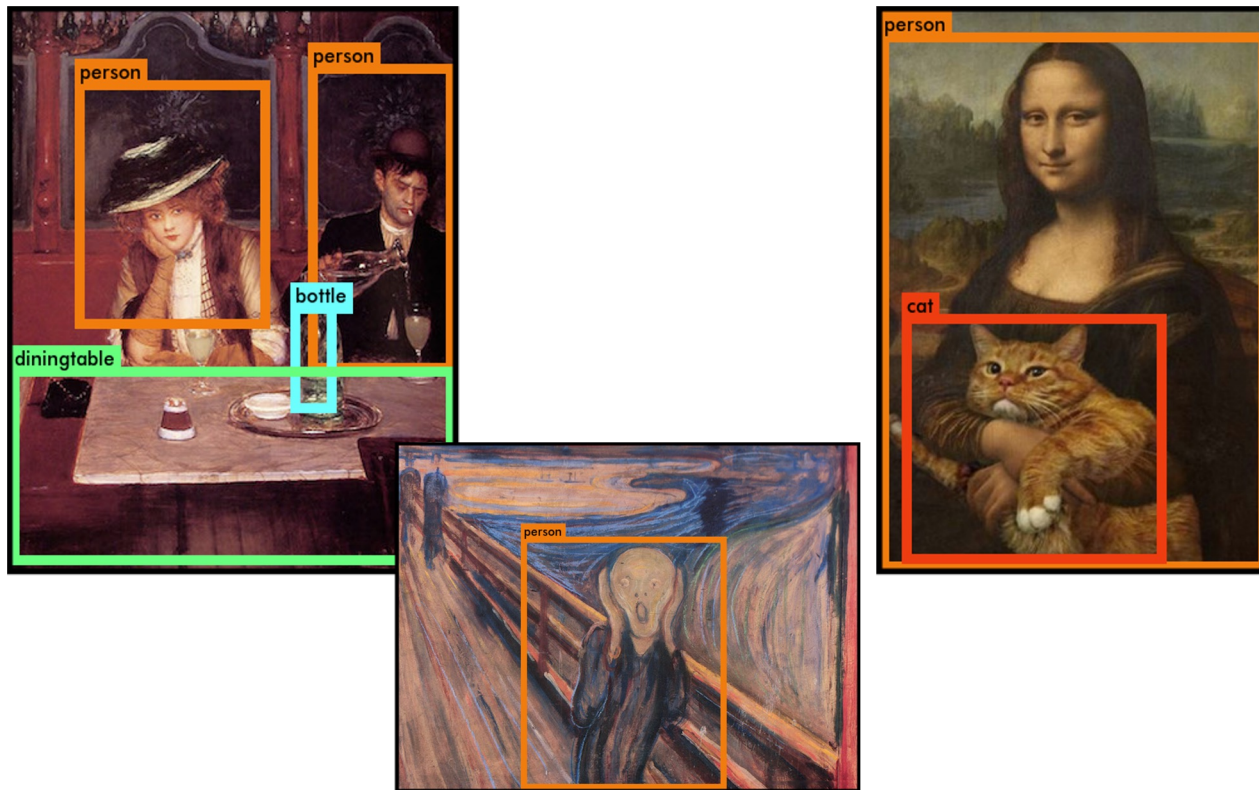
$$+ \sum_{i=0}^{S^2} \mathbb{1}_i^{\text{obj}} \sum_{c \in \text{classes}} (p_i(c) - \hat{p}_i(c))^2$$

YOLO works across many natural images



Redmon et al., "You Only Look Once: Unified, Real-Time Object Detection", CVPR 2016

It also generalizes well to new domains



Redmon et al., "You Only Look Once: Unified, Real-Time Object Detection", CVPR 2016

RETURN IN.....

YOLO9000
Better, *Faster*,
Stronger

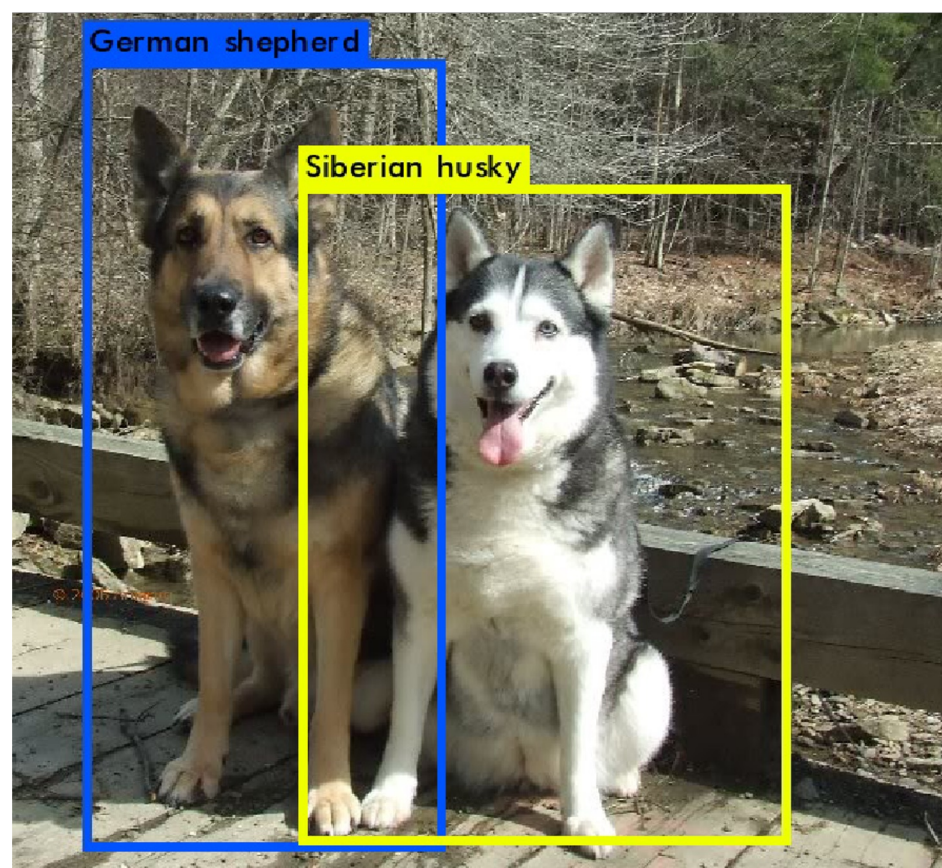
NOW PLAYING IN A DEMO NEAR YOU

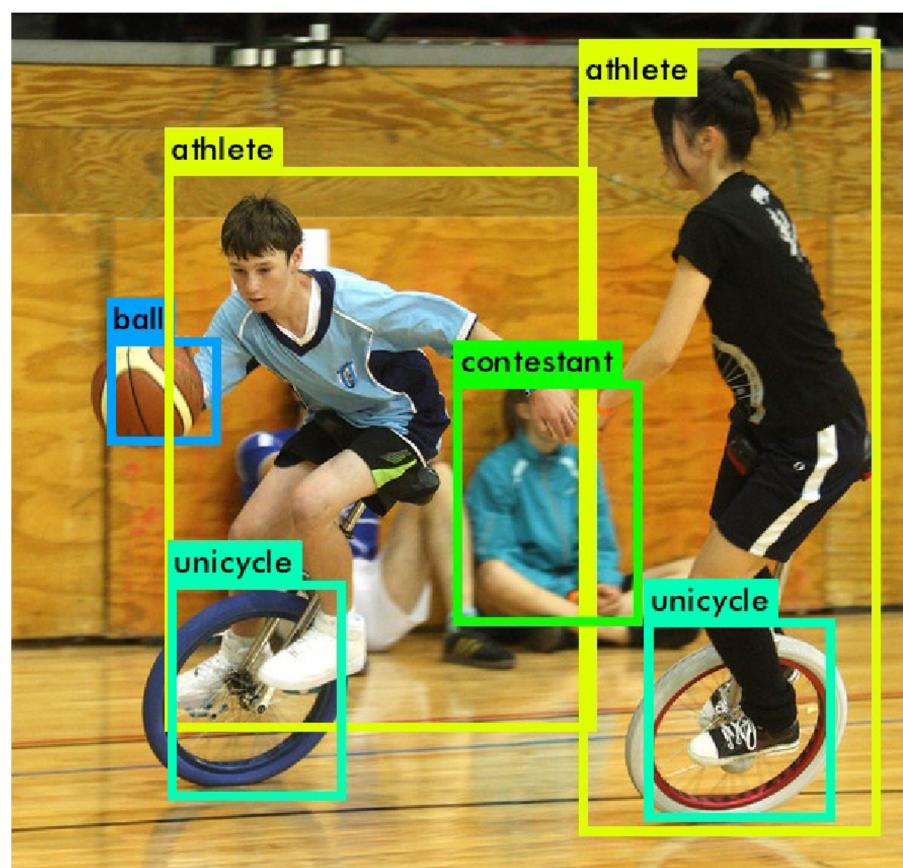
A UNIVERSITY OF WASHINGTON PRODUCTION IN ASSOCIATION WITH XNOR.AI AND THE ALLEN INSTITUTE FOR ARTIFICIAL INTELLIGENCE
MODELS BY DARKNET: OPEN SOURCE NEURAL NETWORKS

@DARKNETFOREVER #YOLO9000

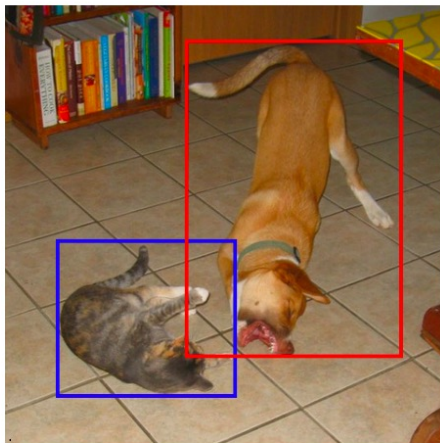
pjreddie.com/yolo



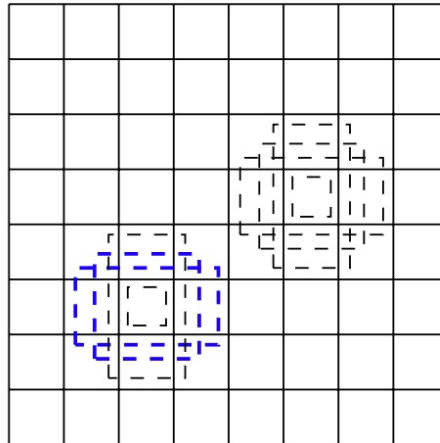




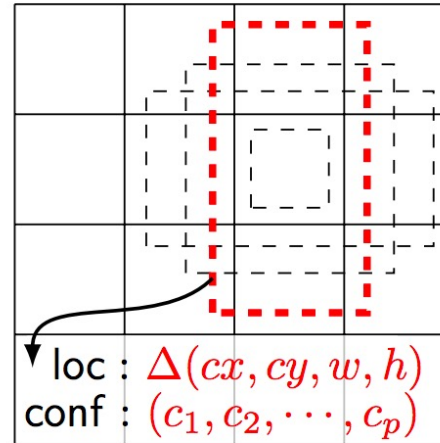
SSD: Single Shot Detector



(a) Image with GT boxes



(b) 8×8 feature map



(c) 4×4 feature map

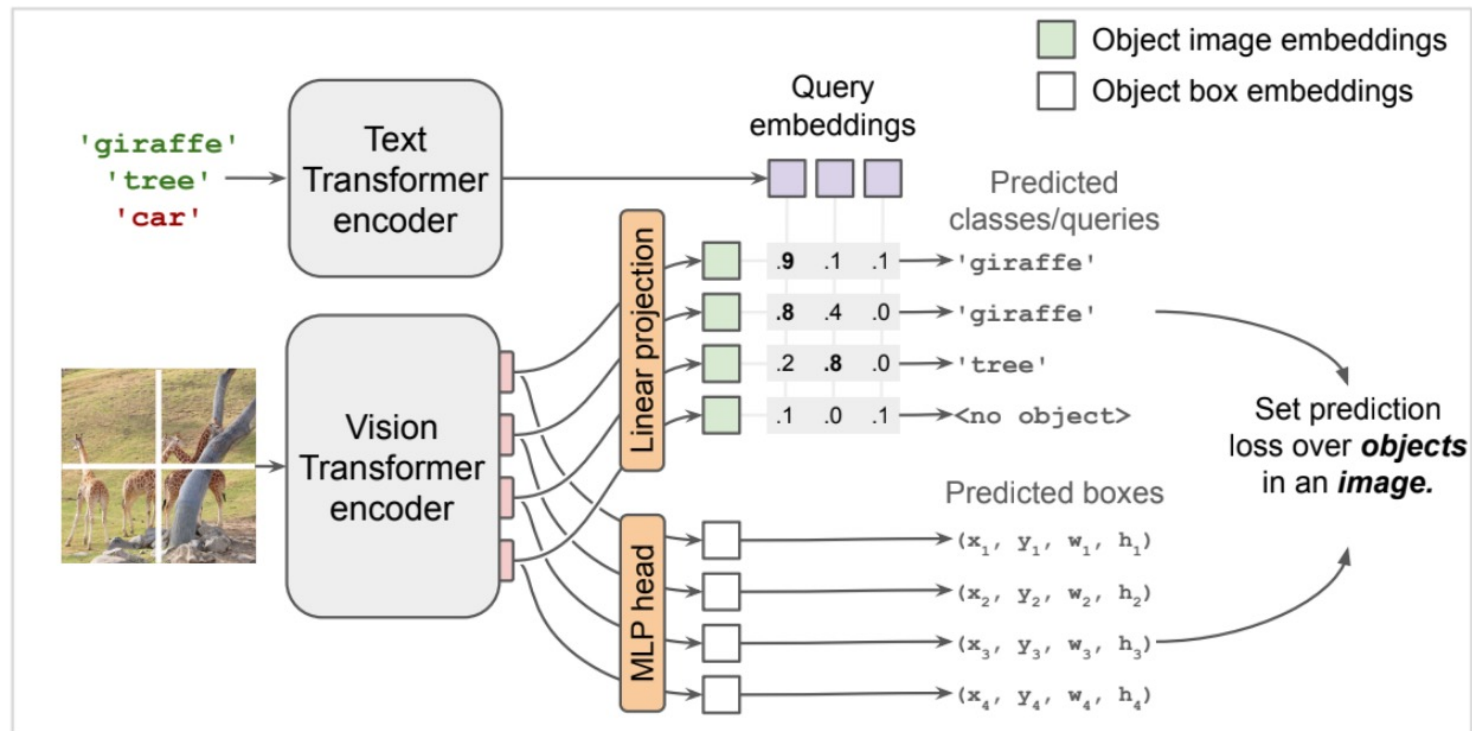
Lab 11a



Idea: Similar to YOLO, but denser grid map, multiscale grid maps. + Data augmentation + Hard negative mining + Other design choices in the network.

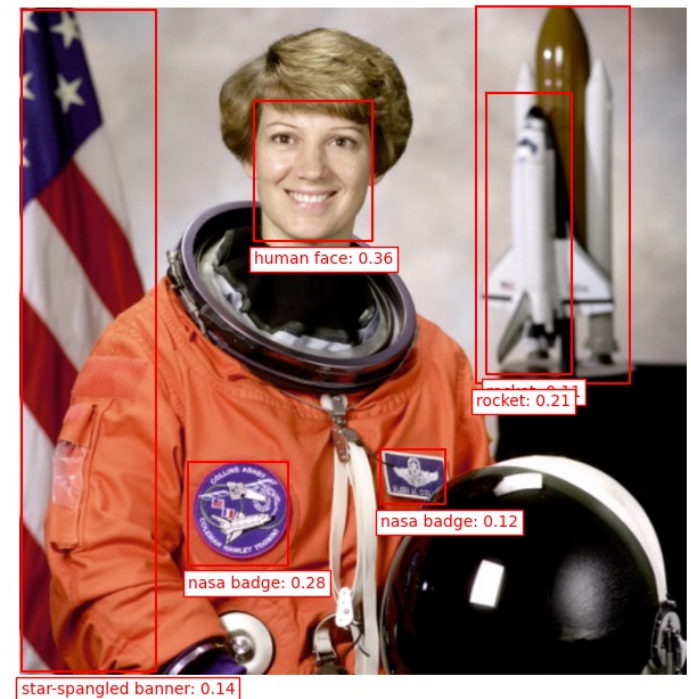
Liu et al. ECCV 2016.

[SOTA] OWL-ViT: open-vocabulary object detector



<https://arxiv.org/pdf/2205.06230>

[SOTA] OWL-ViT: open-vocabulary object detector



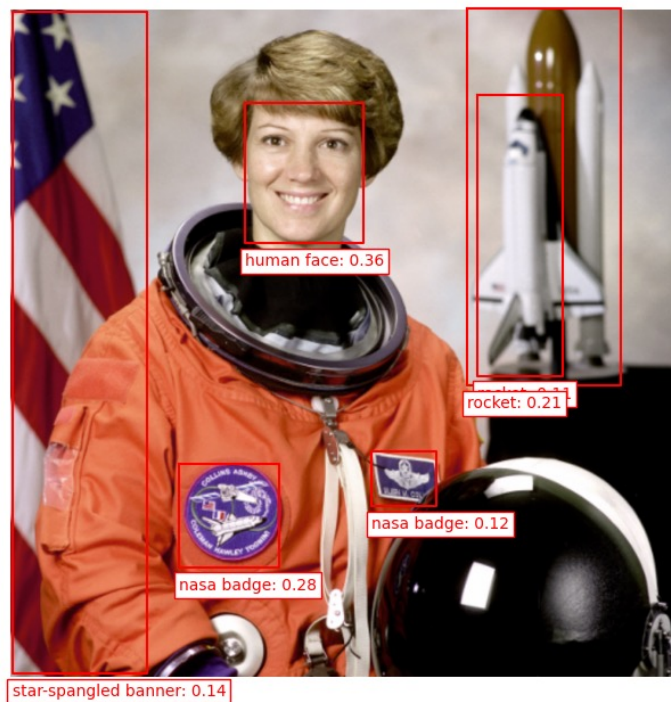
```
text_queries = ["human face", "rocket",  
                "nasa badge", "star-spangled banner"]
```

Lab 11b: Open Vocabulary Object Recognition

Duration: 5 min



[SOTA] OWL-ViT: open-vocabulary object detector

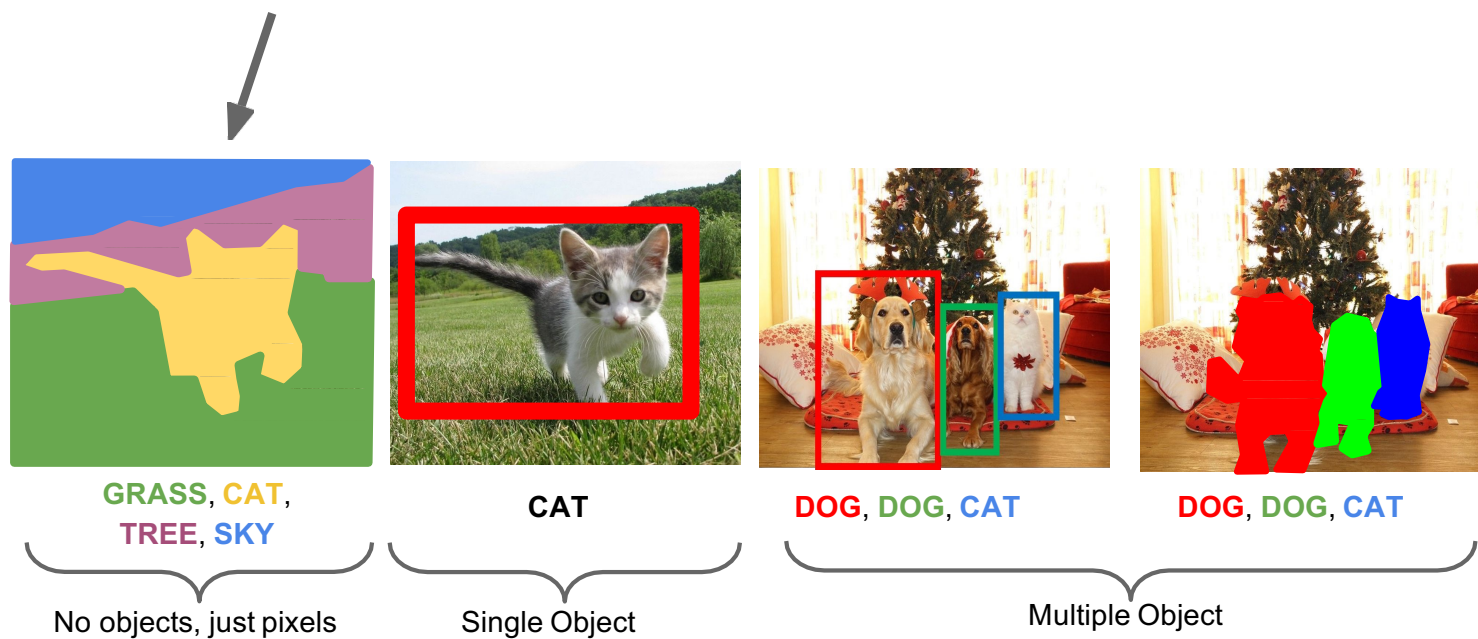


How Bounding Boxes are encoded?

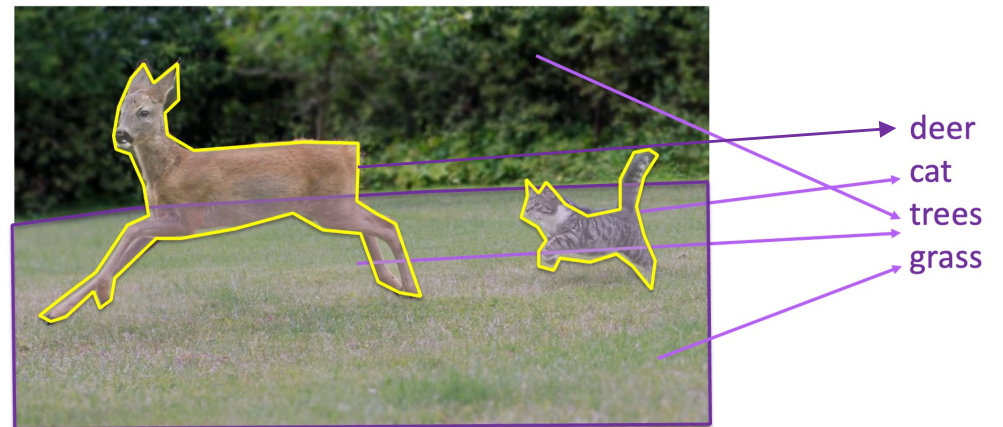
Plan for the next lectures

- Detection approaches
 - Pre-CNNs
 - Detection with whole windows: Pedestrian detection
 - Part-based detection: Deformable Part Models
 - Post-CNNs
 - Detection with region proposals: R-CNN, Fast R-CNN, Faster-R-CNN
 - Detection without region proposals: YOLO, SSD
- Segmentation approaches
 - Semantic segmentation: FCN
 - Instance segmentation: Mask R-CNN

Semantic Segmentation



Semantic Segmentation

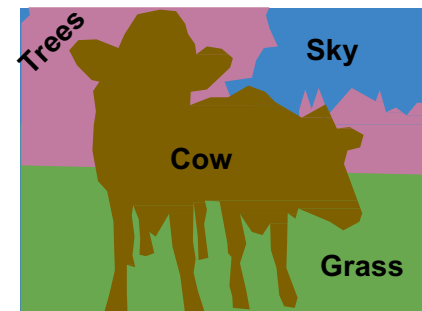
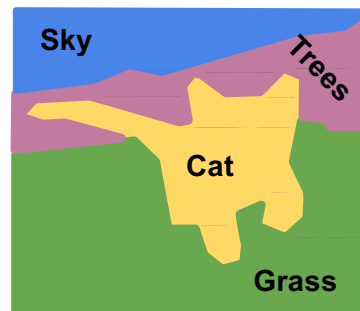


Adapted from Vicente Ordoñez

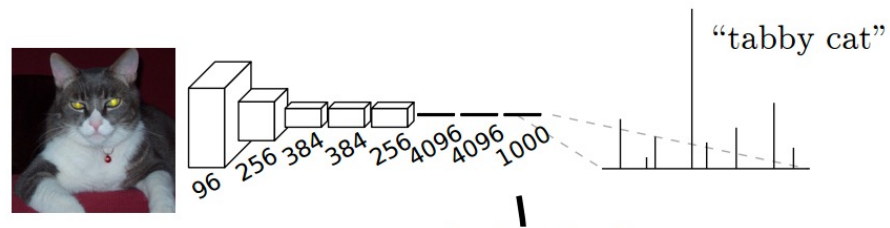
Semantic Segmentation

Label each pixel in the image with a category label

Don't differentiate instances, only care about pixels

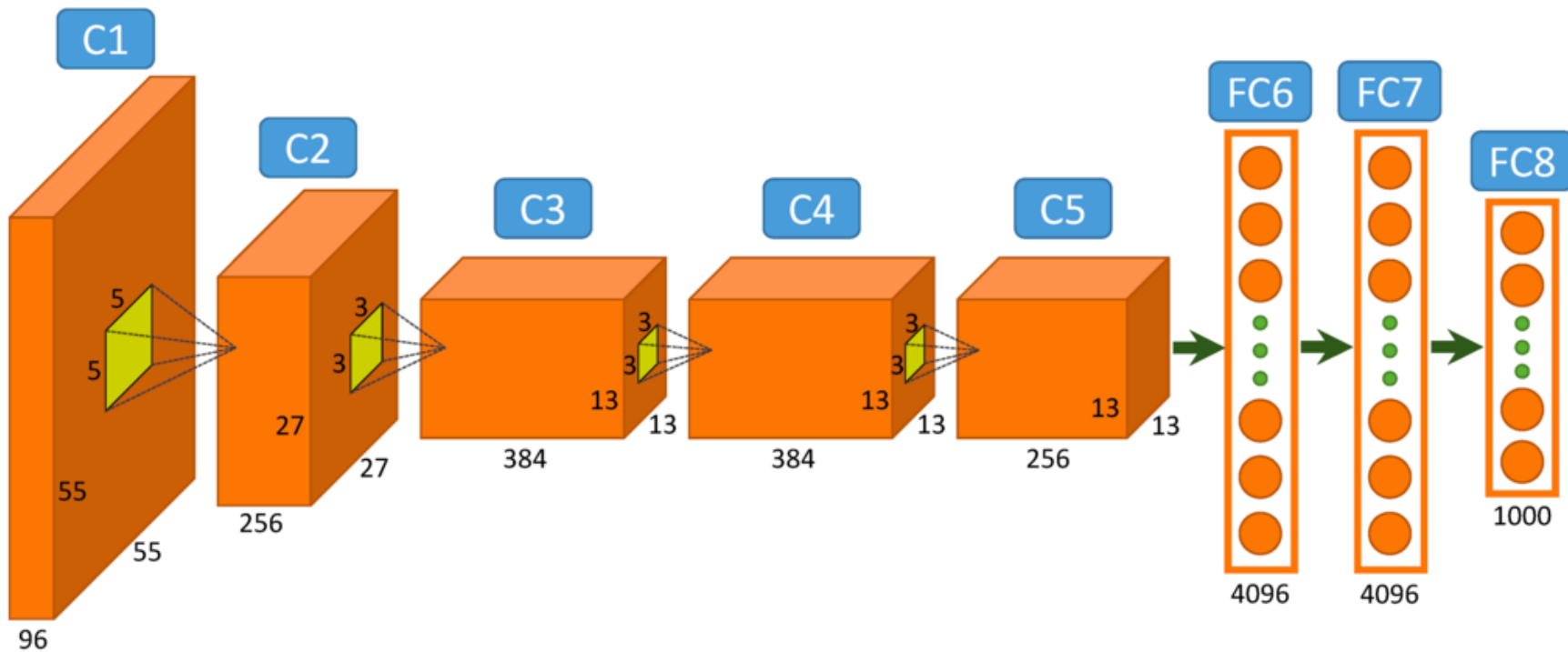


Idea 1: Convolutionalization



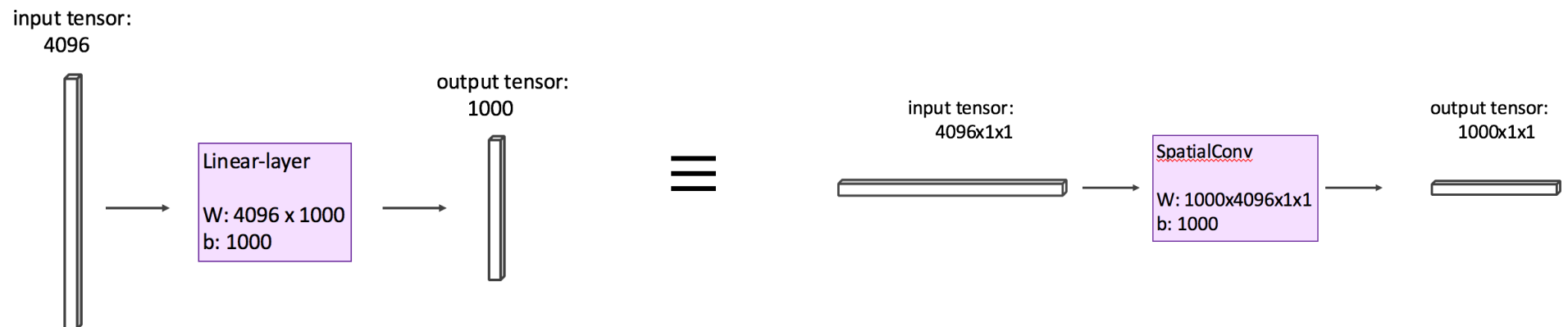
Adapted from Vicente Ordoñez

Idea 1: Convolutionalization



Idea 1: Convolutionalization

`nn.Linear(4096, 1000) == nn.Conv2D(4096, 1000, kernel_size = 1, stride = 1)`



Adapted from Vicente Ordoñez

Idea 2: Fully Convolutional Networks (CVPR 2015)

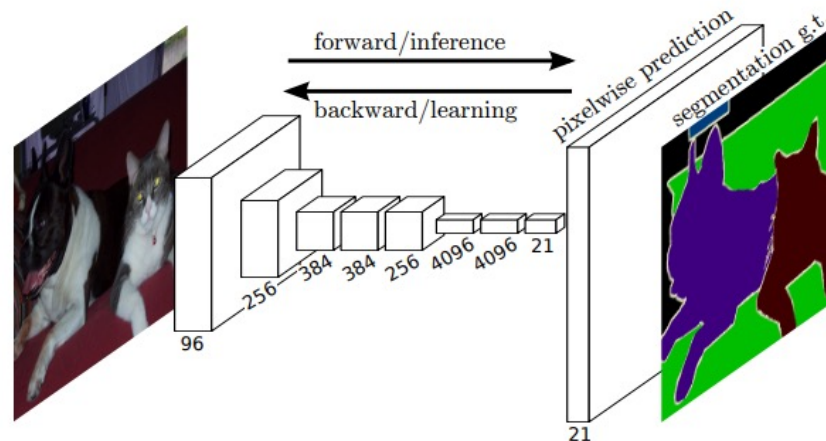
Fully Convolutional Networks for Semantic Segmentation

Jonathan Long*

Evan Shelhamer*
UC Berkeley

Trevor Darrell

{jonlong, shelhamer, trevor}@cs.berkeley.edu

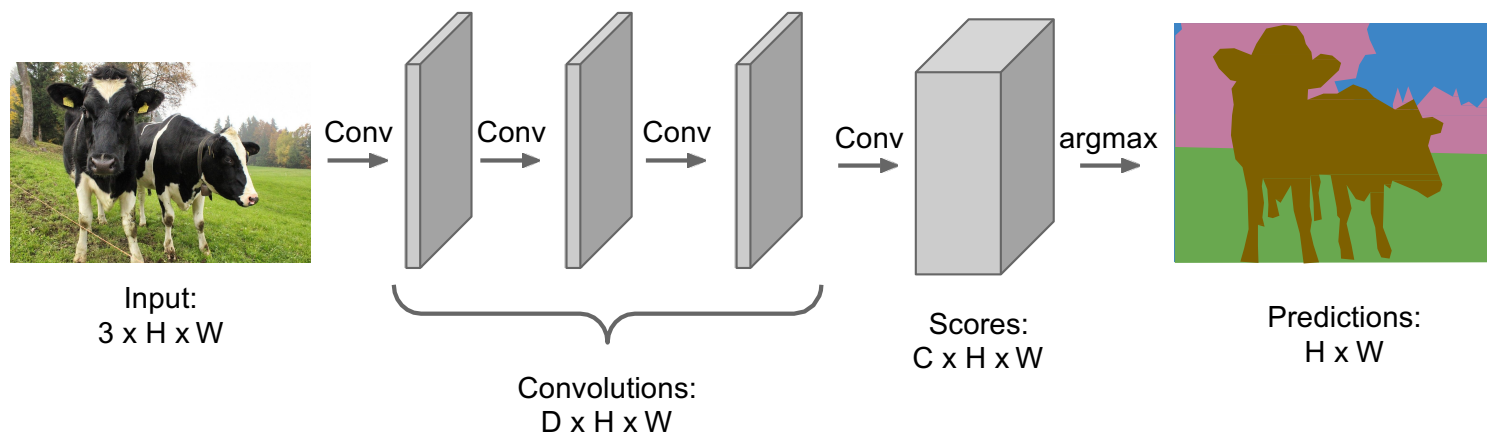


Lab 11a



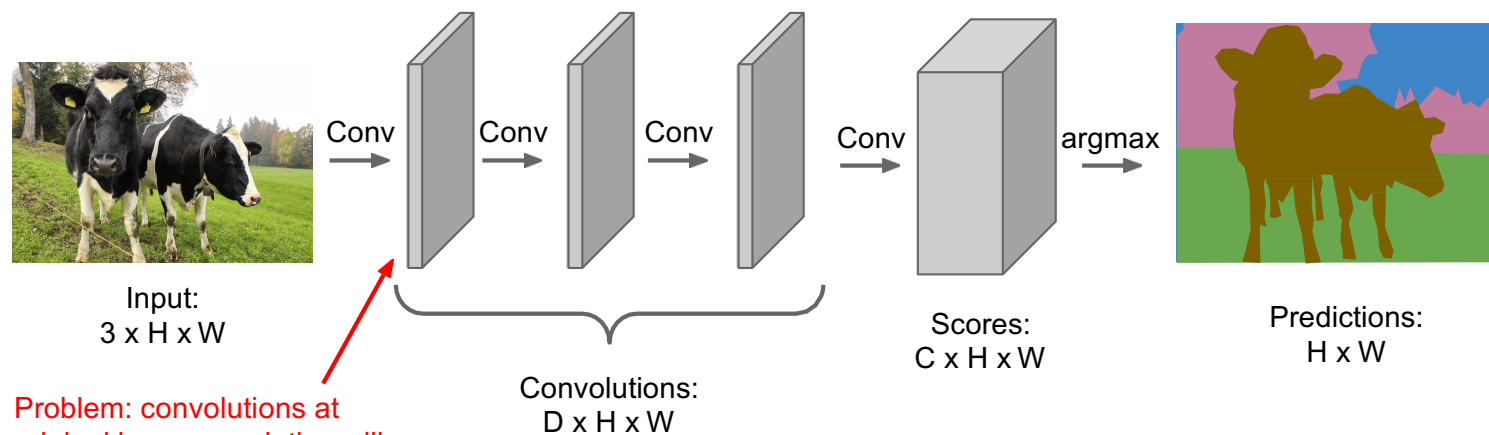
Idea 2: Fully Convolutional

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



Idea 2: Fully Convolutional

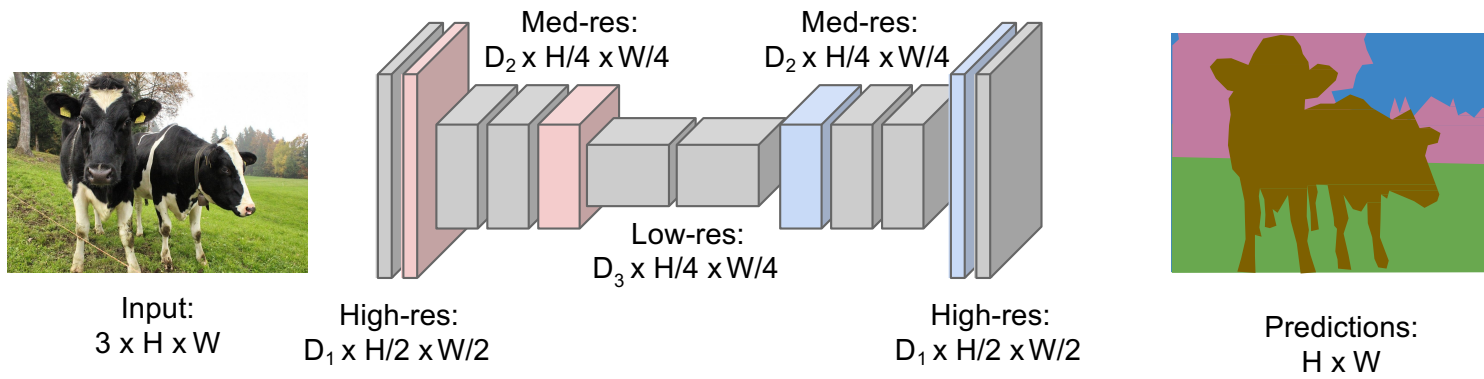
Design a network as a bunch of convolutional layers to make predictions for pixels all at once!



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

Idea 2: Fully Convolutional

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



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

Slide by: Justin Johnson

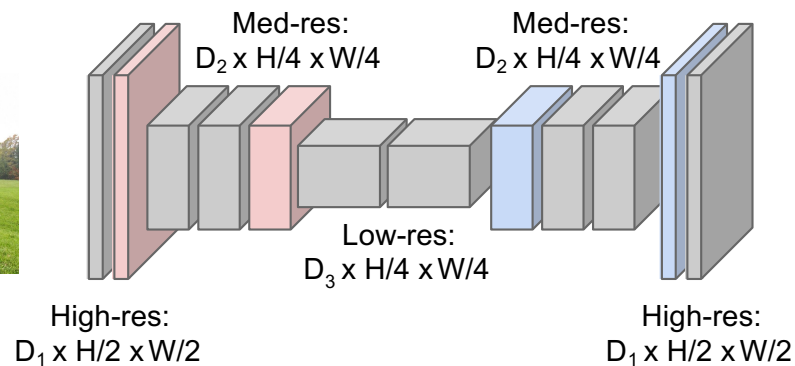
Idea 2: Fully Convolutional

Downsampling:
Pooling, strided
convolution



Input:
 $3 \times H \times W$

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



Upsampling:
???



Predictions:
 $H \times W$

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



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

Input: 2 x 2

Output: 4 x 4

“Bed of Nails”

1	2
3	4

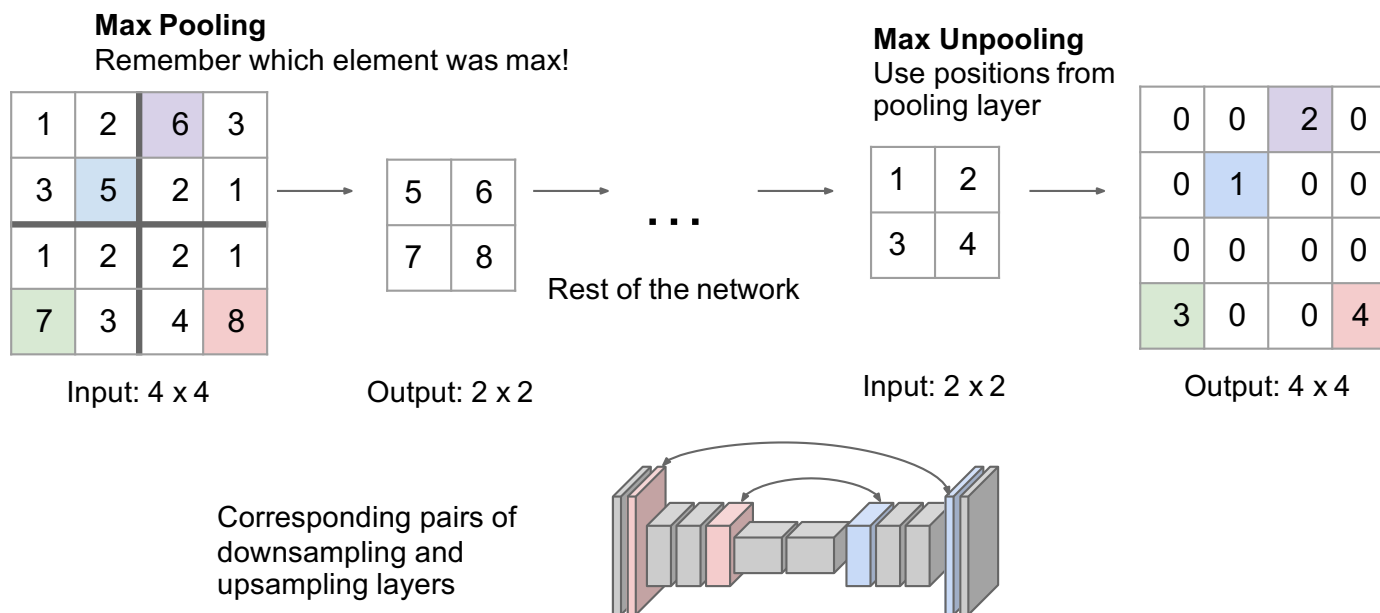


1	0	2	0
0	0	0	0
3	0	4	0
0	0	0	0

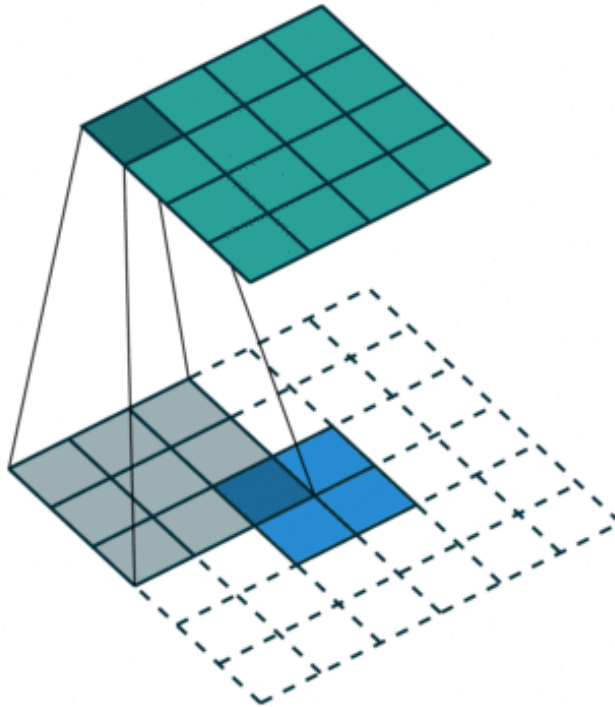
Input: 2 x 2

Output: 4 x 4

In-Network upsampling: “Max Unpooling”

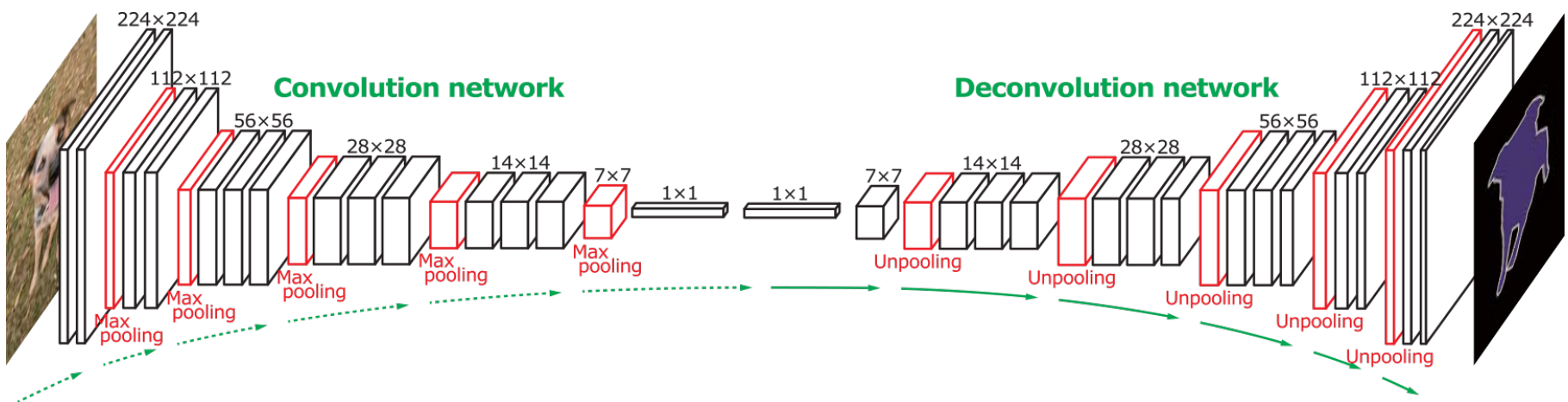


In-Network Up-sampling Convolutions or "Deconvolutions"



https://github.com/vdumoulin/conv_arithmetic

Idea 2: Fully Convolutional Networks



Learning Deconvolution Network for Semantic Segmentation

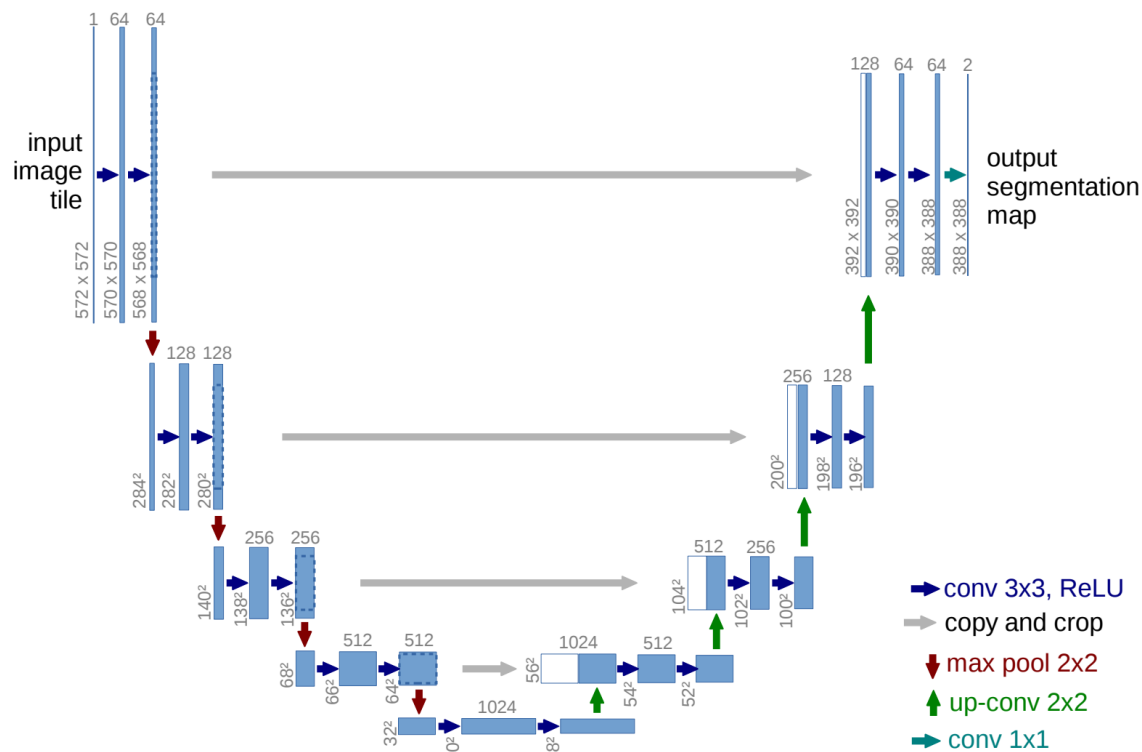
Hyeonwoo Noh Seunghoon Hong Bohyung Han
Department of Computer Science and Engineering, POSTECH, Korea
{hyeonwoonoh-, maga33, bhhan}@postech.ac.kr

<http://cvlab.postech.ac.kr/research/deconvnet/>

U-Net: Convolutional Networks for Biomedical Image Segmentation

Olaf Ronneberger, Philipp Fischer, and Thomas Brox

Medical Signalling Studies,



<https://arxiv.org/abs/1505.04597>

<https://github.com/milesial/Pytorch-UNet>

<https://github.com/usuyama/pytorch-unet>

UNet in Pytorch

```

from .unet_parts import *

class UNet(nn.Module):
    def __init__(self, n_channels, n_classes, bilinear=False):
        super(UNet, self).__init__()
        self.n_channels = n_channels
        self.n_classes = n_classes
        self.bilinear = bilinear

        self.inc = (DoubleConv(n_channels, 64))
        self.down1 = (Down(64, 128))
        self.down2 = (Down(128, 256))
        self.down3 = (Down(256, 512))
        factor = 2 if bilinear else 1
        self.down4 = (Down(512, 1024 // factor))
        self.up1 = (Up(1024, 512 // factor, bilinear))
        self.up2 = (Up(512, 256 // factor, bilinear))
        self.up3 = (Up(256, 128 // factor, bilinear))
        self.up4 = (Up(128, 64, bilinear))
        self.outc = (OutConv(64, n_classes))

    def forward(self, x):
        x1 = self.inc(x)
        x2 = self.down1(x1)
        x3 = self.down2(x2)
        x4 = self.down3(x3)
        x5 = self.down4(x4)
        x = self.up1(x5, x4)
        x = self.up2(x, x3)
        x = self.up3(x, x2)
        x = self.up4(x, x1)
        logits = self.outc(x)
        return logits

```

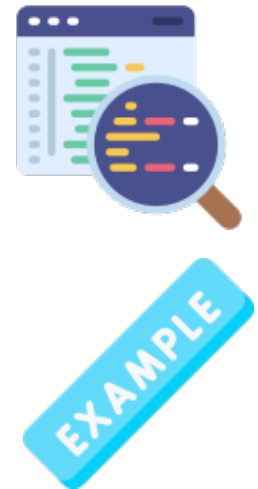
https://github.com/milesial/Pytorch-UNet/blob/master/unet/unet_model.py

Image Segmentation Learning

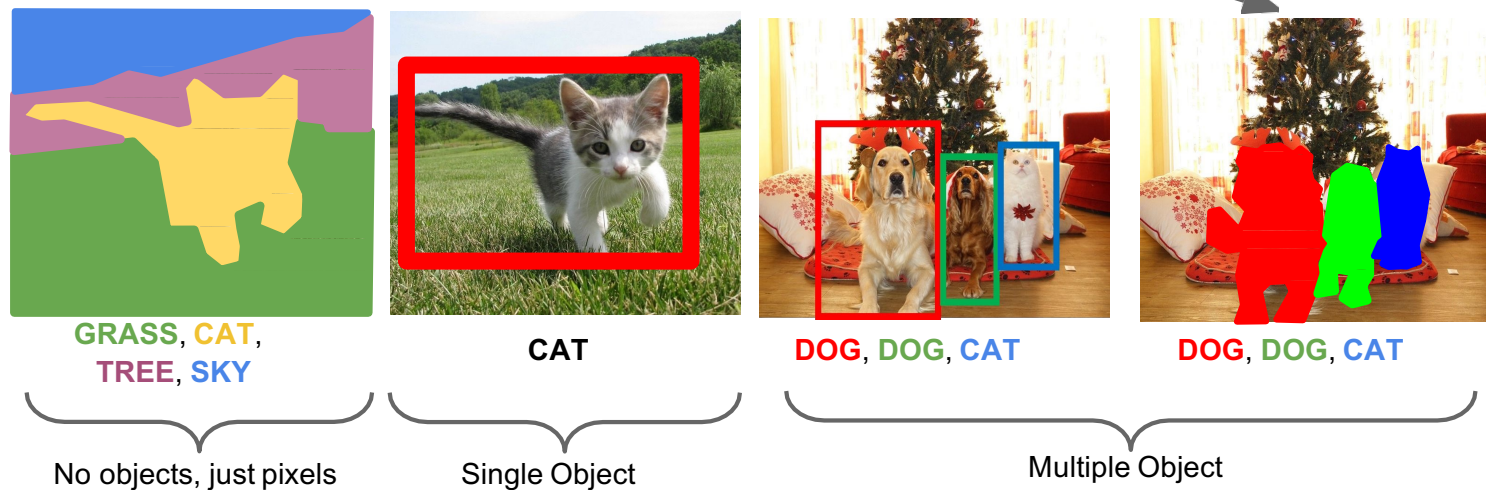
What loss can we use?

Additional Resources

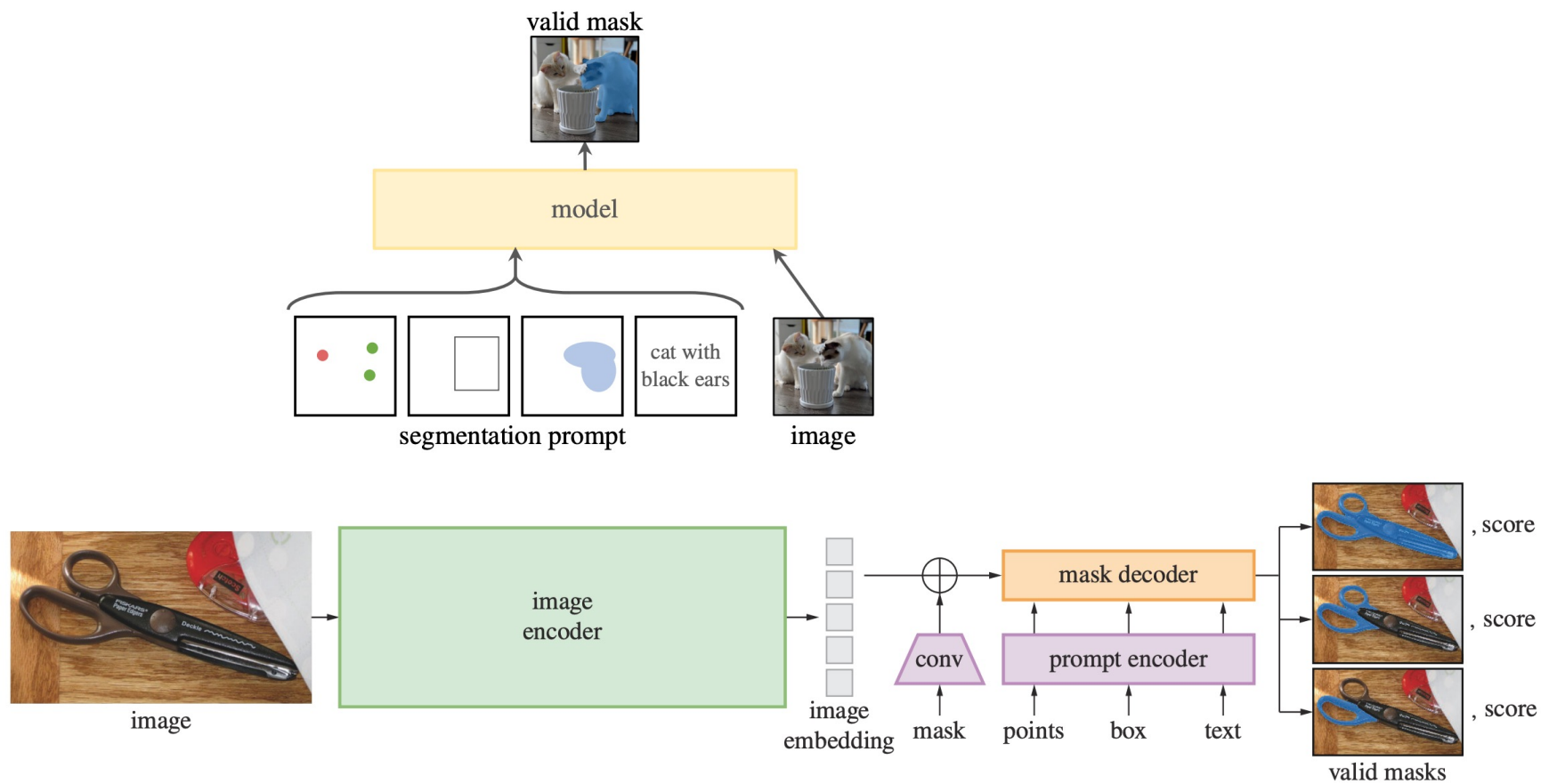
- Image Segmentation with Synthetic Masks [[link](#)]
- Segmentation Models [[link](#)]
- Oxford Pets Image Segmentation [[link](#)]
- Cars Image Segmentation [[link](#)]
- Road Analysis Image Segmentation [[link](#)]



Instance Segmentation



[SOTA] Segment Anything



<https://arxiv.org/abs/2304.02643>

Lab 11c: Segment Anything

Duration: 10 min

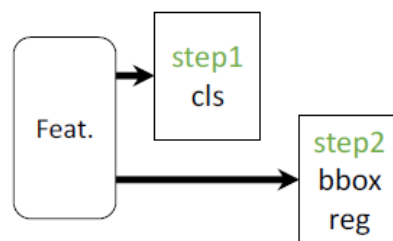


Mask R-CNN

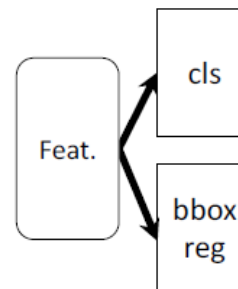
He et al, "Mask R-CNN", ICCV 2017

What is Mask R-CNN: Parallel Heads

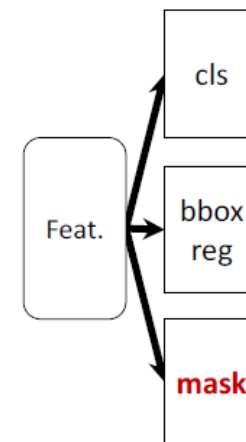
- Easy, fast to implement and use



(slow) R-CNN



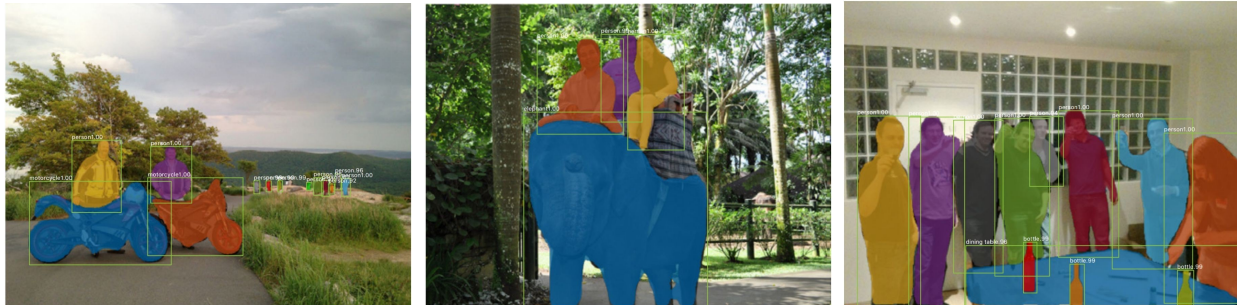
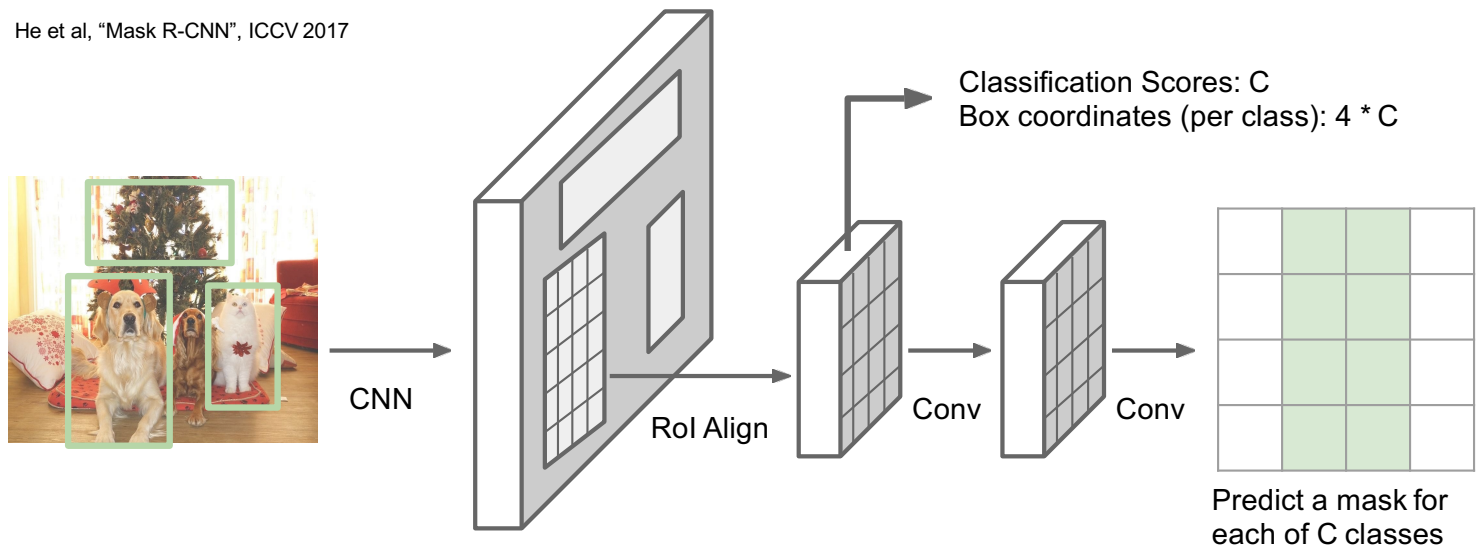
Fast/er R-CNN



Mask R-CNN

Mask R-CNN

He et al, "Mask R-CNN", ICCV 2017



Adapted from Justin Johnson

Lab 11d: Fine-Tuning Mask R-CNN for Object Recognition and Image Segmentation

Duration: 10 min



To join, go to: ahaslides.com/GVB6P 



Please, from Lab 11d and Activity 11, submit your generated image result.

^ Get Feedback

✓ Slide 1 selected for PowerPoint



Group



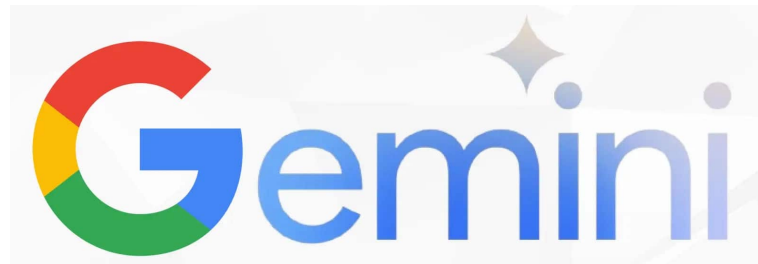
0



0/100



[SOTA] Object Recognition and Segmentation with LLMs



<https://github.com/roboflow/notebooks>

Lab 11e: Zero-Shot Object Detection and Segmentation with Google Gemini 2.5

Duration: 10 min



Plan for the next few lectures

- Detection approaches
 - Pre-CNNs
 - Detection with whole windows: Pedestrian detection
 - Part-based detection: Deformable Part Models
 - Post-CNNs
 - Detection with region proposals: R-CNN, Fast R-CNN, Faster-R-CNN
 - Detection without region proposals: YOLO, SSD
- Segmentation approaches
 - Semantic segmentation
 - Fully-Convolutional Networks (FCN)
 - Instance segmentation
 - Mask R-CNN
 - Segment Anything
- Learning from noisy web image-text data
 - Contrastive Language-Image Pretraining (CLIP)
 - Prompting
 - Open-vocabulary object detection

Learning from noisy web data

- Massive datasets of image-text pairs from the web
 - E.g. alt text, Flickr, Reddit, Wikipedia, etc
- Images and their co-occurring text assumed related (text provides a reasonable description of image?)
- Train text and image feature extractors using the objective that matched (co-occurring) image-text should be more similar than mismatched ones
- Great performance at a low annotation cost (data already existed)

Contrastive Language-Image Pretraining (CLIP)

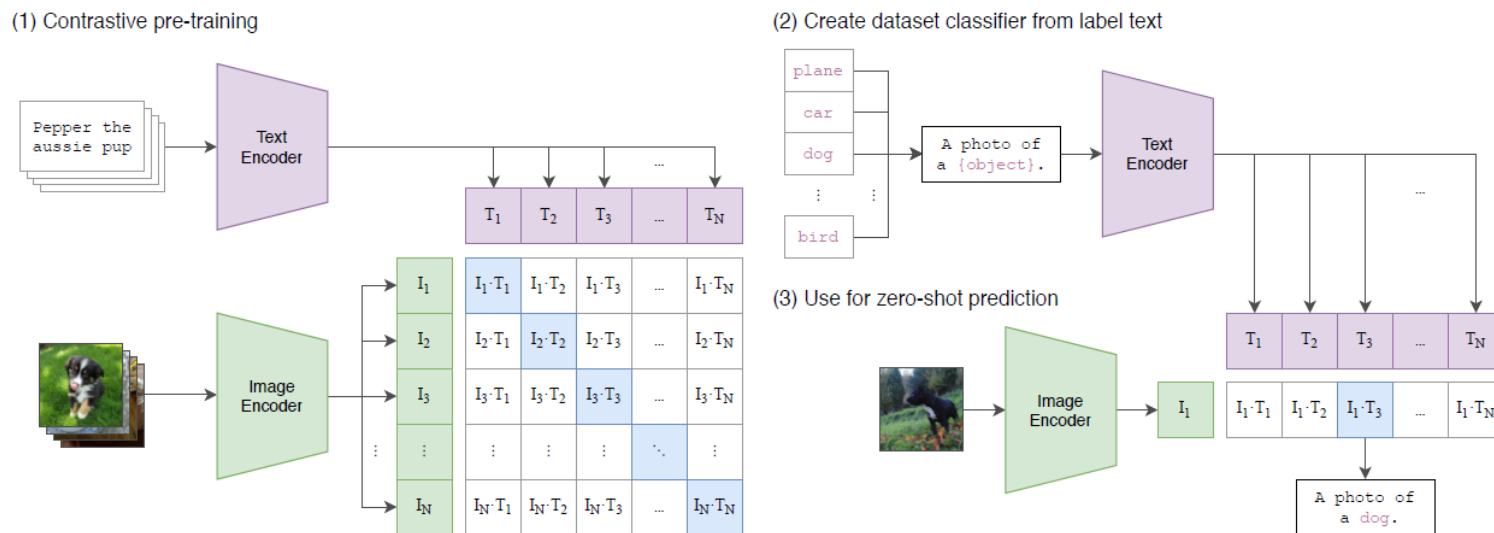
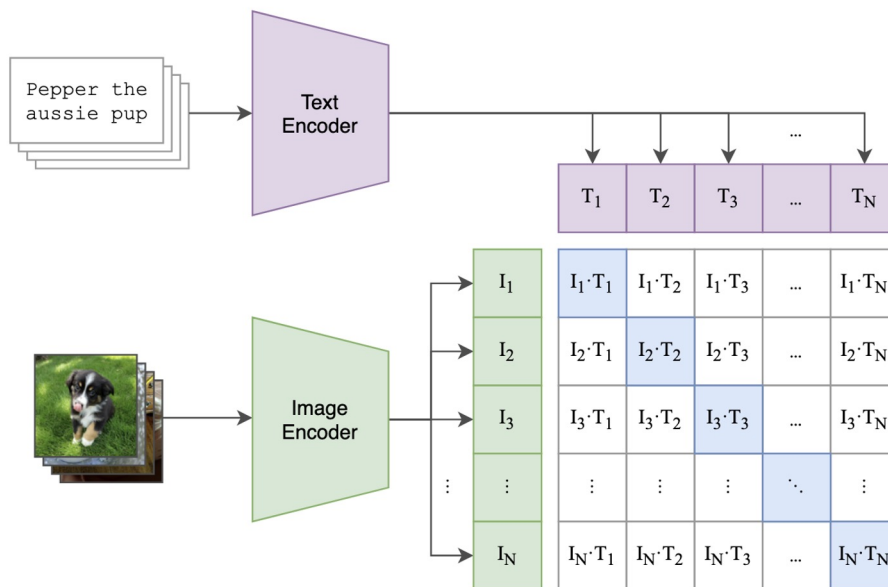


Figure 1. Summary of our approach. While standard image models jointly train an image feature extractor and a linear classifier to predict some label, CLIP jointly trains an image encoder and a text encoder to predict the correct pairings of a batch of (image, text) training examples. At test time the learned text encoder synthesizes a zero-shot linear classifier by embedding the names or descriptions of the target dataset's classes.

Radford et al. "Learning Transferable Visual Models From Natural Language Supervision." ICML 2021.

Contrastive Language-Image Pretraining (CLIP)



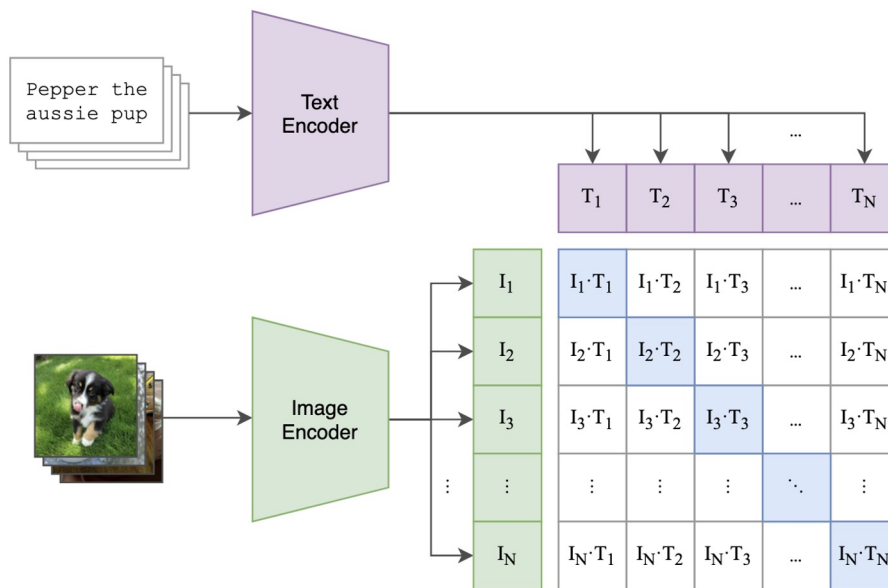
$$L = \sum_k \ell(I_k T_k)$$

$$\ell(I_k T_k) = -\log \left(\frac{\exp(\text{sim}(I_k, T_k))}{\sum_{t=1}^{2N} 1[k \neq i] \exp(\text{sim}(I_k, T_t))} \right)$$

Radford et al. "Learning Transferable Visual Models From Natural Language Supervision." ICML 2021.

Adapted from Vicente Ordoñez

Contrastive Language-Image Pretraining (CLIP)



$$L = \sum_k \ell_1(I_k T_k) + \ell_2(I_k T_k)$$

$$\ell_1(I_k T_k) = -\log \left(\frac{\exp(\text{sim}(I_k, T_k))}{\sum_{t=1}^{2N} 1[k \neq i] \exp(\text{sim}(I_k, T_t))} \right)$$

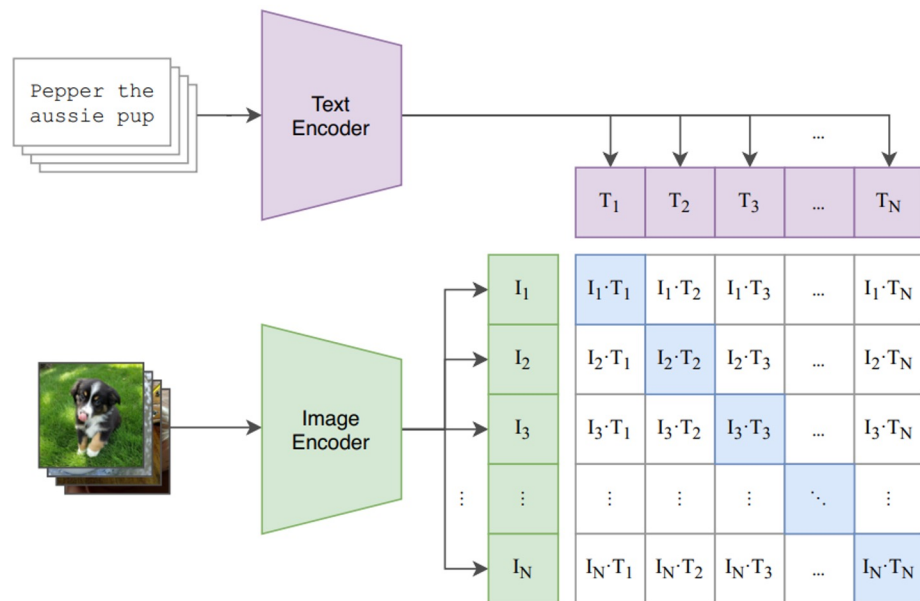
$$\ell_2(I_k T_k) = -\log \left(\frac{\exp(\text{sim}(I_k, T_k))}{\sum_{t=1}^{2N} 1[k \neq i] \exp(\text{sim}(I_t, T_k))} \right)$$

Radford et al. "Learning Transferable Visual Models From Natural Language Supervision." ICML 2021.

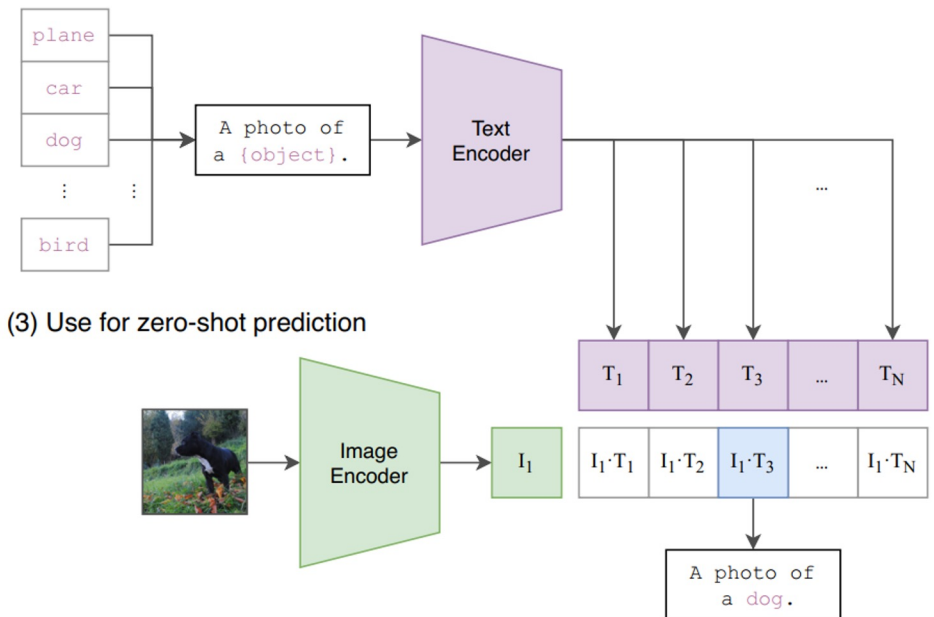
Adapted from Vicente Ordoñez

Zero-shot Image Classification with CLIP

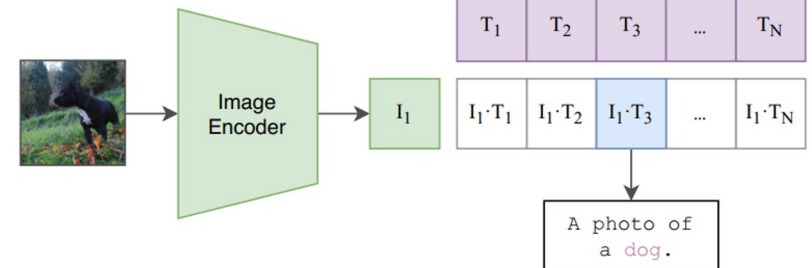
(1) Contrastive pre-training



(2) Create dataset classifier from label text



(3) Use for zero-shot prediction



Radford et al. "Learning Transferable Visual Models From Natural Language Supervision." ICML 2021.

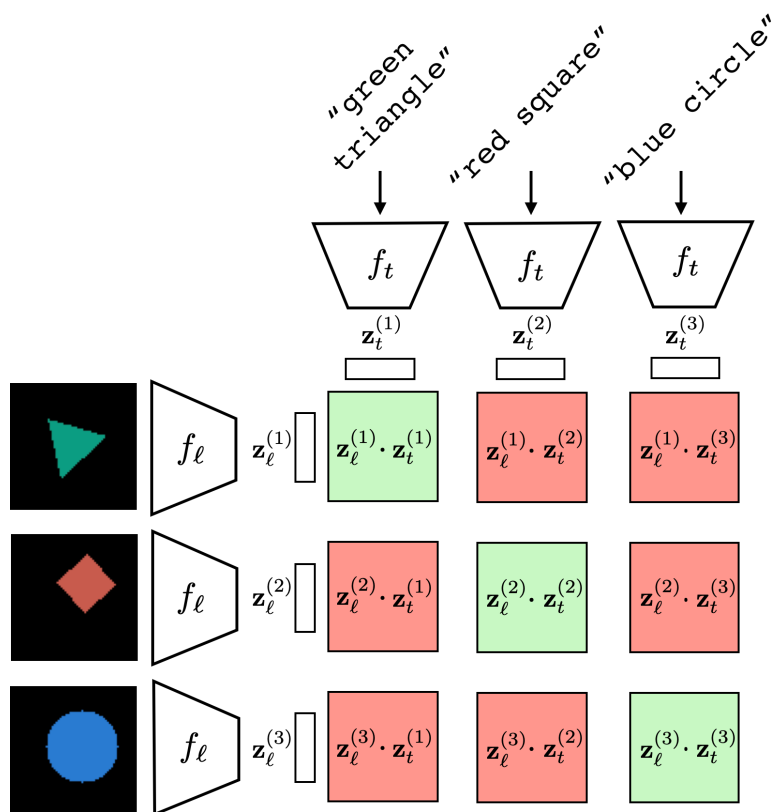
Adapted from Vicente Ordoñez

Zero-shot Image Classification with CLIP

- Image classification: given an image, predict its class name
- Image captioning: given an image, predict its caption
- Contrastive learning: align image and text embeddings that describe the same thing

[Radford*, Kim* et al., ICML 2021]

Zero-shot Image Classification with CLIP



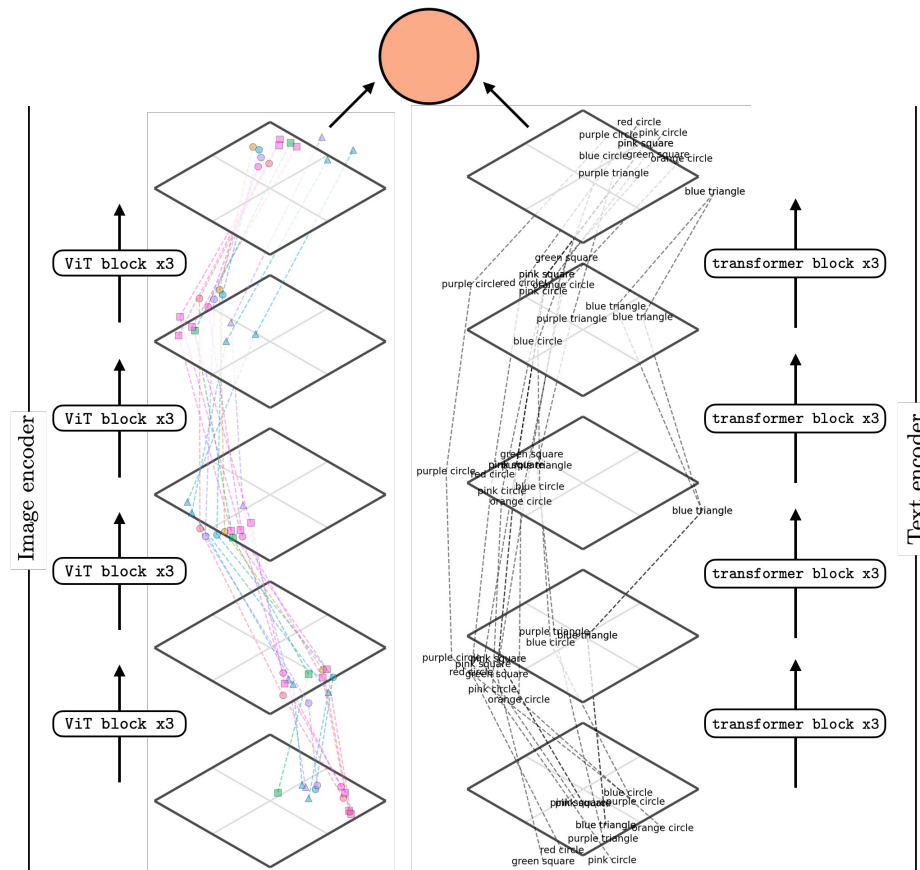
- Positive pairs: image and its caption.
- Negative pairs: image and a different image's caption.
- Learn a representation in which positives are pulled together, negatives are pushed apart.

We learn this representation using Contrastive Learning

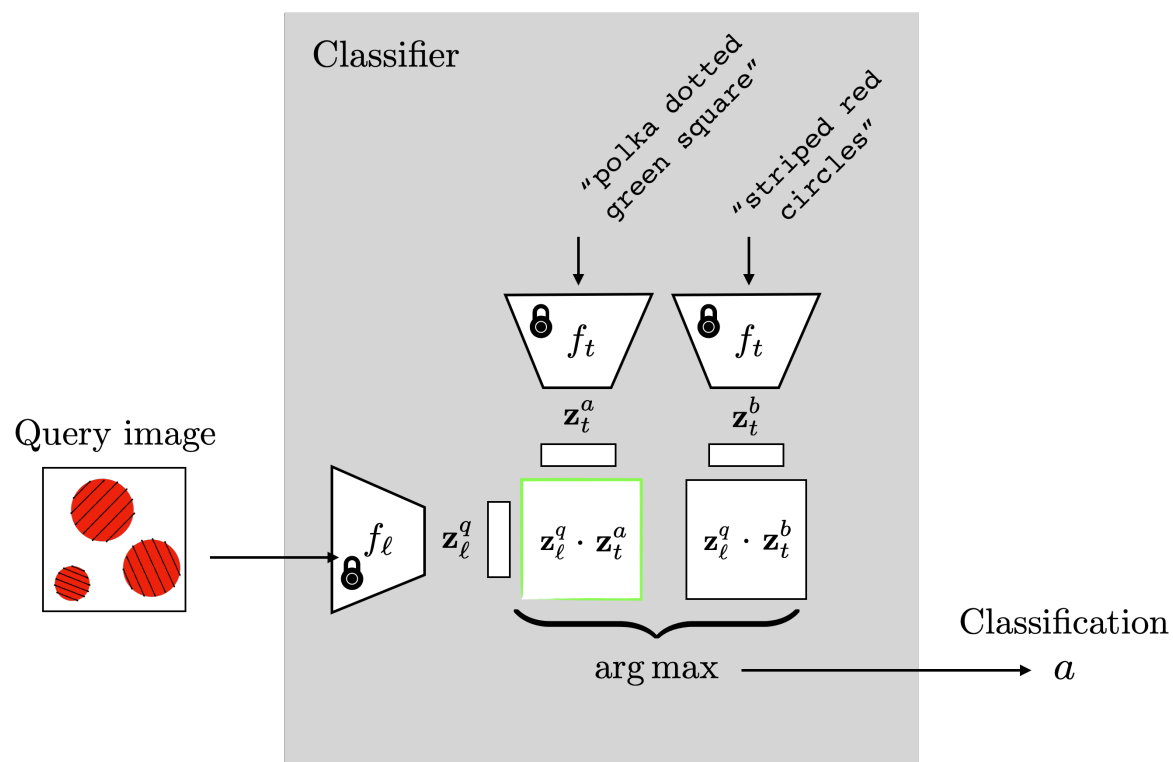
[Radford*, Kim* et al., ICML 2021]

Zero-shot Image Classification with CLIP

Joint embedding space



Zero-shot Image Classification with CLIP



Variants of CLIP

- CLIP: <https://github.com/openai/CLIP>
- OpenCLIP: https://github.com/mlfoundations/open_clip
- MetaCLIP: <https://github.com/facebookresearch/MetaCLIP>
- CLIPA: <https://github.com/UCSC-VLAA/CLIPA>
- SigLIP: <https://github.com/merveenoyan/siglip>
- DFN-5b: <https://huggingface.co/apple/DFN5B-CLIP-ViT-H-14-378>

Using CLIP for Object Recognition

- Compute dot product of image and prompt for each class, e.g. “A photo of dog”
- Return class with highest dot product for each image
- Prompt can be optimized manually or through training
- Can extend idea for object detection

Using CLIP for Object Recognition

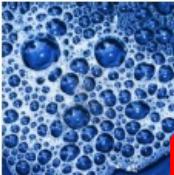
	Prompt	Accuracy
	a [CLASS].	82.68
	a photo of [CLASS].	80.81
	a photo of a [CLASS].	86.29
	$[V]_1 [V]_2 \dots [V]_M$ [CLASS].	91.83
(a)		
	Prompt	Accuracy
	a photo of a [CLASS].	60.86
	a flower photo of a [CLASS].	65.81
	a photo of a [CLASS], a type of flower.	66.14
	$[V]_1 [V]_2 \dots [V]_M$ [CLASS].	94.51
(b)		
	Prompt	Accuracy
	a photo of a [CLASS].	39.83
	a photo of a [CLASS] texture.	40.25
	[CLASS] texture.	42.32
	$[V]_1 [V]_2 \dots [V]_M$ [CLASS].	63.58
(c)		
	Prompt	Accuracy
	a photo of a [CLASS].	24.17
	a satellite photo of [CLASS].	37.46
	a centered satellite photo of [CLASS].	37.56
	$[V]_1 [V]_2 \dots [V]_M$ [CLASS].	83.53
(d)		

Fig. 1 Prompt engineering vs Context Optimization (CoOp). The former needs to use a held-out validation set for words tuning, which is inefficient; the latter automates the process and requires only a few labeled images for learning.

Using CLIP for Object Recognition

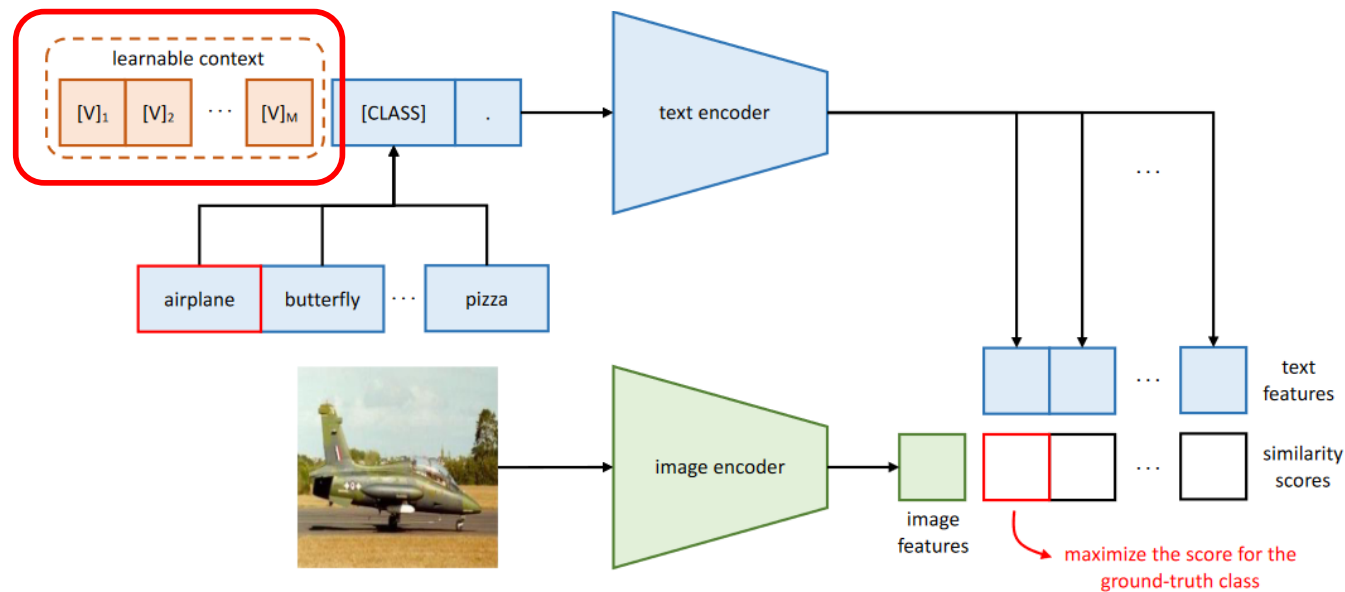


Fig. 2 Overview of Context Optimization (CoOp). The main idea is to model a prompt's context using a set of learnable vectors, which can be optimized through minimizing the classification loss. Two designs are proposed: one is unified context, which shares the same context vectors with all classes; and the other is class-specific context, which learns for each class a specific set of context vectors.

Using CLIP for Object Recognition [Open Vocab]

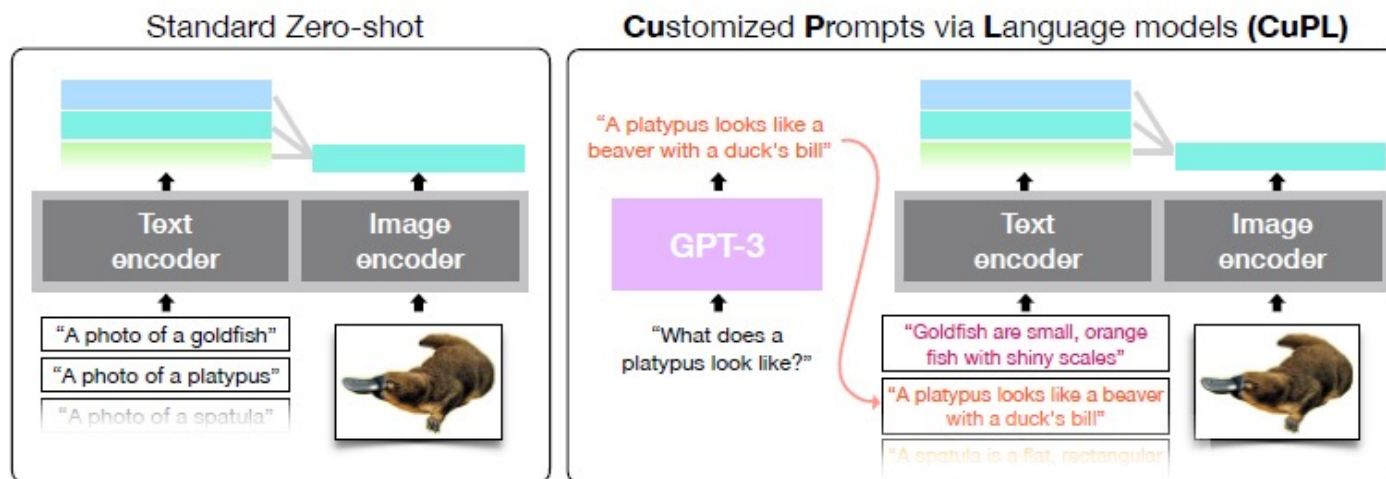


Figure 1: **Schematic of the method.** (Left) The standard method of a zero-shot open vocabulary image classification model (e.g., CLIP (Radford et al., 2021)). (Right) Our method of CuPL. First, an LLM generates descriptive captions for given class categories. Next, an open vocabulary model uses these captions as prompts for performing classification.

Using CLIP for Object Recognition

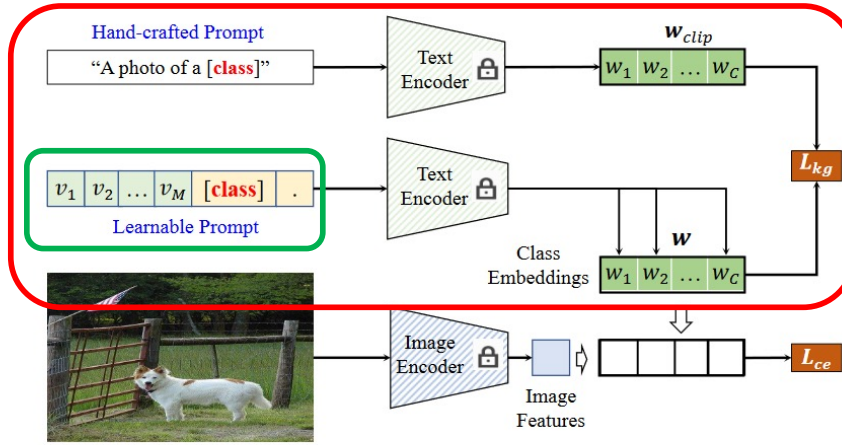


Figure 2. The framework of the Knowledge-guided Context Optimization for prompt tuning. $\mathcal{L}_{ce}$ is the standard cross-entropy loss, and $\mathcal{L}_{kg}$ is the proposed Knowledge-guided Context Optimization constraint to minimize the discrepancy between the special knowledge (learnable textual embeddings) and the general knowledge (the textual embeddings generated by the hand-crafted prompt).

degradation. Therefore, we can minimize the distance between w_i and w_i^{clip} for boosting the generability of the unseen classes,

$$\mathcal{L}_{kg} = \frac{1}{N_c} \sum_{i=1}^{N_c} \|w_i - w_i^{clip}\|_2^2, \quad (3)$$

where $\|\cdot\|$ is the euclidean distance, N_c is the number of seen classes. Meanwhile, the standard contrastive loss is:

$$\mathcal{L}_{ce} = - \sum_{x \in X} \log \frac{\exp(d(x, w_y)/\tau)}{\sum_{i=1}^{N_c} \exp(d(x, w_i)/\tau)}, \quad (4)$$

where y is the corresponding label of the image embedding.

By combining the standard cross-entropy loss $\mathcal{L}_{ce}$, the final objective is:

$$\mathcal{L} = \mathcal{L}_{ce} + \lambda \mathcal{L}_{kg}, \quad (5)$$

where λ is used to balance the effect of $\mathcal{L}_{kg}$.

Using CLIP for Object Recognition [Open Vocab]

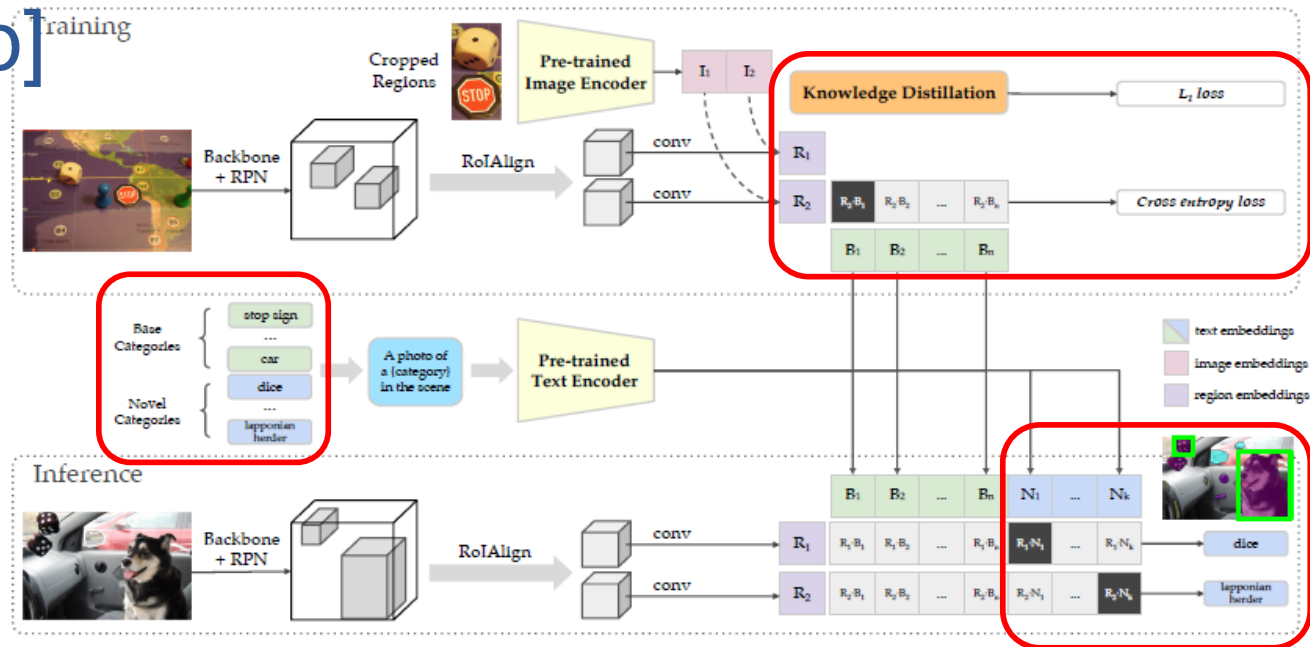


Figure 2: An overview of using ViLD for open-vocabulary object detection. ViLD distills the knowledge from a pretrained open-vocabulary image classification model. First, the category text embeddings and the image embeddings of cropped object proposals are computed, using the text and image encoders in the pretrained classification model. Then, ViLD employs the text embeddings as the region classifier (ViLD-text) and minimizes the distance between the region embedding and the image embedding for each proposal (ViLD-image). During inference, text embeddings of novel categories are used to enable open-vocabulary detection.

Gu et al. "Open-vocabulary Object Detection via Vision and Language Knowledge Distillation." ICLR 2021.