

Graph Neural Networks

Naeemullah Khan

naeemullah.khan@kaust.edu.sa



جامعة الملك عبد الله
للغات والتكنولوجيا
King Abdullah University of
Science and Technology

KAUST Academy
King Abdullah University of Science and Technology

June 15, 2023

Image Classification

- ▶ Previously, we discussed Image Classification
- ▶ A core task in Computer Vision



This image by Nikita is
licensed under CC-BY 2.0

(assume given a set of possible labels)
{dog, cat, truck, plane, ...}



Computer Vision Tasks

Classification



CAT

Semantic Segmentation



GRASS, CAT,
TREE, SKY

Object Detection



DOG, DOG, CAT

Instance Segmentation



DOG, DOG, CAT

No spatial extent

No objects, just pixels

Multiple Object

[This image is CC0 public domain](#)

Semantic Segmentation



GRASS, CAT,
TREE, SKY, ...

Paired training data: for each training image,
each pixel is labeled with a semantic category.



At test time, classify each pixel of a new image.

Semantic Segmentation

Full image



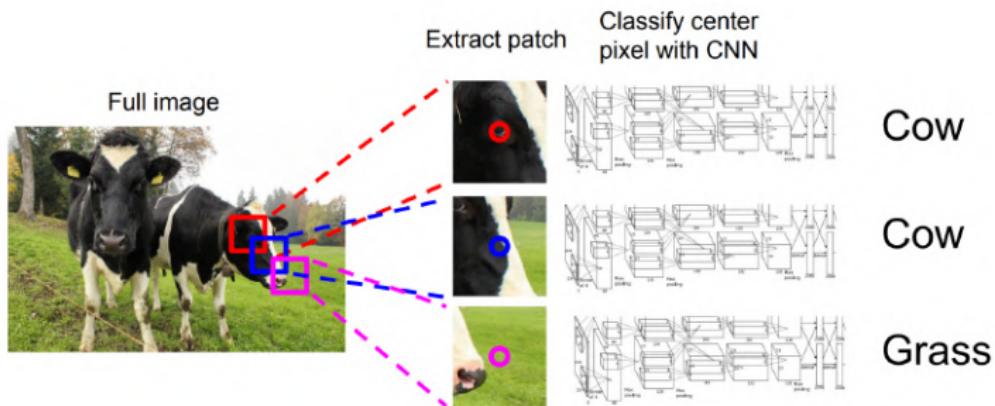
Semantic Segmentation

Full image



- ▶ Impossible to classify without context
- ▶ How do we include context?

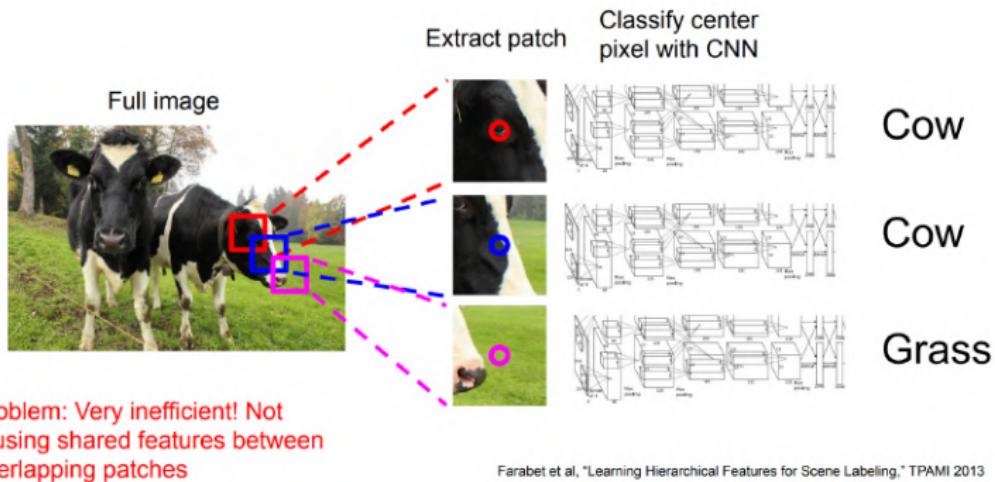
Semantic Segmentation Idea: Sliding Window



Farabet et al, "Learning Hierarchical Features for Scene Labeling," TPAMI 2013

Pinheiro and Collobert, "Recurrent Convolutional Neural Networks for Scene Labeling", ICML 2014

Semantic Segmentation Idea: Sliding Window (cont.)

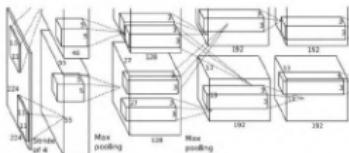


Farabet et al, "Learning Hierarchical Features for Scene Labeling," TPAMI 2013

Pinheiro and Collobert, "Recurrent Convolutional Neural Networks for Scene Labeling", ICML 2014

Semantic Segmentation Idea: Convolution

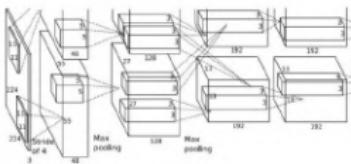
Full image



An intuitive idea: encode the entire image with conv net, and do semantic segmentation on top.

Semantic Segmentation Idea: Convolution (cont.)

Full image

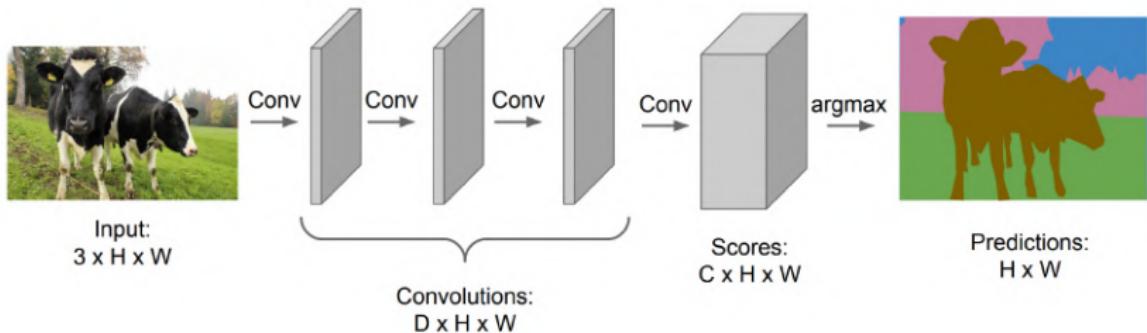


An intuitive idea: encode the entire image with conv net, and do semantic segmentation on top.

Problem: classification architectures often reduce feature spatial sizes to go deeper, but semantic segmentation requires the output size to be the same as input size.

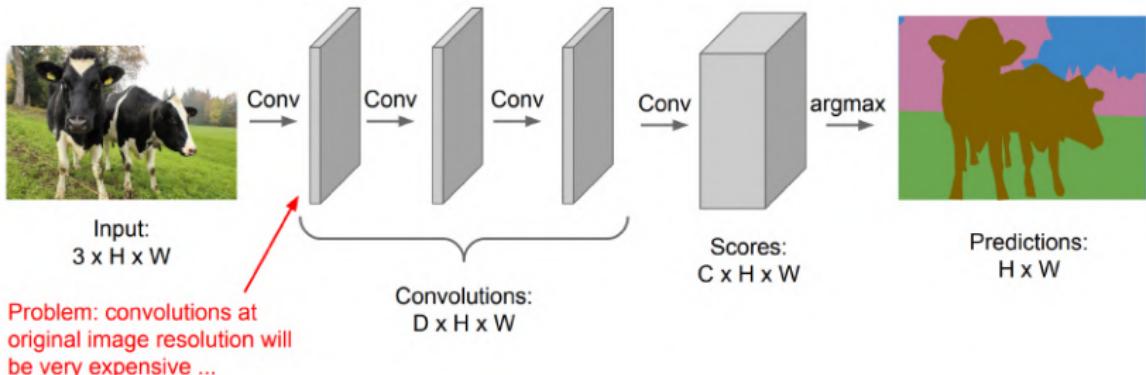
Semantic Segmentation Idea: Fully Convolutional

Design a network with only convolutional layers without downsampling operators to make predictions for pixels all at once!



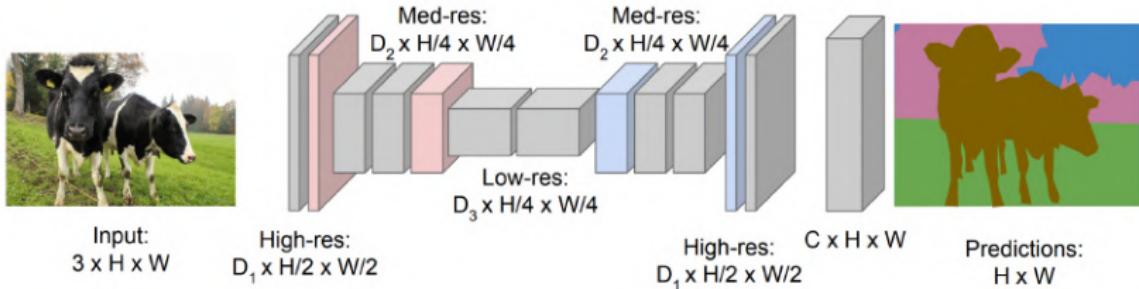
Semantic Segmentation Idea: Fully Convolutional (cont.)

Design a network with only convolutional layers without downsampling operators to make predictions for pixels all at once!



Semantic Segmentation Idea: Fully Convolutional (cont.)

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

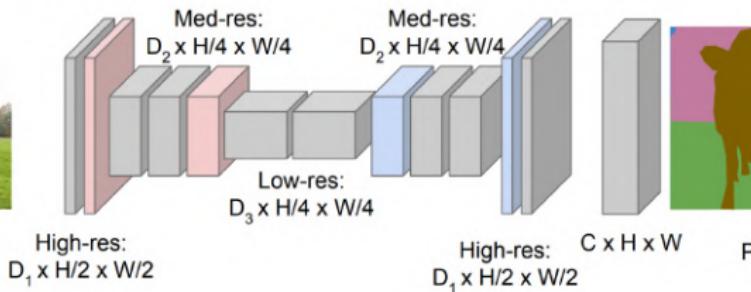
Semantic Segmentation Idea: Fully Convolutional (cont.)

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
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

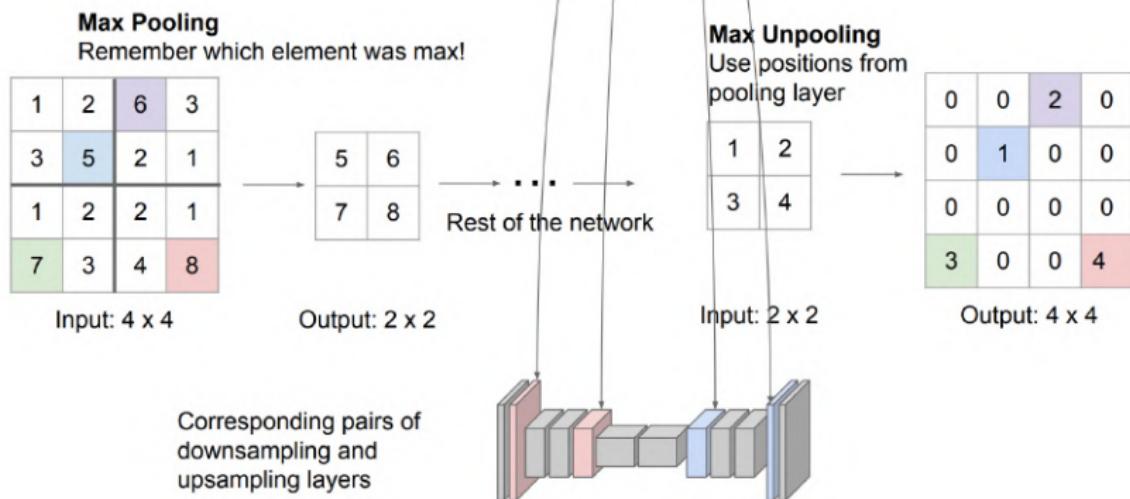


1	0	2	0
0	0	0	0
3	0	4	0
0	0	0	0

Input: 2 x 2

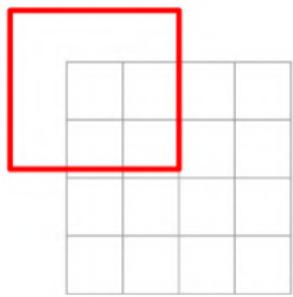
Output: 4 x 4

In-Network Upsampling: Max Unpooling



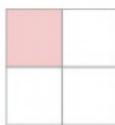
Learnable Upsampling: Transposed Convolution

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



Input: 4×4

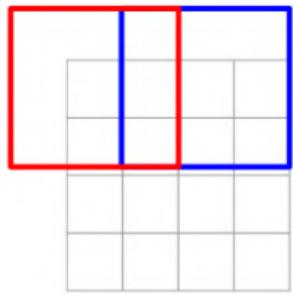
Dot product
between filter
and input



Output: 2×2

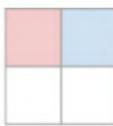
Learnable Upsampling: Transposed Convolution (cont.)

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



Input: 4×4

Dot product
between filter
and input



Output: 2×2

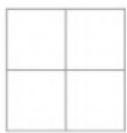
Filter moves 2 pixels in
the input for every one
pixel in the output

Stride gives ratio between
movement in input and
output

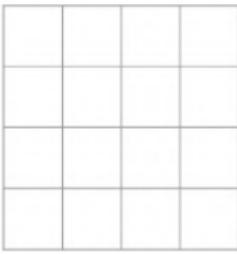
We can interpret strided
convolution as “learnable
downsampling”.

Learnable Upsampling: Transposed Convolution (cont.)

3 x 3 transposed convolution, stride 2 pad 1



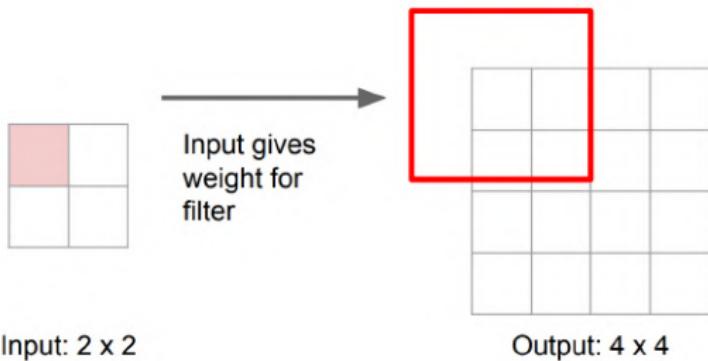
Input: 2 x 2



Output: 4 x 4

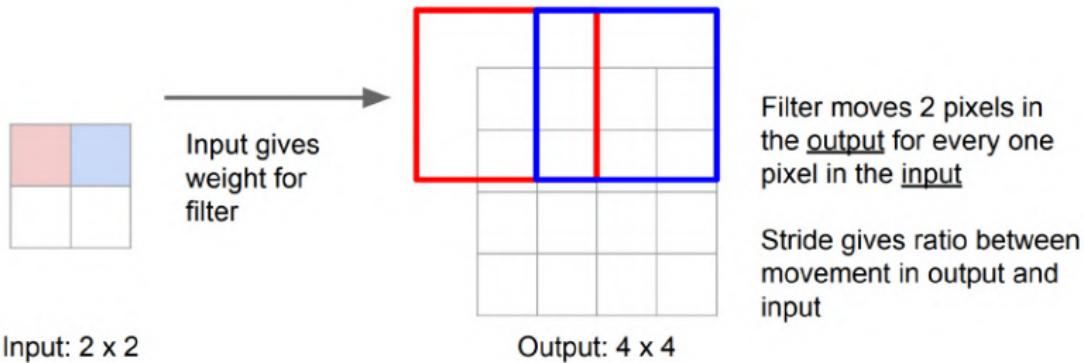
Learnable Upsampling: Transposed Convolution (cont.)

3 x 3 transposed convolution, stride 2 pad 1

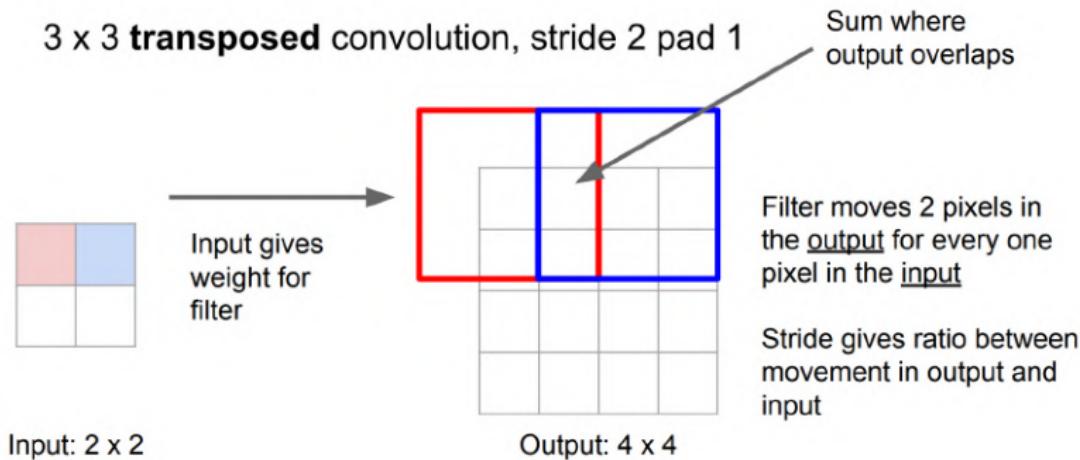


Learnable Upsampling: Transposed Convolution (cont.)

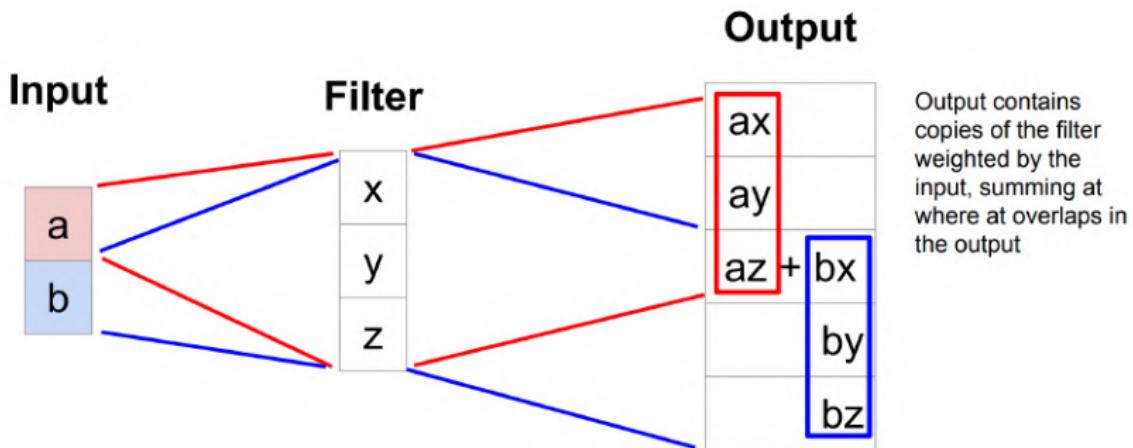
3 x 3 **transposed** convolution, stride 2 pad 1



Learnable Upsampling: Transposed Convolution (cont.)



Transposed Convolution: 1D Example



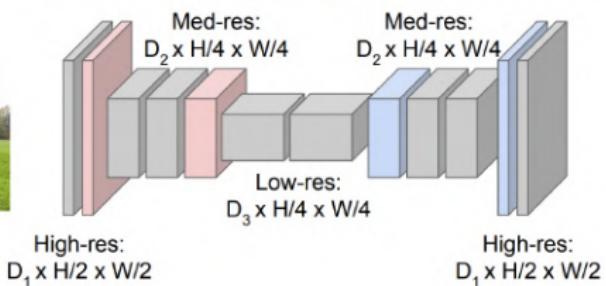
Semantic Segmentation Idea: Fully Convolutional

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:
Unpooling or strided transposed convolution

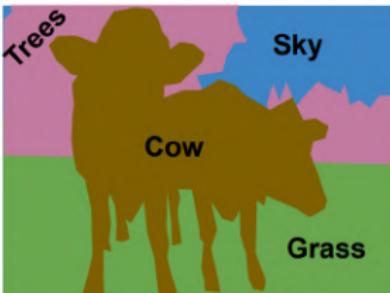
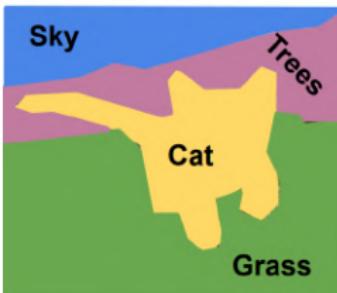


Predictions:
 $H \times W$

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

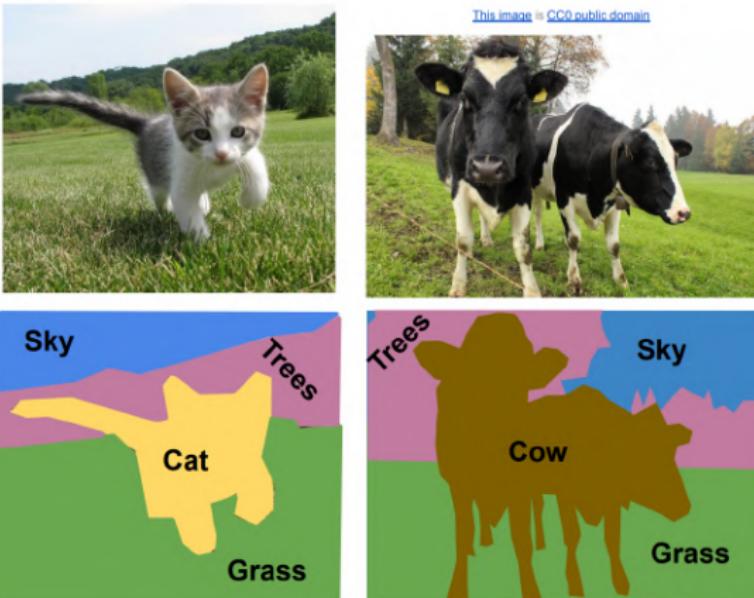
Semantic Segmentation

- ▶ Label each pixel in the image with a category label



Semantic Segmentation

- ▶ Label each pixel in the image with a category label

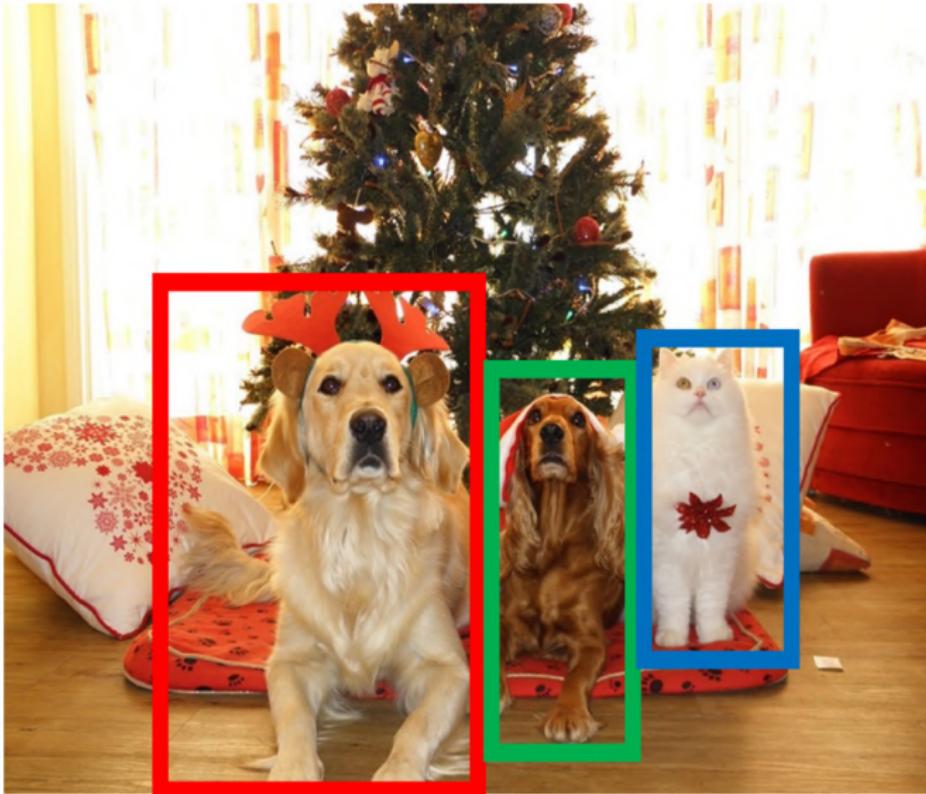


- ▶ Does not differentiate instances, only care about pixels

Object Detection

- ▶ **Input:** Single RGB Image
- ▶ **Output:** A set of detected objects. For each object predict:
 - Category label (from fixed, known set of categories)
 - Bounding box (four numbers: x, y, width, height)

Object Detection (cont.)

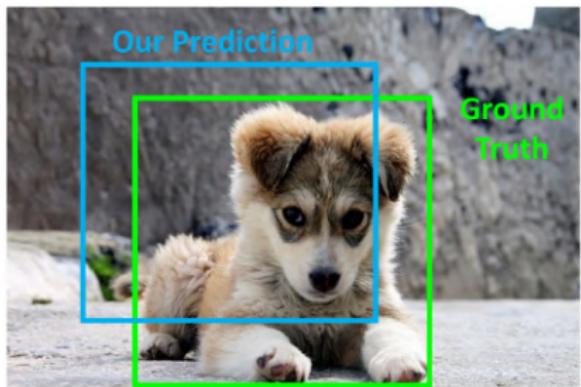


Object Detection: Challenges

- ▶ **Multiple outputs:** Need to output variable numbers of objects per image
- ▶ **Multiple types of output:** Need to predict "what" (category label) as well as "where" (bounding box)
- ▶ **Large images:** Classification works at 224x224; need higher resolution for detection, often $\sim 800 \times 600$

Comparing Boxes: Intersection over Union (IoU)

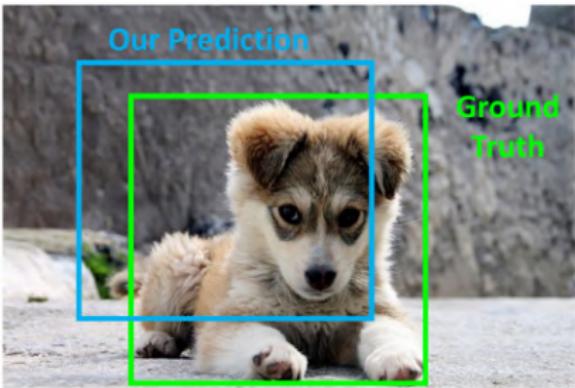
- ▶ How can we compare our prediction to the ground-truth box?



Comparing Boxes: Intersection over Union (IoU)

- ▶ How can we compare our prediction to the ground-truth box?
- ▶ **Intersection over Union (IoU)**
(Also called "Jaccard similarity" or "Jaccard index"):

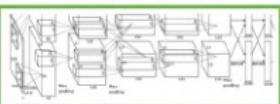
$$\frac{\text{Area of Intersection}}{\text{Area of Union}}$$



Detecting a Single Object



Often pretrained
on ImageNet
(Transfer learning)

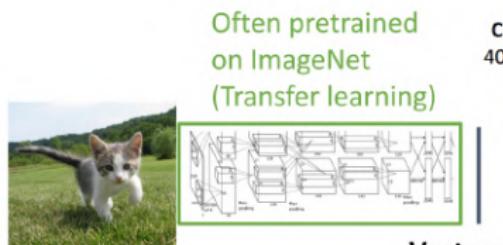


Vector:
4096

This image is CC-B public domain

Treat localization as a
regression problem!

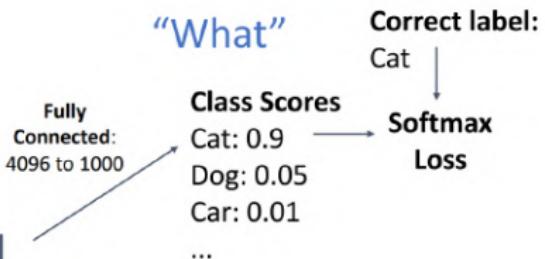
Detecting a Single Object (cont.)



This image is CC0 public domain

Treat localization as a regression problem!

Vector:
4096



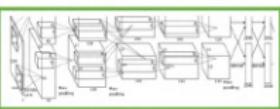
Detecting a Single Object (cont.)

Often pretrained
on ImageNet
(Transfer learning)

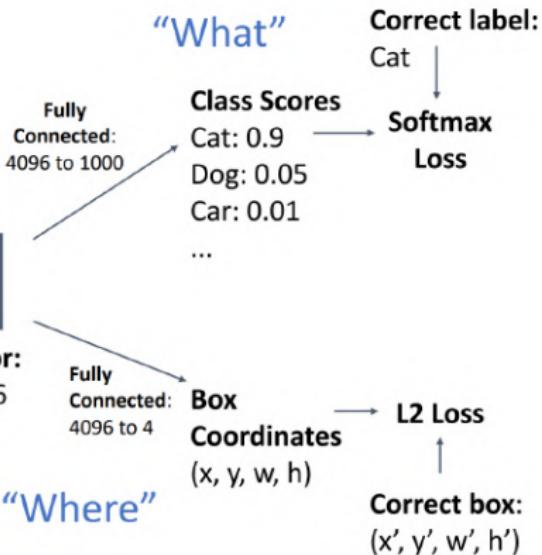


This image is CC0 public domain.

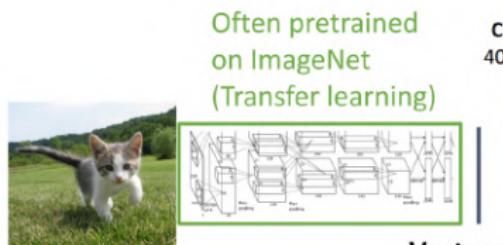
Treat localization as a
regression problem!



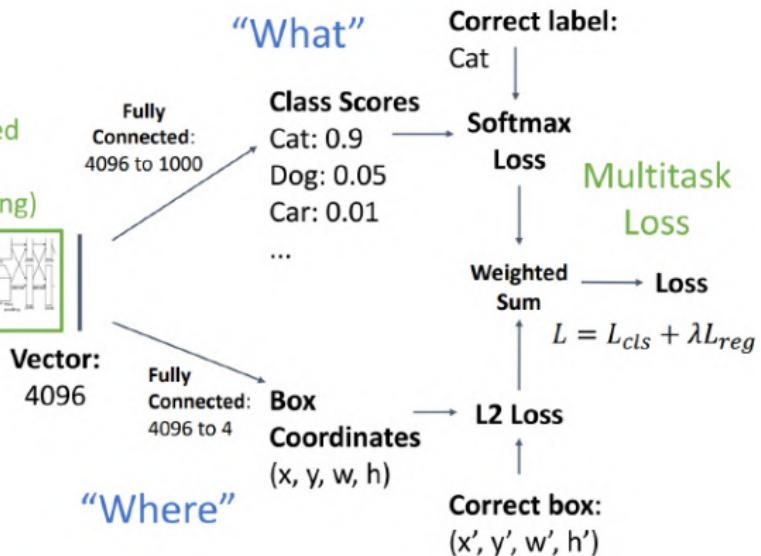
Vector:
4096



Detecting a Single Object (cont.)



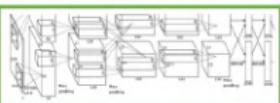
Treat localization as a regression problem!



Detecting a Single Object (cont.)



Often pretrained
on ImageNet
(Transfer learning)

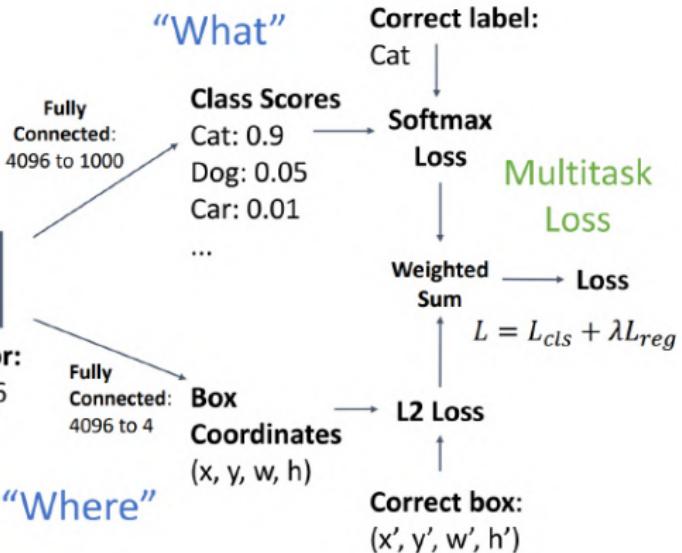


This image is CC-BY public domain.

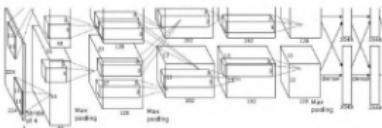
Treat localization as a
regression problem!

Problem: Images can have
more than one object!

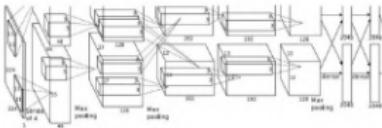
Vector:
4096



Multiple Objects



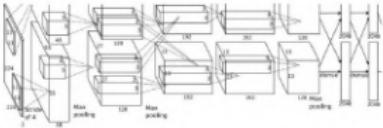
CAT: (x, y, w, h)



DOG: (x, y, w, h)

DOG: (x, y, w, h)

CAT: (x, y, w, h)



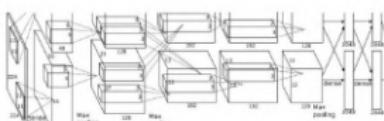
DUCK: (x, y, w, h)

DUCK: (x, y, w, h)

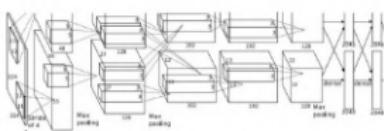
....

Multiple Objects (cont.)

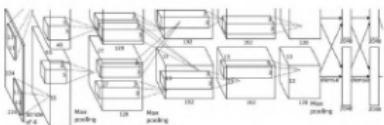
Each image needs a
different number of outputs!



CAT: (x, y, w, h) **4 numbers**



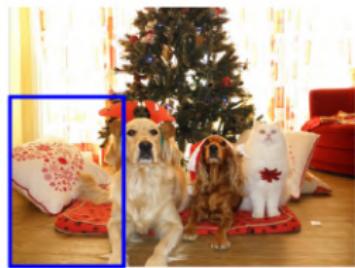
DOG: (x, y, w, h)
DOG: (x, y, w, h) **12 numbers**
CAT: (x, y, w, h)



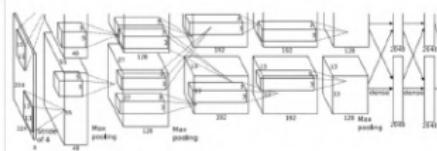
DUCK: (x, y, w, h) **Many numbers!**
DUCK: (x, y, w, h) **Many numbers!**

....

Detecting Multiple Objects: Sliding Window



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

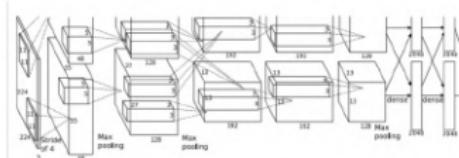


Dog? NO
Cat? NO
Background? YES

Detecting Multiple Objects: Sliding Window (cont.)



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

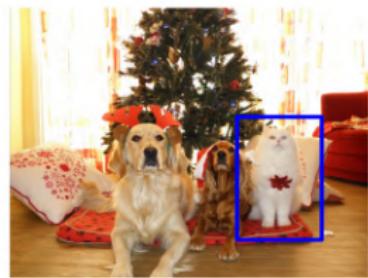


Dog? YES

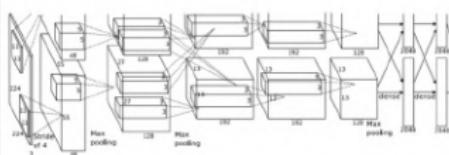
Cat? NO

Background? NO

Detecting Multiple Objects: Sliding Window (cont.)

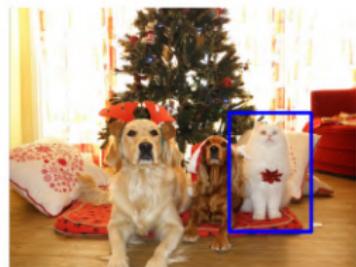


Apply a CNN to many different crops of the image, CNN classifies each crop as object or background



Dog? NO
Cat? YES
Background? NO

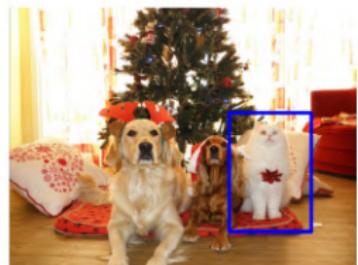
Detecting Multiple Objects: Sliding Window (cont.)



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

Question: How many possible boxes are there in an image of size $H \times W$?

Detecting Multiple Objects: Sliding Window (cont.)



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

Question: How many possible boxes are there in an image of size $H \times W$?

Consider a box of size $h \times w$:

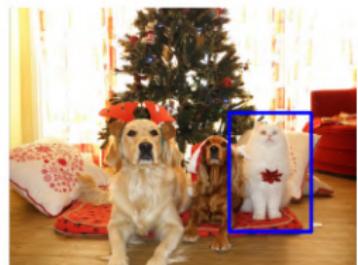
Possible x positions: $W - w + 1$

Possible y positions: $H - h + 1$

Possible positions:

$$(W - w + 1) * (H - h + 1)$$

Detecting Multiple Objects: Sliding Window (cont.)



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

Question: How many possible boxes are there in an image of size $H \times W$?

Consider a box of size $h \times w$:

Possible x positions: $W - w + 1$

Possible y positions: $H - h + 1$

Possible positions:

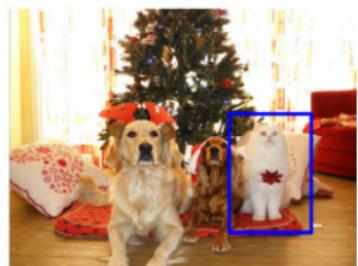
$$(W - w + 1) * (H - h + 1)$$

Total possible boxes:

$$\sum_{h=1}^H \sum_{w=1}^W (W - w + 1)(H - h + 1)$$

$$= \frac{H(H + 1)}{2} \frac{W(W + 1)}{2}$$

Detecting Multiple Objects: Sliding Window (cont.)



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

800 x 600 image
has ~58M boxes!
No way we can evaluate them all

Question: How many possible boxes are there in an image of size $H \times W$?

Consider a box of size $h \times w$:

Possible x positions: $W - w + 1$

Possible y positions: $H - h + 1$

Possible positions:

$$(W - w + 1) * (H - h + 1)$$

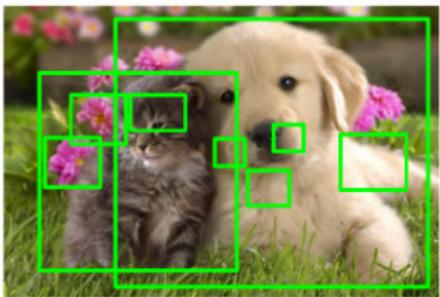
Total possible boxes:

$$\sum_{h=1}^H \sum_{w=1}^W (W - w + 1)(H - h + 1)$$

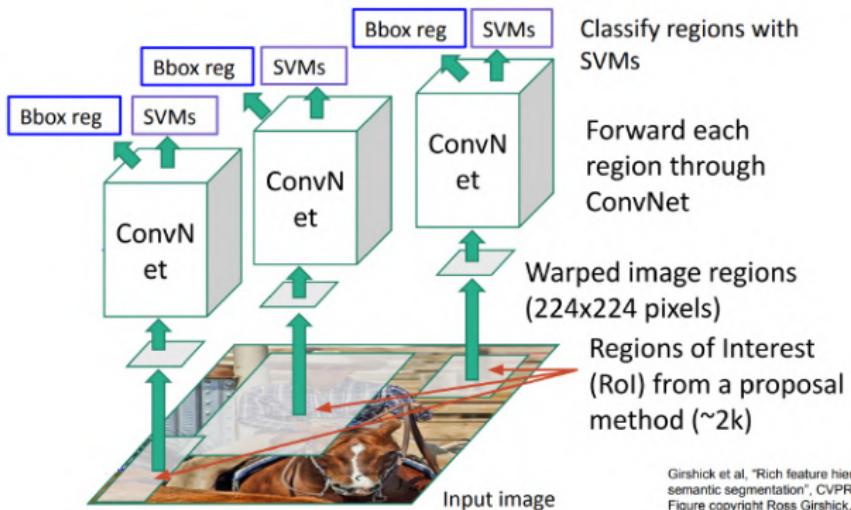
$$= \frac{H(H + 1)}{2} \frac{W(W + 1)}{2}$$

Region Proposal

- ▶ Find a small set of boxes that are likely to cover all objects
- ▶ Often based on heuristics: e.g. look for “blob-like” image regions
- ▶ Relatively fast to run; e.g. Selective Search gives 2000 region proposals in a few seconds on CPU



Predict “corrections” to the RoI: 4 numbers: (dx, dy, dw, dh)

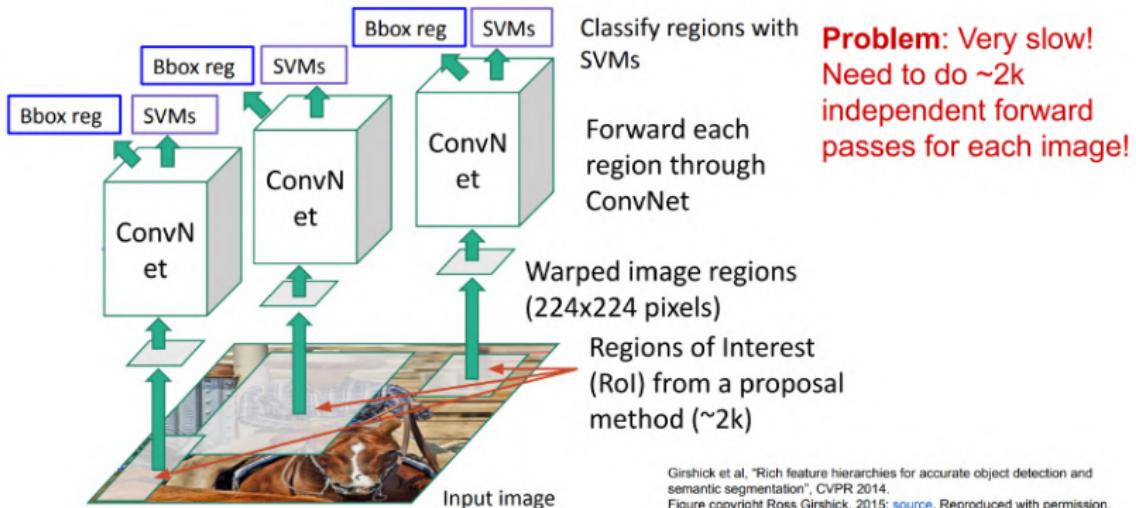


Girshick et al, “Rich feature hierarchies for accurate object detection and semantic segmentation”, CVPR 2014.

Figure copyright Ross Girshick, 2015; [source](#). Reproduced with permission.

R-CNN (cont.)

Predict “corrections” to the RoI: 4 numbers: (dx, dy, dw, dh)

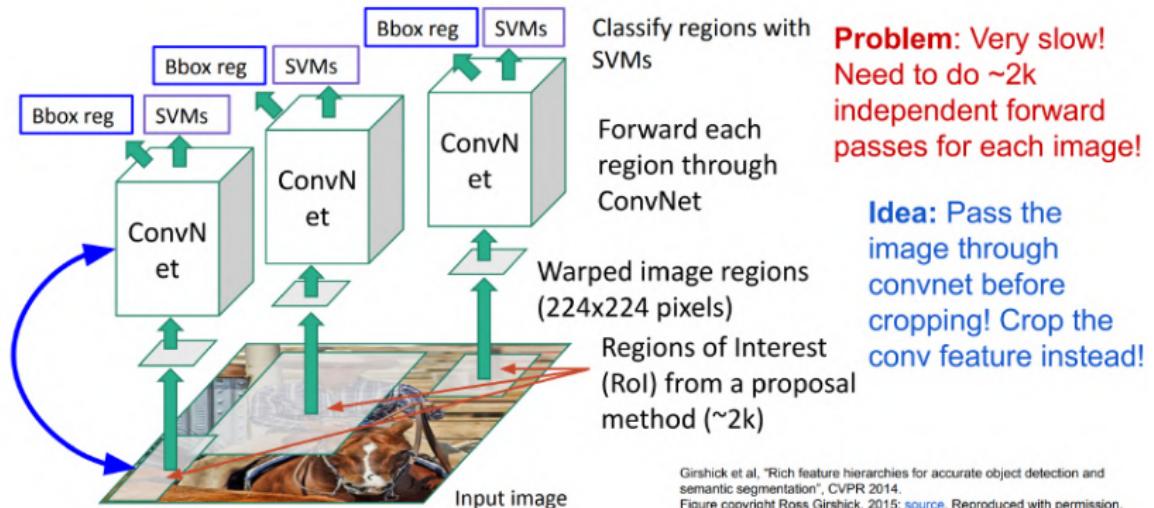


Girshick et al, "Rich feature hierarchies for accurate object detection and semantic segmentation", CVPR 2014.

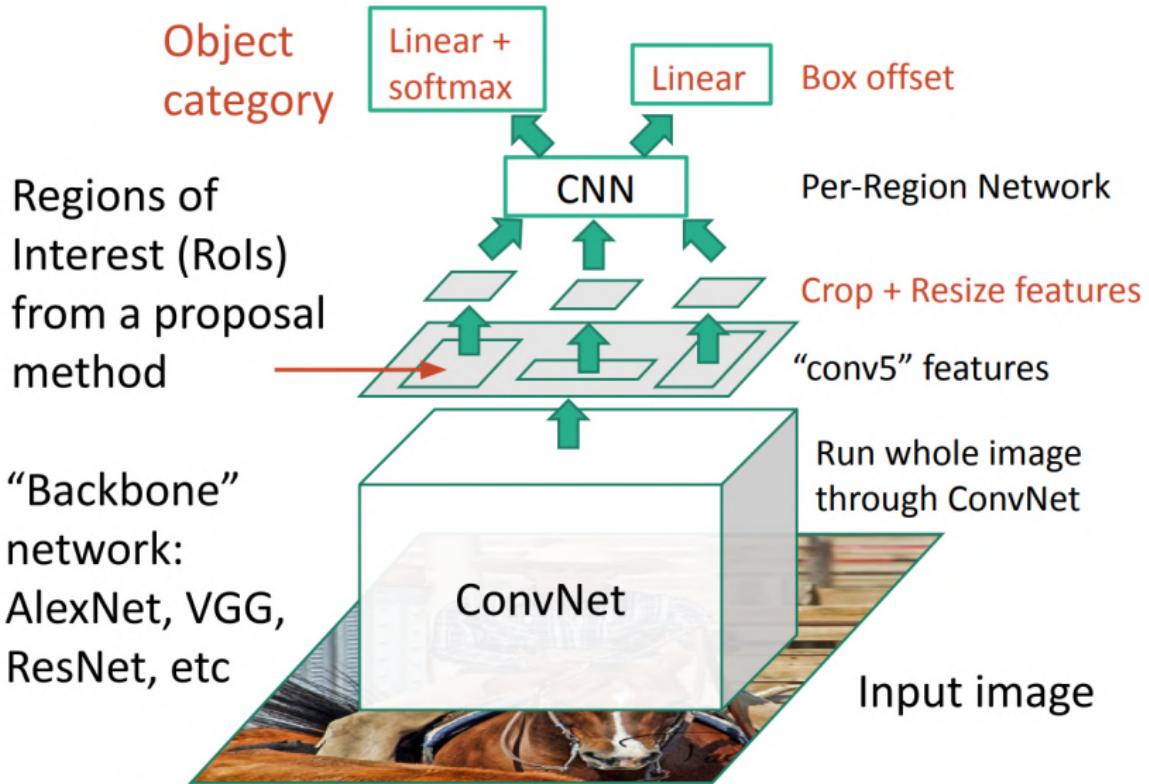
Figure copyright Ross Girshick, 2015; [source](#). Reproduced with permission.

R-CNN (cont.)

Predict “corrections” to the RoI: 4 numbers: (dx, dy, dw, dh)



Girshick et al, ‘Rich feature hierarchies for accurate object detection and semantic segmentation’, CVPR 2014.
Figure copyright Ross Girshick, 2015; [source](#). Reproduced with permission.

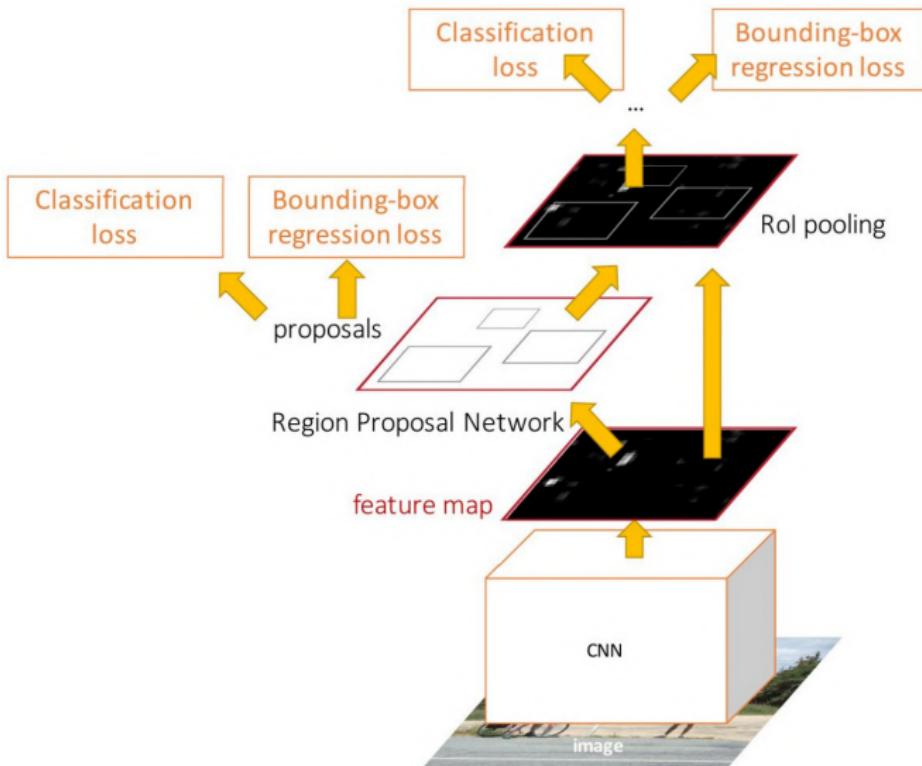


Faster R-CNN

- ▶ Make CNN do proposals!
- ▶ Insert Region Proposal Network (RPN) to predict proposals from features

- ▶ Make CNN do proposals!
- ▶ Insert Region Proposal Network (RPN) to predict proposals from features
- ▶ Jointly train on 4 losses:
 - **RPN classification:** anchor box is object / not an object
 - **RPN regression:** predict transform from anchor box to proposal box
 - **Object classification:** classify proposals as background / object class
 - **Object regression:** predict transform from proposal box to object box

Faster R-CNN



Faster R-CNN: Make CNN do proposals!

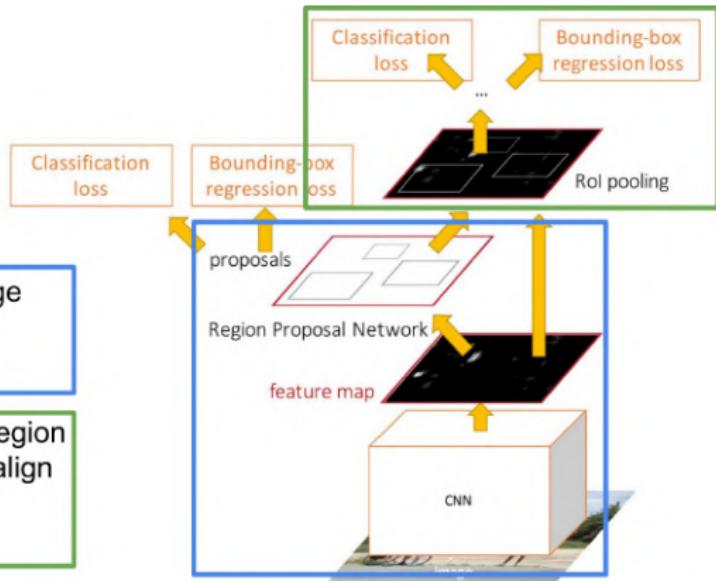
Faster R-CNN is a
Two-stage object detector

First stage: Run once per image

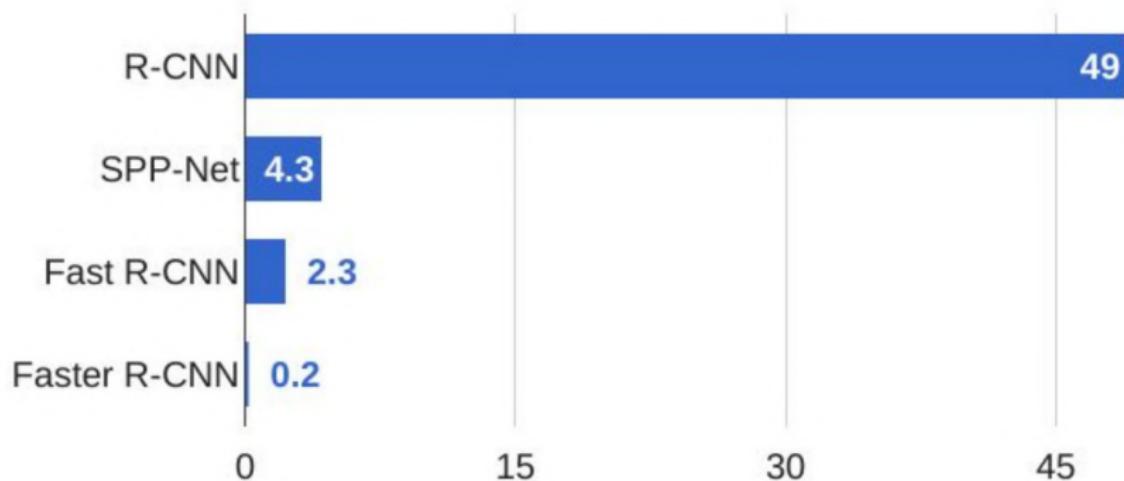
- Backbone network
- Region proposal network

Second stage: Run once per region

- Crop features: RoI pool / align
- Predict object class
- Prediction bbox offset

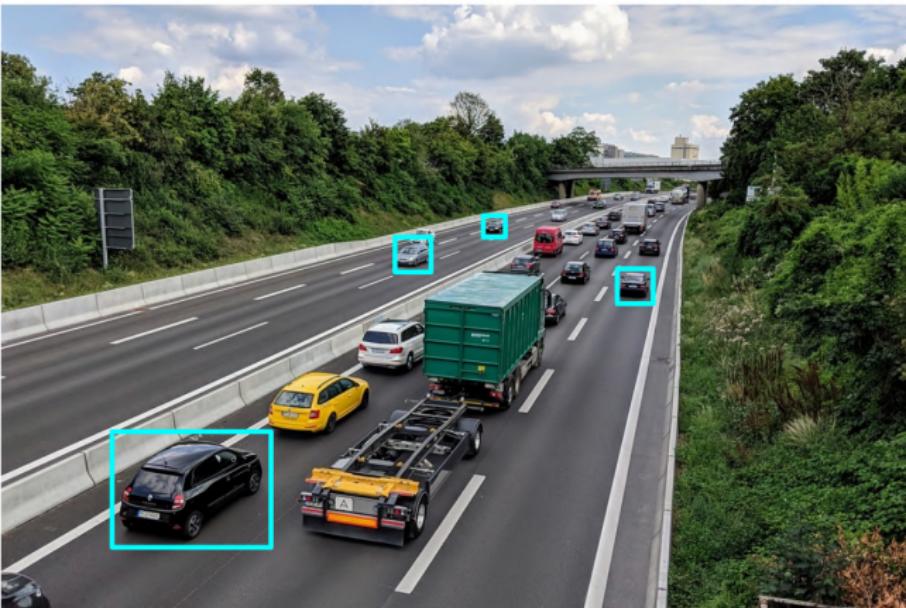


R-CNN Test-Time Speed



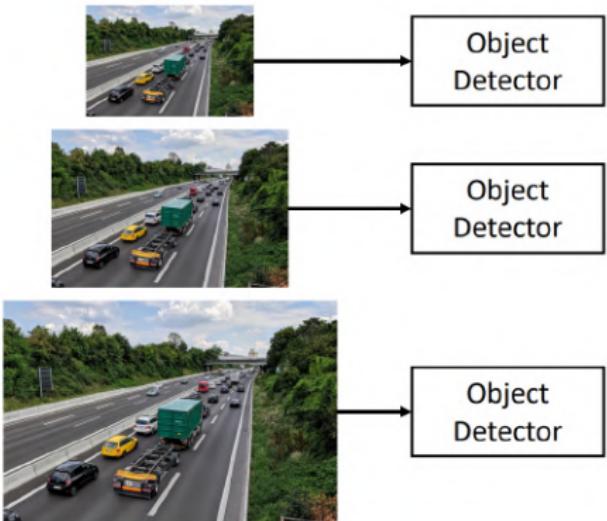
Dealing with Scale

- ▶ We need to detect objects of many different scales.
- ▶ How to improve scale invariance of the detector



Dealing with Scale: Image Pyramid

Classic idea: build an *image pyramid* by resizing the image to different scales, then process each image scale independently.



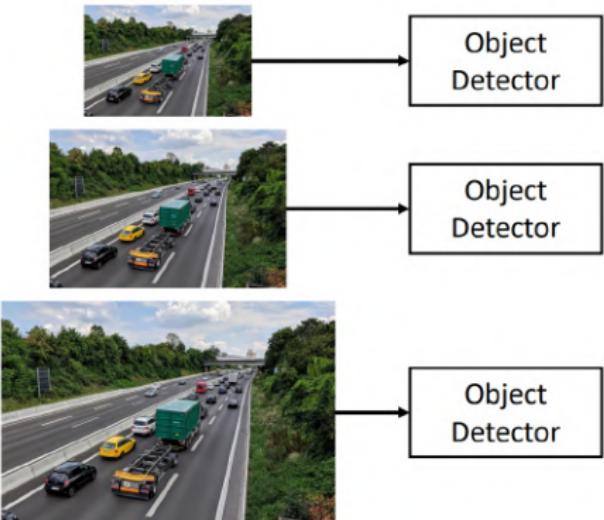
Lin et al, "Feature Pyramid Networks for Object Detection", ICCV 2017

Dealing with Scale: Image Pyramid (cont.)

Classic idea: build an *image pyramid* by resizing the image to different scales, then process each image scale independently.

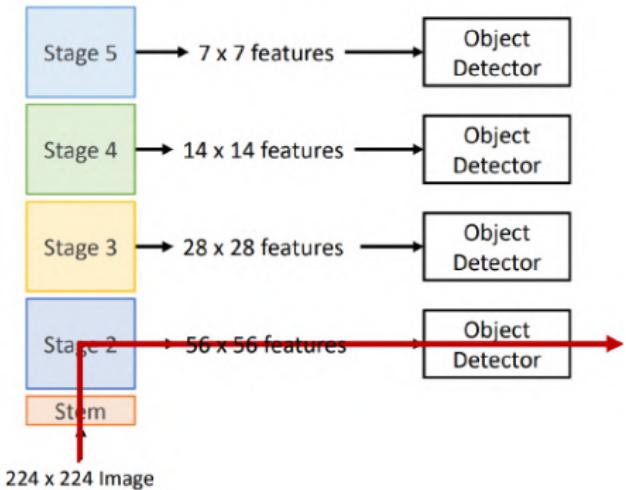
Problem: Expensive! Don't share any computation between scales

Lin et al, "Feature Pyramid Networks for Object Detection", ICCV 2017



Dealing with Scale: Image Pyramid

CNNs have multiple *stages* that operate at different resolutions. Attach an independent detector to the features at each level



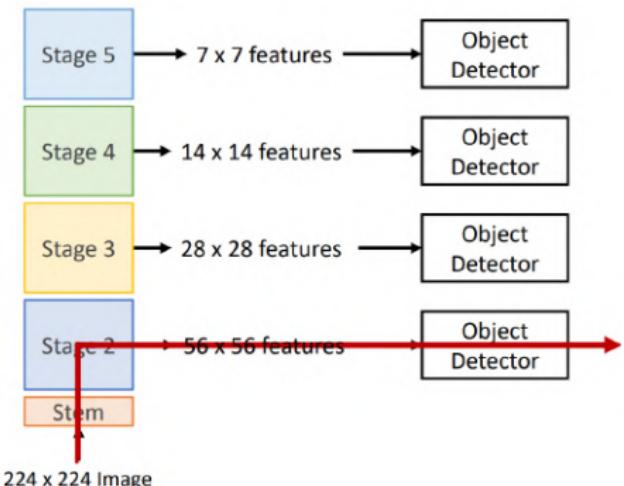
Lin et al, "Feature Pyramid Networks for Object Detection", ICCV 2017

Dealing with Scale: Image Pyramid (cont.)

CNNs have multiple *stages* that operate at different resolutions. Attach an independent detector to the features at each level

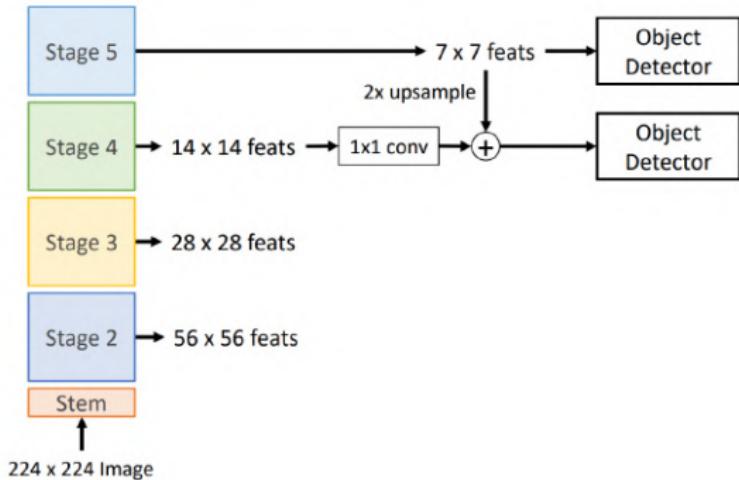
Problem: detector on early features doesn't make use of the entire backbone; doesn't get access to high-level features

Lin et al, "Feature Pyramid Networks for Object Detection", ICCV 2017



Dealing with Scale: Feature Pyramid Network

Add *top down connections* that feed information from high level features back down to lower level features



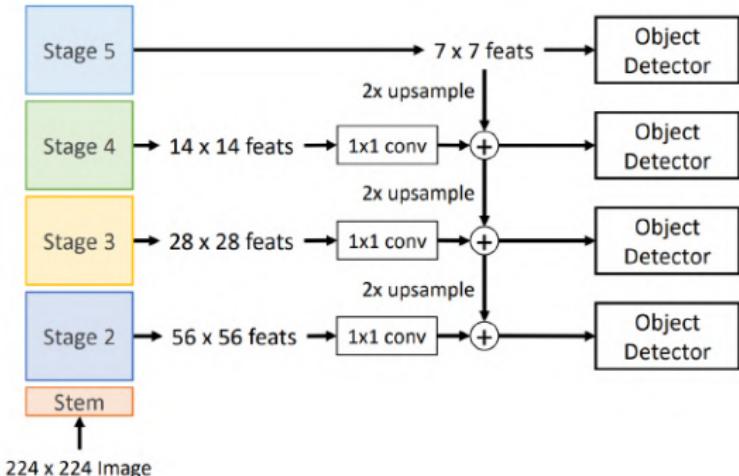
Lin et al, "Feature Pyramid Networks for Object Detection", ICCV 2017

Dealing with Scale: Feature Pyramid Network (cont.)

Add *top down connections* that feed information from high level features back down to lower level features

Efficient multiscale features where all levels benefit from the whole backbone! Widely used in practice

Lin et al, "Feature Pyramid Networks for Object Detection", ICCV 2017

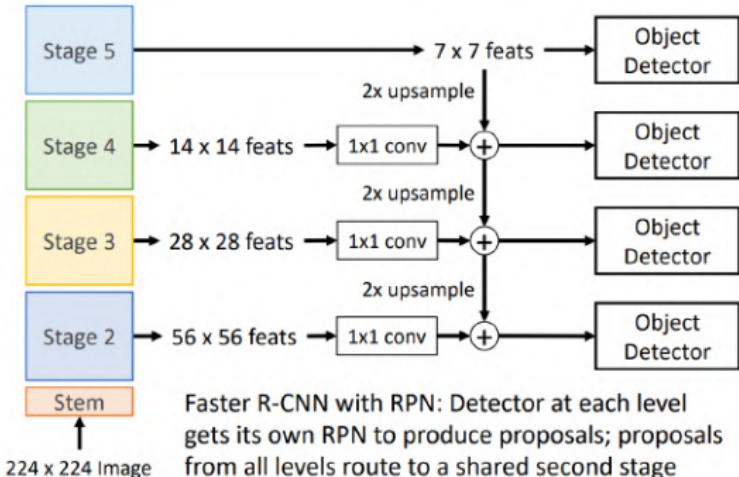


Dealing with Scale: Feature Pyramid Network (cont.)

Add *top down connections* that feed information from high level features back down to lower level features

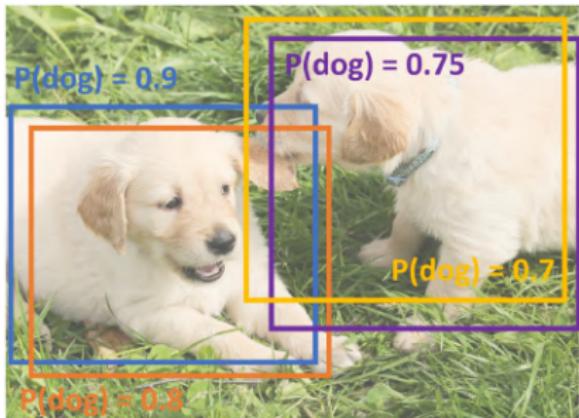
Efficient multiscale features where all levels benefit from the whole backbone! Widely used in practice

Lin et al, "Feature Pyramid Networks for Object Detection", ICCV 2017



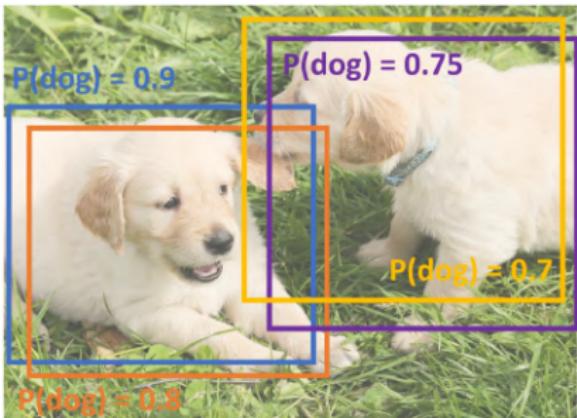
Overlapping Boxes: Non-Max Suppression (NMS)

- ▶ **Problem:** Object detectors often output many overlapping detections



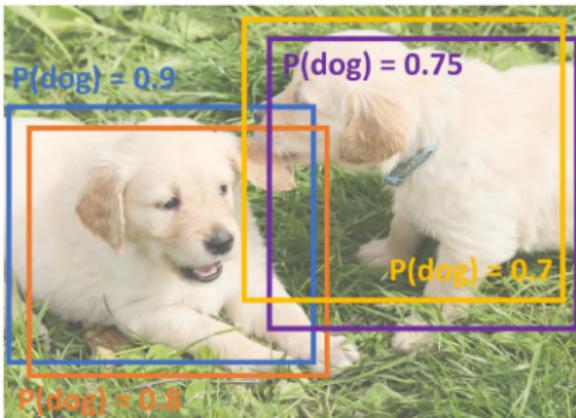
Overlapping Boxes: Non-Max Suppression (NMS)

- ▶ **Problem:** Object detectors often output many overlapping detections
- ▶ **Solution:** Post-process raw detections using Non-Max Suppression (NMS)



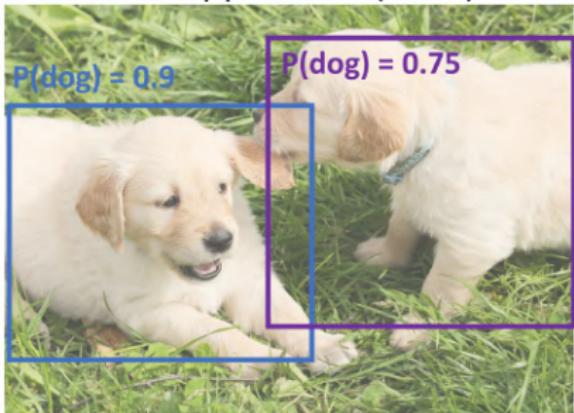
Overlapping Boxes: Non-Max Suppression (NMS)

- ▶ **Problem:** Object detectors often output many overlapping detections
- ▶ **Solution:** Post-process raw detections using Non-Max Suppression (NMS)
 1. Select next highest-scoring box
 2. Eliminate lower-scoring boxes
 3. with $\text{IoU} > \text{threshold}$ (e.g. 0.7)
 4. If any boxes remain, GOTO 1



Overlapping Boxes: Non-Max Suppression (NMS)

- ▶ **Problem:** Object detectors often output many overlapping detections
- ▶ **Solution:** Post-process raw detections using Non-Max Suppression (NMS)
 1. Select next highest-scoring box
 2. Eliminate lower-scoring boxes
 3. with $\text{IoU} > \text{threshold}$ (e.g. 0.7)
 4. If any boxes remain, GOTO 1



Overlapping Boxes: Non-Max Suppression (NMS)

- ▶ **Problem:** Object detectors often output many overlapping detections
- ▶ **Solution:** Post-process raw detections using Non-Max Suppression (NMS)
 1. Select next highest-scoring box
 2. Eliminate lower-scoring boxes
 3. with $\text{IoU} > \text{threshold}$ (e.g. 0.7)
 4. If any boxes remain, GOTO 1
- ▶ **Problem:** NMS may eliminate "good" boxes when objects are highly overlapping... no good solution =(



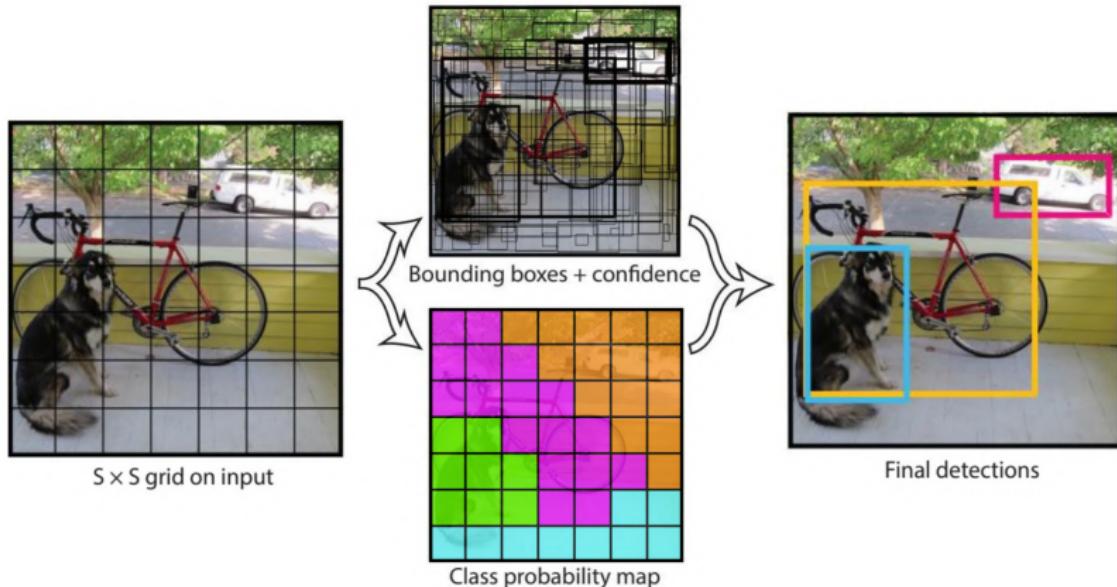
Single Shot Object Detection



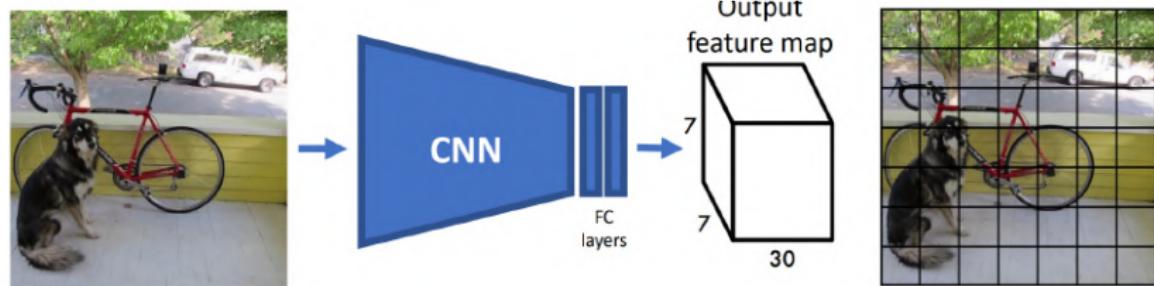
Single Shot:
SSD, YOLO ...

Fast
High false rate

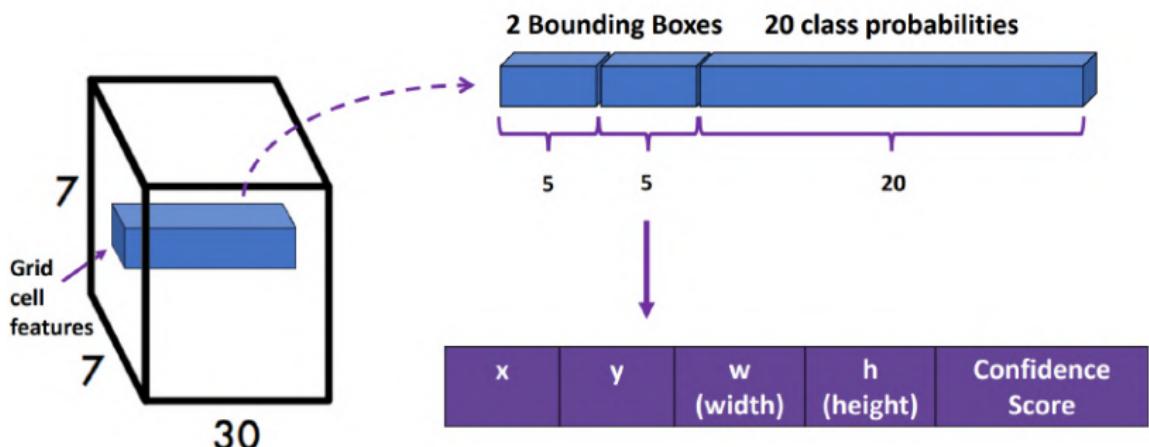
YOLO - Overview

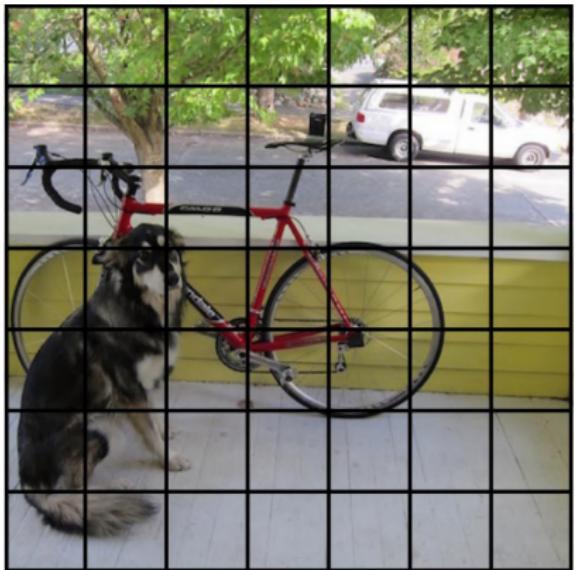


YOLO - Overview

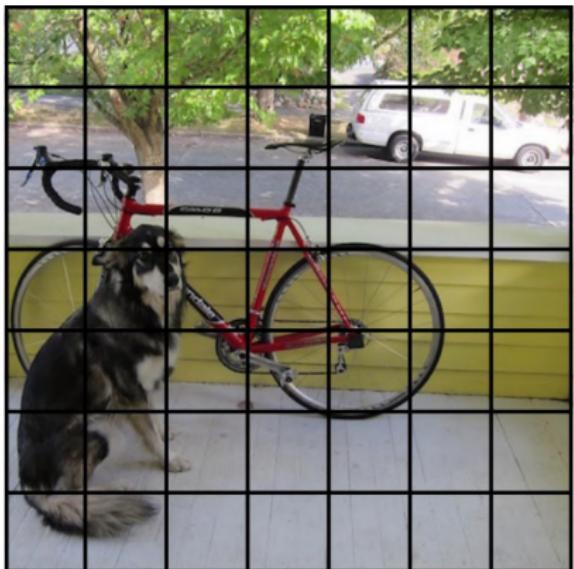


YOLO - Overview





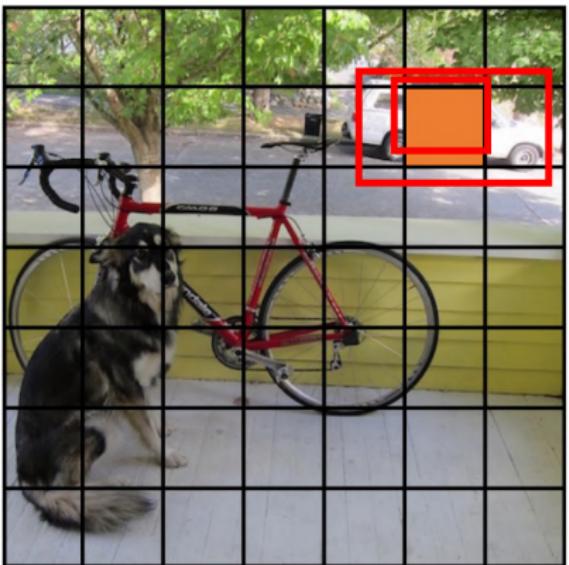
Each cell predicts





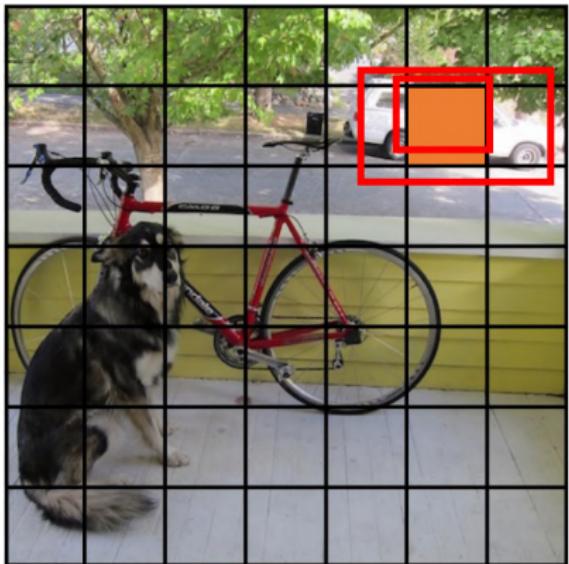
Each cell predicts

- ▶ $B = 2$ bounding boxes
 $(x, y, w, h) + \text{confidence score}$



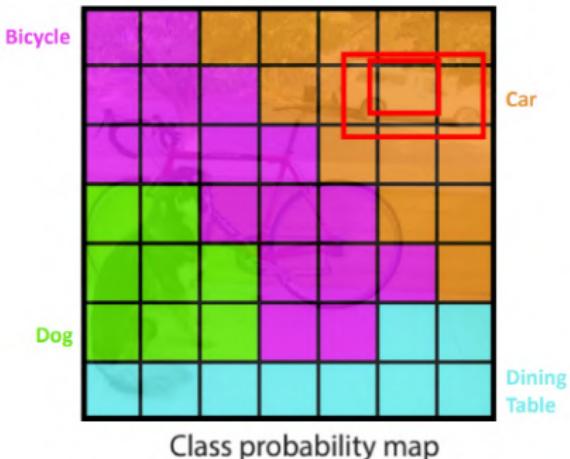
Each cell predicts

- ▶ $B = 2$ bounding boxes
 $(x, y, w, h) +$ confidence score
- ▶ $C = 20$ class probabilities



Each cell predicts

- ▶ $B = 2$ bounding boxes
 $(x, y, w, h) +$ confidence score
- ▶ $C = 20$ class probabilities



Each cell predicts

- ▶ $B = 2$ bounding boxes
 $(x, y, w, h) +$ confidence score
- ▶ $C = 20$ class probabilities

SxSxB Bounding-Boxes ($S=7, B=2 \rightarrow 96$ Bboxes)

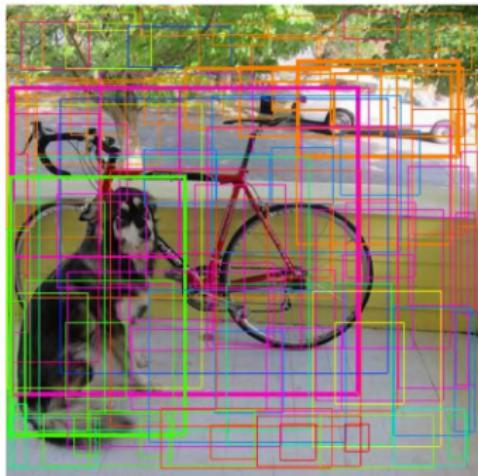


7×7 grid on input

Each cell predicts

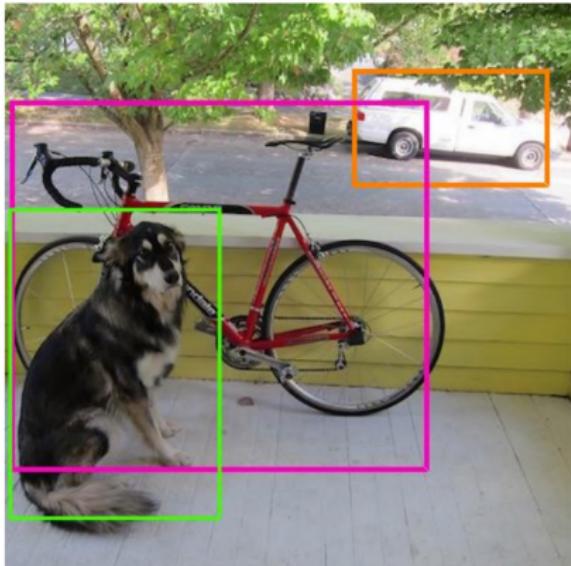
- ▶ $B = 2$ bounding boxes
 $(x, y, w, h) +$ confidence score
- ▶ $C = 20$ class probabilities

SxSxB Bounding-Boxes ($S=7, B=2 \rightarrow 96$ Bboxes)



Each cell predicts

- ▶ $B = 2$ bounding boxes
 $(x, y, w, h) +$ confidence score
- ▶ $C = 20$ class probabilities
- ▶ Apply Non-Maximum Suppression



YOLO - Loss Function

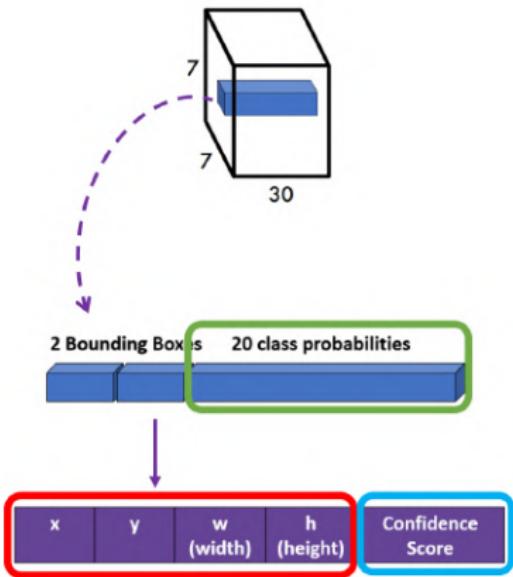


YOLO – Loss function

$$\mathcal{L} = \mathcal{L}_{Localization\ Loss}$$

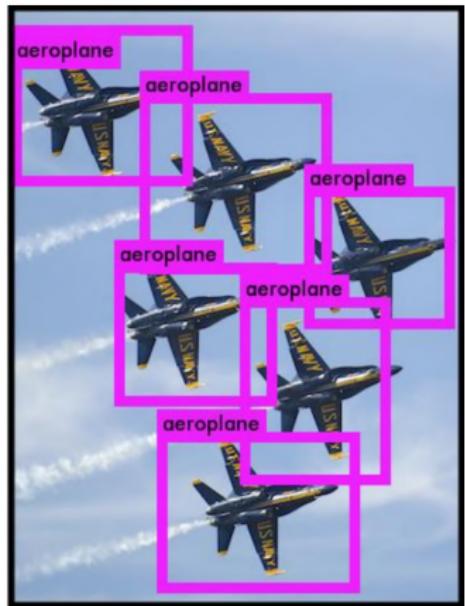
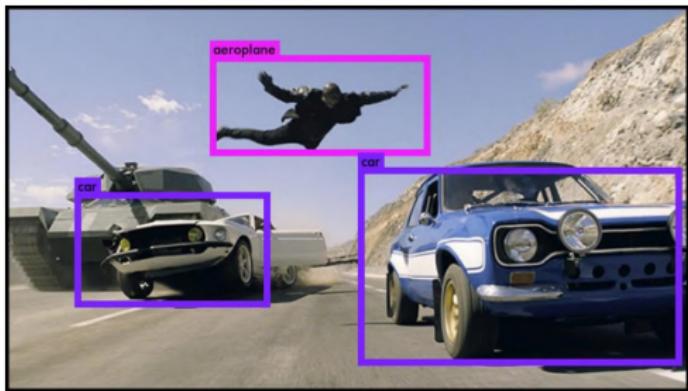
$$+ \mathcal{L}_{Confidence\ Loss}$$

$$+ \mathcal{L}_{Classification\ Loss}$$

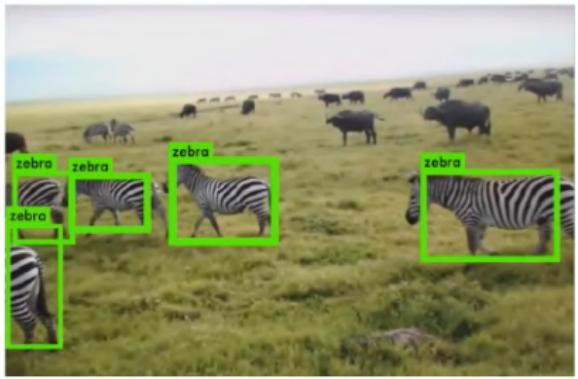


YOLO - Benefits

- ▶ Fast. Good for real-time processing
- ▶ End-to-end training

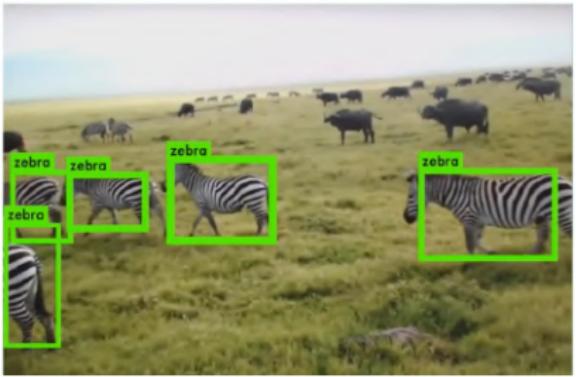


YOLO - Limitations



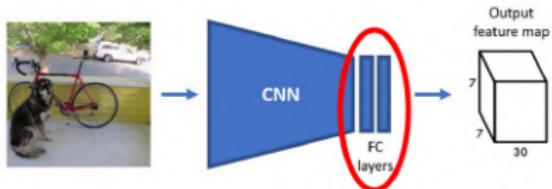
YOLO - Limitations

- ▶ Difficult to detect small objects
- ▶ Coarse predictions



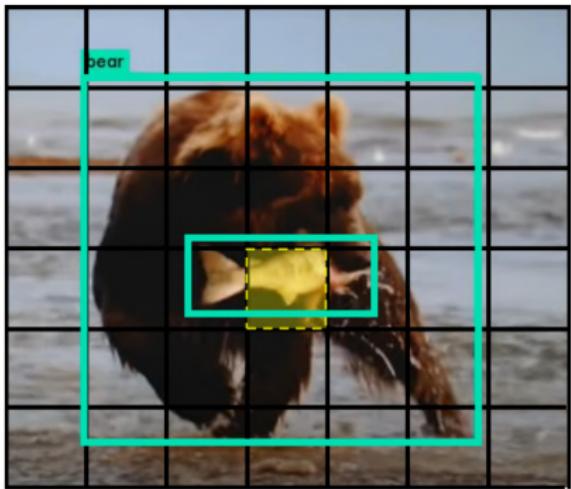
YOLO - Limitations

- ▶ Difficult to detect small objects
- ▶ Coarse predictions
- ▶ Fixed input size



YOLO - Limitations

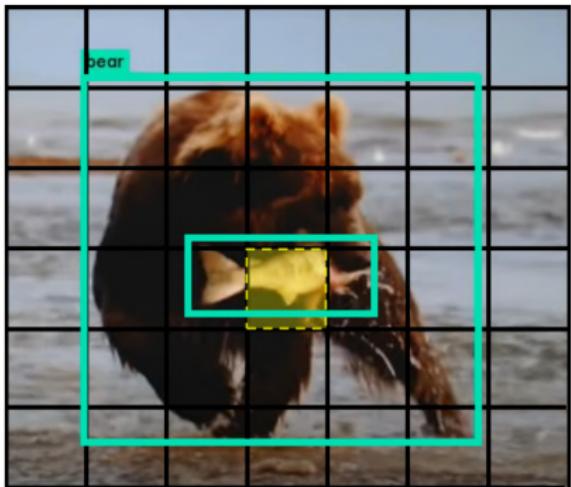
- ▶ Difficult to detect small objects
- ▶ Coarse predictions
- ▶ Fixed input size
- ▶ A grid cell can predict only one class



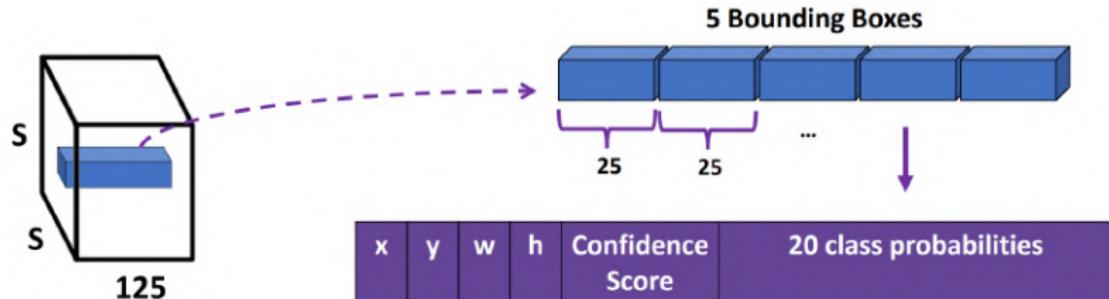
YOLO - Limitations

- ▶ Difficult to detect small objects
- ▶ Coarse predictions
- ▶ Fixed input size
- ▶ A grid cell can predict only one class

- ▶ Solutions:
 - Remove fc layers!
 - Predict class per bbox (not per cell)



- ▶ Removed fully connected layers
- ▶ A grid cell predicts class probabilities for each box



► YOLOv3

- J. Redmon, A. Farhadi. Yolov3: An incremental improvement, 2018

► YOLOv4

- A. Bochkovskiy, C. Wang, H. Liao. Yolov4: Optimal speed and accuracy of object detection (Feb. 2020)

► YOLOv5

- YOLOv5 by ultralytics (June 2020)

► PP-YOLO

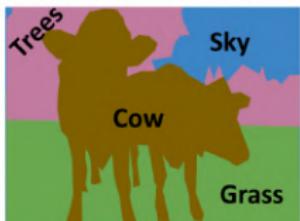
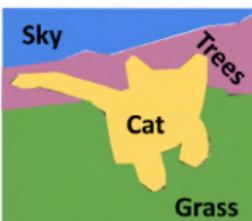
- X. Long, K. Deng, G. Wang, Y. Zhang, Q. Dang, Y. Gao, H. Shen, J. Ren, S. Han, E. Ding, S. Wen. Pp-yolo: An effective and efficient implementation of object detector (June 2020)

► PP-YOLOv2 (2021)

- J. X. Huang, X. Wang, W. Lv, X. Bai, X. Long, K. Deng, Q. Dang, S. Han, Q. Liu, X. Hu, D. Yu, Y. Ma, O. Yoshie. PP-YOLOv2: A Practical Object Detector (2021)

Things and Stuff

- ▶ **Things:** Object categories that can be separated into object instances (e.g. cats, cars, person)
- ▶ **Stuff:** Object categories that cannot be separated into instances (e.g. sky, grass, water, trees)

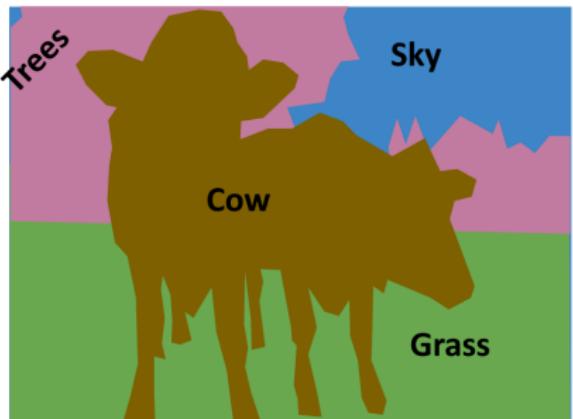


Computer Vision Tasks

- ▶ **Object Detection:** Detects individual object instances, but only gives box(Only things!)

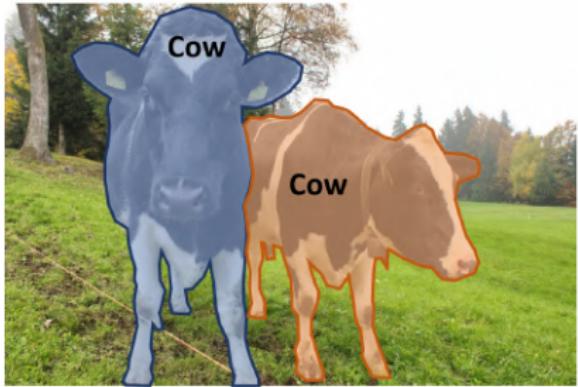


- ▶ **Semantic Segmentation:** Gives per-pixel labels, but merges instances (Both things and stuff)



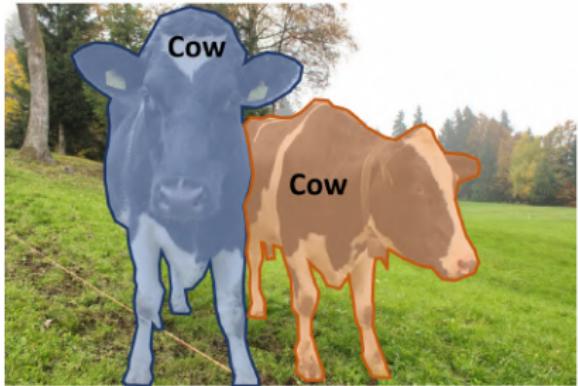
Instance Segmentation

- ▶ Detect all objects in the image, and identify the pixels that belong to each object (Only things!)

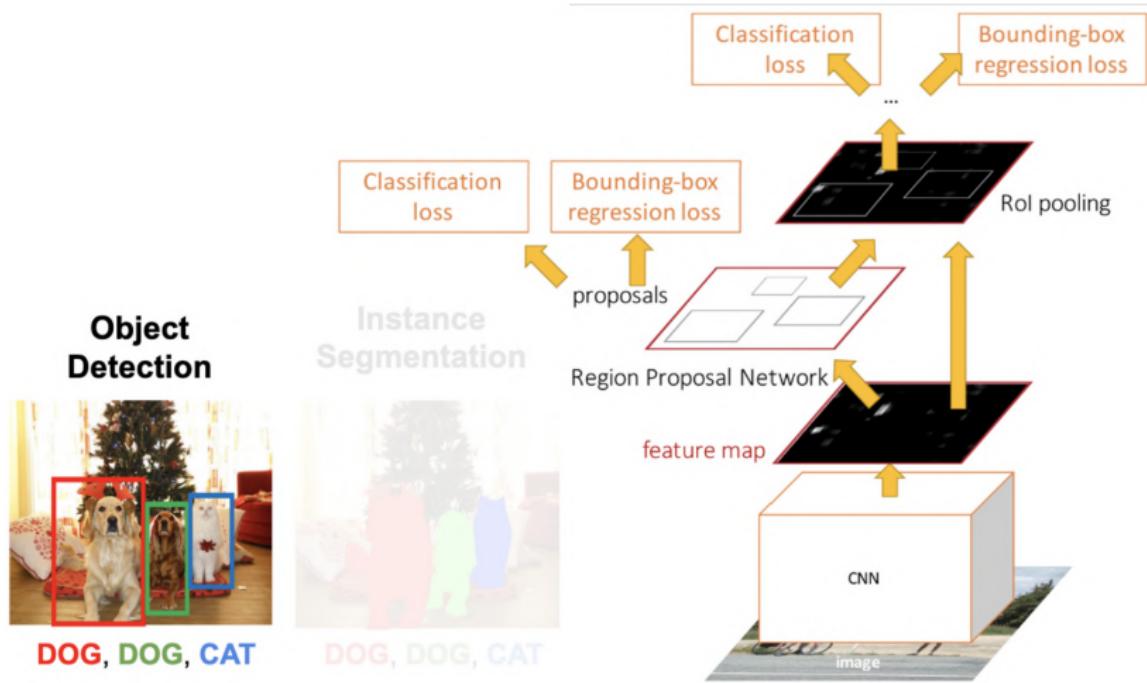


Instance Segmentation

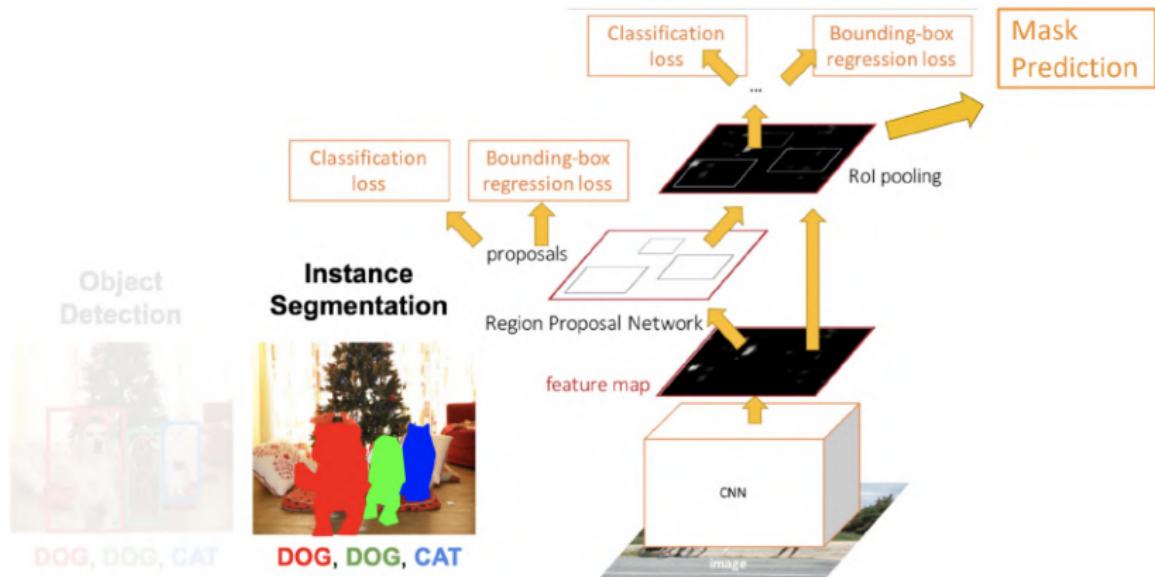
- ▶ Detect all objects in the image, and identify the pixels that belong to each object (Only things!)
- ▶ **Approach:** Perform object detection, then predict a segmentation mask for each object!



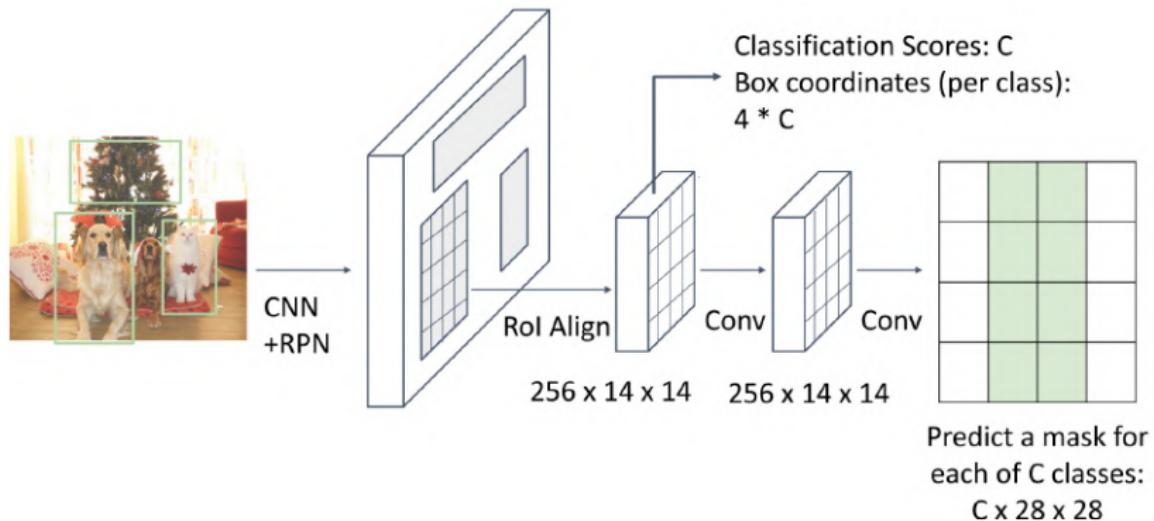
Object Detection: Faster R-CNN



Instance Segmentation: Mask R-CNN

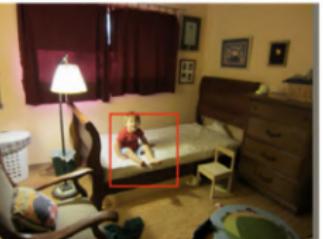
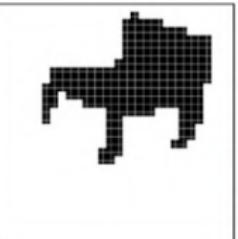
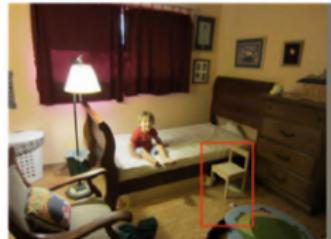
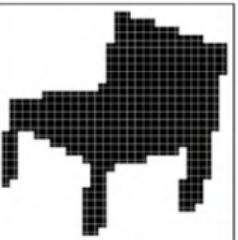


Instance Segmentation: Mask R-CNN

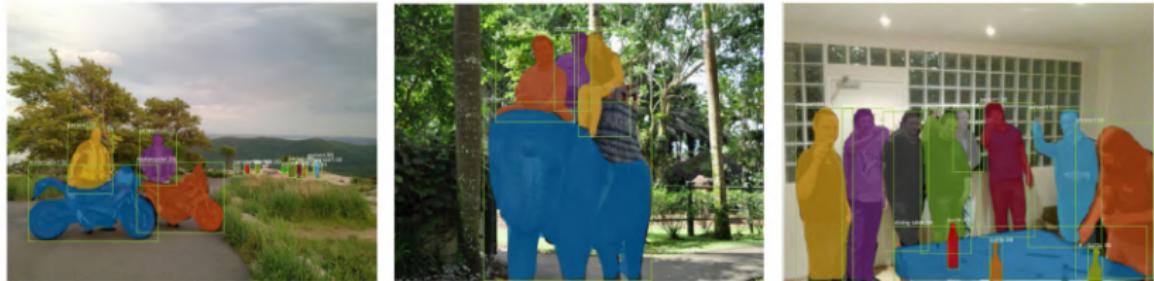


Mask R-CNN: Example Training Targets

Mask R-CNN: Example Training Targets

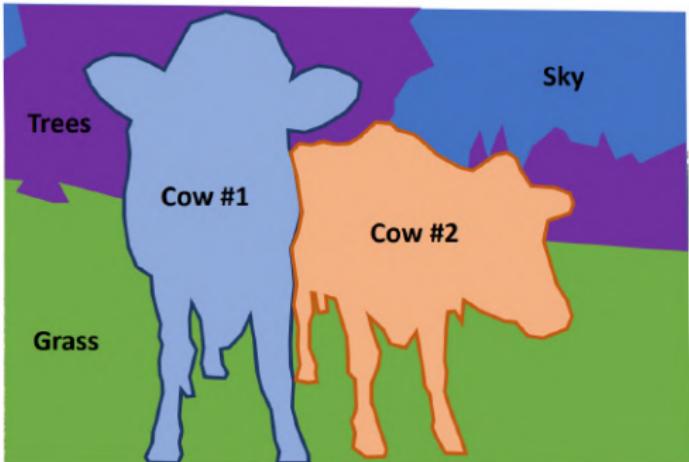


Mask R-CNN: Very Good Results!



Beyond Instance Segmentation: Panoptic Segmentation

- ▶ Label all pixels in the image (both things and stuff)
- ▶ For "thing" categories also separate into instances



Beyond Instance Segmentation: Panoptic Segmentation



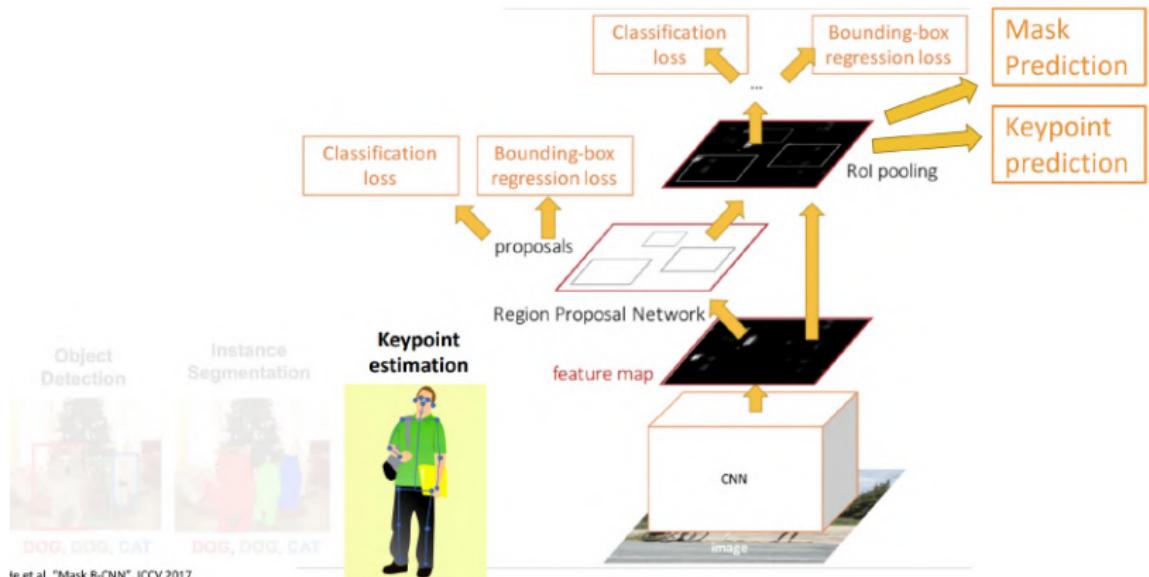
⁰Kirillov et al, "Panoptic Feature Pyramid Networks", CVPR 2019

Beyond Instance Segmentation: Human Keypoints

- ▶ Represent the pose of a human by locating a set of keypoint se.g. 17 keypoints:
 - ▶ Nose
 - ▶ Left / Right eye
 - ▶ Left / Right ear
 - ▶ Left / Right shoulder
 - ▶ Left / Right elbow
 - ▶ Left / Right wrist

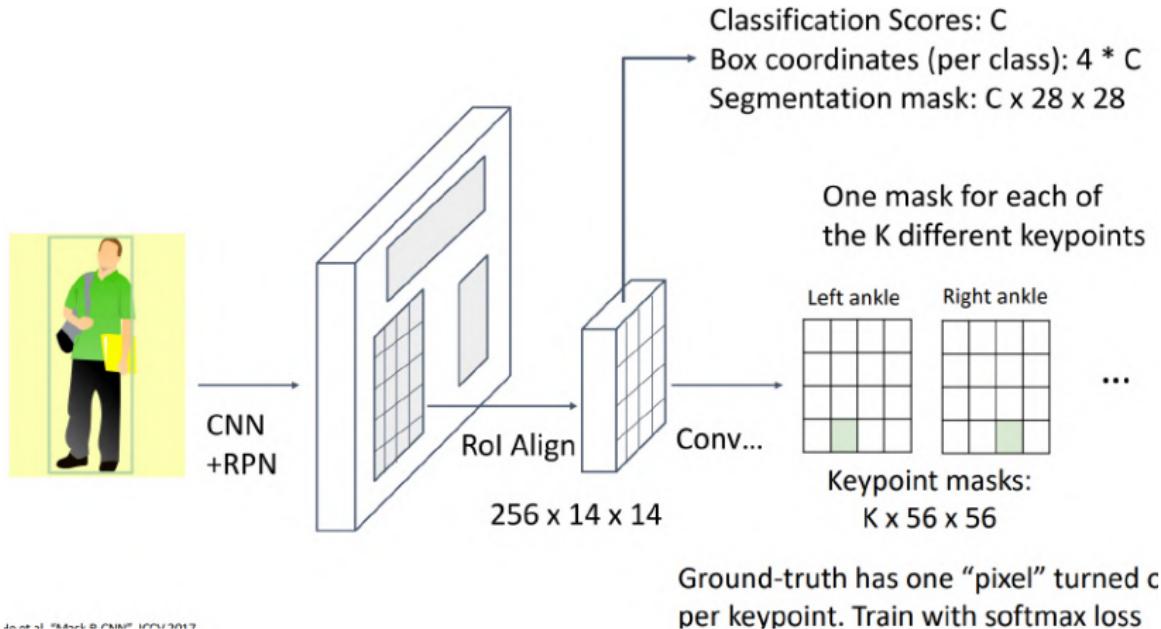


Mask R-CNN: Keypoint Estimation



Ie et al., "Mask R-CNN", ICCV 2017

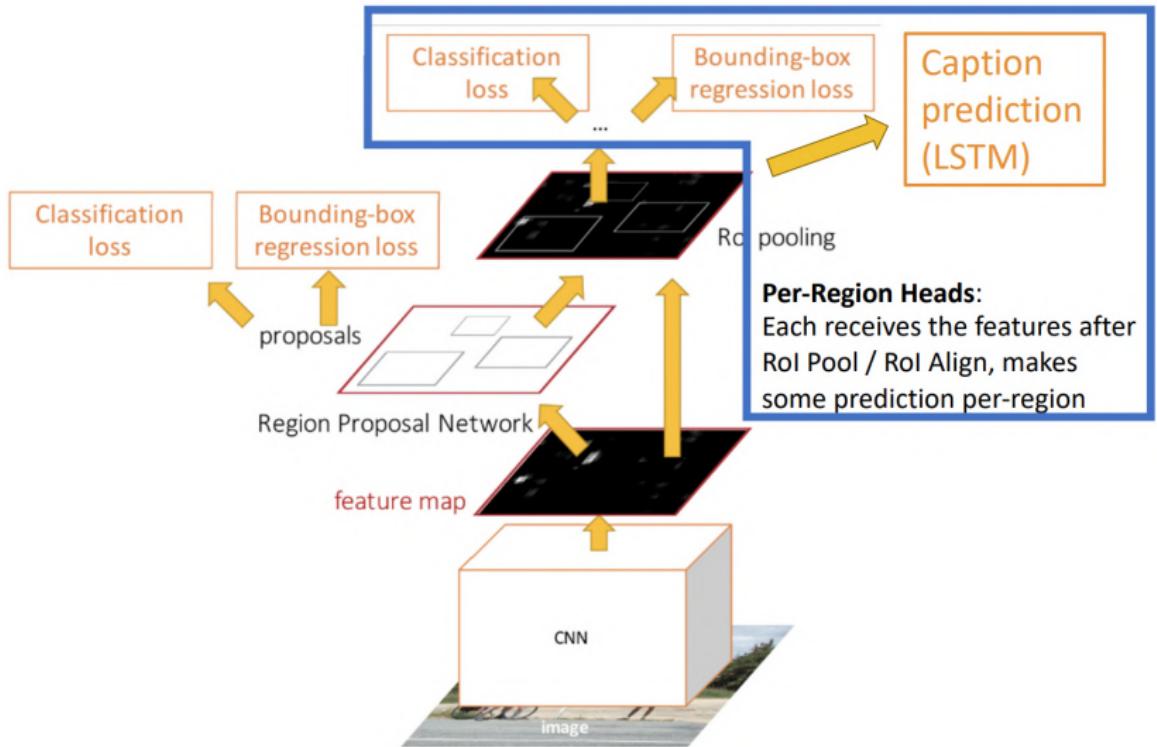
Mask R-CNN: Keypoint Estimation



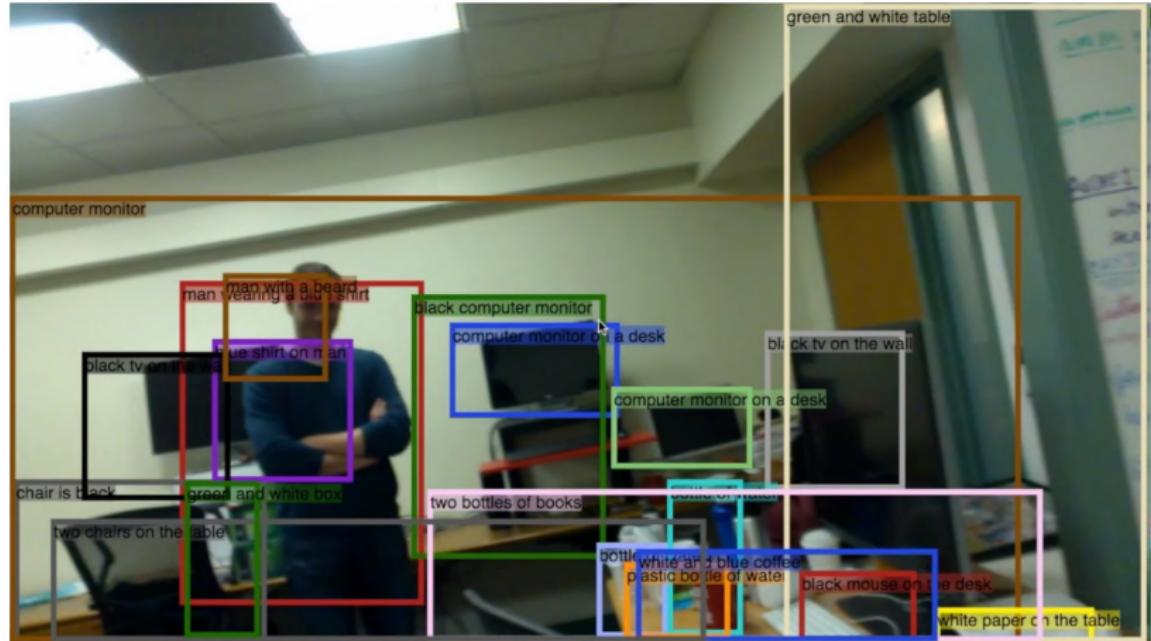
Joint Instance Segmentation and Pose Estimation



Captioning: Predict a caption per region!

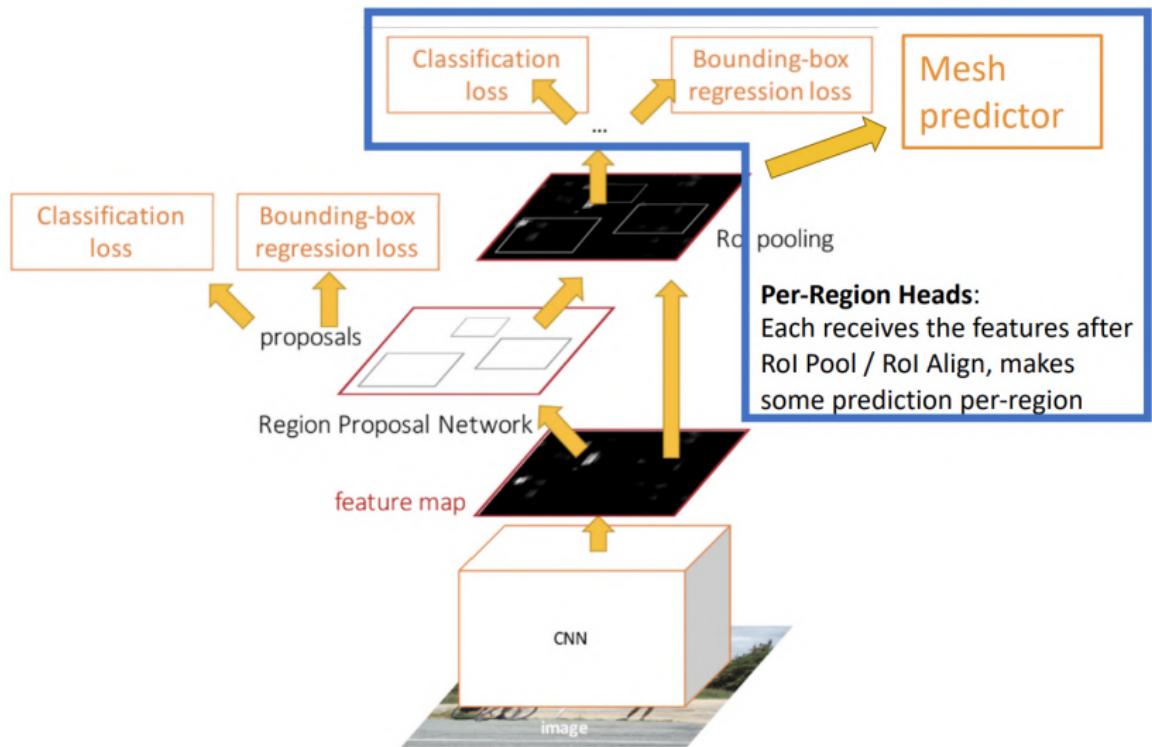


Captioning: Predict a caption per region!



Johnson, Karpathy, and Fei-Fei, "DenseCap: Fully Convolutional Localization Networks for Dense Captioning", CVPR 2016

3D Shape Prediction

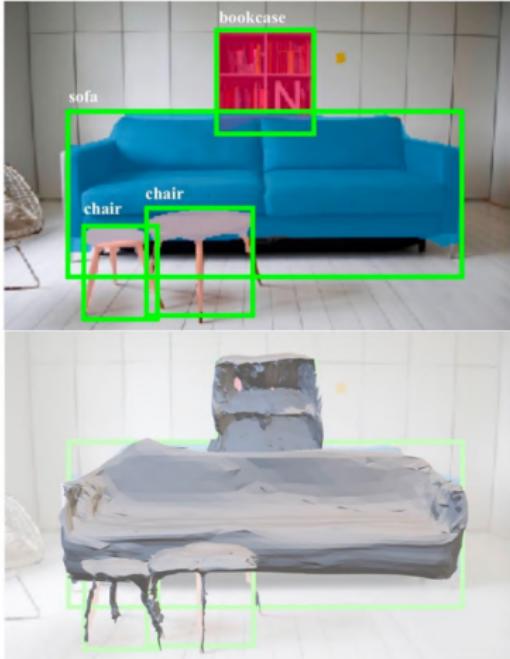


3D Shape Prediction

Mask R-CNN:
2D Image -> 2D shapes



Mesh R-CNN:
2D Image -> 3D shapes



Gkioxari, Malik, and Johnson, "Mesh R-CNN", ICCV 2019

These slides have been adapted from

- ▶ Fei-Fei Li, Yunzhu Li & Ruohan Gao, Stanford CS231n: Deep Learning for Computer Vision
- ▶ Assaf Shocher, Shai Bagon, Meirav Galun & Tali Dekel, WAIC DL4CV Deep Learning for Computer Vision: Fundamentals and Applications
- ▶ Justin Johnson, UMich EECS 498.008/598.008: Deep Learning for Computer Vision