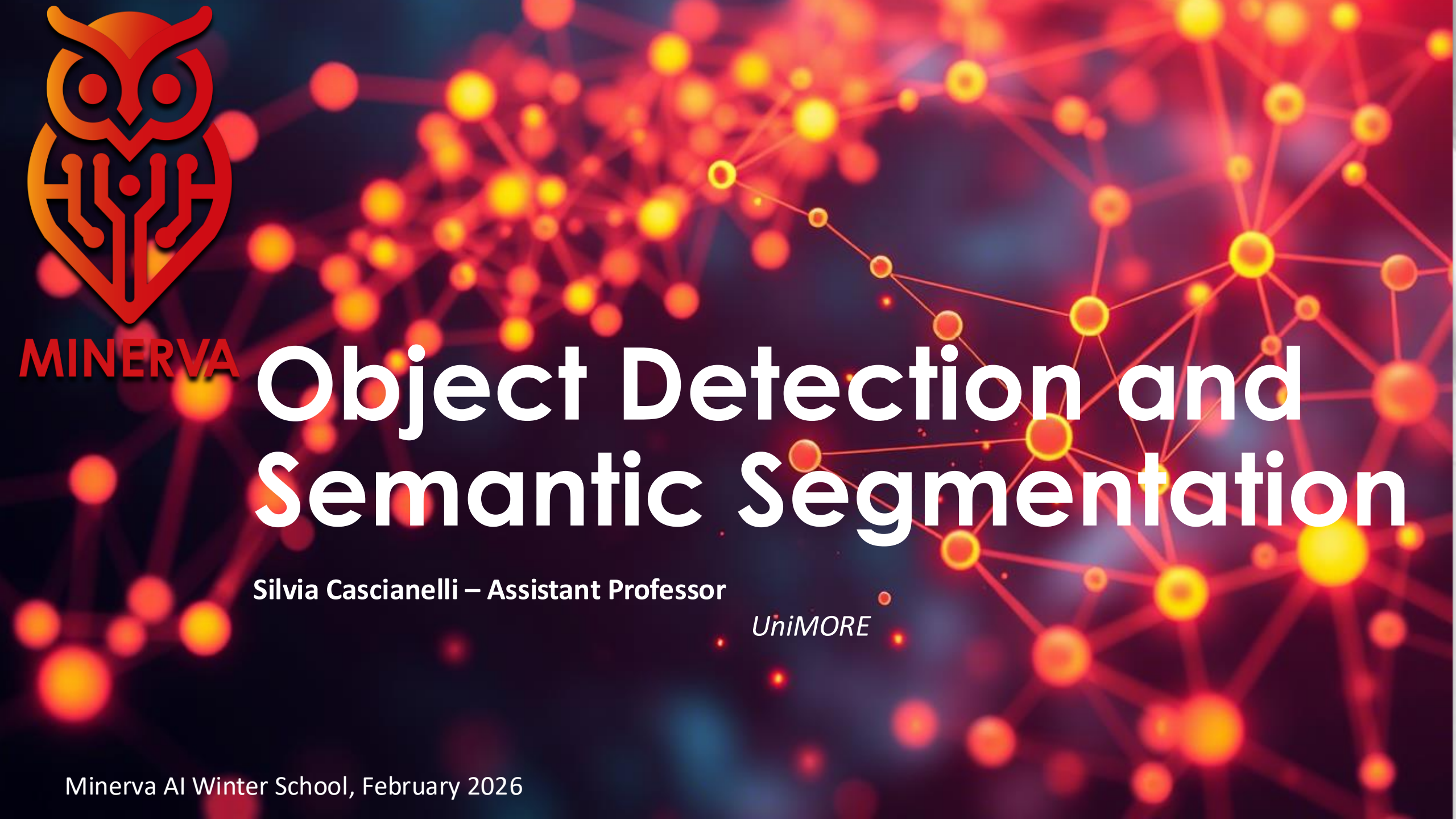


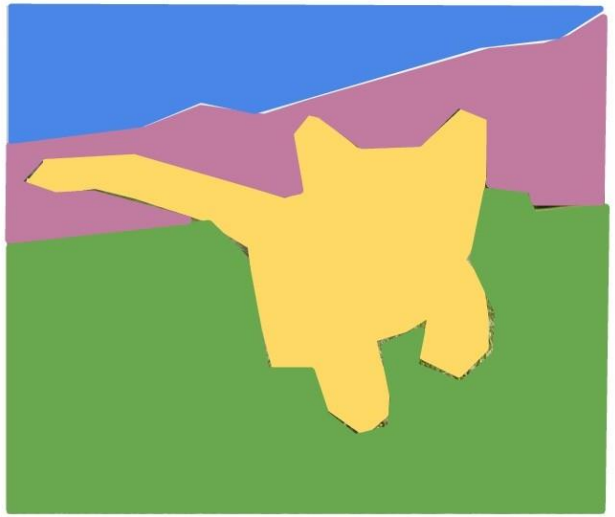
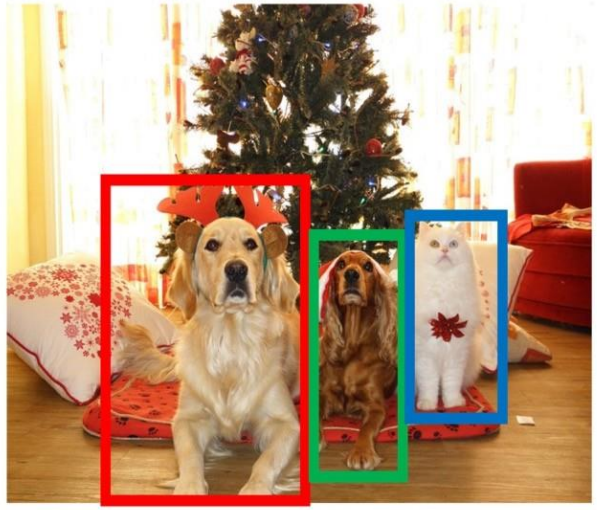
Object Detection and Semantic Segmentation

Silvia Cascianelli – Assistant Professor

UniMORE

Minerva AI Winter School, February 2026

A bit of Taxonomy

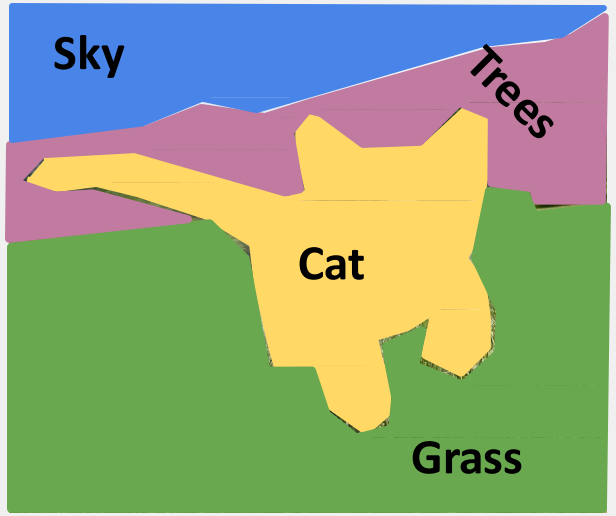
Classification	Semantic Segmentation	Object Detection	Instance Segmentation
			
CAT	GRASS, CAT, TREE, SKY	DOG, DOG, CAT	DOG, DOG, CAT
No spatial extent	No objects, just pixels	Multiple Objects	

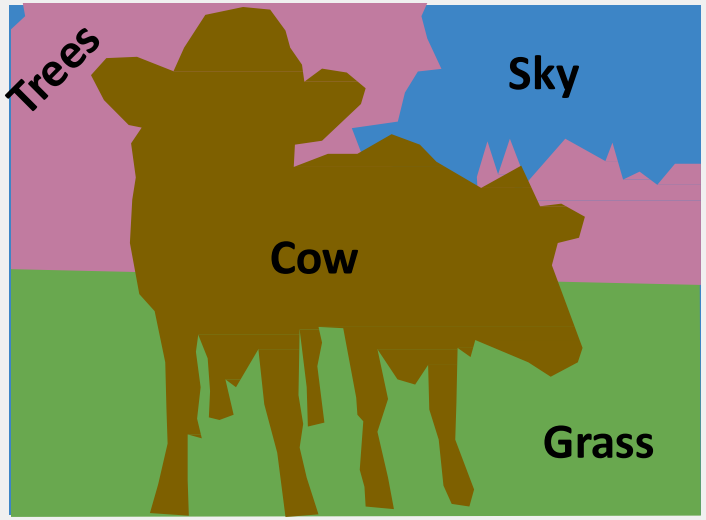
Semantic Segmentation: a definition

Input: Single image (can also be a video frame)

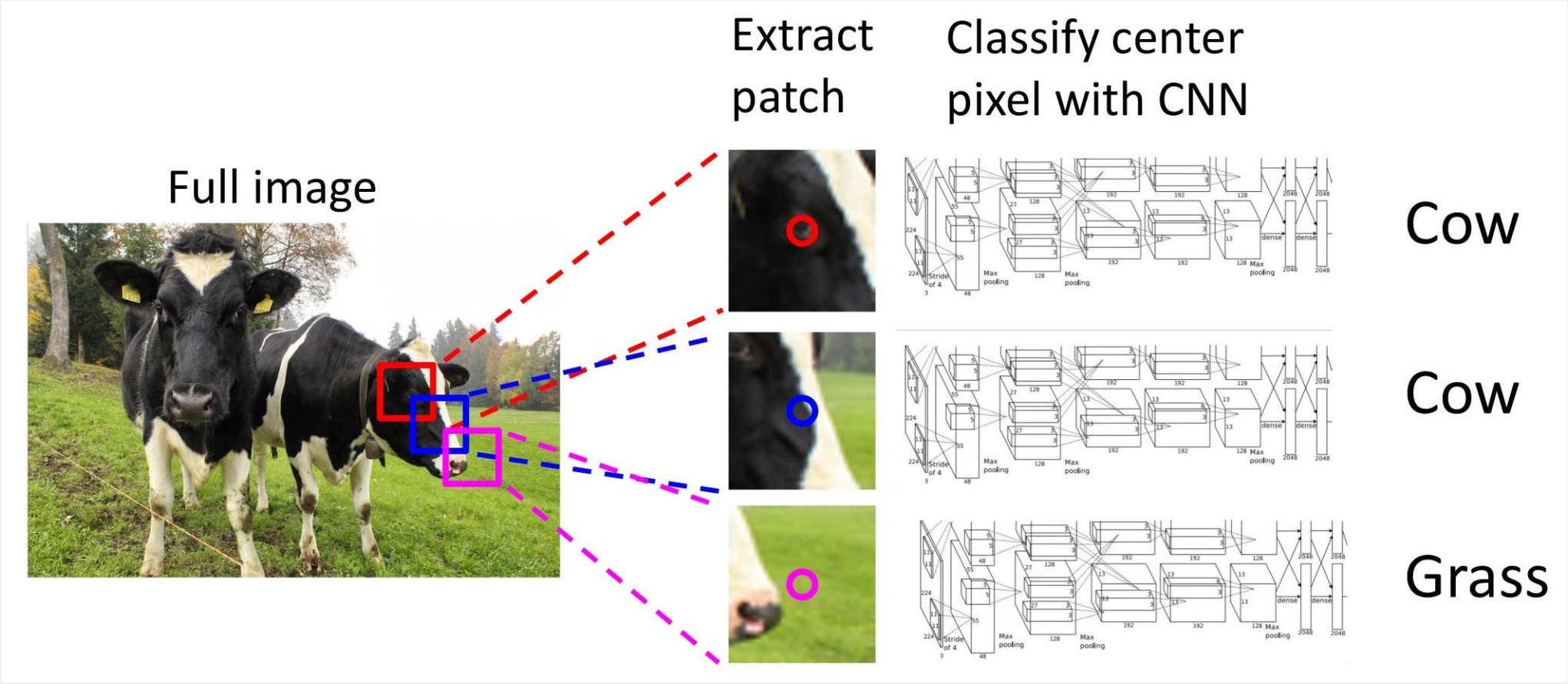
Output: a set of labels, one for each pixel in the image, assigning them to a category

Note that we do not want to differentiate instances, we only care about pixels





Semantic Segmentation with sliding windows

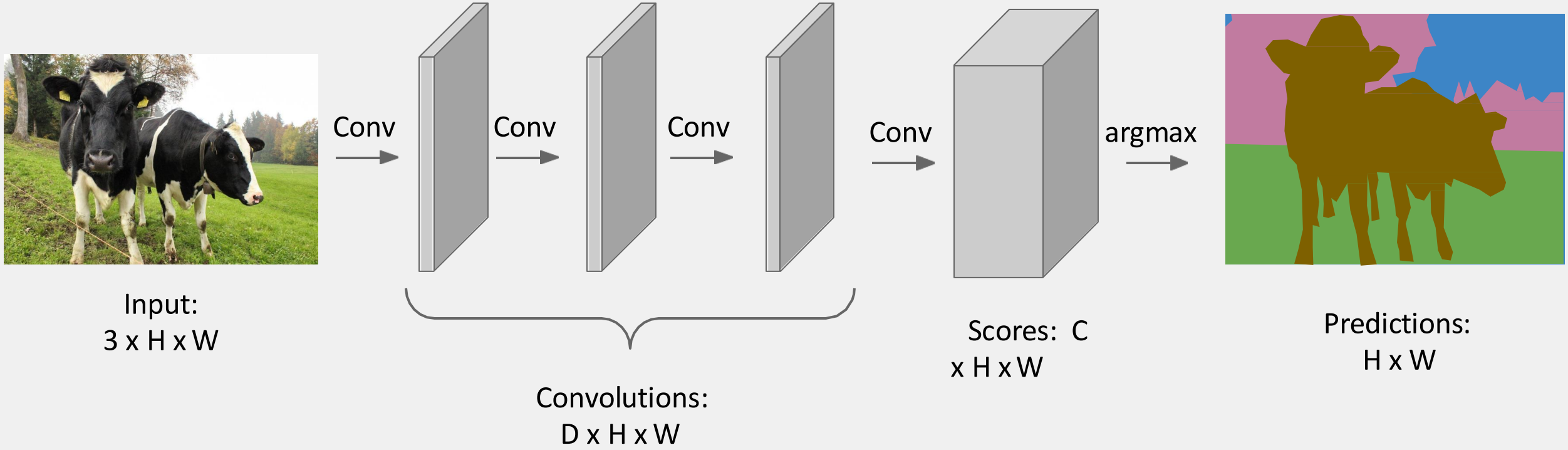


Not reusing shared features between overlapping patches → highly inefficient

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

Semantic Segmentation in one go

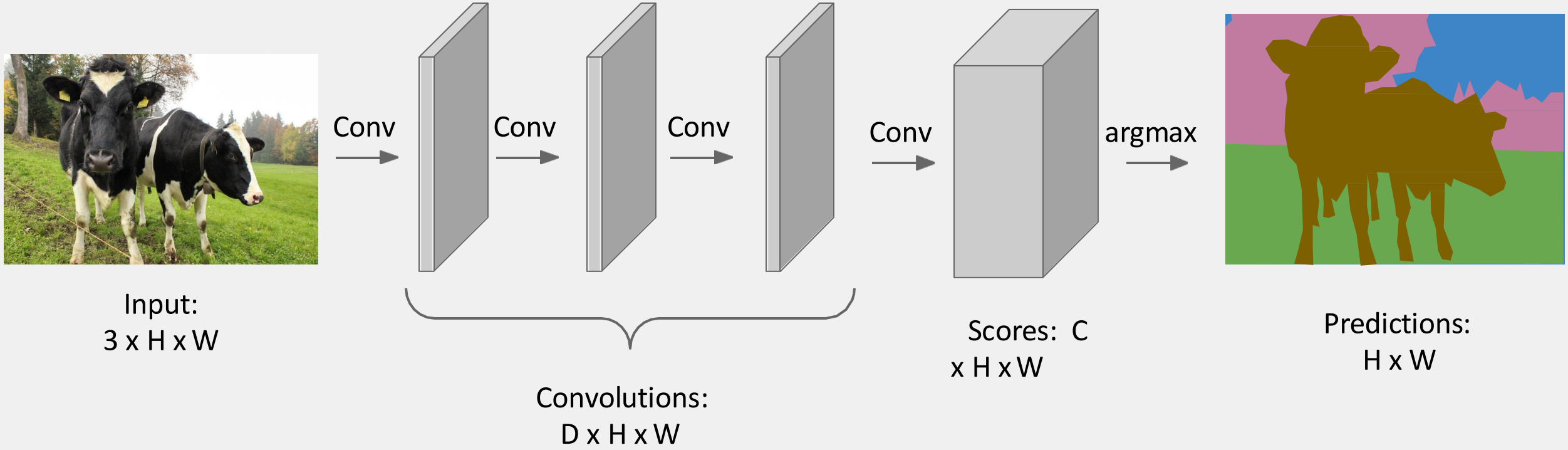
Design a network as a bunch of convolutional layers to make predictions for pixels all at once!



Semantic Segmentation in one go

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

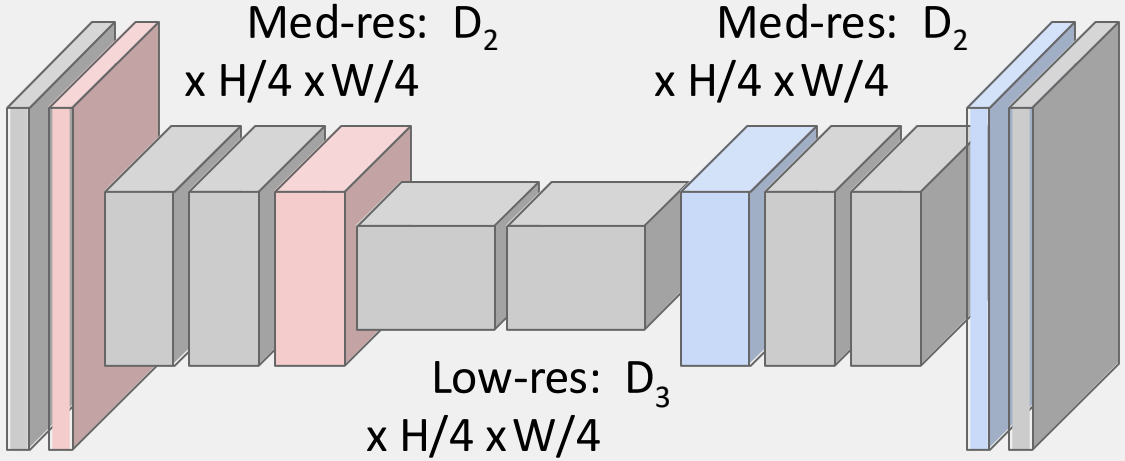
Problem: convolutions at original image resolution will be very expensive …



Semantic Segmentation in one go

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

Input:
3 x H x W



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

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

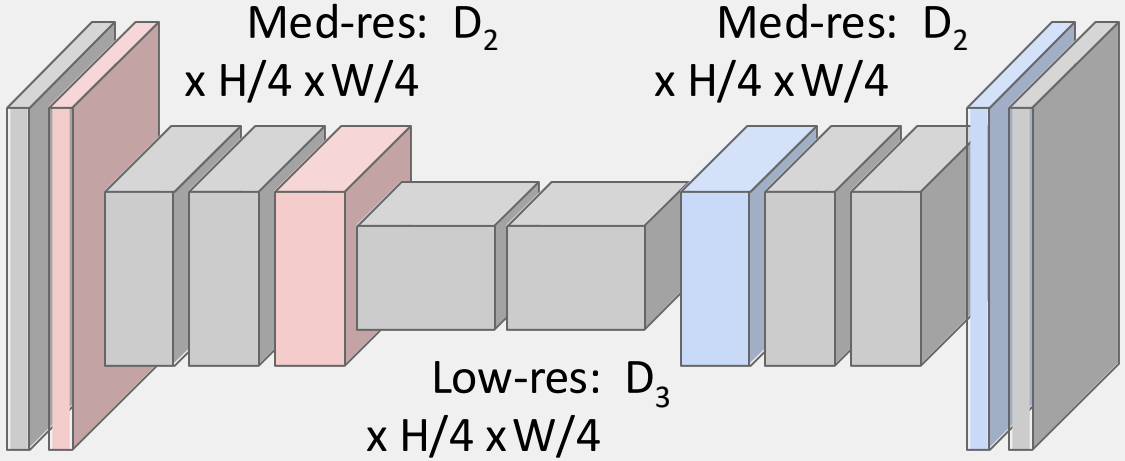
Predictions:
H x W

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

Semantic Segmentation in one go

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

Input:
3 x H x W



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

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

Predictions:
H x W

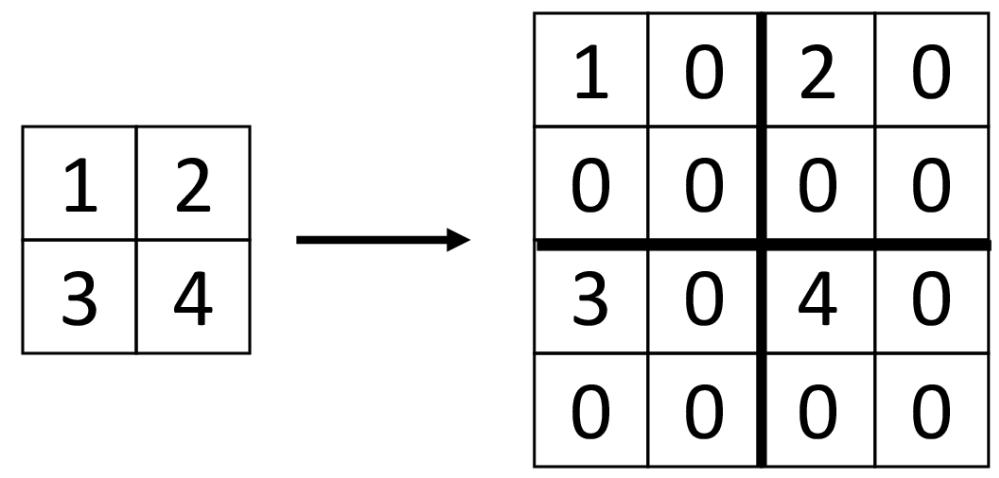
Downsampling:
Pooling, strided convolution

Upsampling:
???

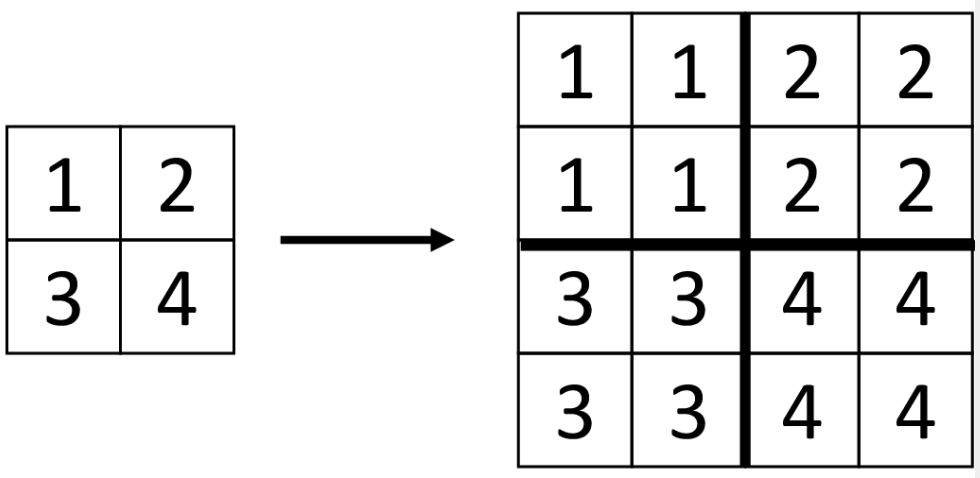
Long, Shelhamer, and Darrell, "Fully Convolutional Networks for Semantic Segmentation", CVPR 2015
Noh et al, "Learning Deconvolution Network for Semantic Segmentation", ICCV 2015

Semantic Segmentation in one go

Upsampling with Unpooling



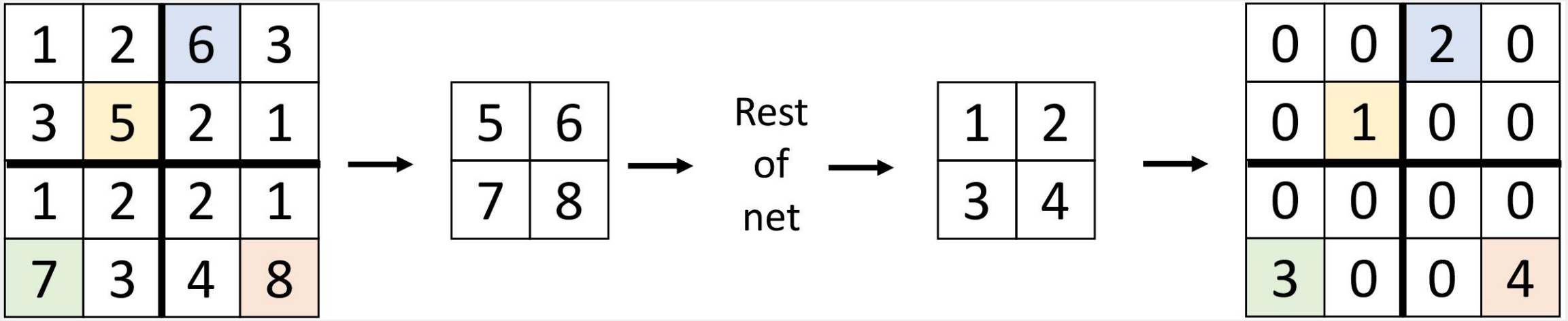
Bed of Nails Unpooling



Nearest Neighbor Unpooling (instead of copying, we can also interpolate via **Bilinear** of **Bicubic** interpolation)

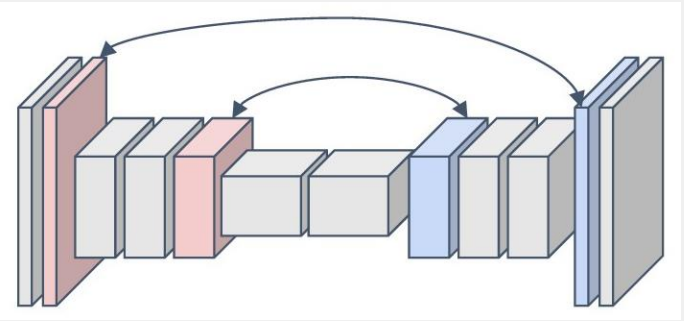
Semantic Segmentation in one go

Upsampling with Unpooling



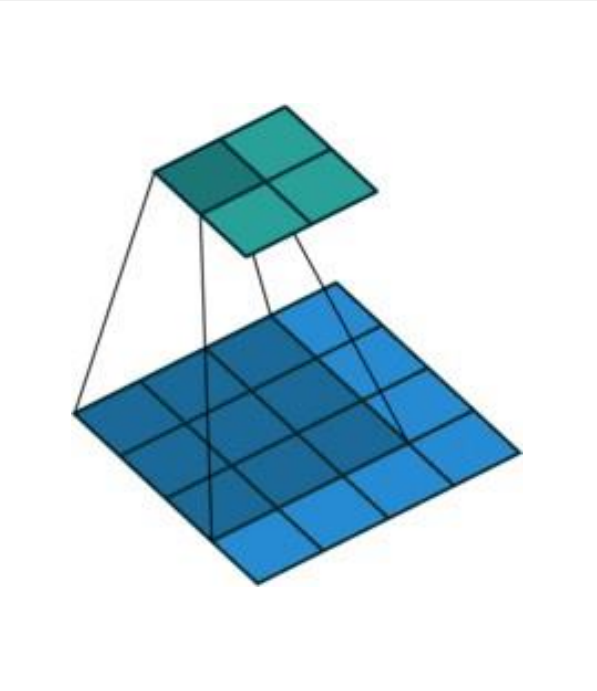
Max Unpooling:

- Pair each downsampling layer with a corresponding upsampling layer
- Remember where the max was in the downsampling filter
- Place the new values in the same position

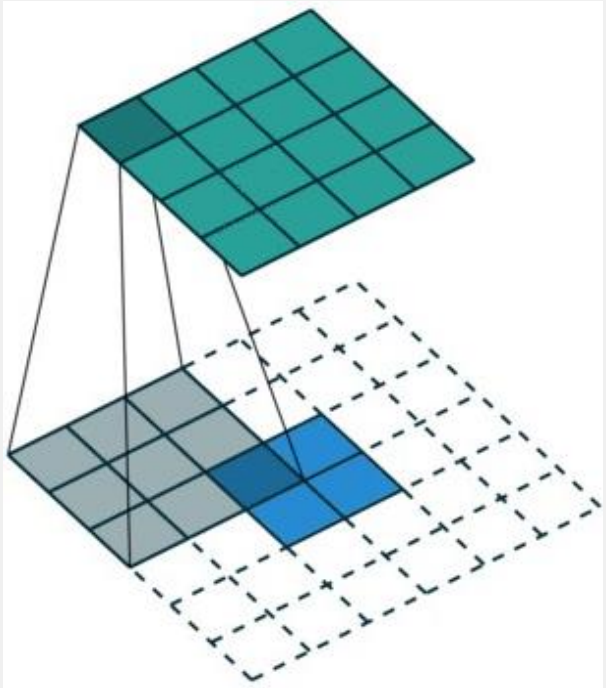


Semantic Segmentation in one go

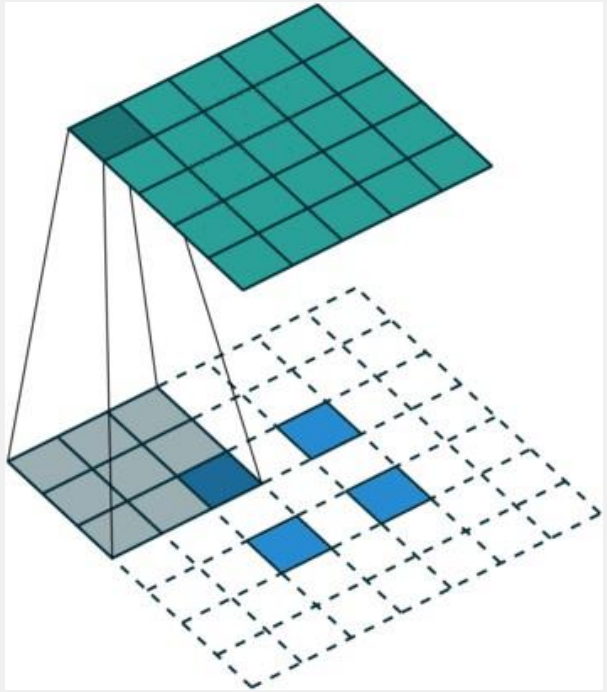
Upsampling with Transpose Convolution



Regular Convolution



Transposed convolution:
Apply a learnable kernel to each input pixel.
Each pixel generates a weighted output patch.
When patches overlap, their values are summed.

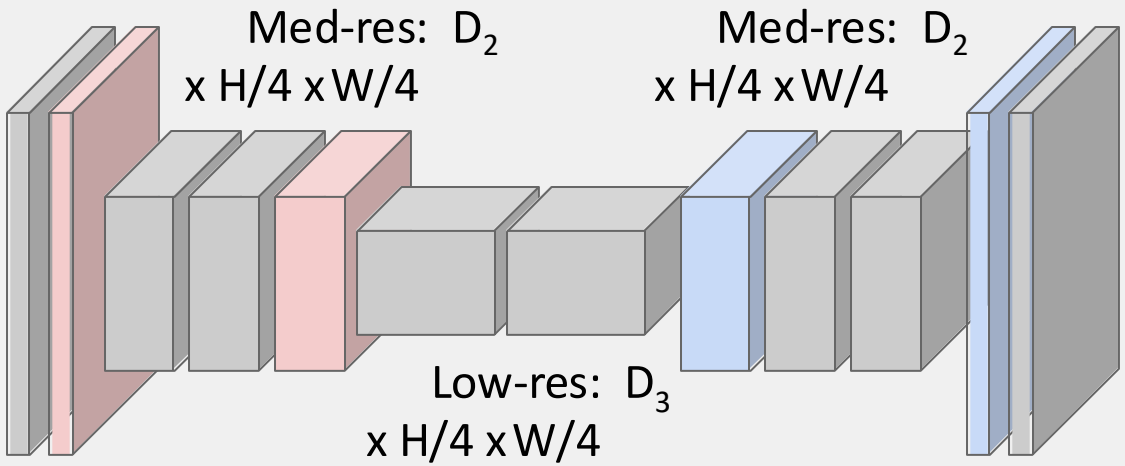


Strided transpose convolution:
With stride > 1, the output patches are placed farther apart, increasing the spatial resolution

Semantic Segmentation in one go

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

Input:
3 x H x W



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

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

Predictions:
H x W

Downsampling:
Pooling, strided convolution

Upsampling:
Unpooling (non learnable), strided transpose convolution (learnable)

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

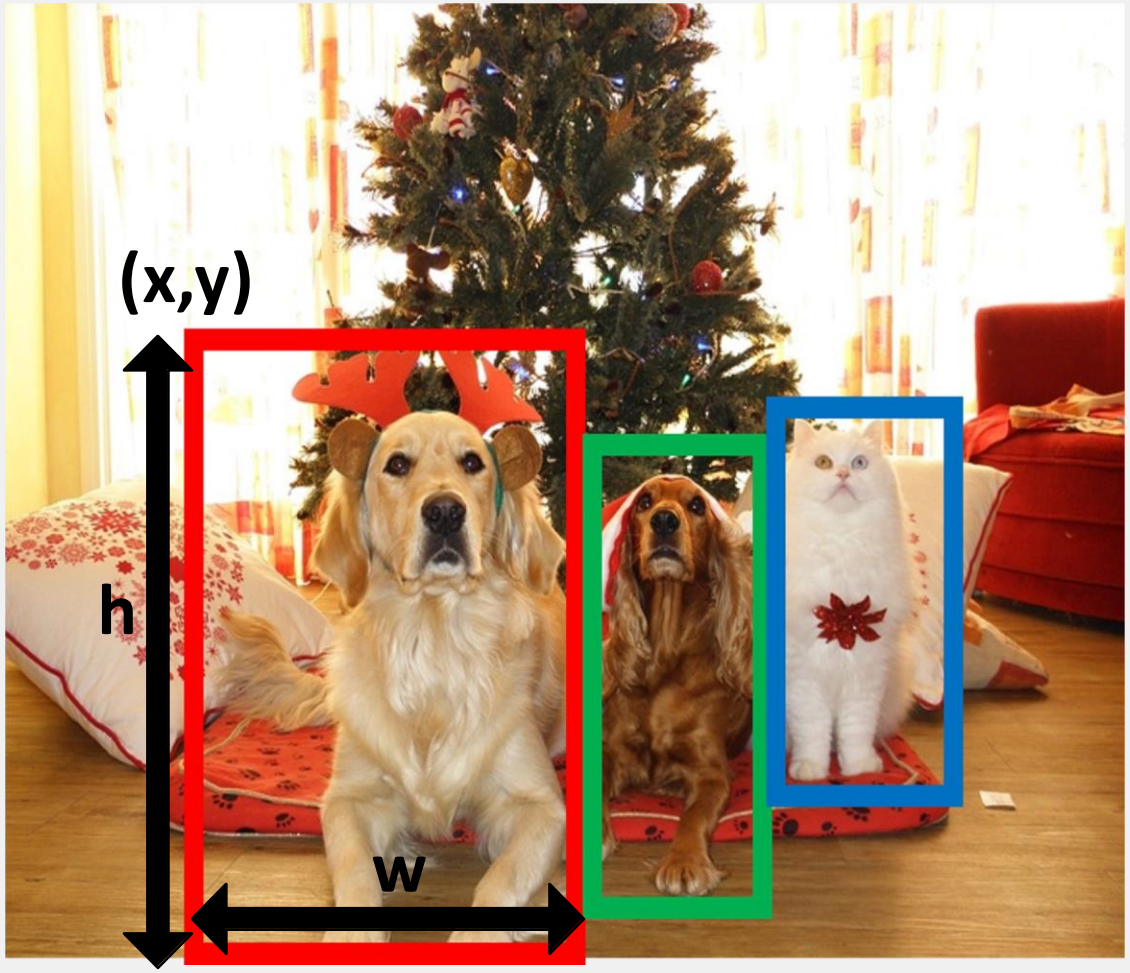
Object Detection: a definition

Input: Single image (can also be a video frame)

Output: A set of detected objects;

For each object predict:

1. Category label (from a fixed, known set of categories – but some models relax this constraint)
2. Bounding box (identified via e.g., coodinates and extent: <x, y, width, height>)
3. Confidence score (this is a currently-usual nice to have)



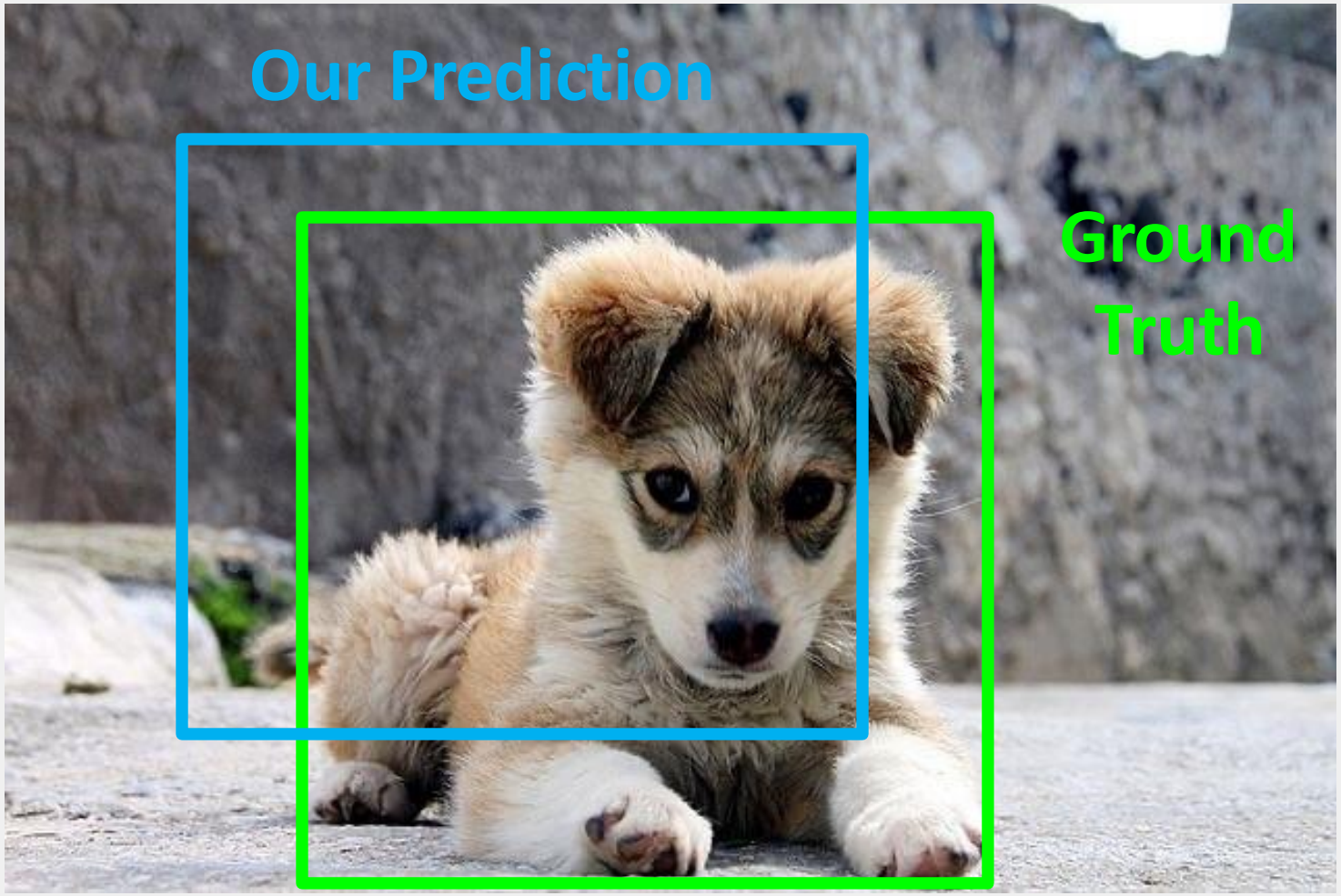
Object Detection: a definition

A few notes on the bounding boxes:

- Bounding boxes are typically axis-aligned, oriented boxes are much less common
- Bounding boxes can also be bounding cubes in 3D object detection
- In modal detection, bounding boxes (usually) cover only the visible portion of the object
- In amodal detection, box covers the entire extent of the object, even occluded parts

How's my model doing?

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

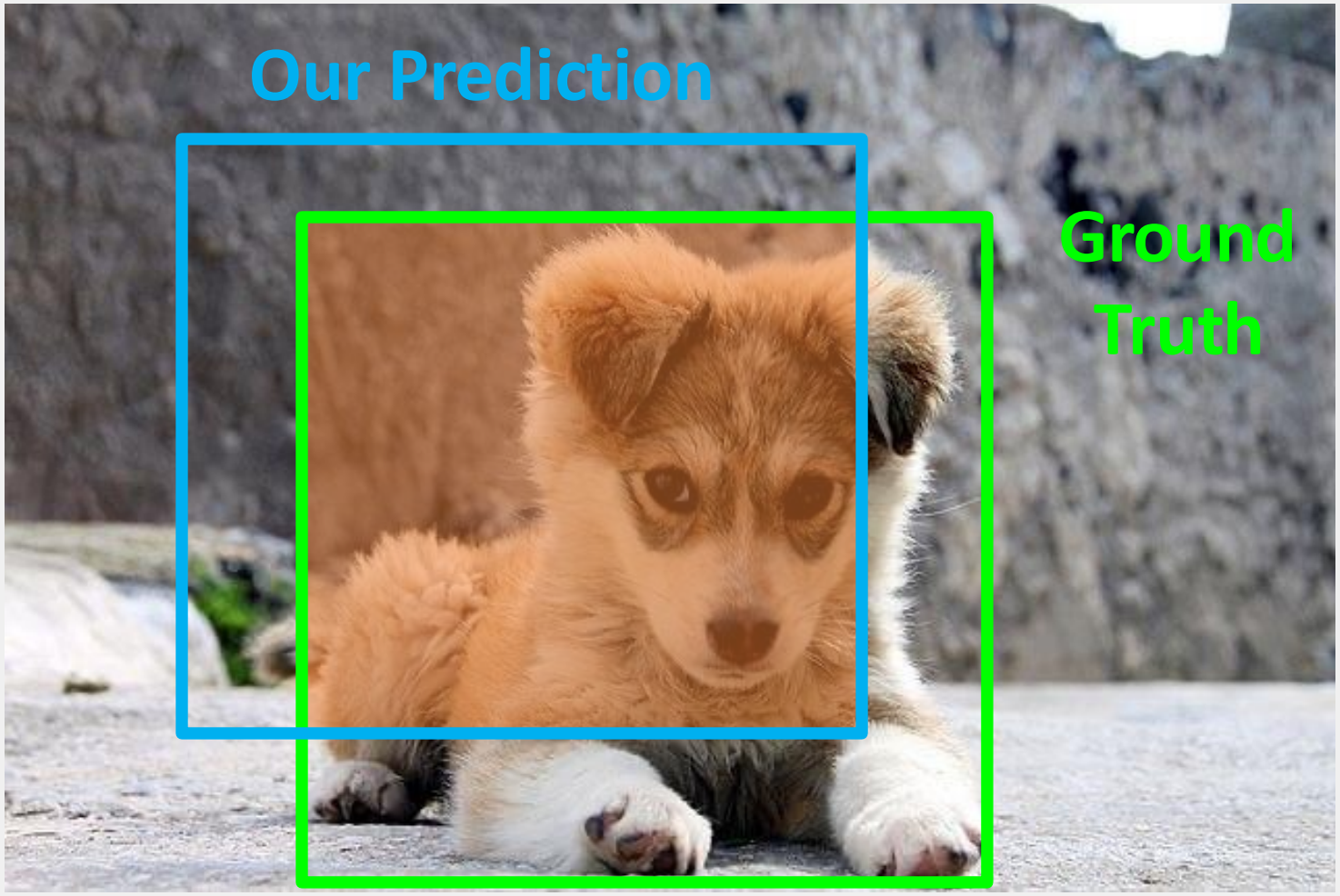


How's my model doing?

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

Intersection over Union (IoU) (Also called "Jaccard similarity" or "Jaccard index"):

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

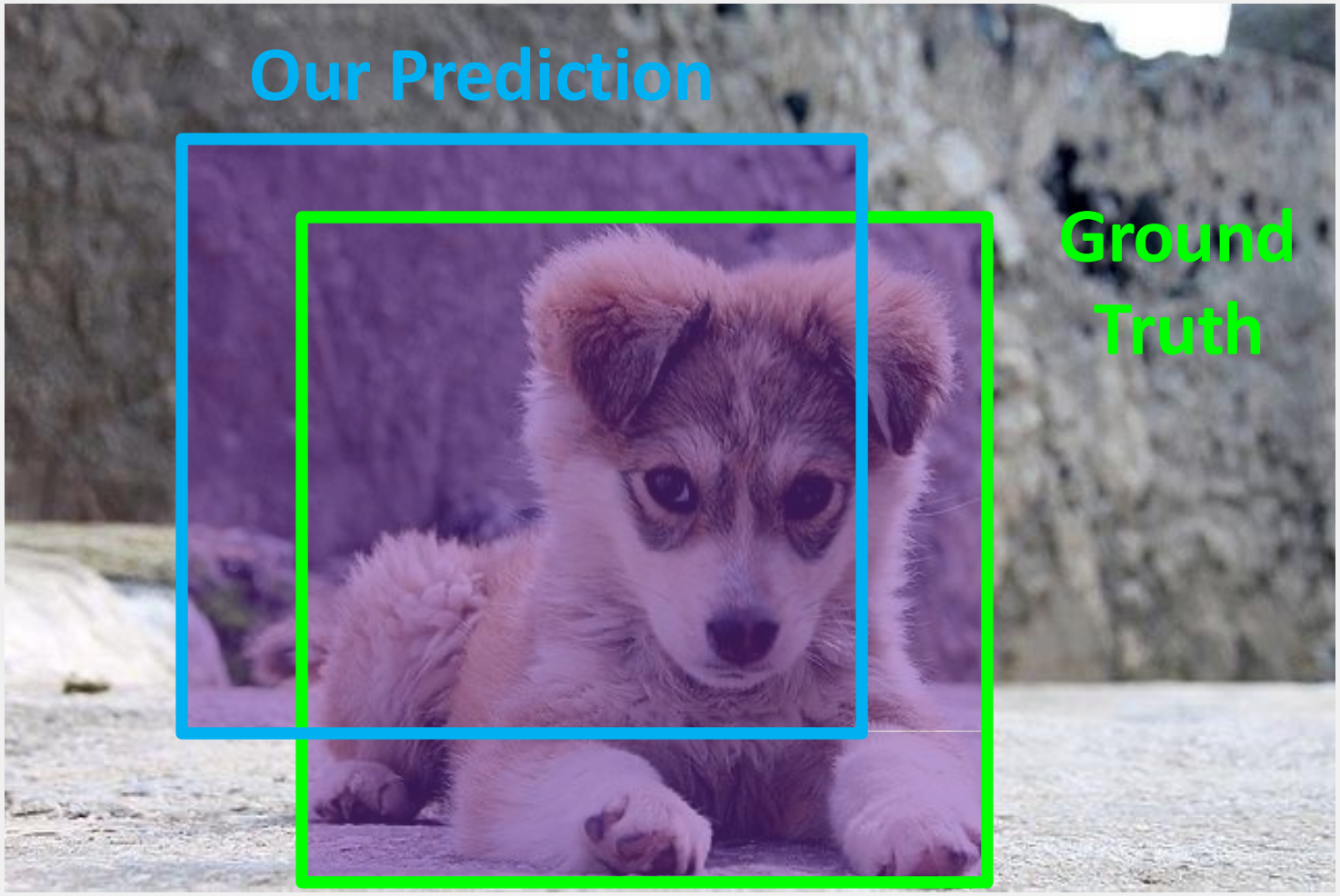


How's my model doing?

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

Intersection over Union (IoU) (Also called "Jaccard similarity" or "Jaccard index"):

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



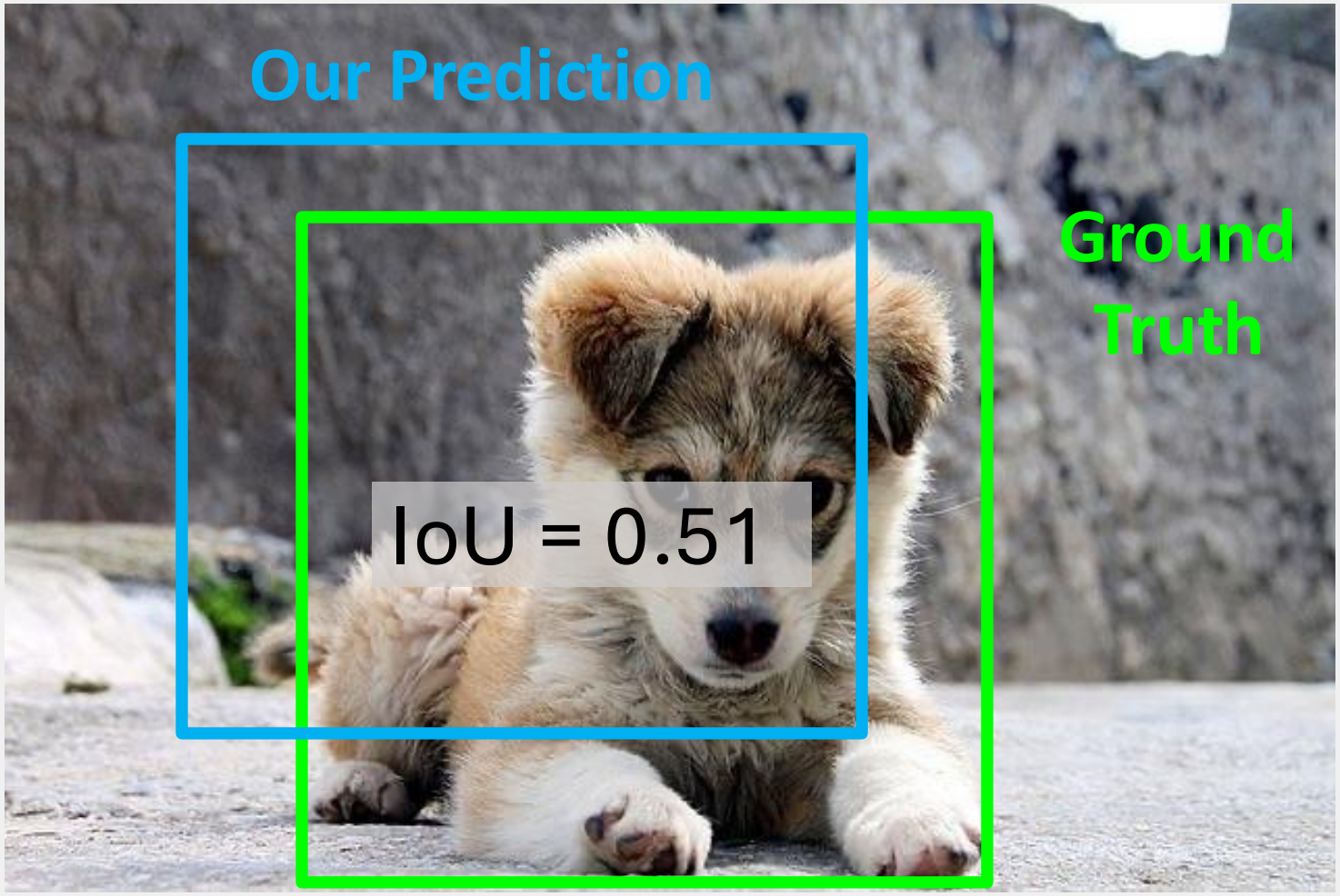
How's my model doing?

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

Intersection over Union (IoU) (Also called "Jaccard similarity" or "Jaccard index"):

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

IoU > 0.5 is "decent"



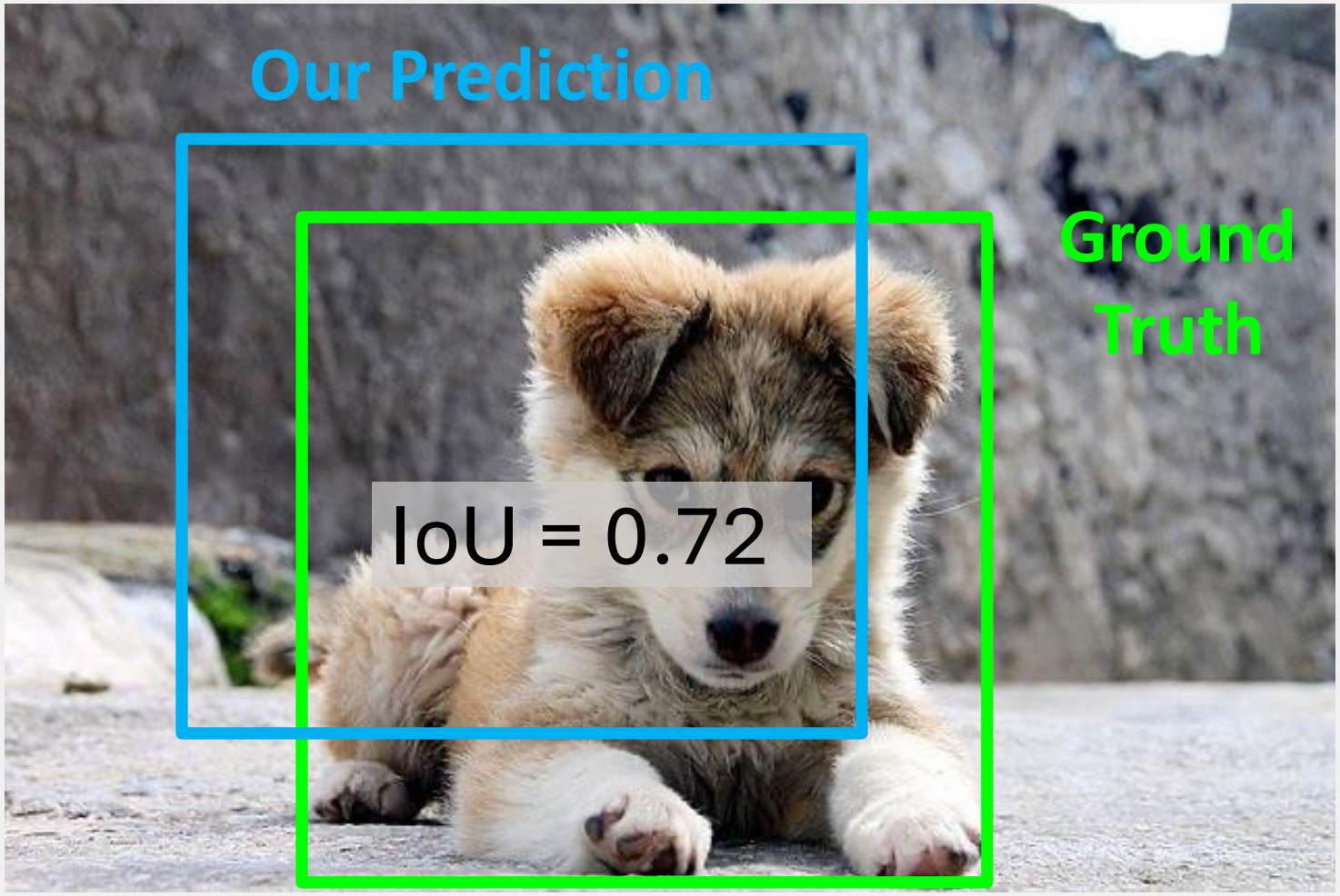
How's my model doing?

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

Intersection over Union (IoU) (Also called "Jaccard similarity" or "Jaccard index"):

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

IoU > 0.5 is "decent",
IoU > 0.7 is "pretty good",



How's my model doing?

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

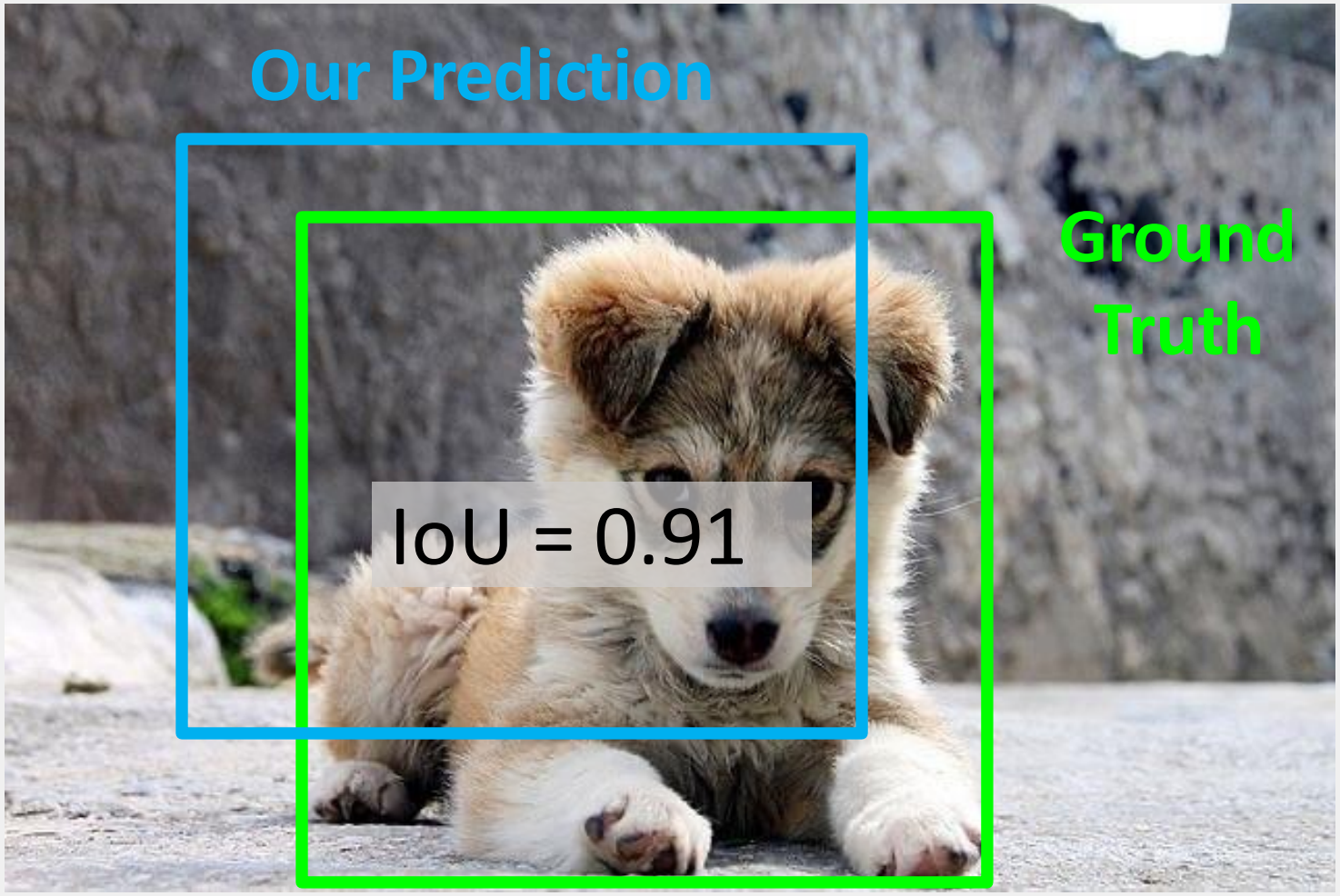
Intersection over Union (IoU) (Also called "Jaccard similarity" or "Jaccard index"):

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

IoU > 0.5 is "decent",

IoU > 0.7 is "pretty good",

IoU > 0.9 is "almost perfect"

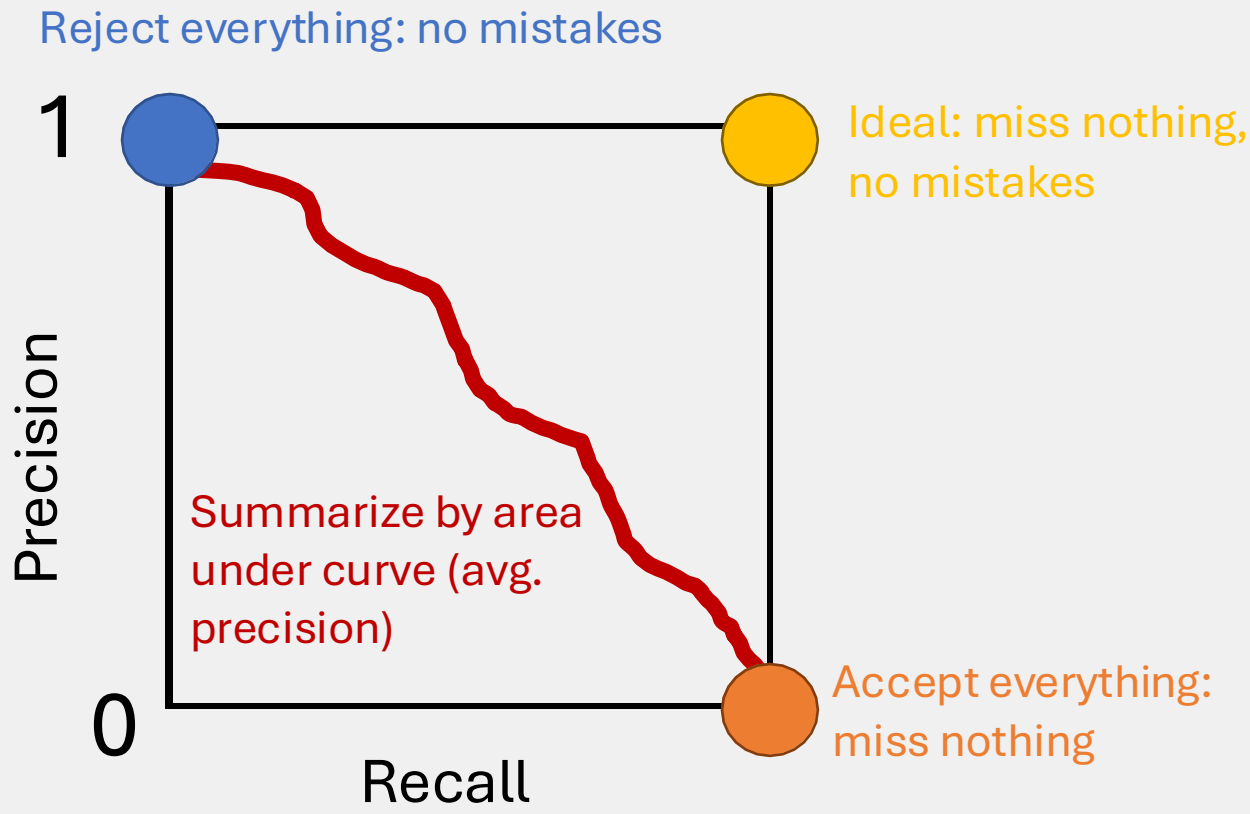


How's my model doing?

- True detection: high intersection over union based on a threshold
- Precision:

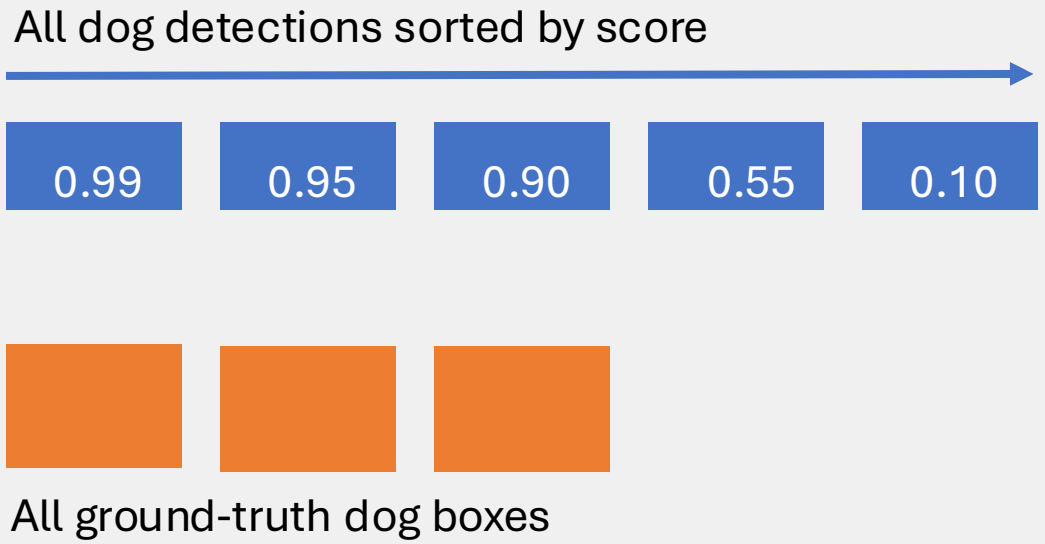
  #true detections / #detections
- Recall:

  #true detections / #true positives



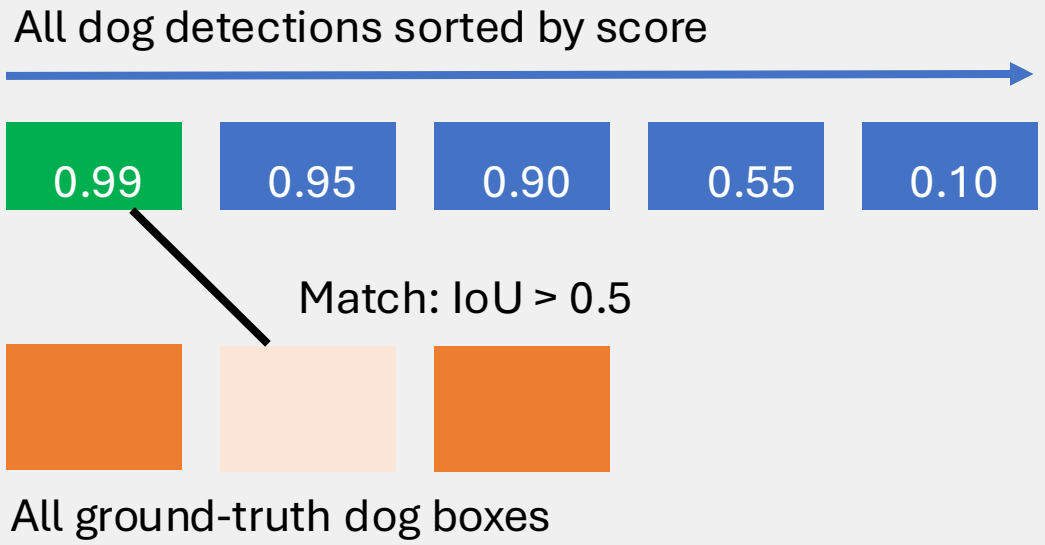
How’s my model doing?

1. For each category, compute Average Precision (AP)
   = area under Precision vs Recall Curve



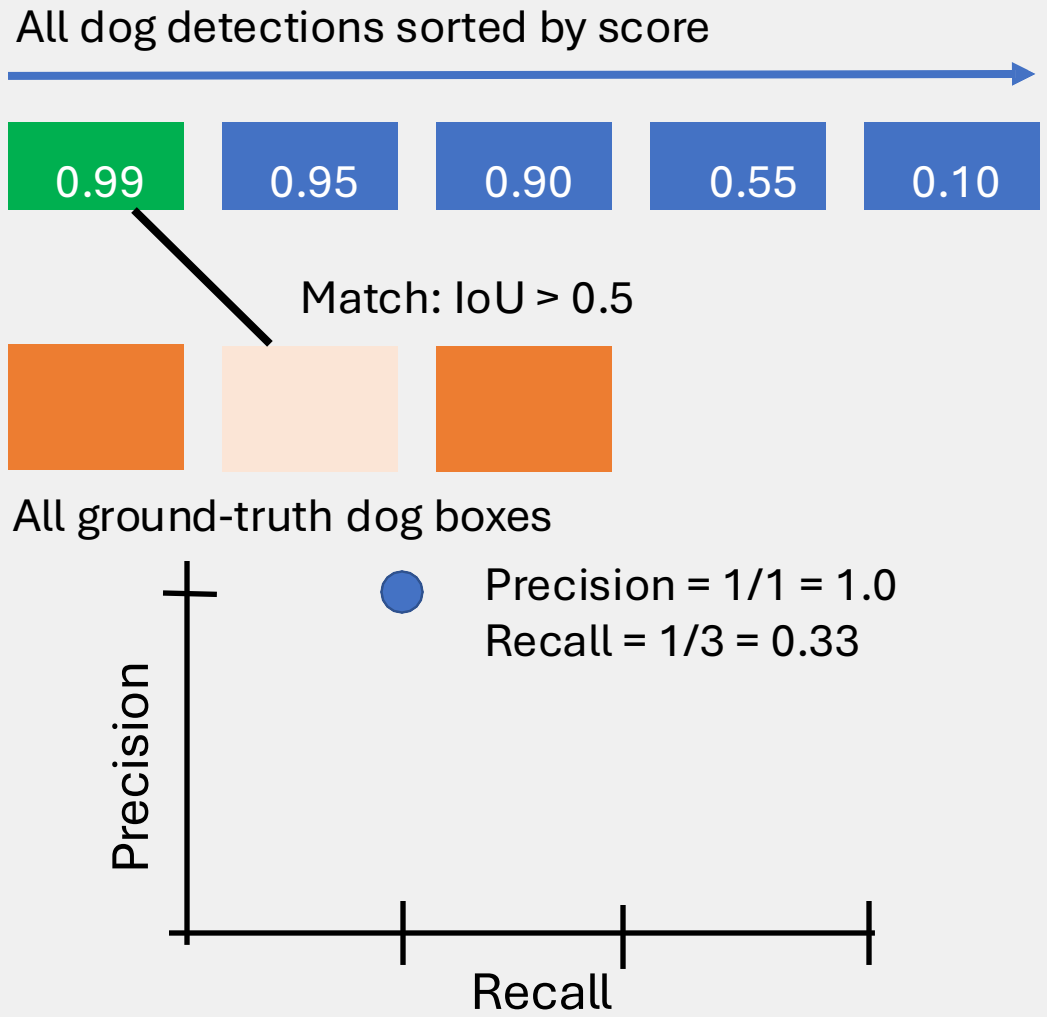
How's my model doing?

1. For each category, compute Average Precision (AP) = area under Precision vs Recall Curve
    1. For each detection (highest score to lowest score):
        1. If it matches some GT box with IoU > 0.5, mark it as positive and eliminate the GT
        2. Otherwise mark it as negative



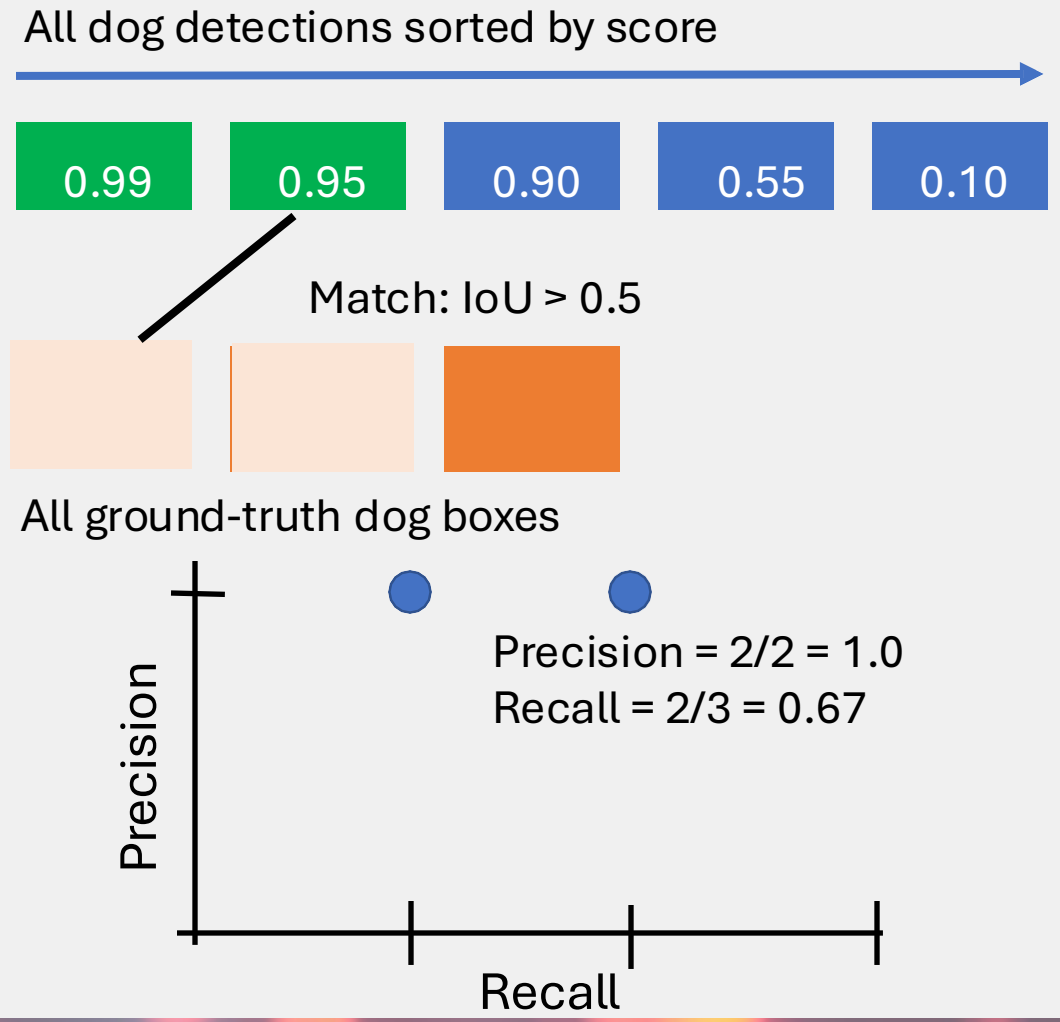
How's my model doing?

1. For each category, compute Average Precision (AP) = area under Precision vs Recall Curve
    1. For each detection (highest score to lowest score):
        1. If it matches some GT box with IoU > 0.5, mark it as positive and eliminate the GT
        2. Otherwise mark it as negative
        3. Plot a point on PR Curve



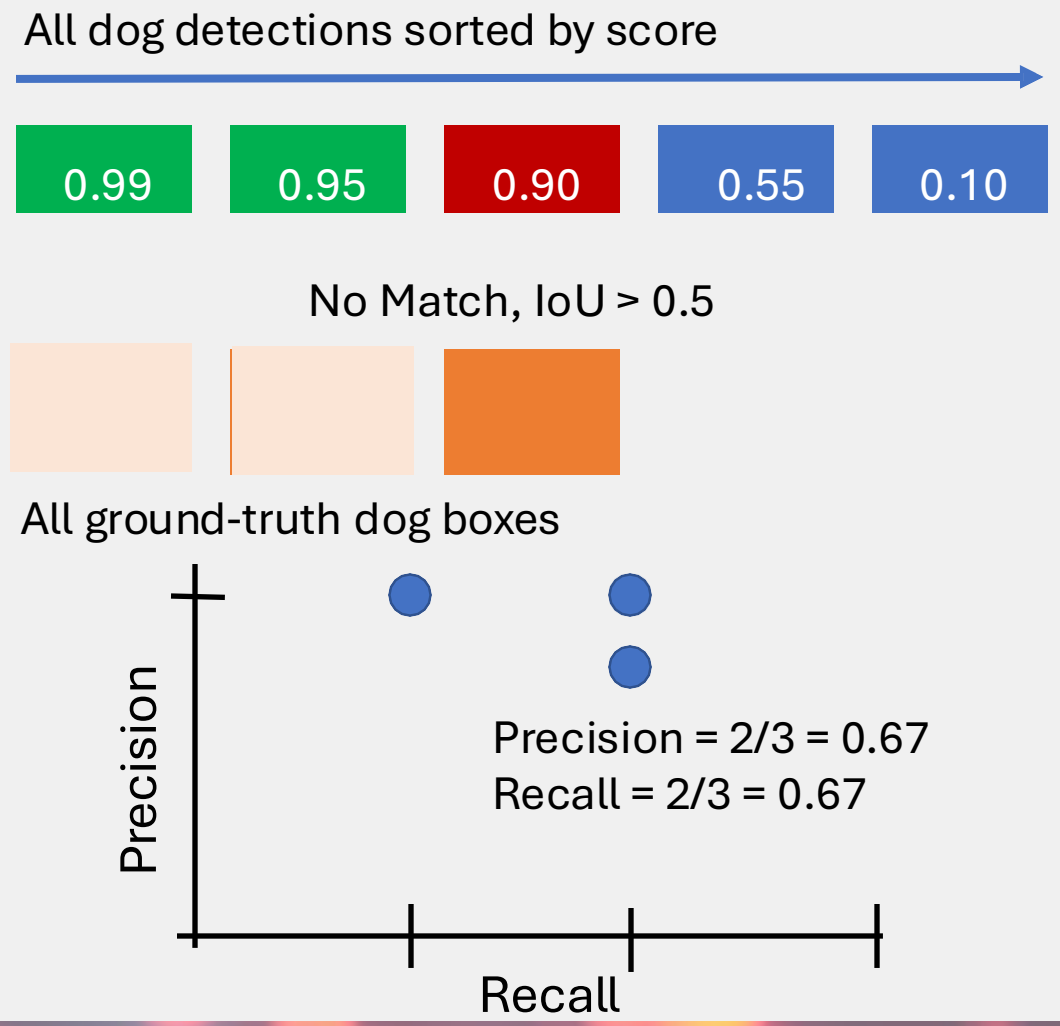
How's my model doing?

1. For each category, compute Average Precision (AP) = area under Precision vs Recall Curve
    1. For each detection (highest score to lowest score):
        1. If it matches some GT box with IoU > 0.5, mark it as positive and eliminate the GT
        2. Otherwise mark it as negative
        3. Plot a point on PR Curve



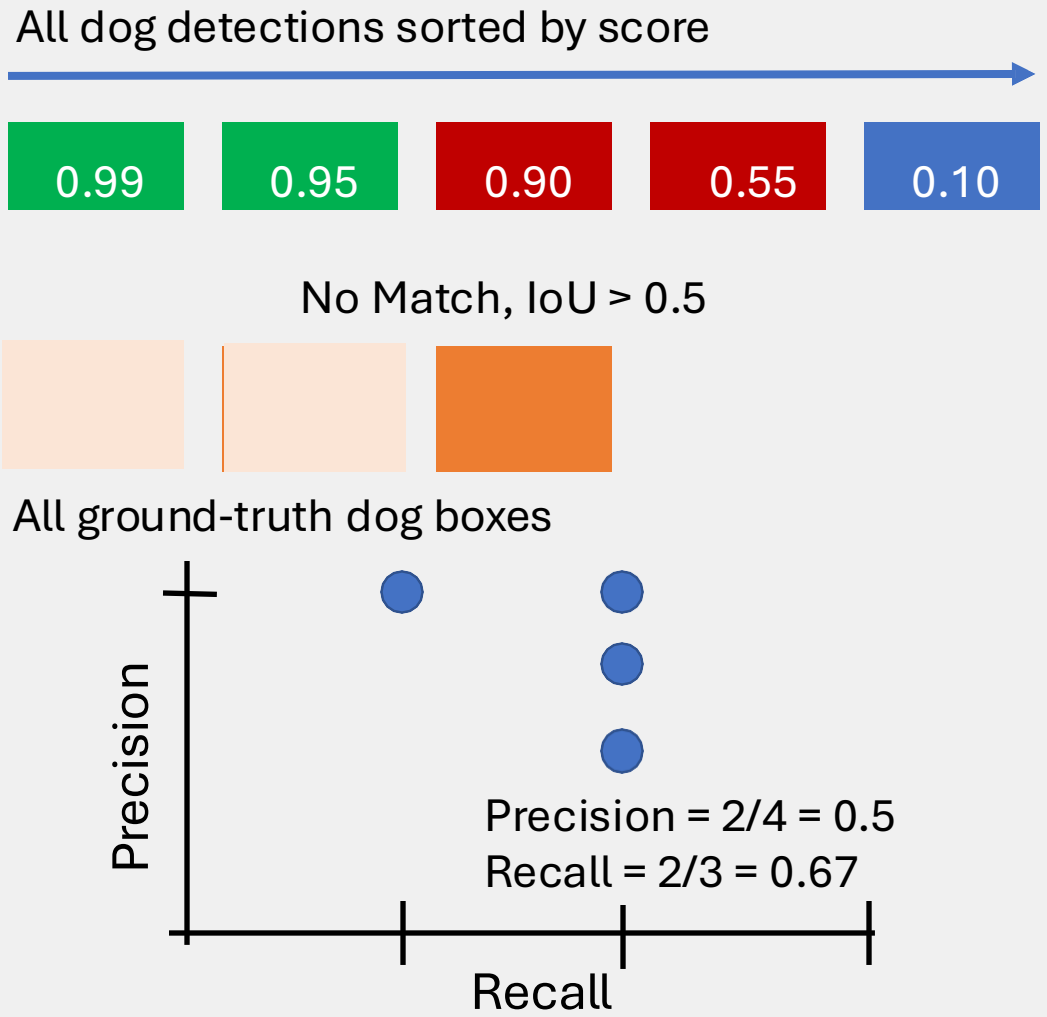
How's my model doing?

1. For each category, compute Average Precision (AP) = area under Precision vs Recall Curve
    1. For each detection (highest score to lowest score):
        1. If it matches some GT box with IoU > 0.5, mark it as positive and eliminate the GT
        2. Otherwise mark it as negative
        3. Plot a point on PR Curve



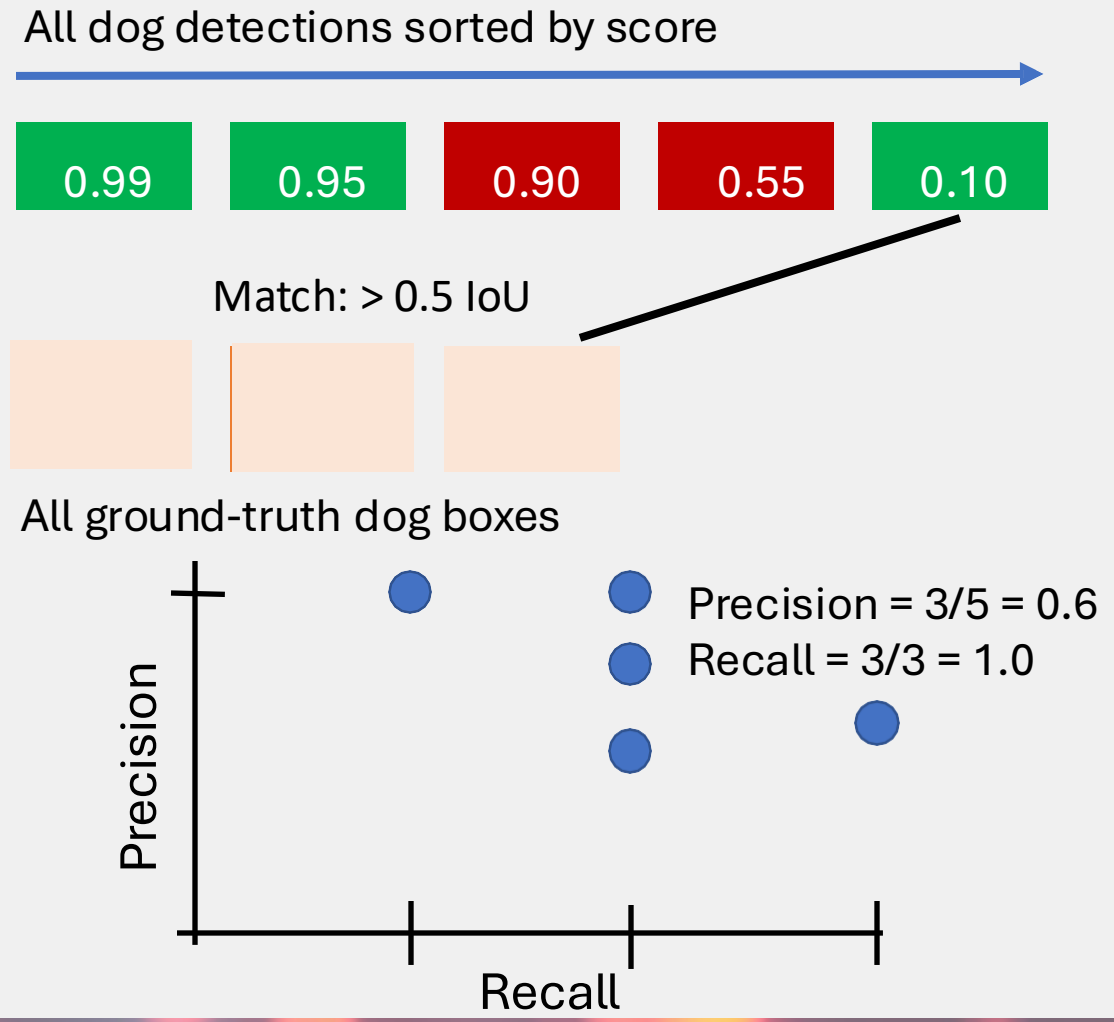
How's my model doing?

1. For each category, compute Average Precision (AP) = area under Precision vs Recall Curve
    1. For each detection (highest score to lowest score):
        1. If it matches some GT box with IoU > 0.5, mark it as positive and eliminate the GT
        2. Otherwise mark it as negative
        3. Plot a point on PR Curve



How's my model doing?

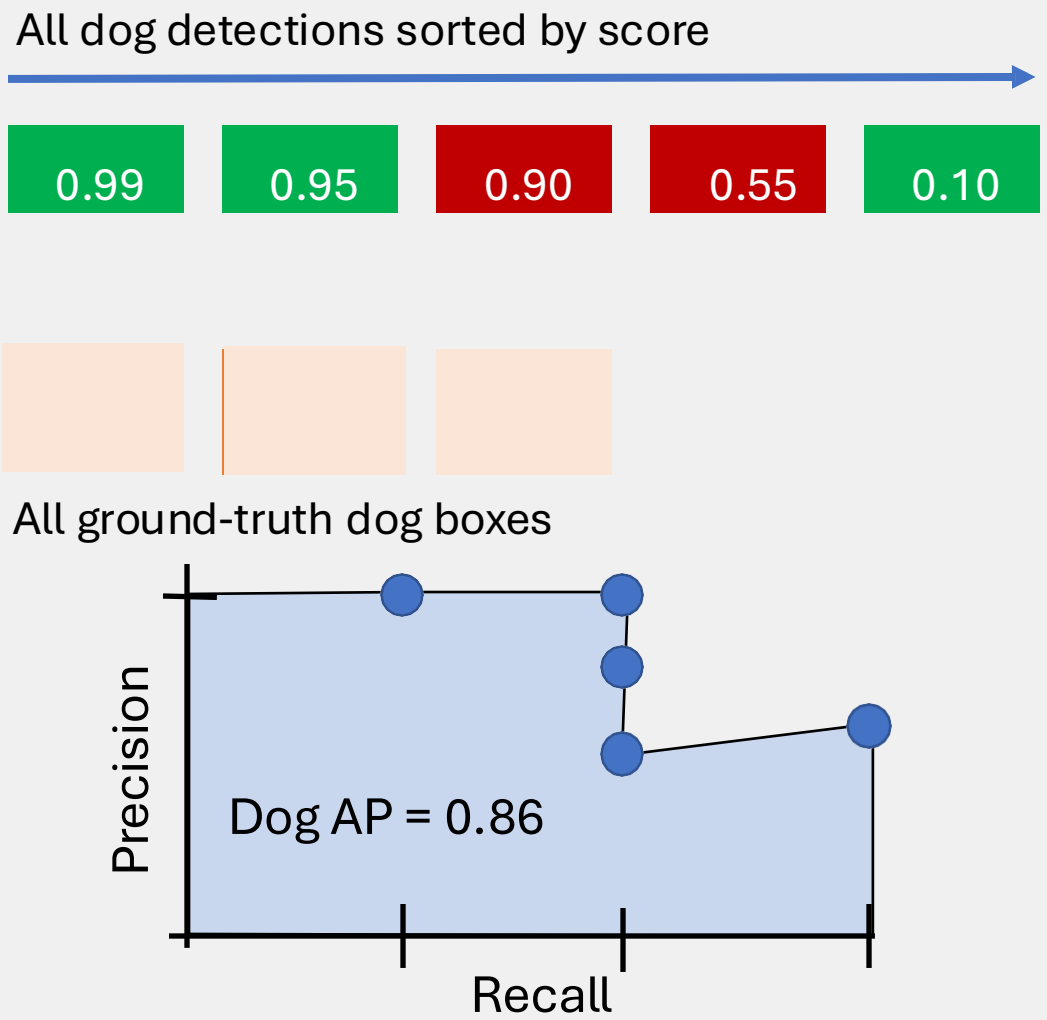
1. For each category, compute Average Precision (AP) = area under Precision vs Recall Curve
    1. For each detection (highest score to lowest score):
        1. If it matches some GT box with IoU > 0.5, mark it as positive and eliminate the GT
        2. Otherwise mark it as negative
        3. Plot a point on PR Curve



How's my model doing?

1. For each category, compute Average Precision (AP) = area under Precision vs Recall Curve
    1. For each detection (highest score to lowest score):
        1. If it matches some GT box with IoU > 0.5, mark it as positive and eliminate the GT
        2. Otherwise mark it as negative
        3. Plot a point on PR Curve
2. Average Precision (AP) = area under PR curve

How to get AP = 1.0:
Hit all GT boxes with IoU > 0.5, and have no "false positive" detections ranked above any "true positives"



How's my model doing?

1. For each category, compute Average Precision (AP)
   = area under Precision vs Recall Curve
   1. For each detection (highest score to lowest score):
      1. If it matches some GT box with IoU > 0.5, mark it as positive and eliminate the GT
      2. Otherwise mark it as negative
      3. Plot a point on PR Curve
2. Average Precision (AP) = area under PR curve
3. Repeat for each class and compute the mean over your dataset

Car AP = 0.65

Cat AP = 0.80

Dog AP = 0.86

mAP@0.5 = 0.77

How's my model doing?

1. For each category, compute Average Precision (AP)
   = area under Precision vs Recall Curve
   1. For each detection (highest score to lowest score):
      1. If it matches some GT box with IoU > 0.5, mark it as positive and eliminate the GT
      2. Otherwise mark it as negative
      3. Plot a point on PR Curve
2. Average Precision (AP) = area under PR curve
3. Repeat for each class and compute the mean over your dataset

- For "COCO mAP": Compute mAP@thresh for each IoU threshold (0.5, 0.55, 0.6, …, 0.95) and take average

mAP@0.5 = 0.77

mAP@0.55 = 0.71

mAP@0.60 = 0.65

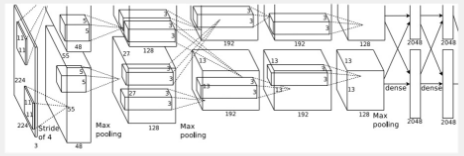
…

mAP@0.95 = 0.2

COCO mAP = 0.4

Detection as regression

This image is CC0 public domain



Vector:
4096

Fully Connected:
4096 to 1000

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

Correct label:
Cat

Softmax Loss

Fully Connected:
4096 to 4

Box Coordinates
(x, y, w, h)

L2 Loss

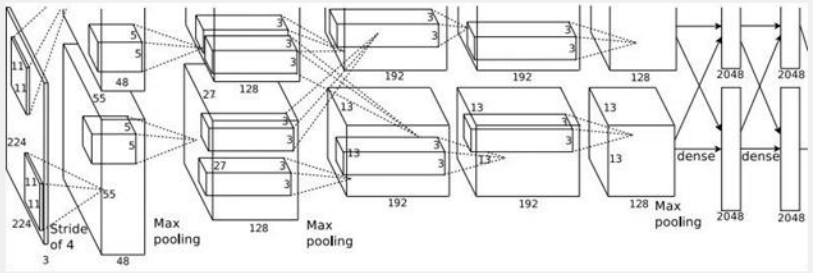
Correct box:
(x', y', w', h')

Weighted Sum → **Loss**

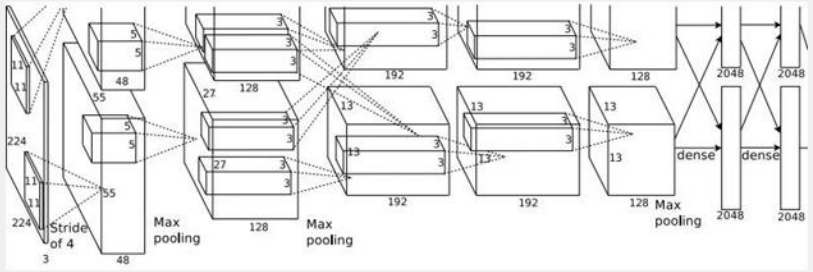
WHAT
→ One class

WHERE
→ Four values

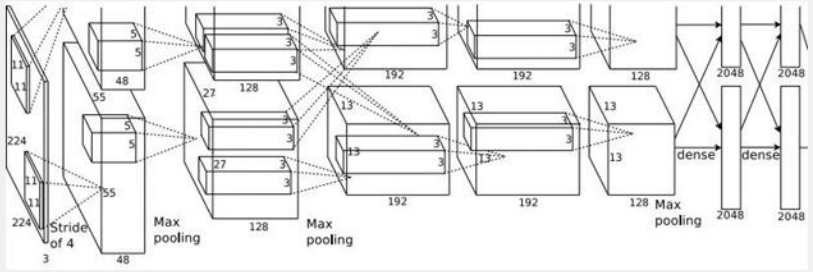
Detection as regression



CAT: (x, y, w, h)

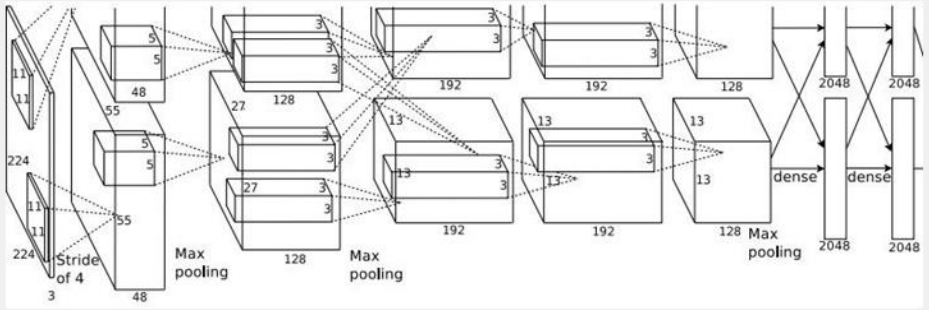


DOG: (x, y, w, h)
DOG: (x, y, w, h)
CAT: (x, y, w, h)



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

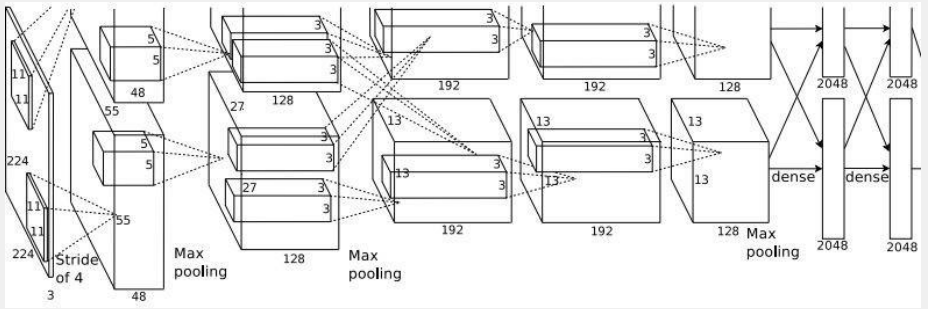
Detection as classification



Dog? NO
Cat? NO
Background? YES

Sliding window approach:
Apply a CNN to many different crops of the image to classify each crop as object or background

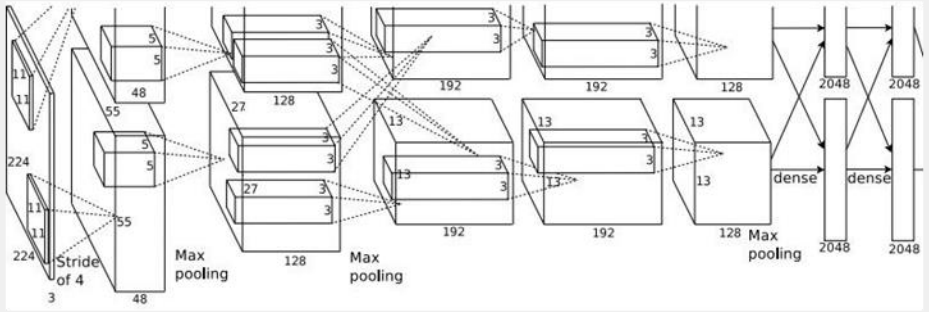
Detection as classification



Dog? YES
Cat? NO
Background? NO

Sliding window approach:
Apply a CNN to many different crops of the image to classify each crop as object or background

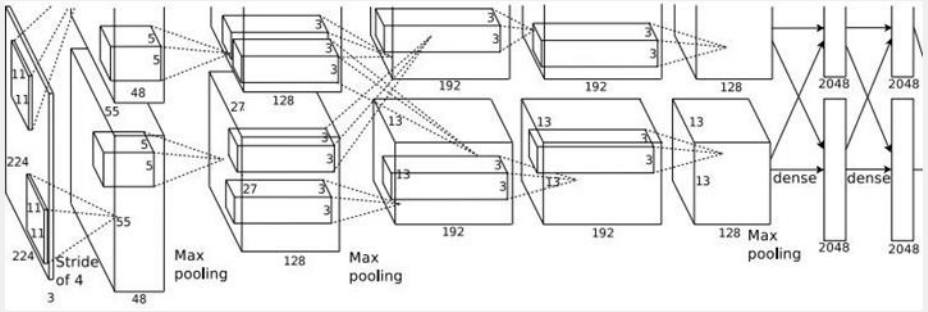
Detection as classification



Dog? YES
Cat? NO
Background? NO

Sliding window approach:
Apply a CNN to many different crops of the image to classify each crop as object or background

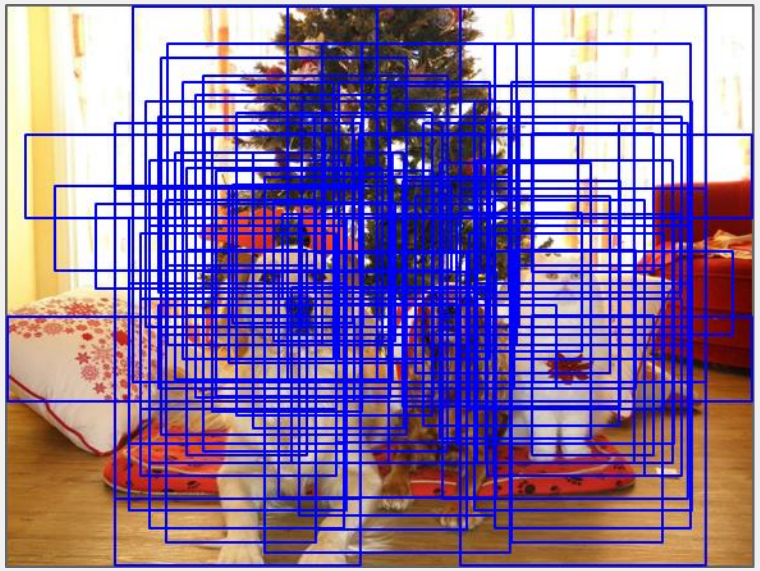
Detection as classification

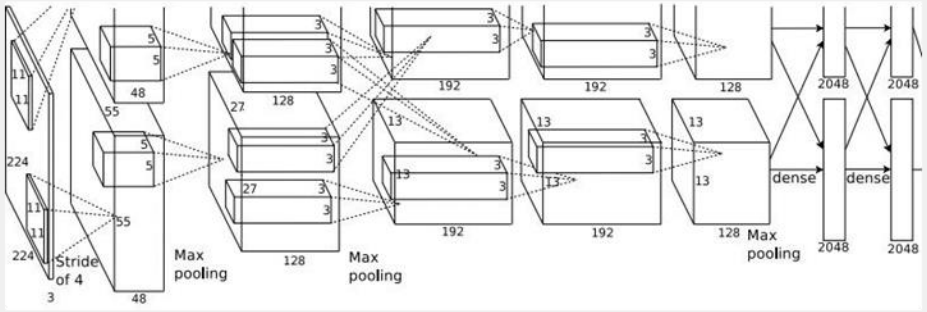


Dog? NO
Cat? YES
Background? NO

Sliding window approach:
Apply a CNN to many different crops of the image to classify each crop as object or background

Detection as classification





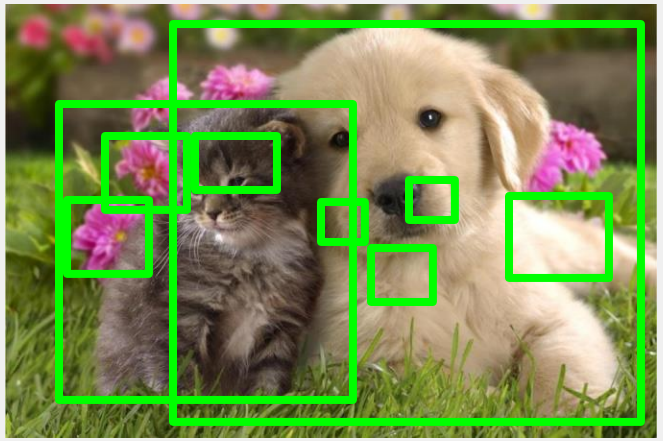
Dog? NO
Cat? YES
Background? NO

Sliding window approach:
Apply a CNN to many different crops of the image to classify each crop as object or background
→ Need to apply CNN to huge number of locations, scales, and aspect ratios, very computationally expensive!

Detection with Region Proposals

Core idea: Find a small set of boxes that are likely to cover all objects, then ‘refine’

- Often based on heuristics: e.g. look for “blob-like” image regions
- Relatively fast to run: e.g. **Selective Search algorithm** gives 2000 region proposals in a few seconds on CPU



Detection with Region Proposals

R-CNN

Input image

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

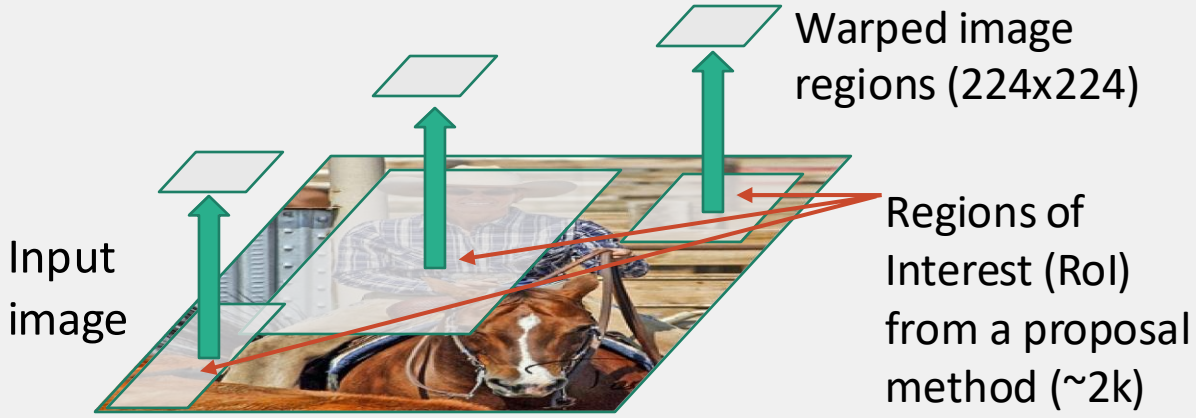
Detection with Region Proposals

R-CNN

Girshick et al, "Rich feature hierarchies for accurate object detection and semantic segmentation", CVPR 2014.
Figure copyright Ross Girshick, 2015; source. Reproduced with permission.

Detection with Region Proposals

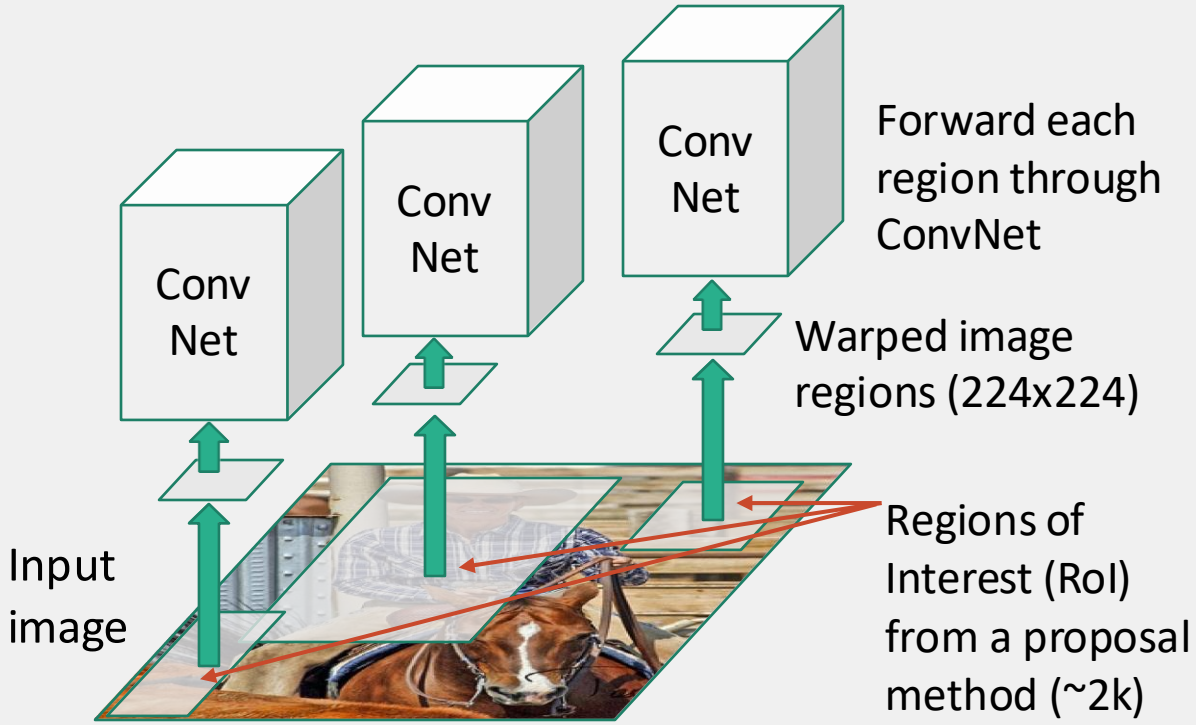
R-CNN



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

Detection with Region Proposals

R-CNN

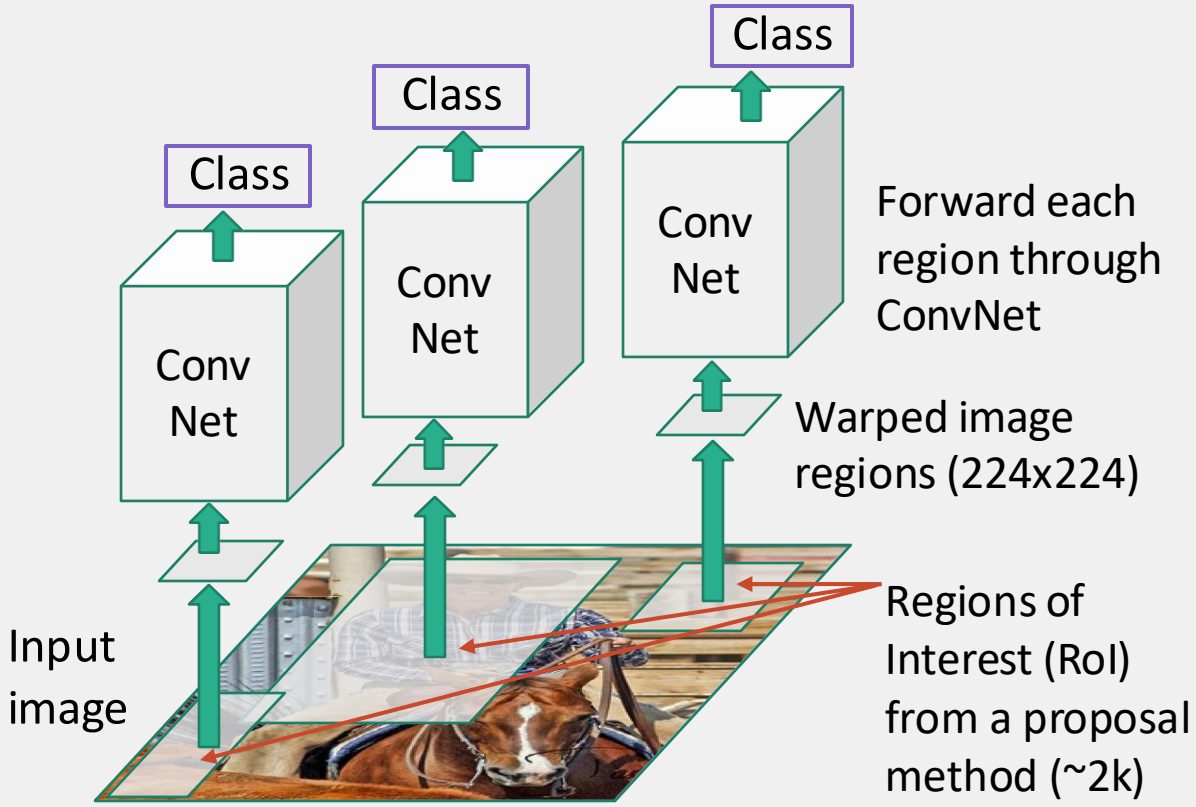


Girshick et al, "Rich feature hierarchies for accurate object detection and semantic segmentation", CVPR 2014.

Detection with Region Proposals

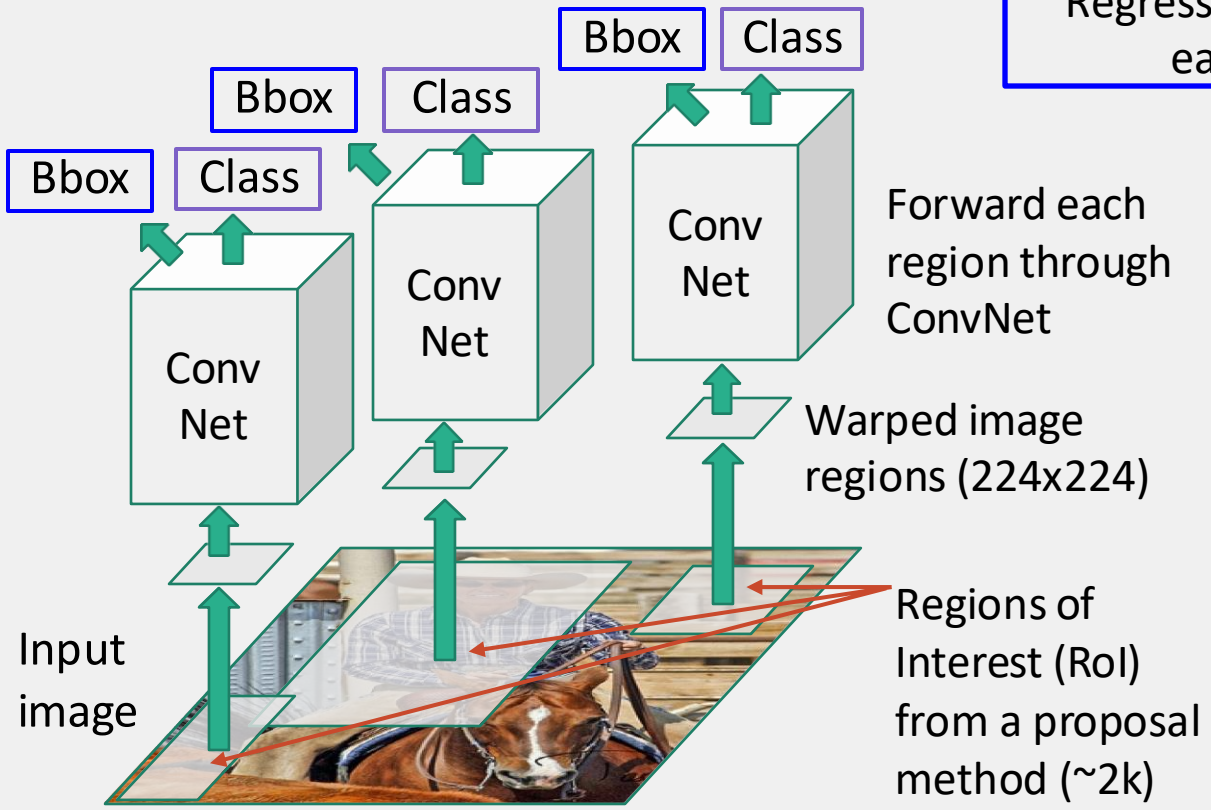
Classify each region

R-CNN



Girshick et al, "Rich feature hierarchies for accurate object detection and semantic segmentation", CVPR 2014.
Figure copyright Ross Girshick, 2015; source. Reproduced with permission.

Detection with Region Proposals



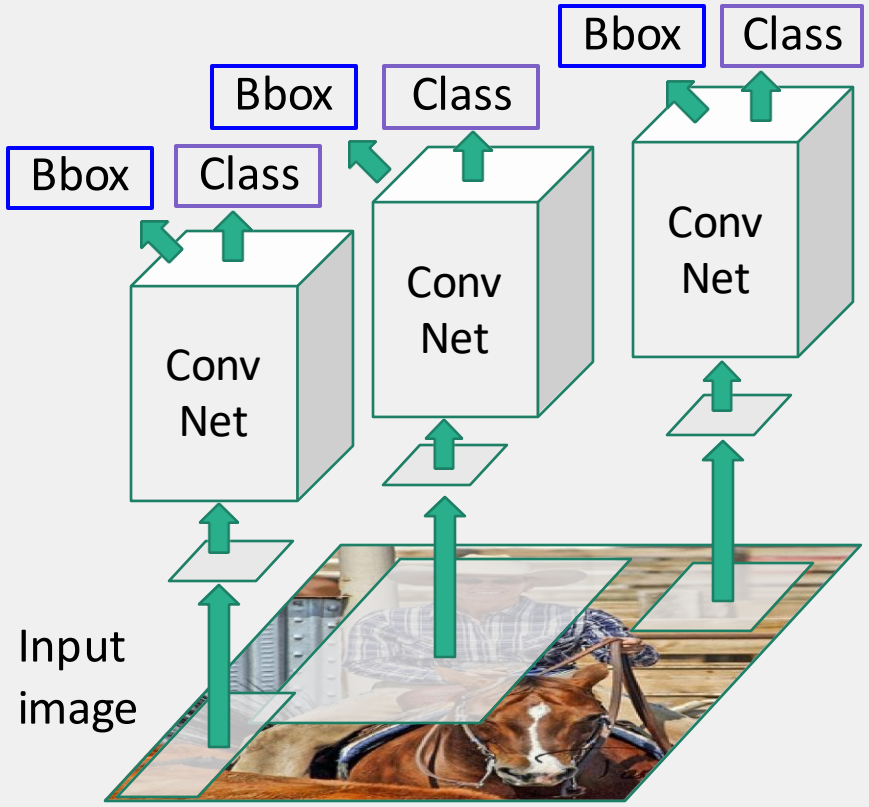
Classify each region

Regress the parameters to turn each RoI into a Bbox

R-CNN

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

Detection with Region Proposals



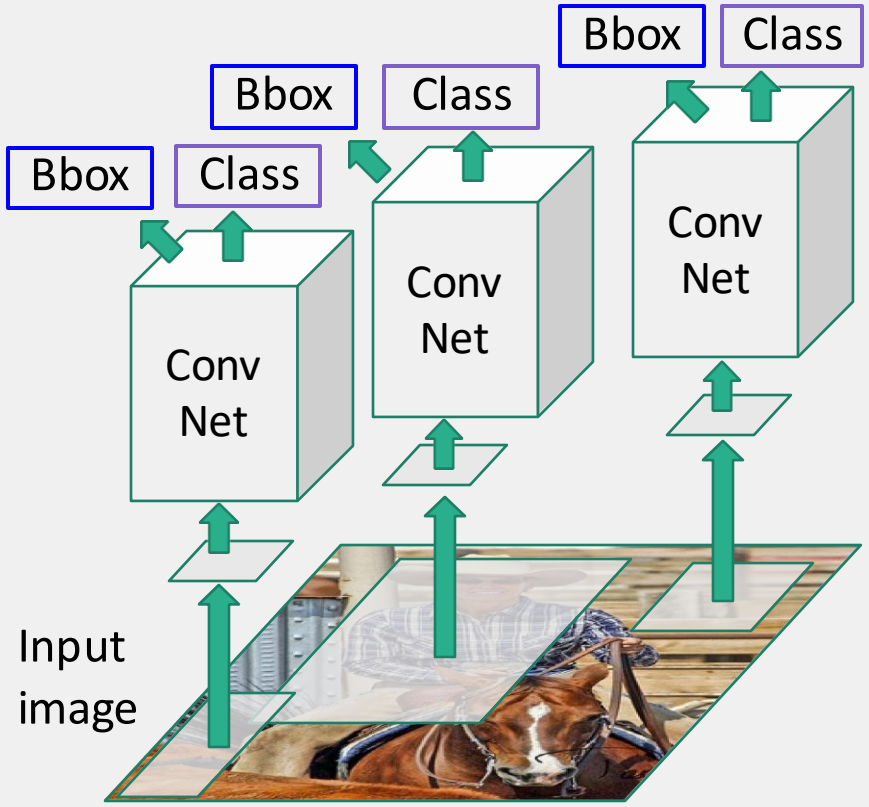
R-CNN

During training:

- Categorize each region proposal as positive, negative, or neutral based on overlap with ground-truth boxes:
  - Positive: > 0.5 IoU with a GT box
  - Negative: < 0.3 IoU with all GT boxes
  - Neutral: between 0.3 and 0.5 IoU with GT boxes
- Crop the non-Neutral regions and run each of them through the CNN
  - Positive regions: predict class and transform
  - Negative regions: just predict class

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

Detection with Region Proposals



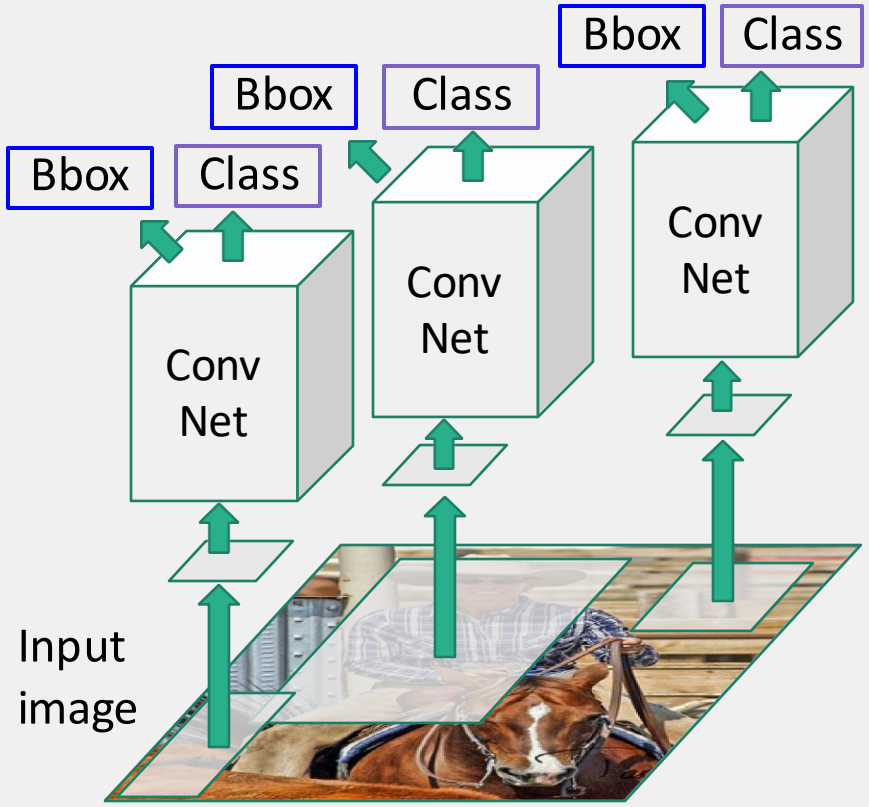
R-CNN

During inference:

- Run the proposal method
- Run the CNN on each proposal to get class scores, transforms
- Threshold class scores to get a set of detections

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

Detection with Region Proposals



R-CNN

During inference:

- Run the proposal method
- Run the CNN on each proposal to get class scores, transforms
- Threshold class scores to get a set of detections

But…

- CNN often outputs overlapping boxes → Non-maximal suppression
- How to set the thresholds? → Hyper-parameters
- Very slow (~2k forward passes for each image) → ?

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

Detection with Region Proposals

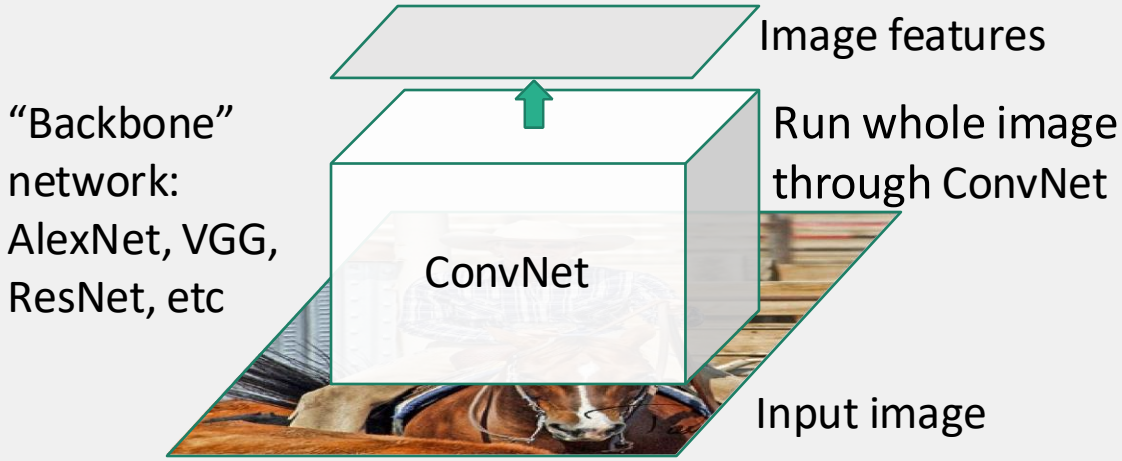
Fast R-CNN

Input image

Girshick, "Fast R-CNN", ICCV 2015. Figure copyright Ross Girshick, 2015; source. Reproduced with permission.

Detection with Region Proposals

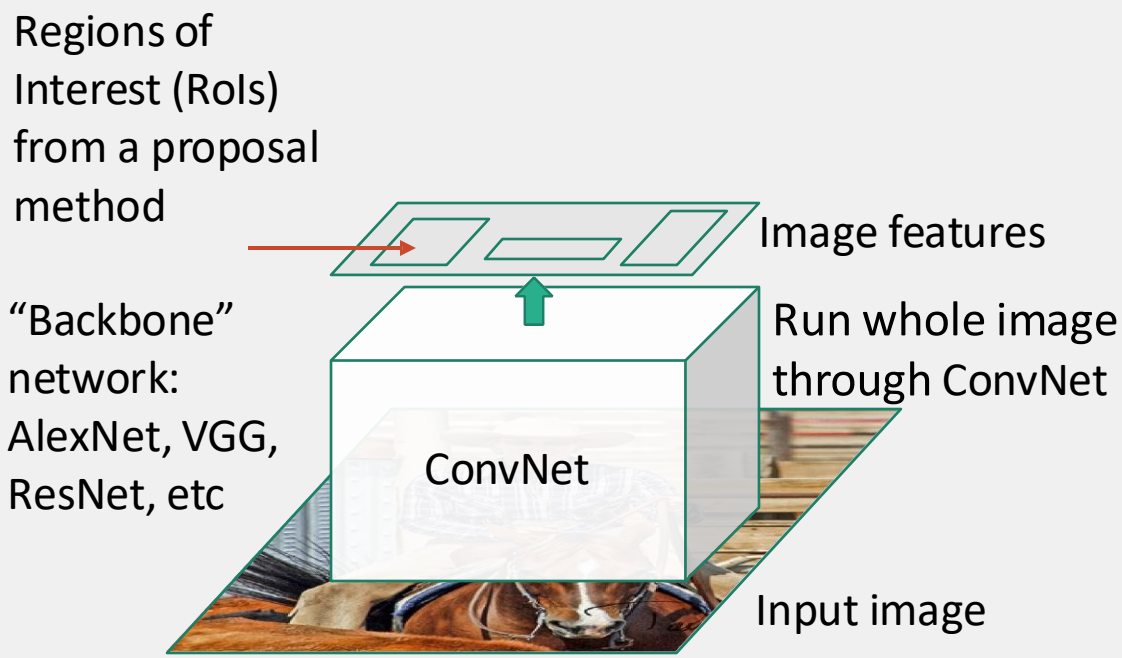
Fast R-CNN



Girshick, “Fast R-CNN”, ICCV 2015. Figure copyright Ross Girshick, 2015; source. Reproduced with permission.

Detection with Region Proposals

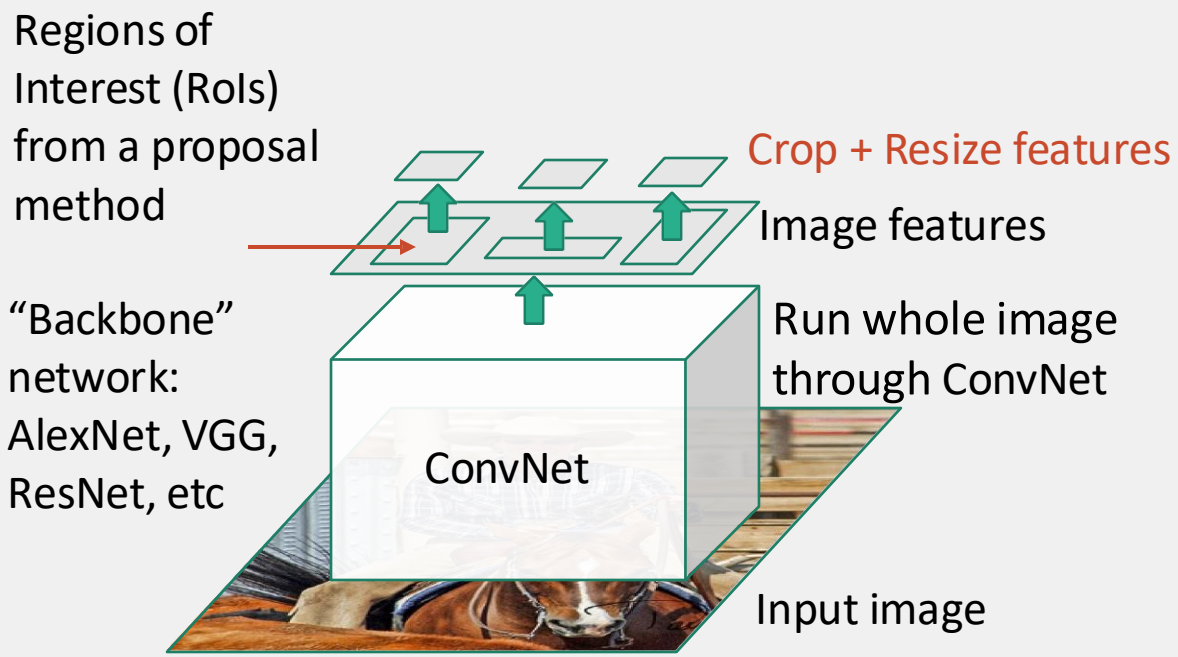
Fast R-CNN



Girshick, "Fast R-CNN", ICCV 2015. Figure copyright Ross Girshick, 2015; source. Reproduced with permission.

Detection with Region Proposals

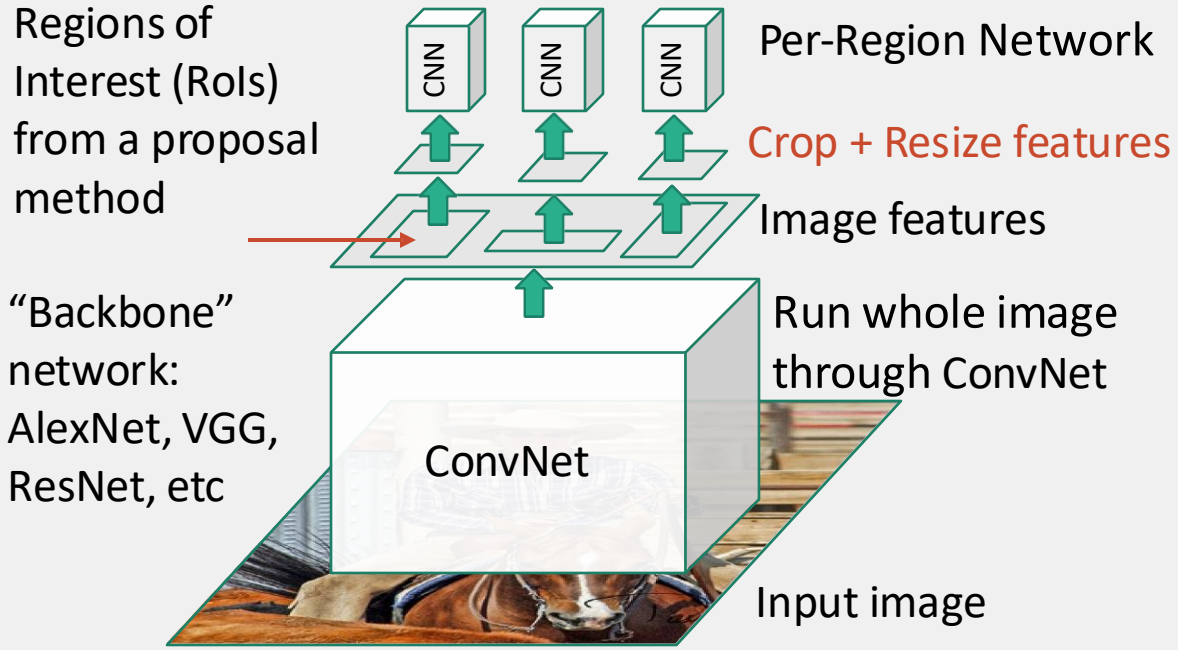
Fast R-CNN



Girshick, "Fast R-CNN", ICCV 2015. Figure copyright Ross Girshick, 2015; source. Reproduced with permission.

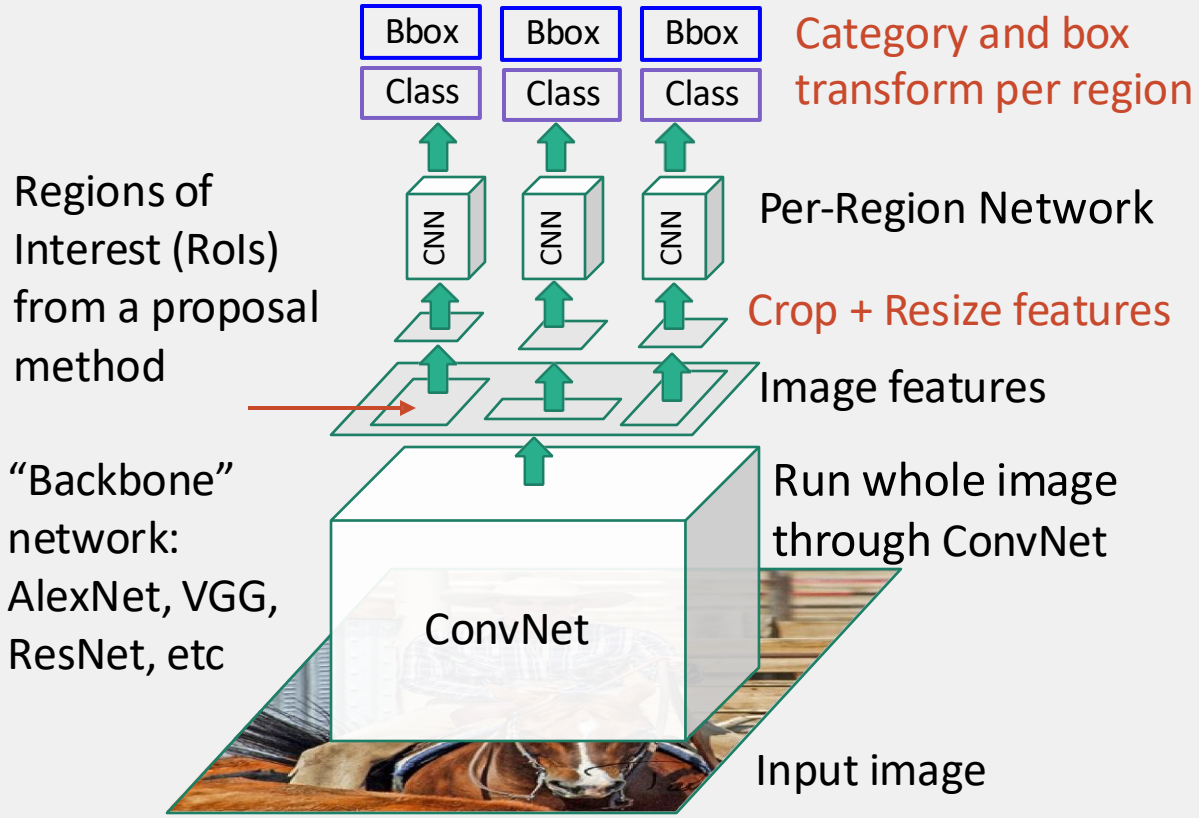
Detection with Region Proposals

Fast R-CNN



Girshick, "Fast R-CNN", ICCV 2015. Figure copyright Ross Girshick, 2015; source. Reproduced with permission.

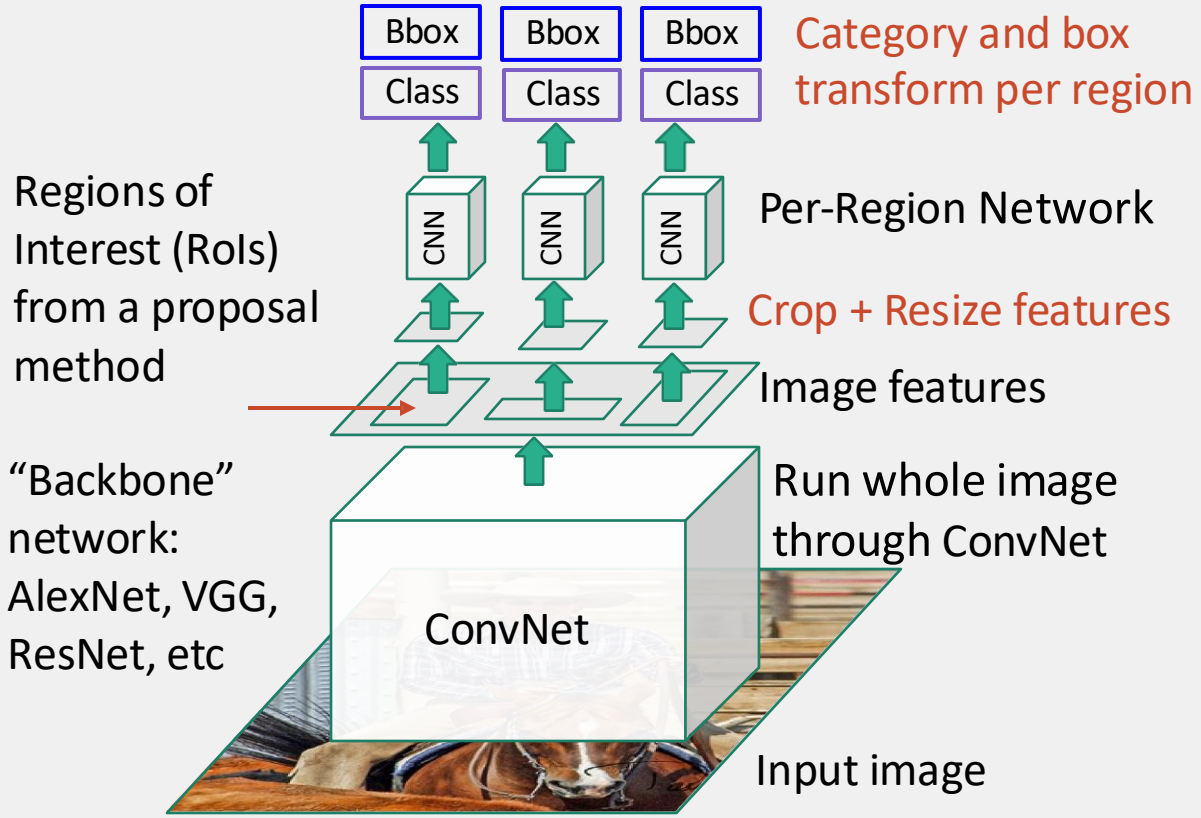
Detection with Region Proposals



Fast R-CNN

Girshick, "Fast R-CNN", ICCV 2015. Figure copyright Ross Girshick, 2015; source. Reproduced with permission.

Detection with Region Proposals



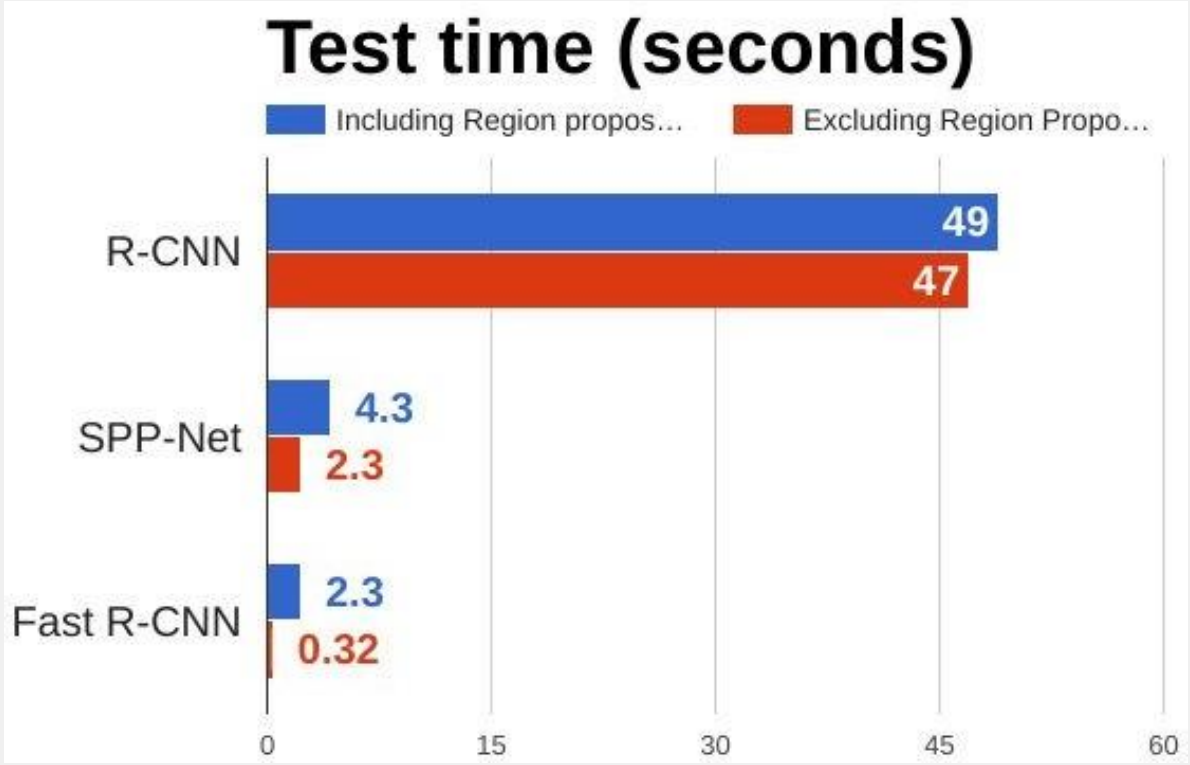
Fast R-CNN

R-CNN was slow since it processed each region independently

In Fast R-CNN, most of the computation happens in backbone network (this saves work for overlapping region proposals) & the Per-Region network is relatively lightweight

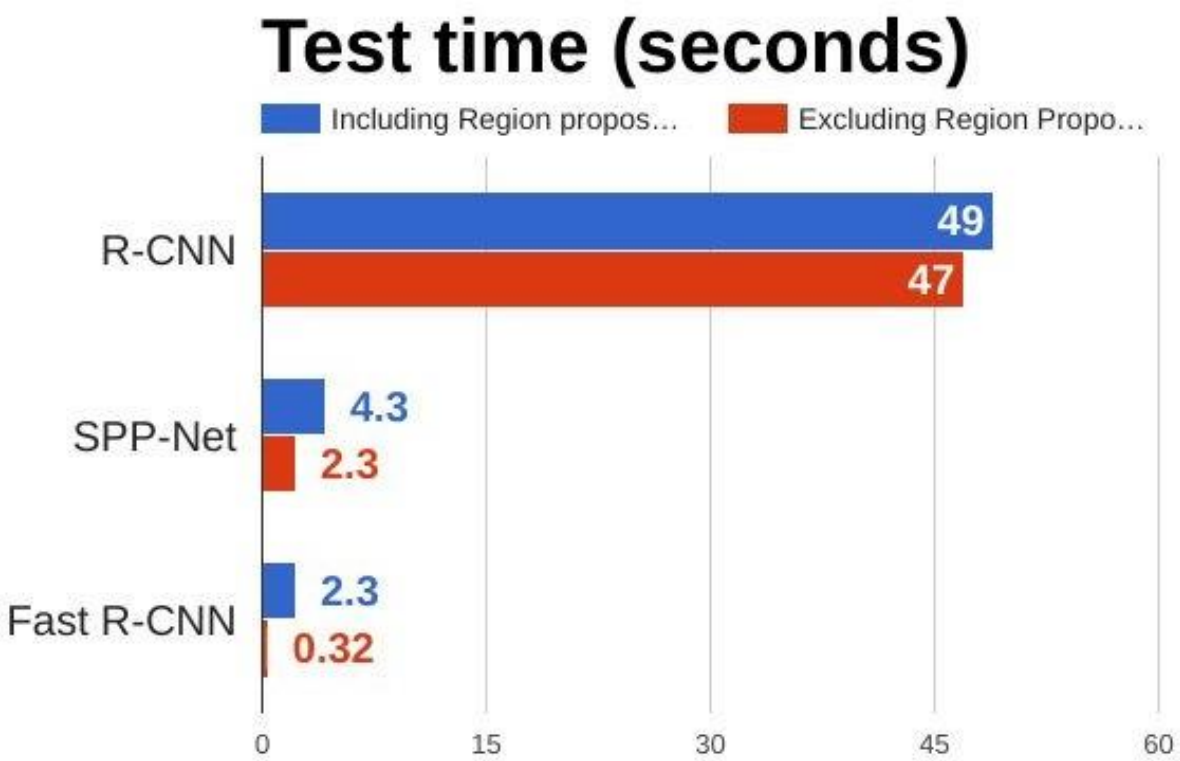
Girshick, "Fast R-CNN", ICCV 2015. Figure copyright Ross Girshick, 2015; source. Reproduced with permission.

Detection with Region Proposals



Girshick et al, "Rich feature hierarchies for accurate object detection and semantic segmentation", CVPR 2014. He et al, "Spatial pyramid pooling in deep convolutional networks for visual recognition", ECCV 2014
Girshick, "Fast R-CNN", ICCV 2015

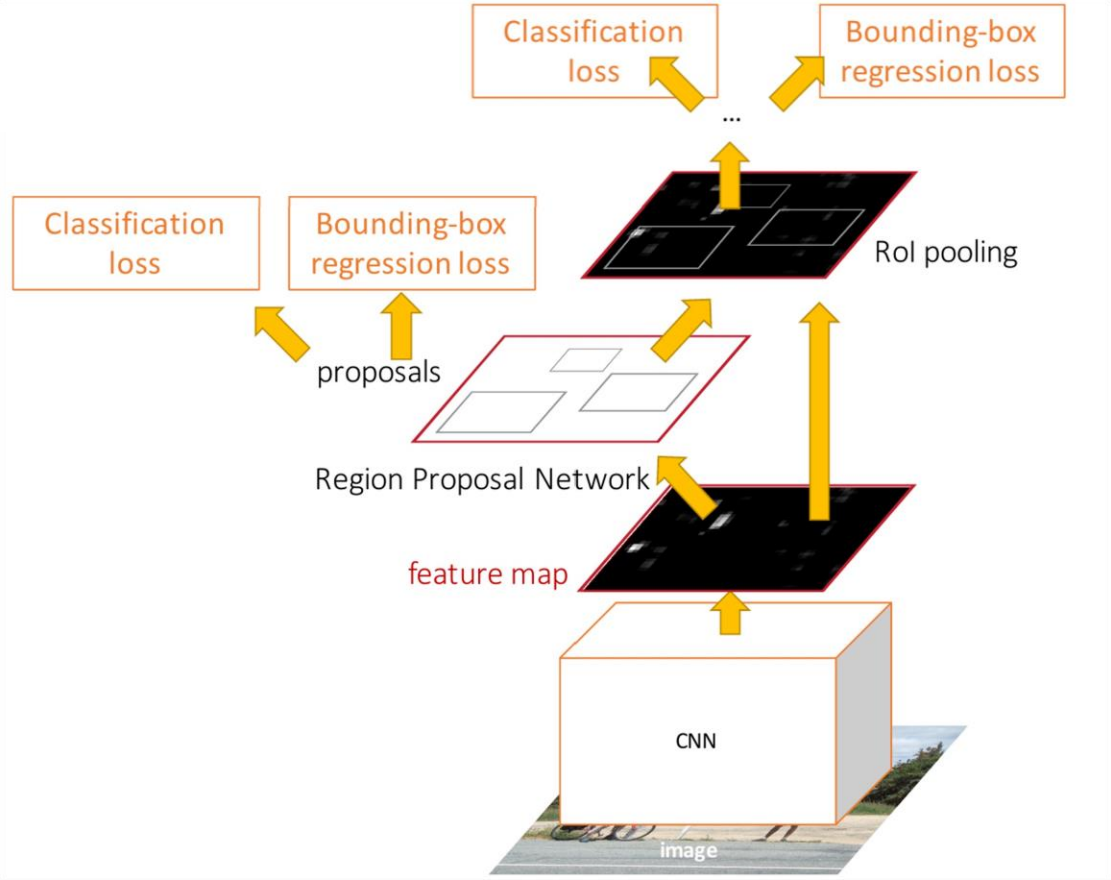
Detection with Region Proposals



The runtime is dominated by the region proposal (recall that the region proposals are computed by heuristic "Selective Search" algorithm on CPU)
→ let's learn them with a CNN!

Girshick et al, "Rich feature hierarchies for accurate object detection and semantic segmentation", CVPR 2014. He et al, "Spatial pyramid pooling in deep convolutional networks for visual recognition", ECCV 2014
Girshick, "Fast R-CNN", ICCV 2015

Detection with Region Proposals



Faster R-CNN

Resort to a **Region Proposal Network (RPN)** to predict proposals from features.
Otherwise same as Fast R-CNN:
Crop features for each proposal & classify them

→ Jointly train with four losses:

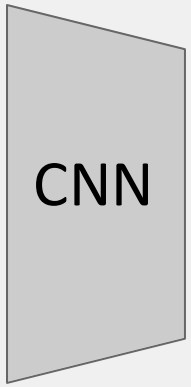
- **RPN classification**: anchor box is object /not an object
- **RPN regression**: predict transform from anchor box to proposal box
- **Object classification**: classify proposals as background / object class
- **Object regression**: predict transform from proposal box to object box

Ren et al, "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks", NIPS 2015 Figure copyright 2015, Ross Girshick; reproduced with permission

Detection with Region Proposals - RPN

Run backbone CNN to get features aligned to input image

Input Image
(e.g. 3 x 640 x 480)



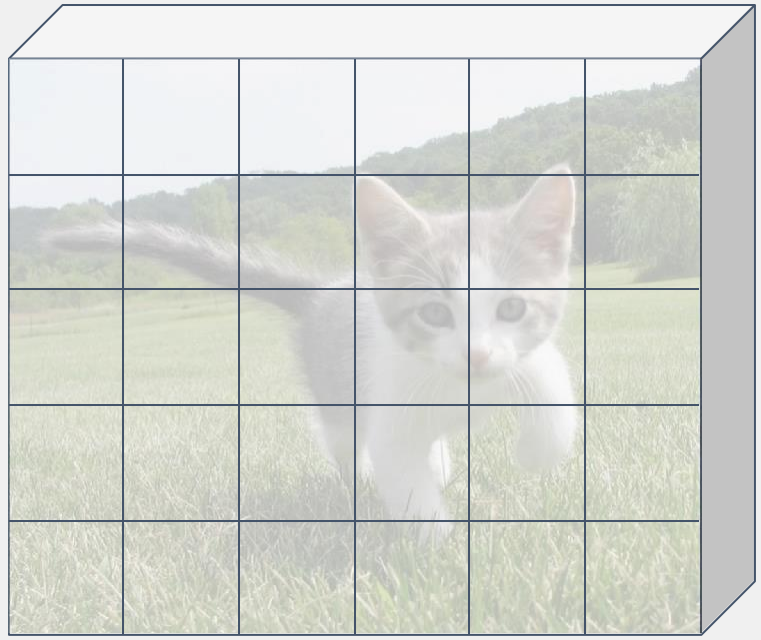


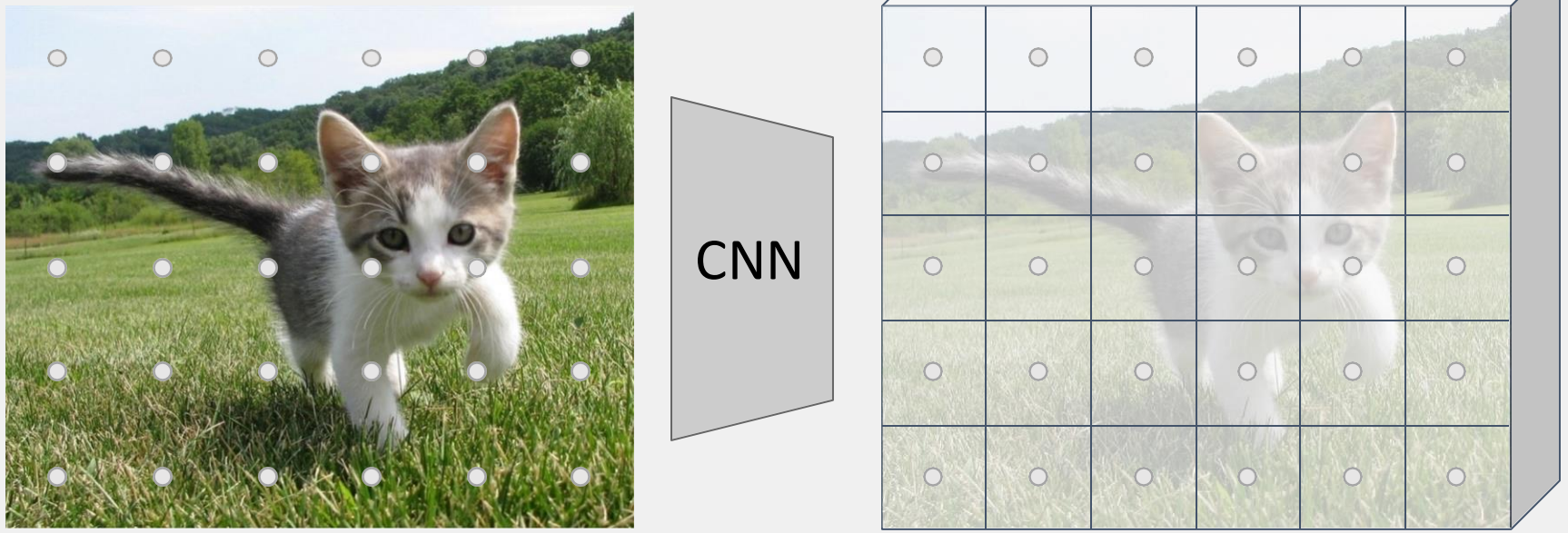
Image features
(e.g. 512 x 5 x 6)

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

Detection with Region Proposals - RPN

Run backbone CNN to get features aligned to input image

Each feature corresponds to a point in the input



Input Image
(e.g. 3 x 640 x 480)

Image features
(e.g. 512 x 5 x 6)

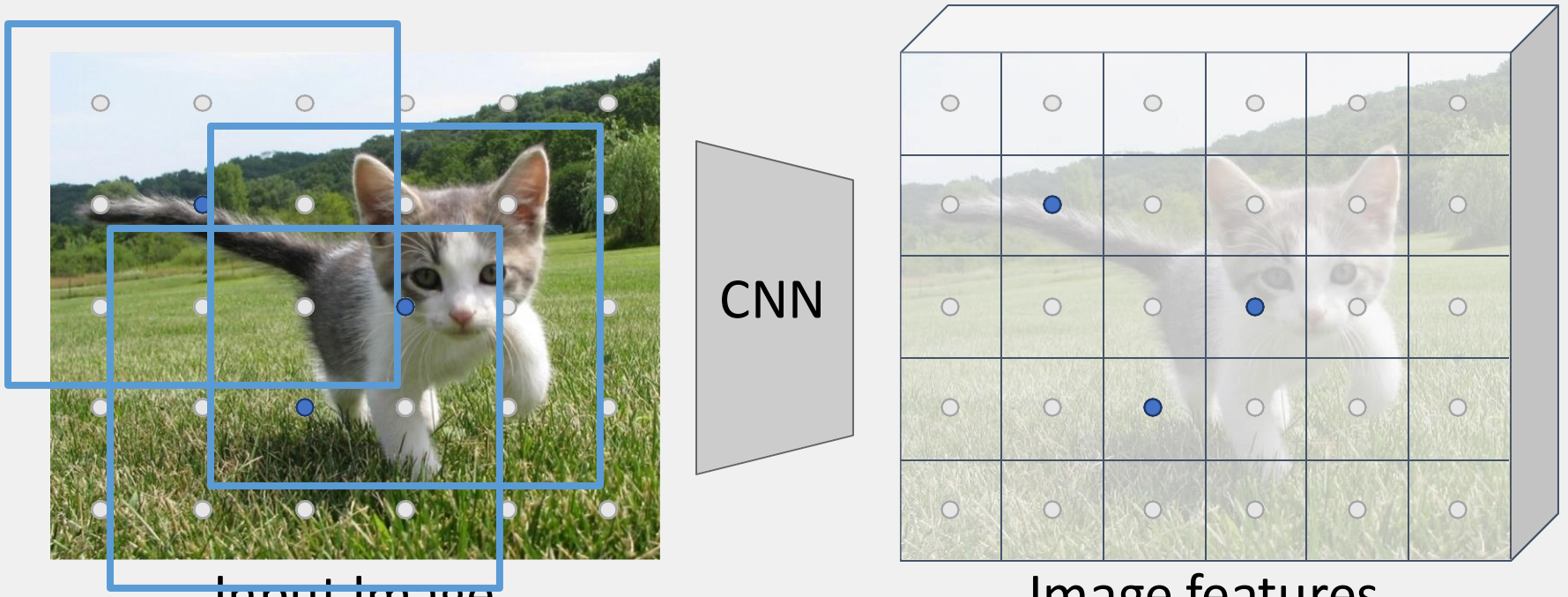
Ren et al, "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks", NIPS 2015

Detection with Region Proposals - RPN

Run backbone CNN to get features aligned to input image

Each feature corresponds to a point in the input

Imagine an **anchor box** of fixed size at each point in the feature map



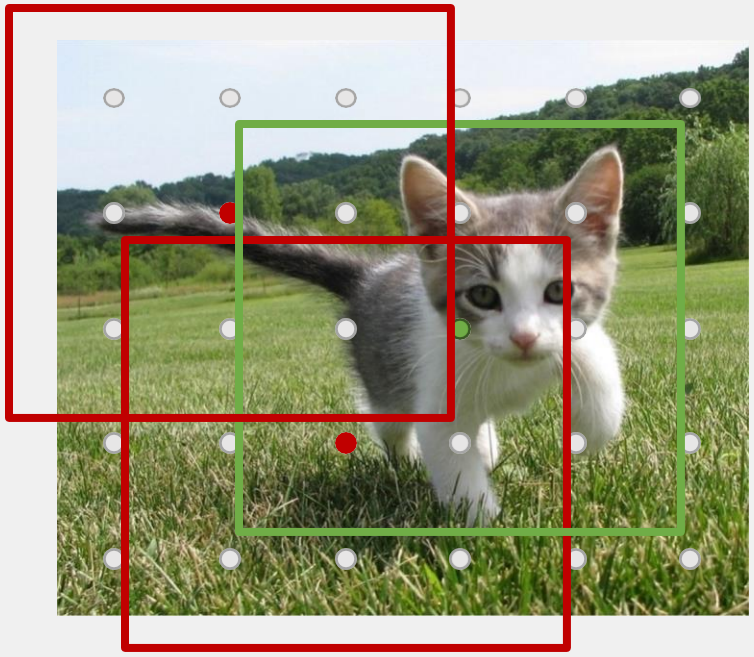
Input Image
(e.g. 3 x 640 x 480)

Image features
(e.g. 512 x 5 x 6)

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

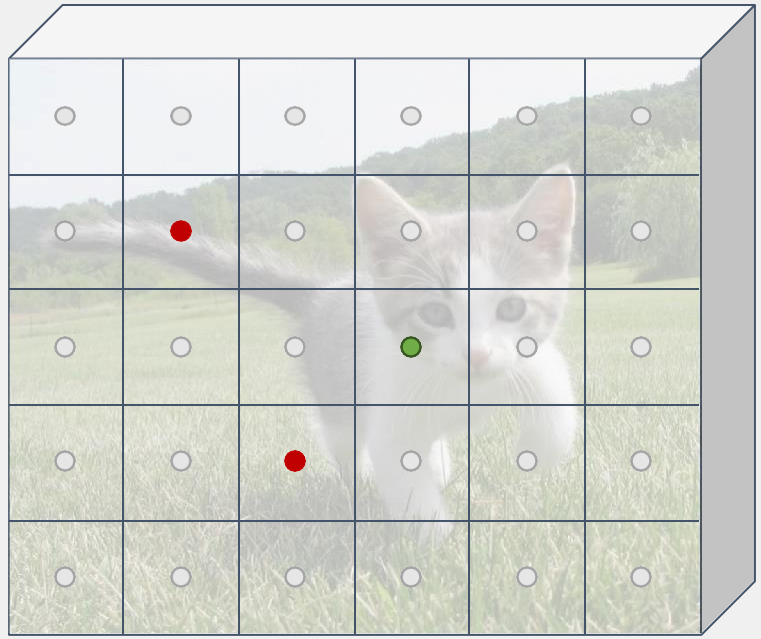
Detection with Region Proposals - RPN

Run backbone CNN to get features aligned to input image



CNN

Each feature corresponds to a point in the input



Conv

Imagine an **anchor box** of fixed size at each point in the feature map

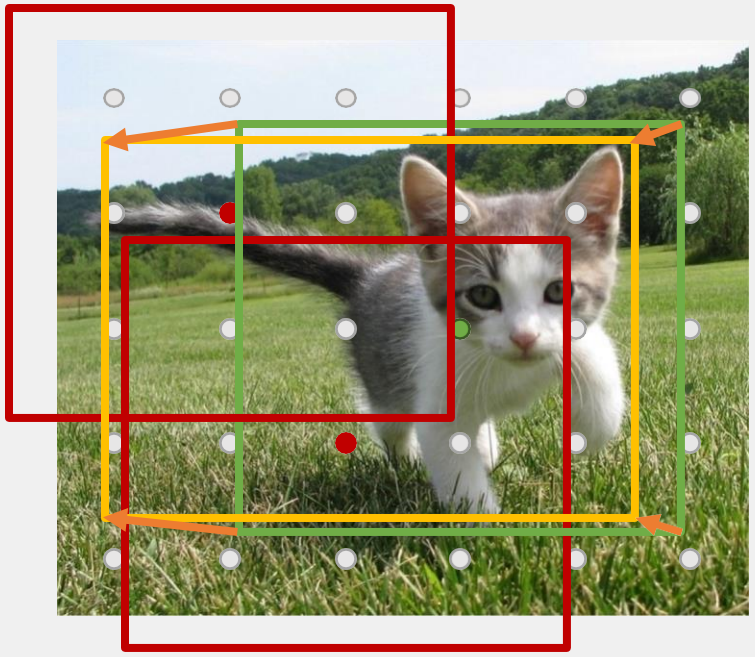
Anchor is object?
2 x 5 x 6

Classify each anchor as positive (object) or negative (no object)

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

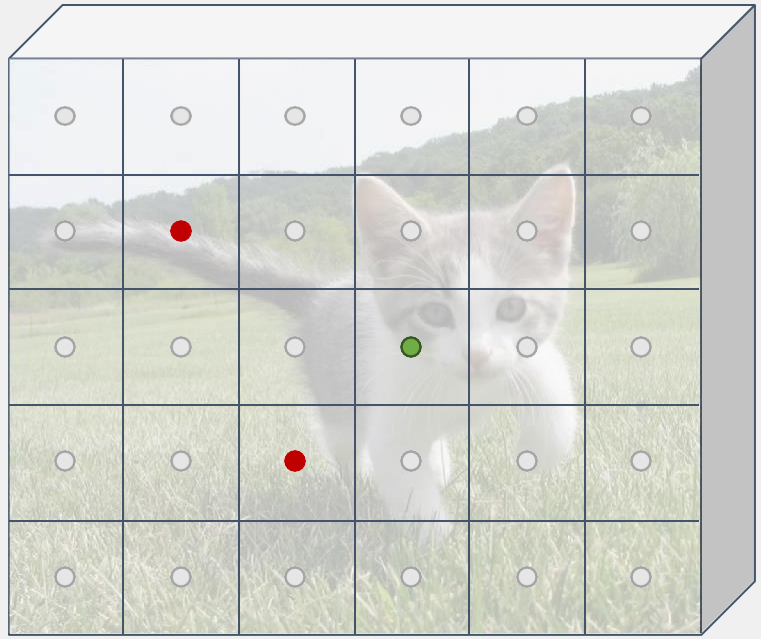
Detection with Region Proposals - RPN

Run backbone CNN to get features aligned to input image



CNN

Each feature corresponds to a point in the input



Conv

Imagine an **anchor box** of fixed size at each point in the feature map

Anchor is object?
2 x 5 x 6

Anchor transforms
4 x 5 x 6

Classify each anchor as positive (object) or negative (no object)
For positive anchors, also predict a transform that convers the anchor to the GT box (like R-CNN)

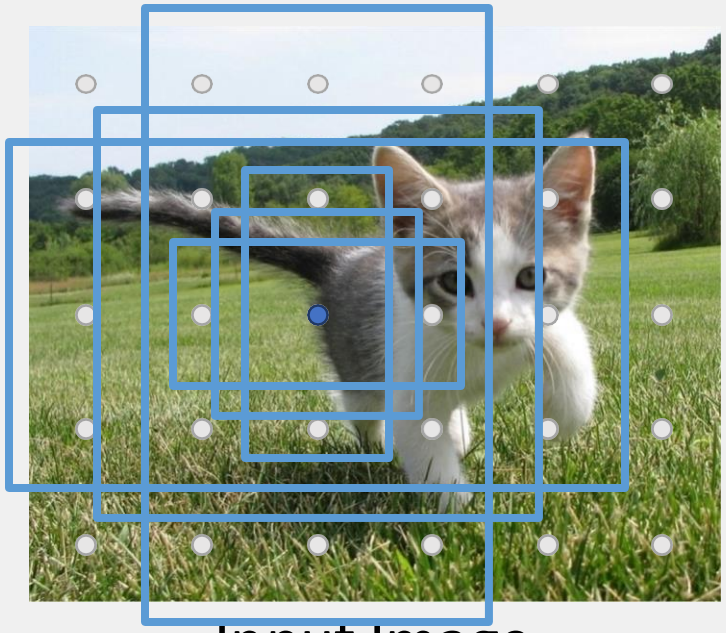
Ren et al, "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks", NIPS 2015

Detection with Region Proposals - RPN

Run backbone CNN to get features aligned to input image

Each feature corresponds to a point in the input

In practice, we use **K anchor boxes** with different size and scale at each point in the feature map



Input Image
(e.g. 3 x 640 x 480)

CNN

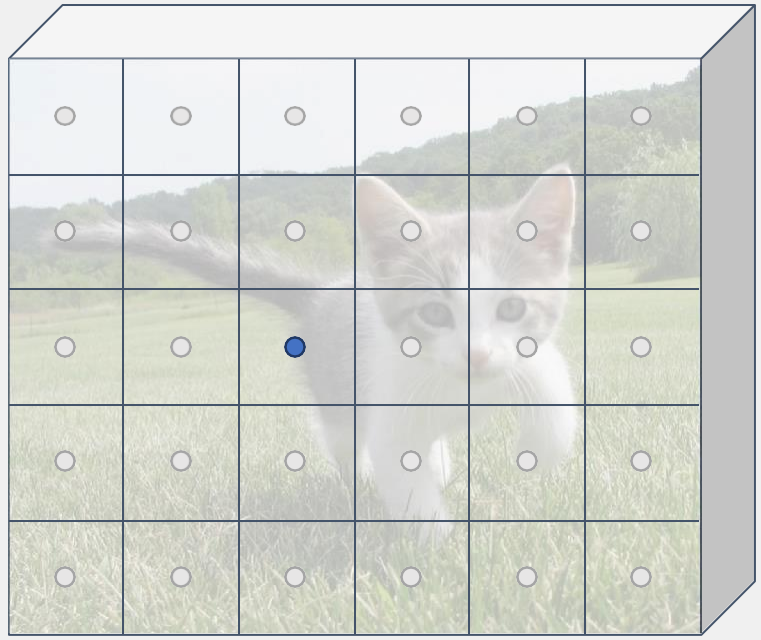


Image features
(e.g. 512 x 5 x 6)

Conv

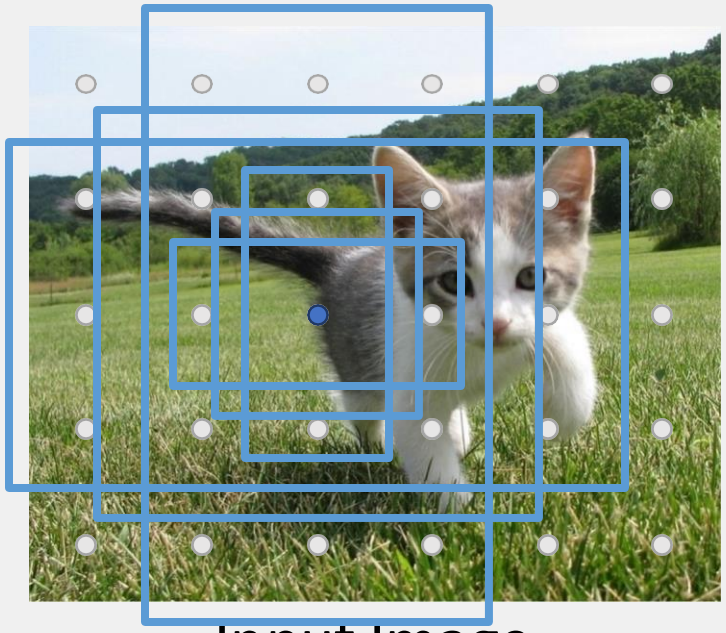
→ Anchor is object?
2K x 5 x 6

→ Anchor transforms
4K x 5 x 6

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

Detection with Region Proposals - RPN

Run backbone CNN to get features aligned to input image



Input Image
(e.g. 3 x 640 x 480)

CNN

Each feature corresponds to a point in the input

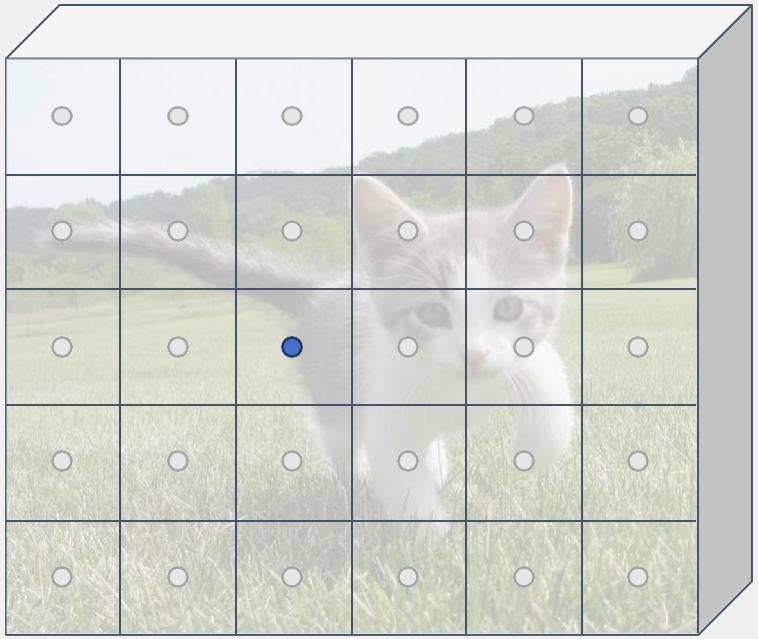


Image features
(e.g. 512 x 5 x 6)

In practice, we use **K anchor boxes** with different size and scale at each point in the feature map

Conv

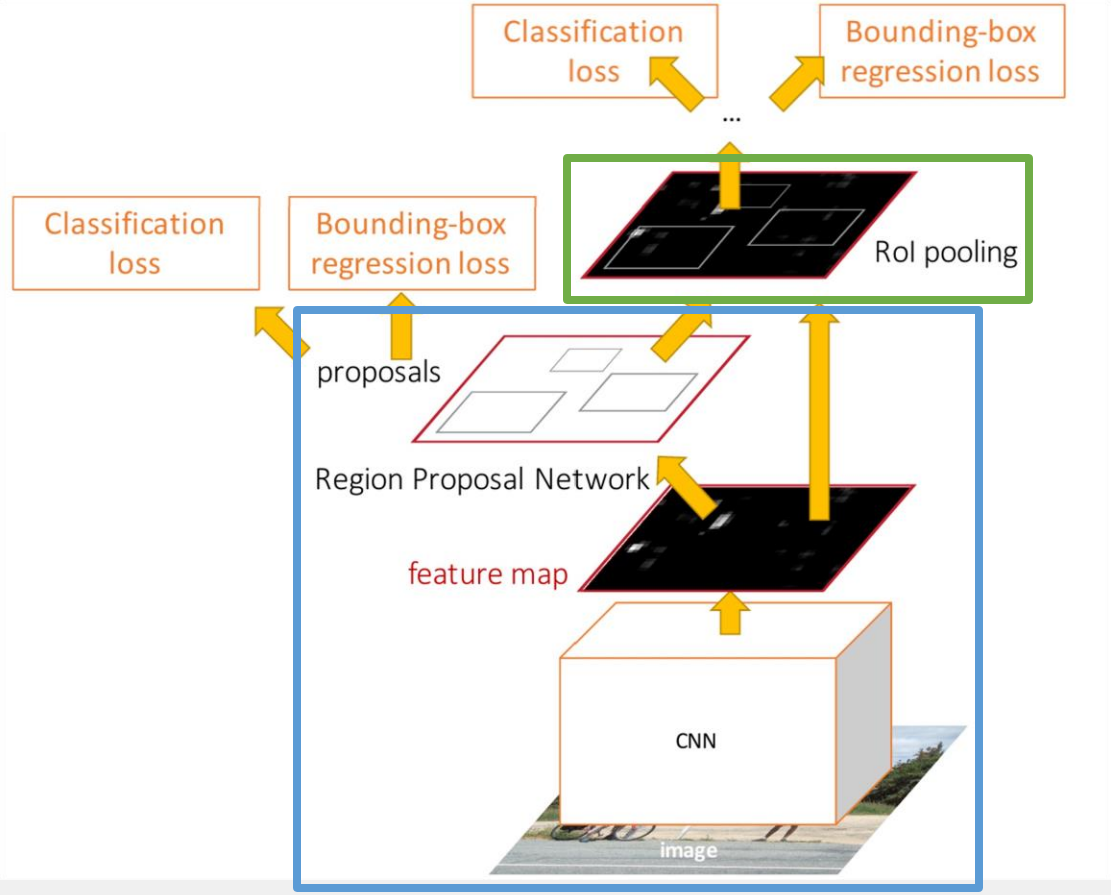
→ Anchor is object?
2K x 5 x 6

→ Anchor transforms
4K x 5 x 6

At test-time, sort all K*5*6 boxes by their 'objectness' score and take the top 300 as region proposals

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

Detection with Region Proposals



Faster R-CNN

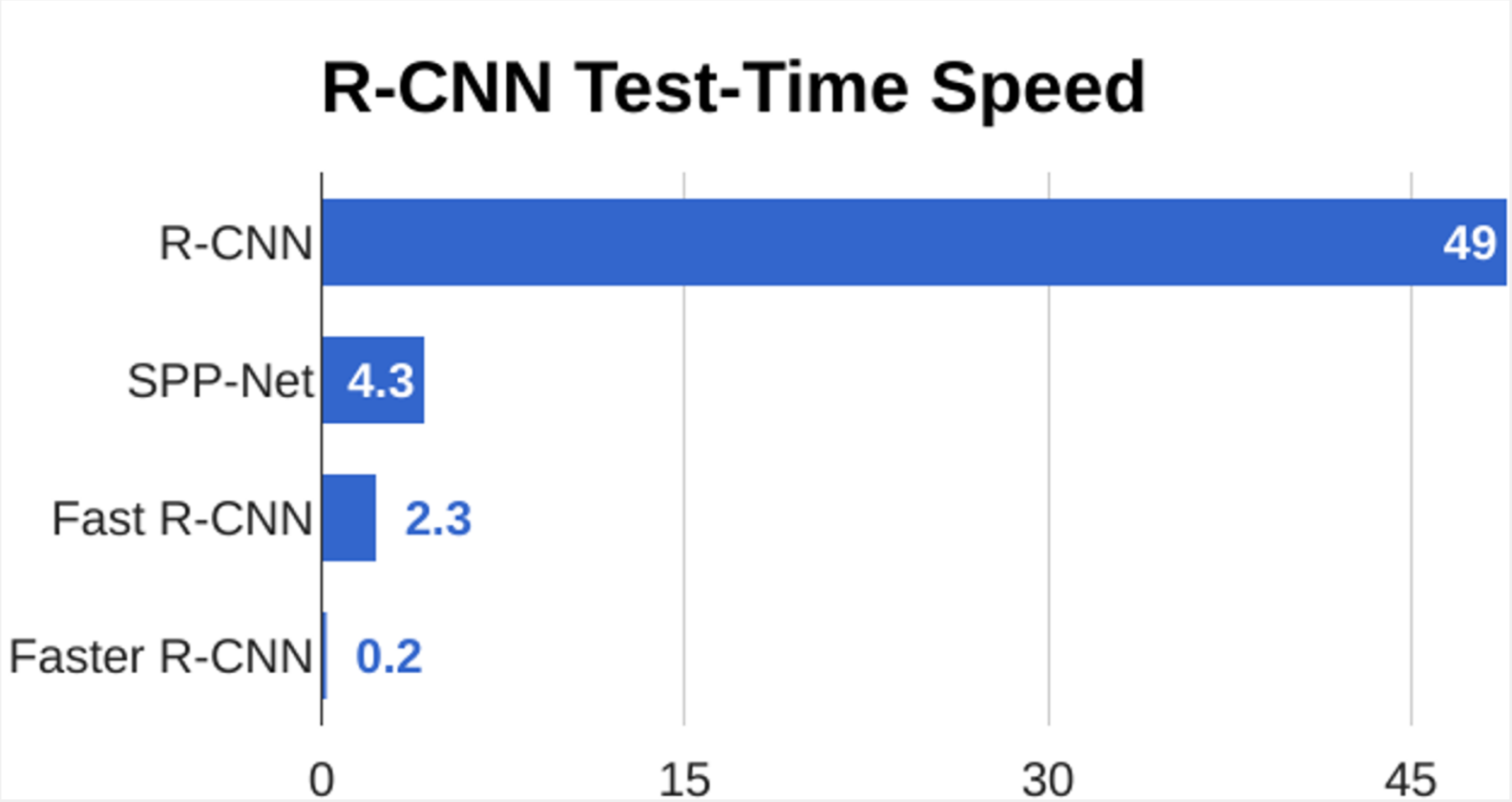
Faster R-CNN is a **Two-stage object detector**

First stage: Run once per image
- Backbone network
- Region proposal network

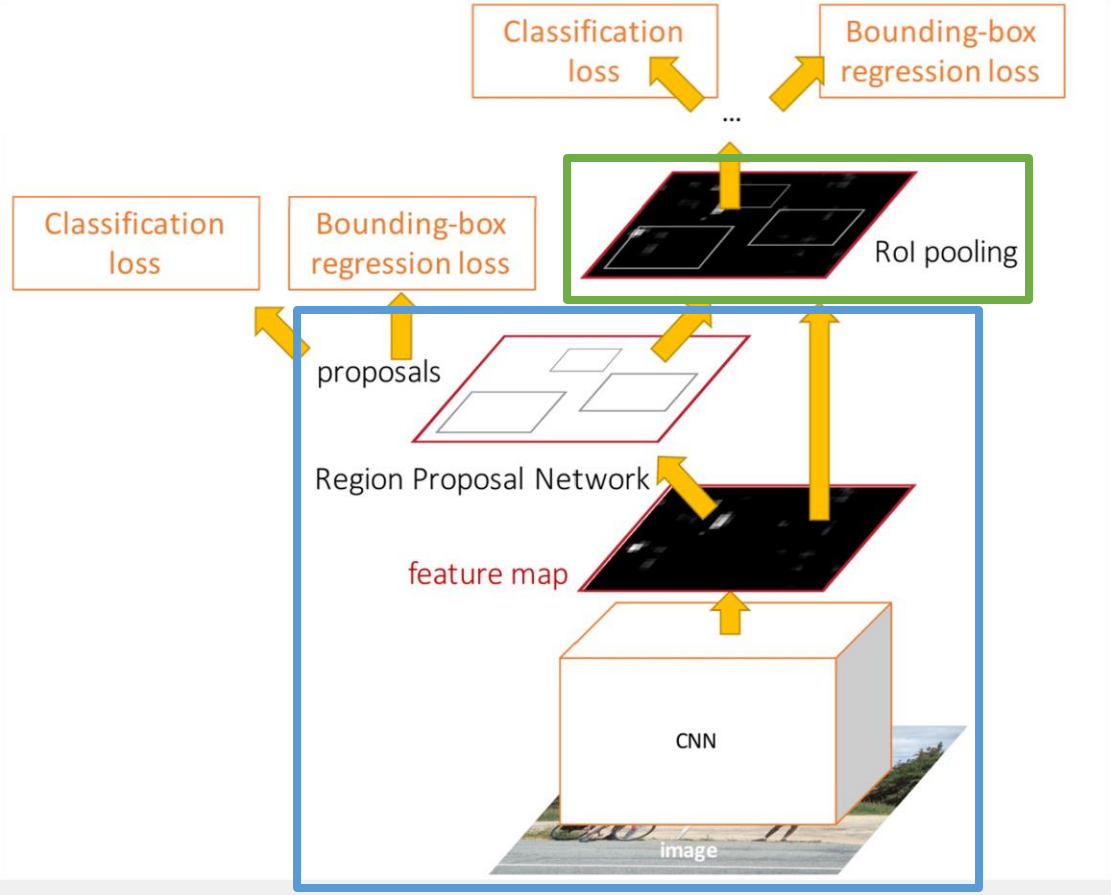
Second stage: Run once per region
- Crop features: RoI pool / align
- Predict object class
- Prediction bbox offset

Ren et al, "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks", NIPS 2015 Figure copyright 2015, Ross Girshick; reproduced with permission

Detection with Region Proposals



Detection with Region Proposals



Faster R-CNN

Faster R-CNN is a **Two-stage object detector**

First stage: Run once per image
- Backbone network
- Region proposal network

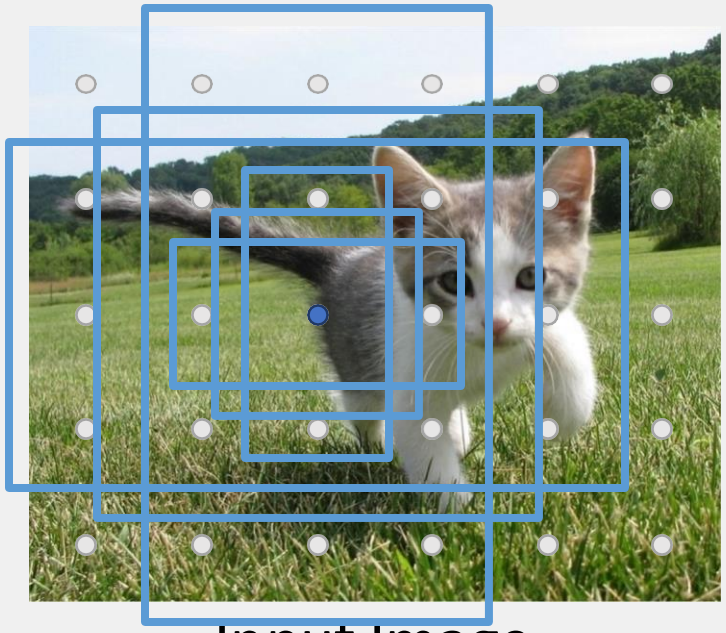
Second stage: Run once per region
- Crop features: RoI pool / align
- Predict object class
- Prediction bbox offset

←Do we really need this?

Ren et al, "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks", NIPS 2015 Figure copyright 2015, Ross Girshick; reproduced with permission

Single-stage detection

RetinaNet



Input Image
(e.g. 3 x 640 x 480)

CNN

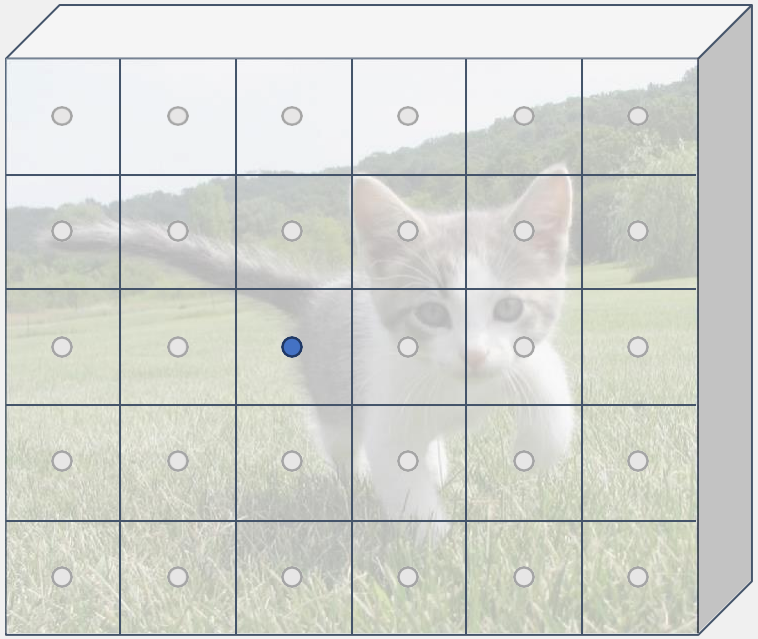


Image features
(e.g. 512 x 5 x 6)

Conv

→ Anchor classification
2K*(C+1) x 5 x 6

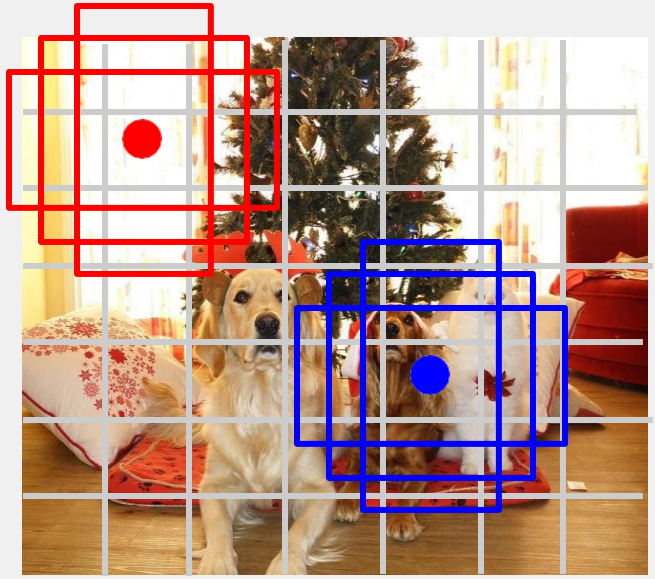
→ Anchor transforms
4K x 5 x 6

Similar to RPN – but rather than classify anchors as object/no object, directly predict object category (among C categories) or background

Lin et al, "Focal Loss for Dense Object Detection", ICCV 2017

Single-stage detection

Input image
3 x H x W



Divide the image into a grid 7 x 7

Consider a set of B **base boxes** centered at each grid cell (here B = 3)

YOLO

Within each grid cell:

- Regress from each of the B base boxes to a final box with 5 values: (dx, dy, dh, dw, confidence)
- Predict scores for each of the C classes (including background as a class)

Output:
7 x 7 x (5 * B + C)

→ Go from input image to tensor of scores with one big convolutional network!

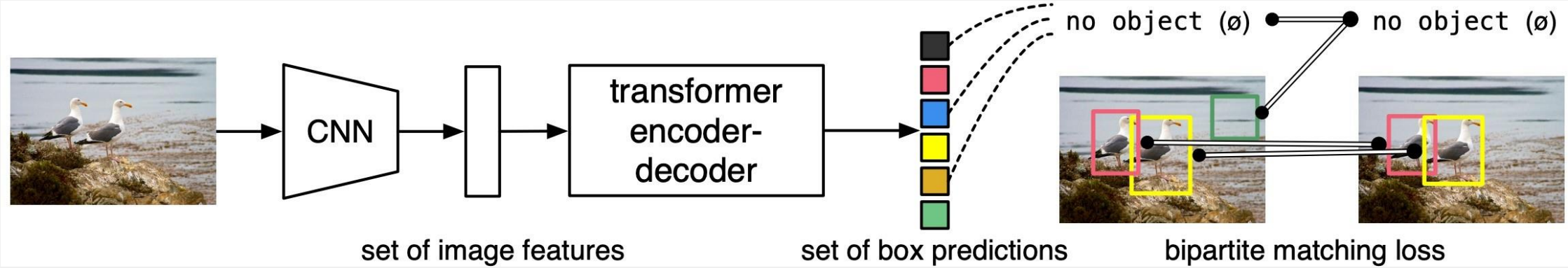
Redmon et al, "You Only Look Once: Unified, Real-Time Object Detection", CVPR 2016
Liu et al, "SSD: Single-Shot MultiBox Detector", ECCV 2016

Detection Transformer

Treat Detection as a sequence prediction task: take an image as input and directly output a set of boxes from a Transformer

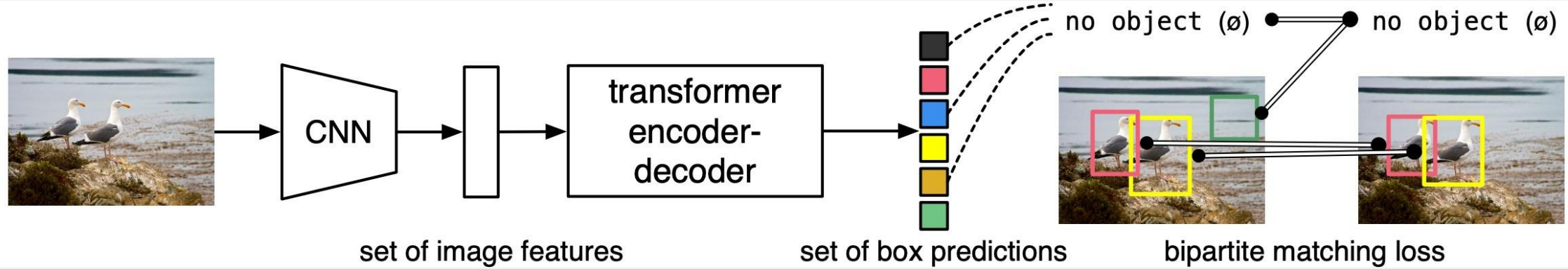
→ No anchors, no regression of box transforms

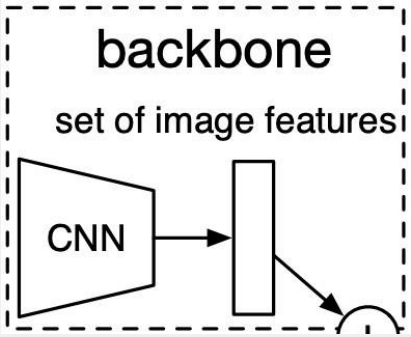
In training, train to regress box coordinates (match the predicted boxes to GT boxes with bipartite matching)



Carion et al, "End-to-End Object Detection with Transformers", ECCV 2020

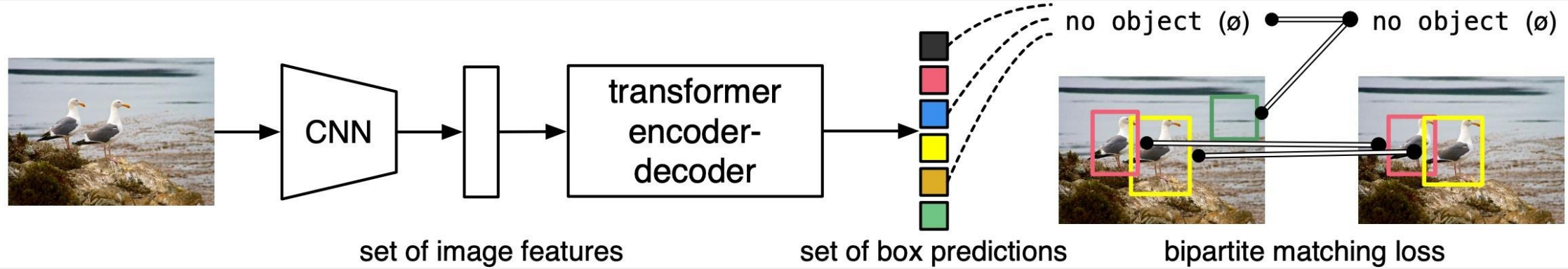
Detection Transformer

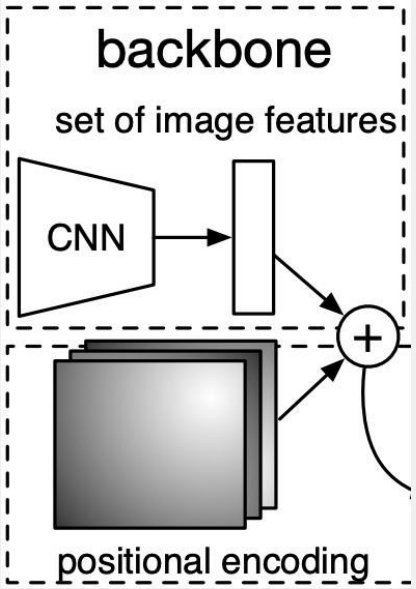




Carion et al, "End-to-End Object Detection with Transformers", ECCV 2020

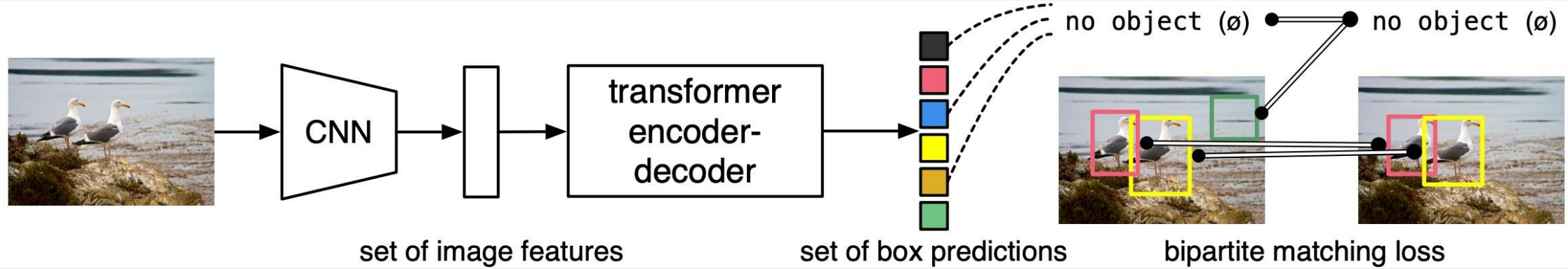
Detection Transformer

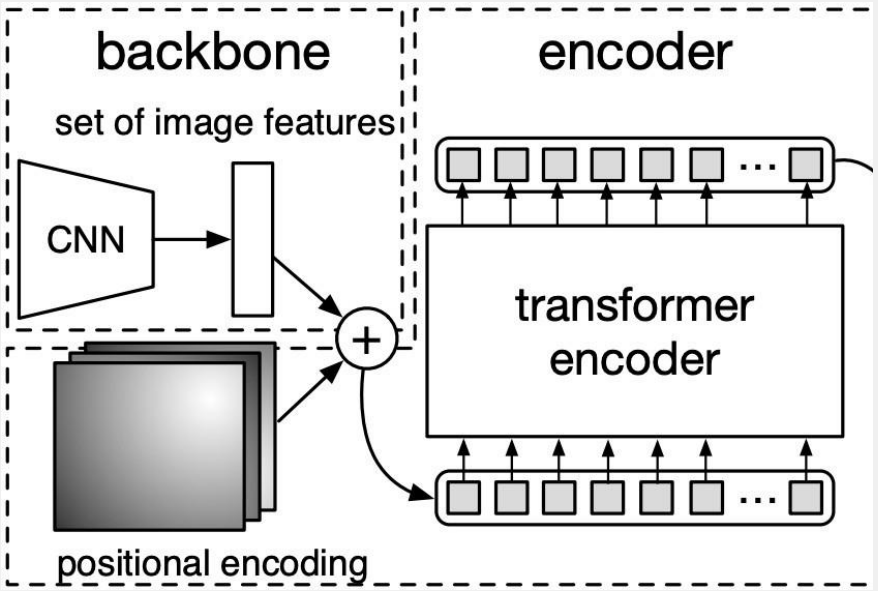




Carion et al, "End-to-End Object Detection with Transformers", ECCV 2020

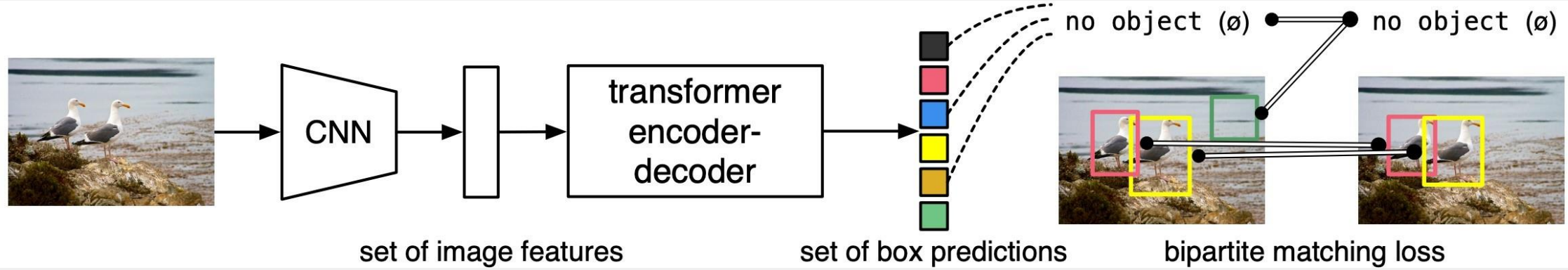
Detection Transformer

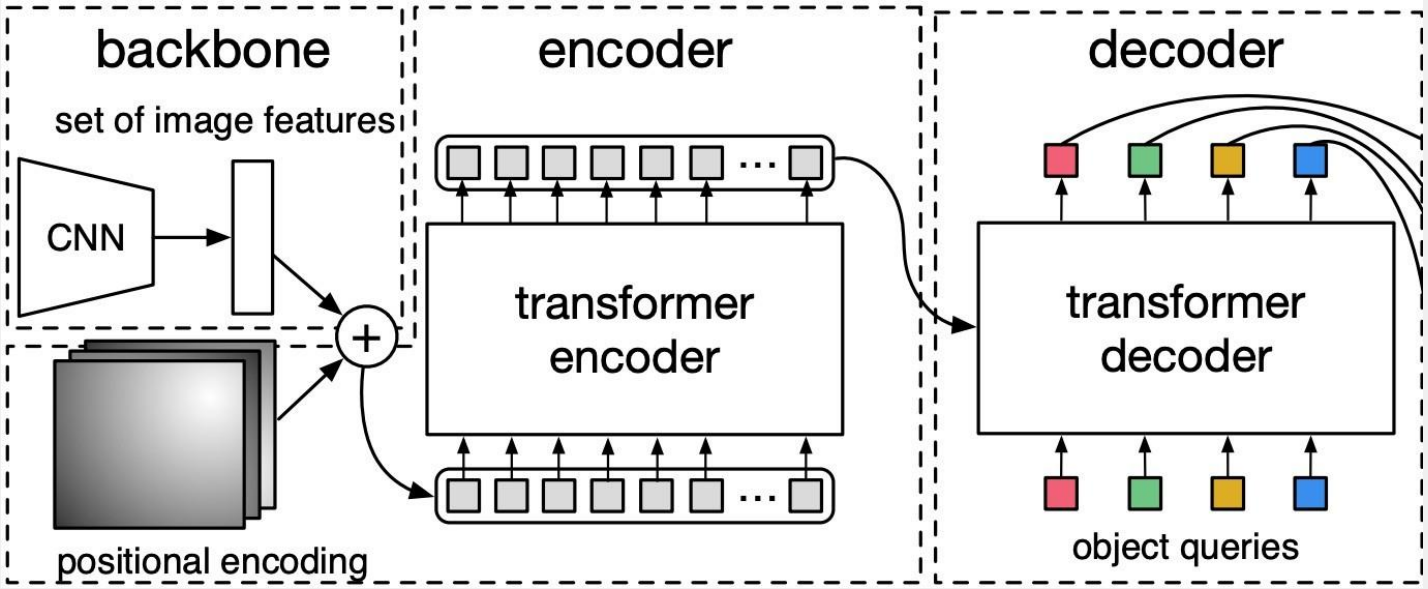




Carion et al, "End-to-End Object Detection with Transformers", ECCV 2020

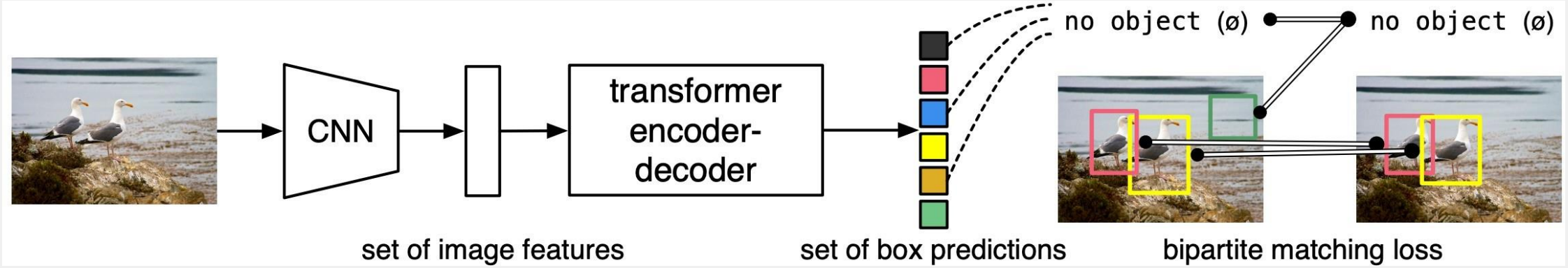
Detection Transformer

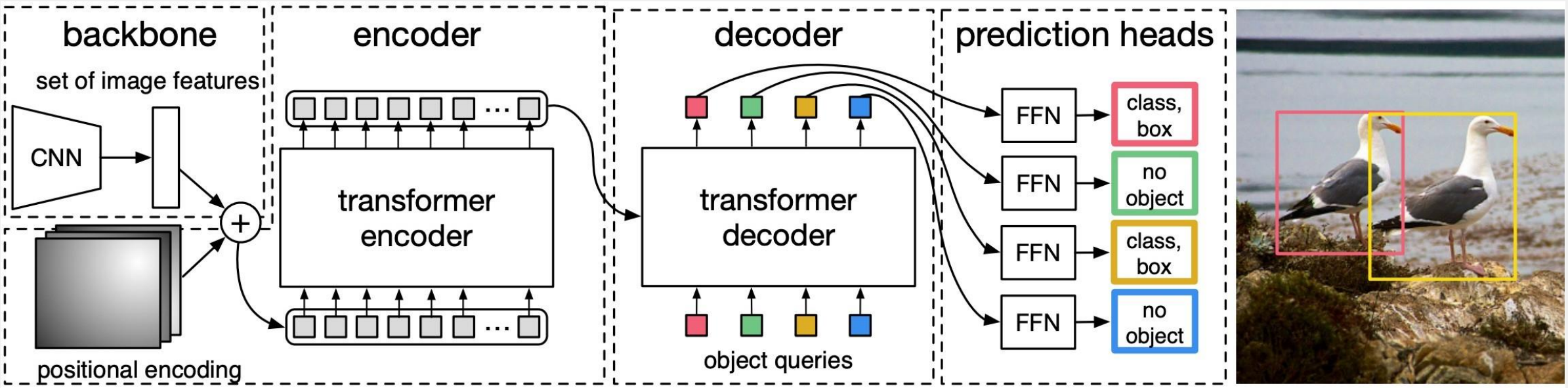




Carion et al, "End-to-End Object Detection with Transformers", ECCV 2020

Detection Transformer





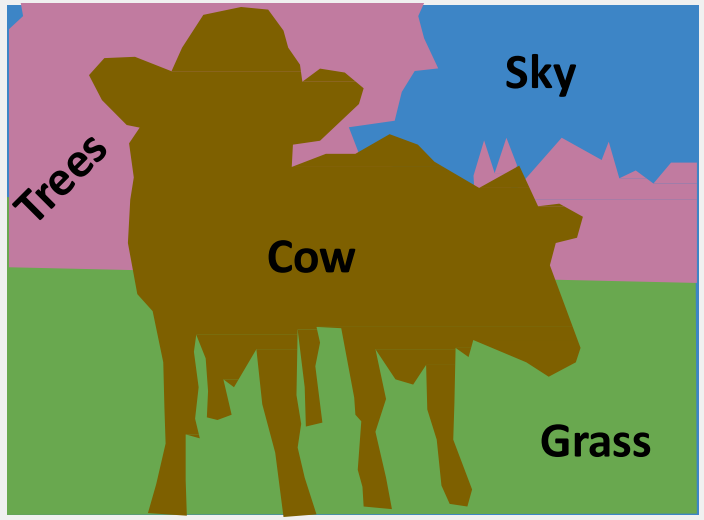
Carion et al, "End-to-End Object Detection with Transformers", ECCV 2020

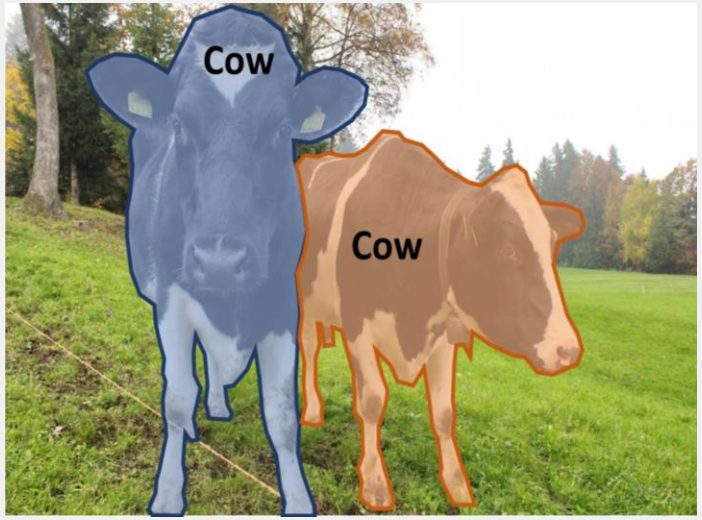
Instance Segmentation: a definition

Input: Single image (can also be a video frame)

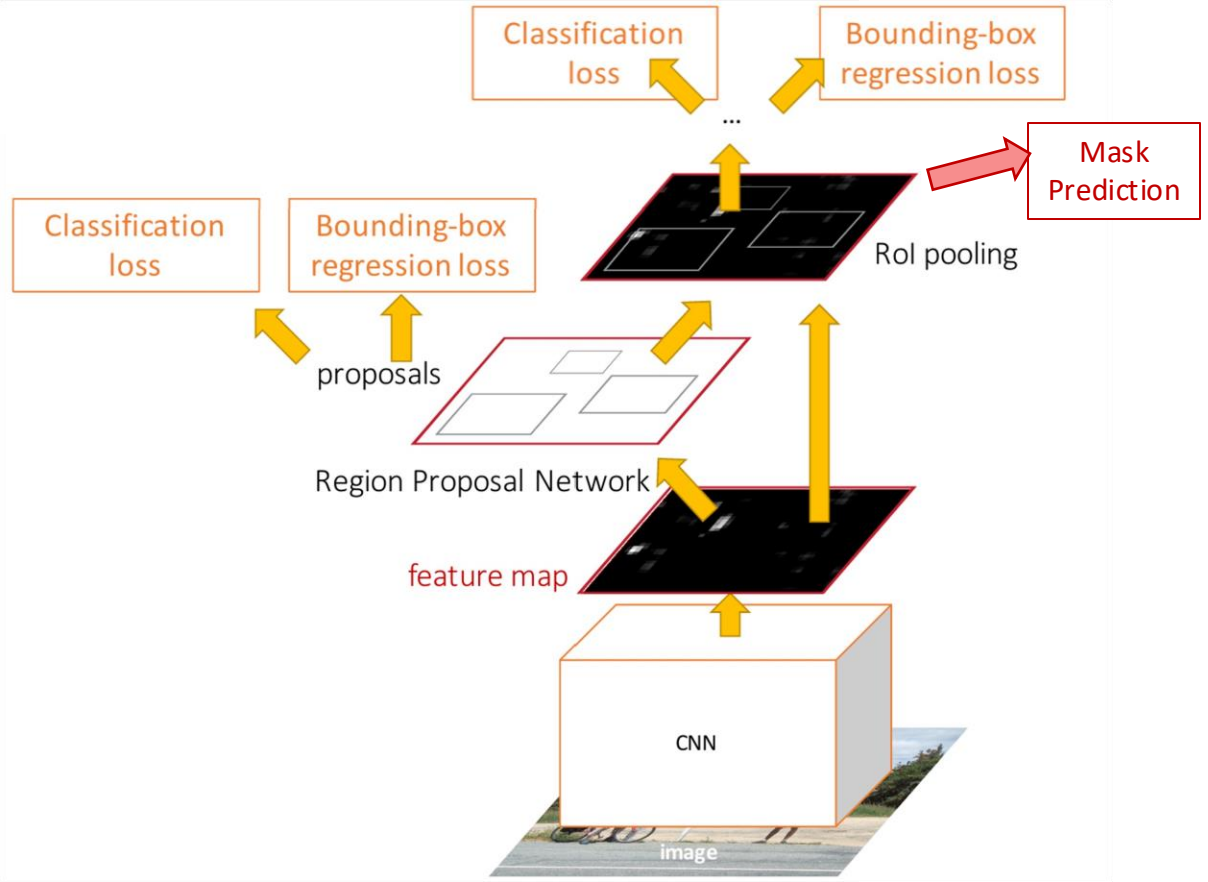
Output: Detection of each *object* in the image and class label for of each of their pixels

Side note:
Semantic Segmentation +
Instance Segmentation =
Panoptic Segmentation





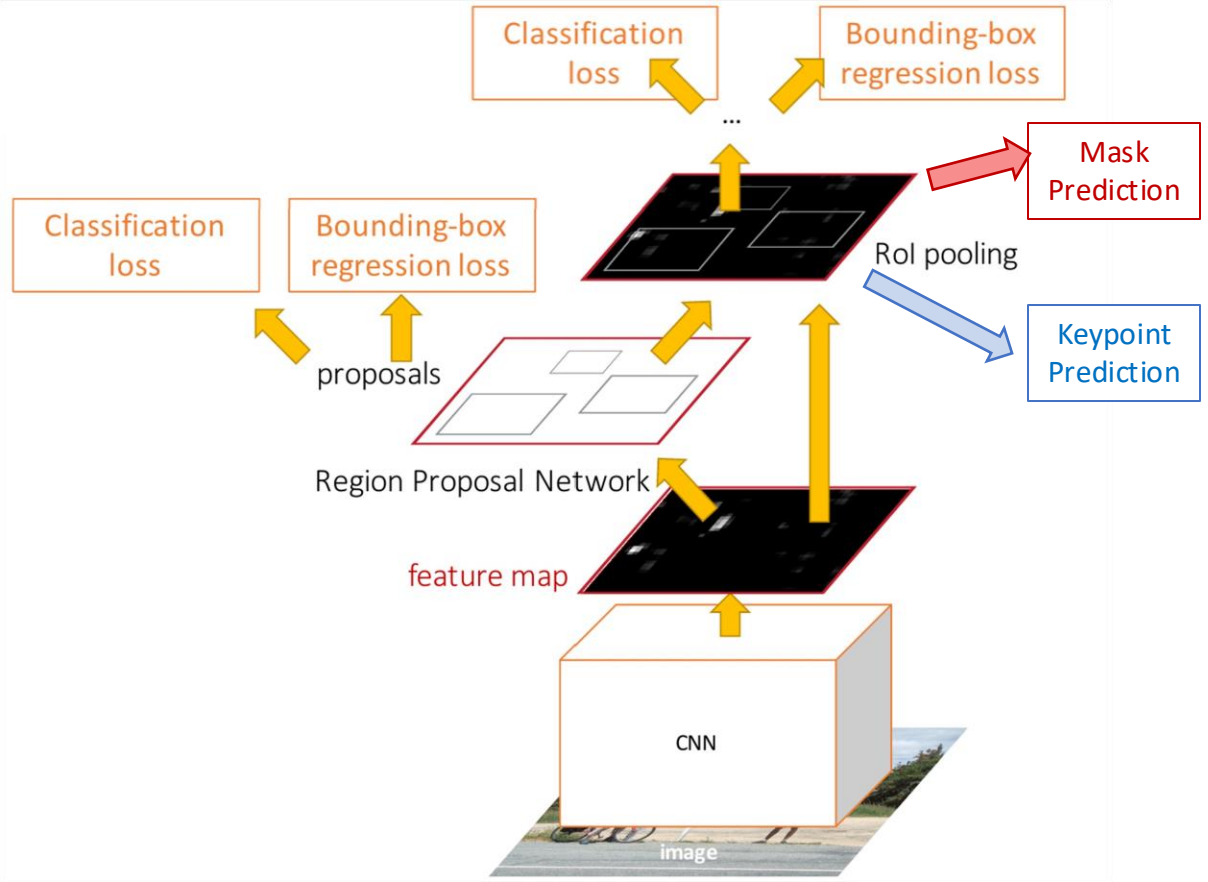
Instance Segmentation by extending detection



Mask R-CNN

Add a small mask network that operates on each RoI and predicts a 28x28 binary mask for each of the C classes in your dataset

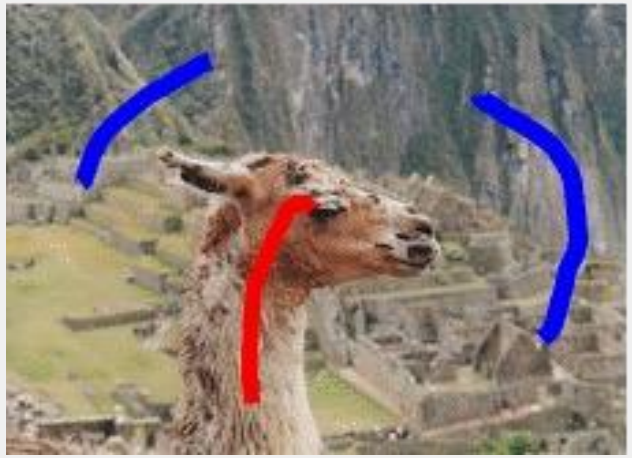
Not only Instance Segmentation by extending detection



Mask R-CNN

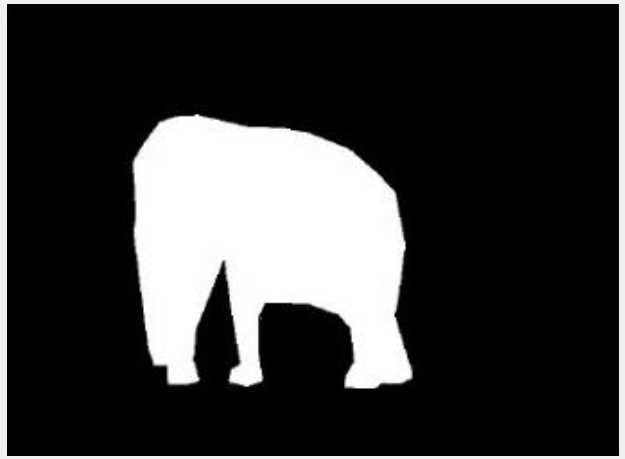
Add a small keypoint prediction network that operates on each RoI and predict a 56x56 binary mask for each of the K keypoints defining the pose

Interactive Segmentation



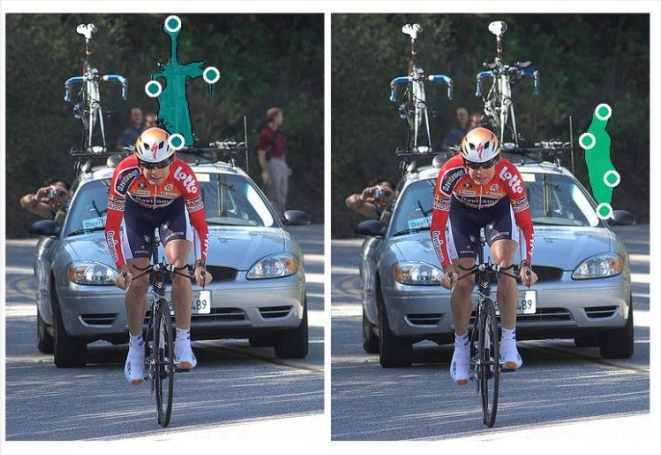
Scribbles (e.g., GraphCut)

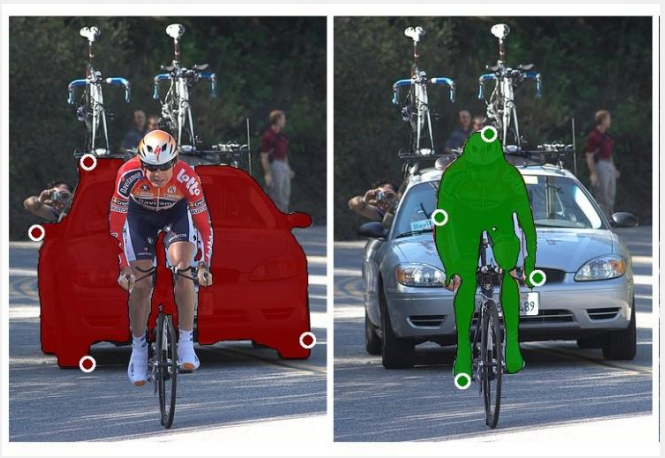
Background/foreground



Two points

Background/foreground

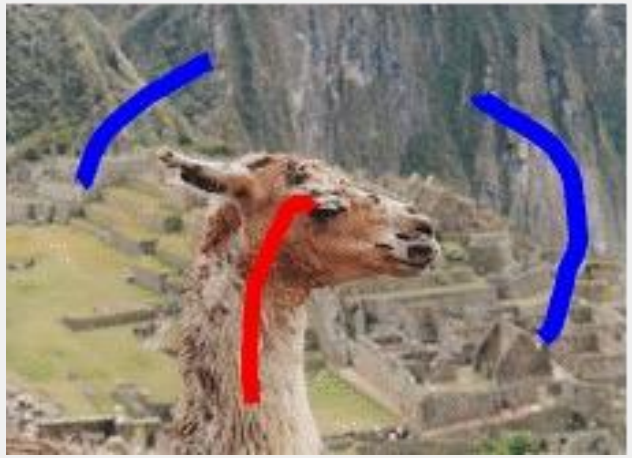




Few points

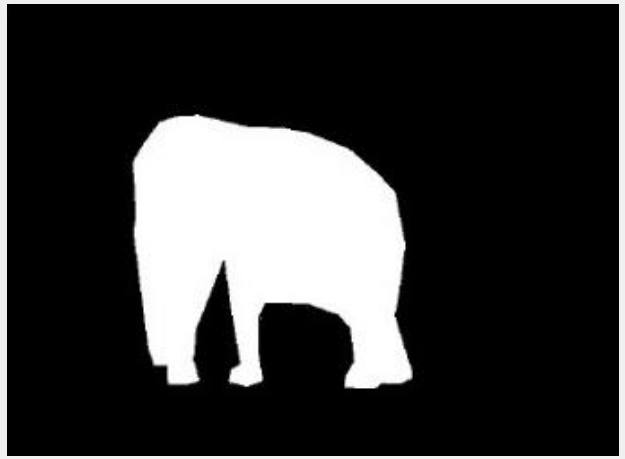
obj1/obj2/obj3/.../objN

Interactive Segmentation



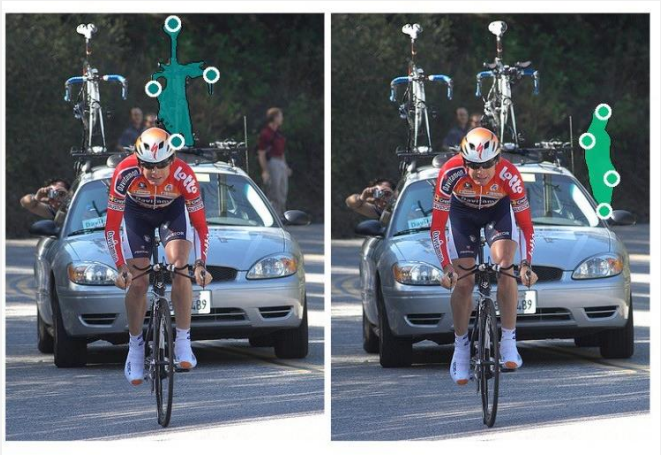
Scribbles (e.g., GraphCut)

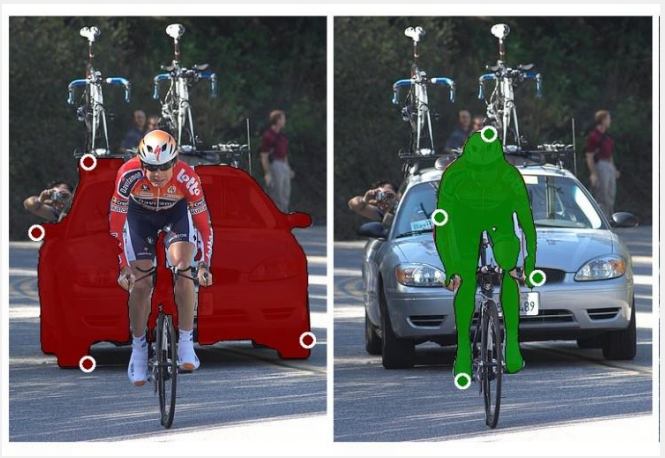
Background/foreground



Two points

Background/foreground





Few points

obj1/obj2/obj3/.../objN

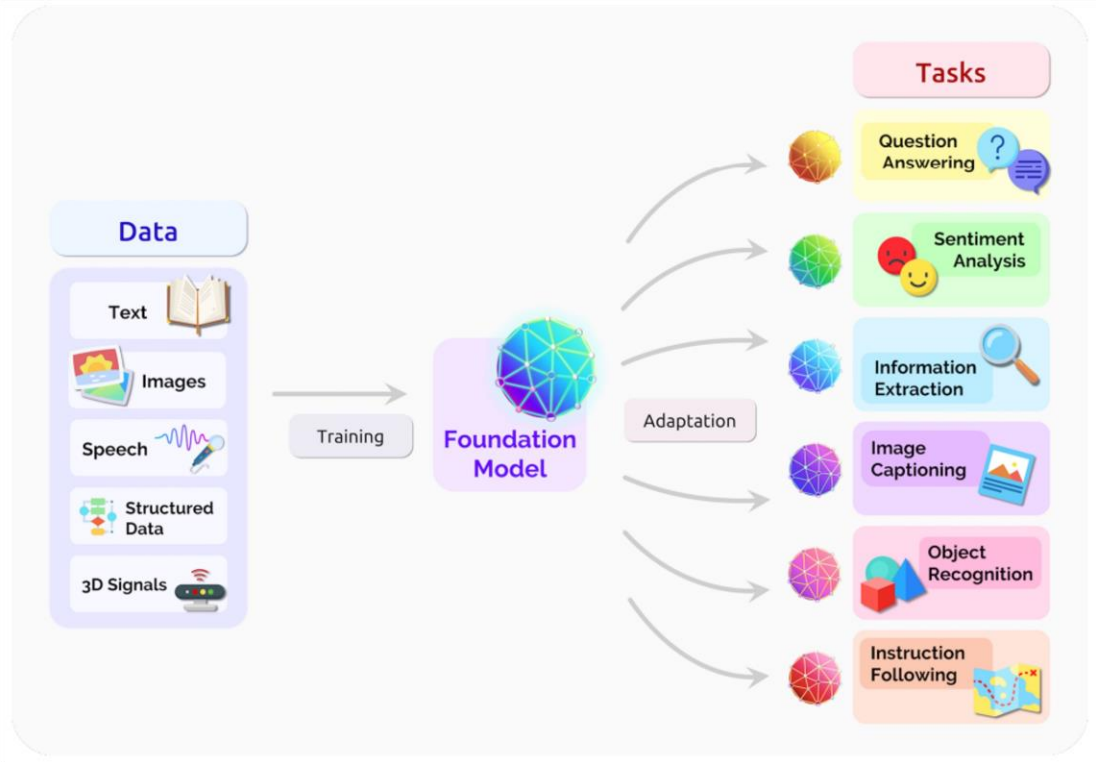
Promptable Segmentation

Foundation models are large Artificial Intelligence models trained on broad data that can:

- produce/generate wide variety of outputs.
- adapt to a wide range of downstream tasks.
- generalize beyond training data distributions.

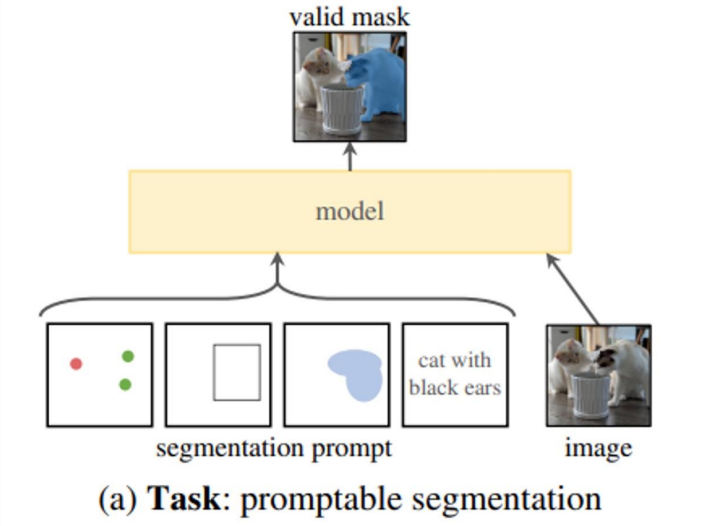
→ A foundation model can centralize the information from various modalities.

Segment Anything Model (SAM): the first foundation model for promptable segmentation

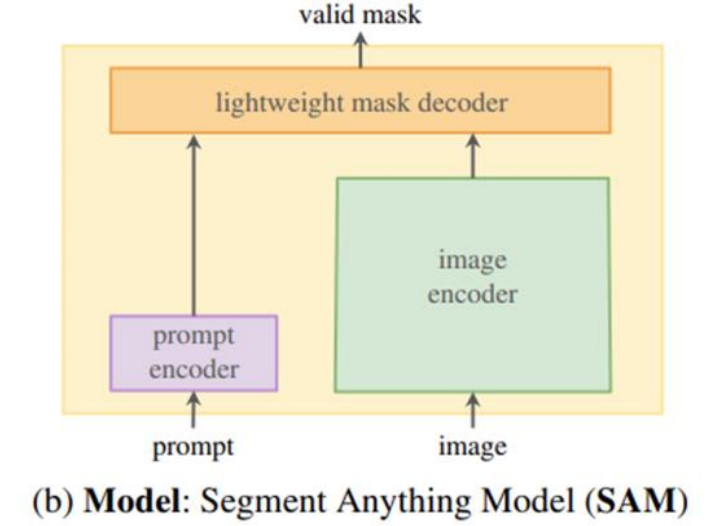


Promptable Segmentation

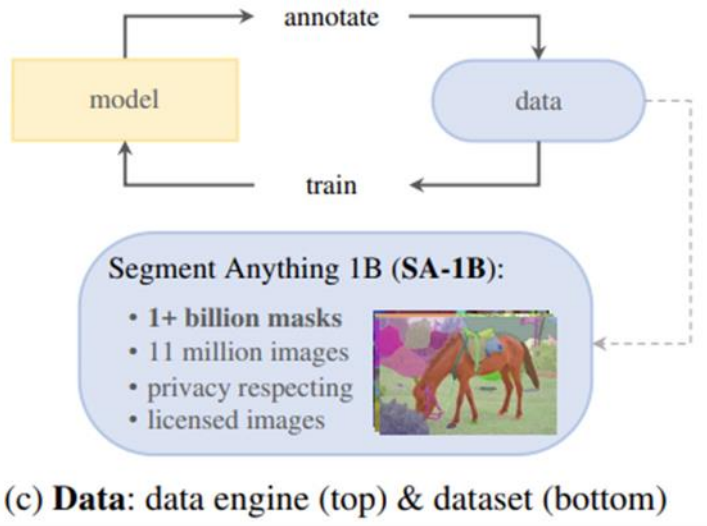
SAM is built with three interconnected components: A **task**, a **model**, and a **data engine**.



(a) **Task**: promptable segmentation

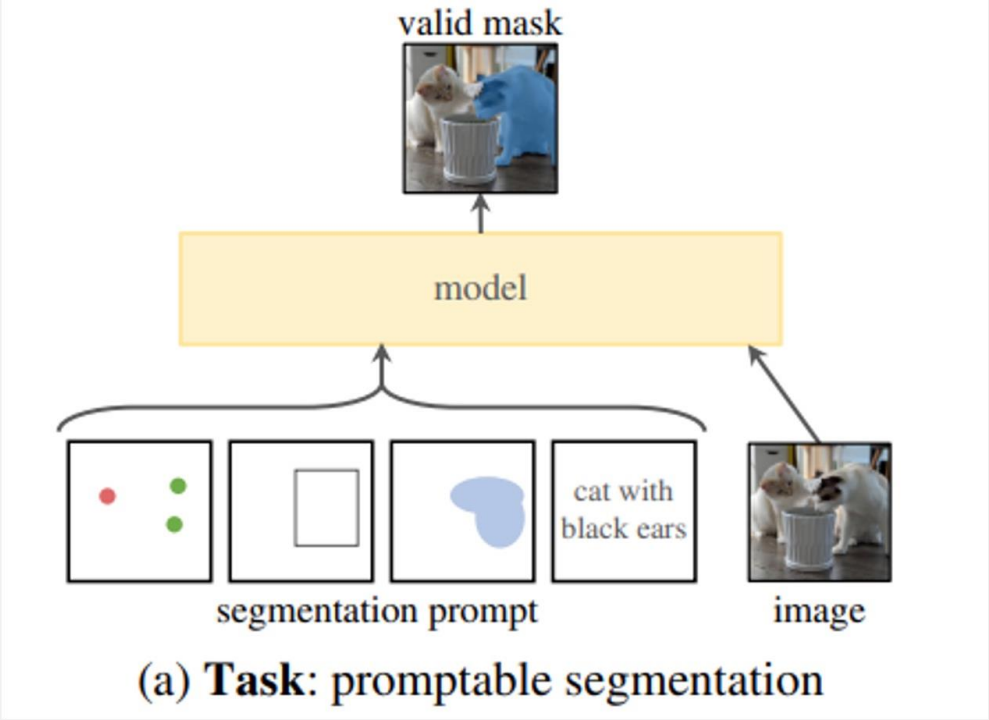


(b) **Model**: Segment Anything Model (**SAM**)



(c) **Data**: data engine (top) & dataset (bottom)

Promptable Segmentation



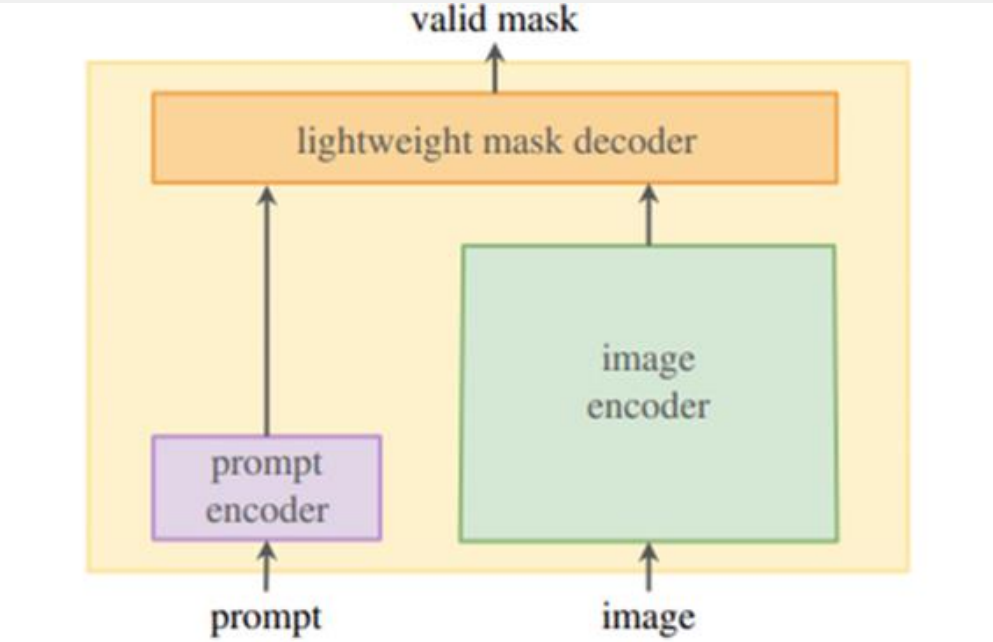
(a) **Task**: promptable segmentation

Promptable Segmentation Task: Given any segmentation prompt that specifies what to segment in an image, the goal is to return a valid segmentation mask.

SAM considers two sets of prompts: **sparse** (clicks, boxes, text) and **dense** (masks).

SAM's promptable design enables flexible integration with other systems (i.e., used as component in larger systems).

Promptable Segmentation



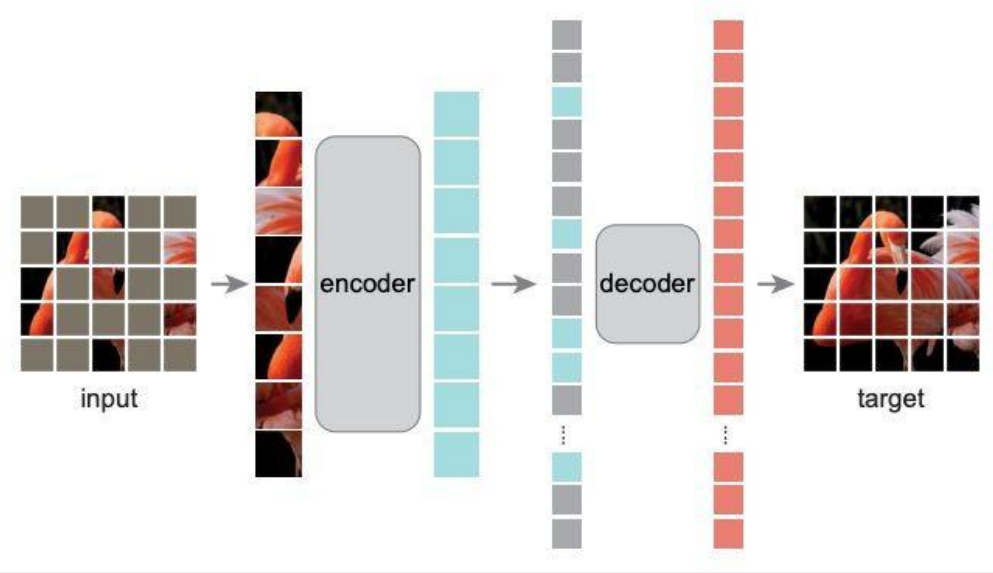
(b) **Model**: Segment Anything Model (**SAM**)

Segment Anything Model:

- A heavyweight image encoder outputs an image embedding.
- A lightweight prompt encoder embeds prompts and efficiently queries the image embedding.
- A lightweight mask decoder produces object masks and confidence scores by combining the two information sources.

SAM supports flexible prompts, works in real-time to allow interactive use, and is ambiguity-aware

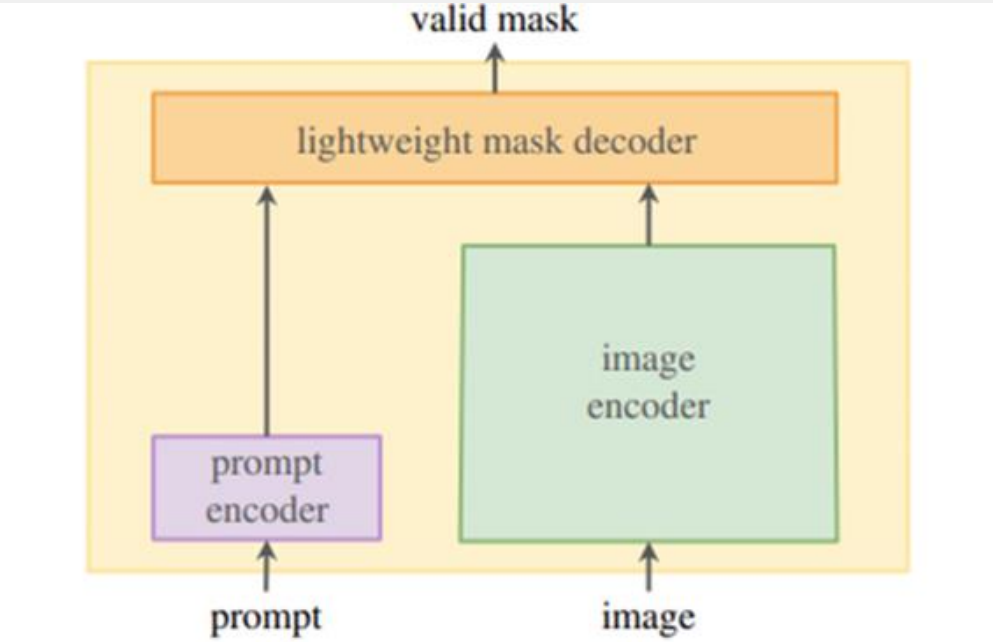
Promptable Segmentation



Segment Anything Model:

- A heavyweight image encoder outputs an image embedding.

Promptable Segmentation

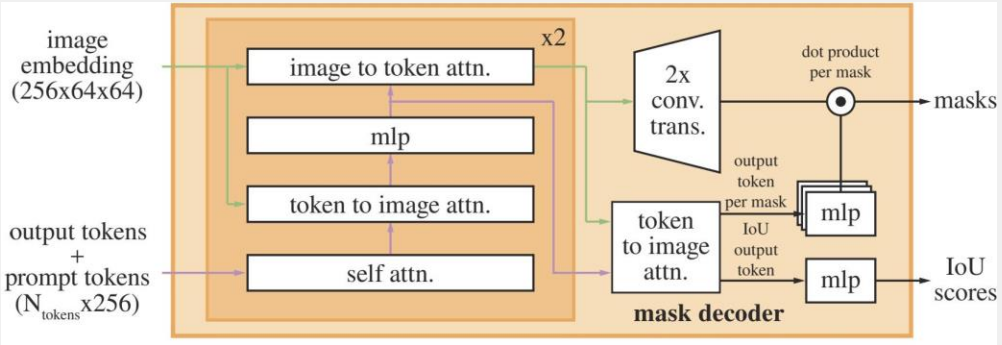


(b) **Model**: Segment Anything Model (**SAM**)

Segment Anything Model:

- A heavyweight image encoder outputs an image embedding.
- A lightweight prompt encoder embeds prompts and efficiently queries the image embedding:
  - Points and boxes: positional encodings + embeddings for each prompt type.
  - Free-form text: text encoder from CLIP.
  - Masks: convolutions and summed element-wise with the image embedding.

Promptable Segmentation



Segment Anything Model:

- A heavyweight image encoder outputs an image embedding.
- A lightweight prompt encoder embeds prompts and efficiently queries the image embedding.
- A lightweight mask decoder produces object masks and confidence scores by combining the two information sources.

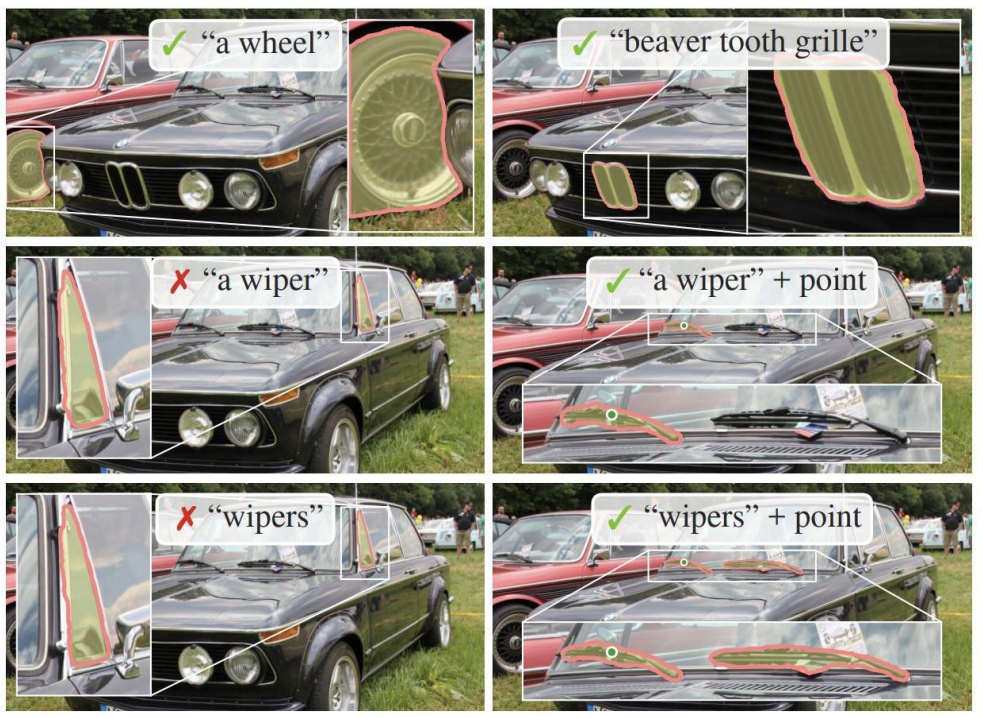
Promptable Segmentation

Segment Anything Model:

- A heavyweight image encoder outputs an image embedding.
- A lightweight prompt encoder embeds prompts and efficiently queries the image embedding.
- A lightweight mask decoder produces object masks and confidence scores by combining the two information sources.

SAM supports flexible prompts, works in real-time to allow interactive use, and is **ambiguity-aware** (it is designed to predict multiple masks for a single prompt, so to naturally handles ambiguity, such as the shirt vs. person).

Promptable Segmentation

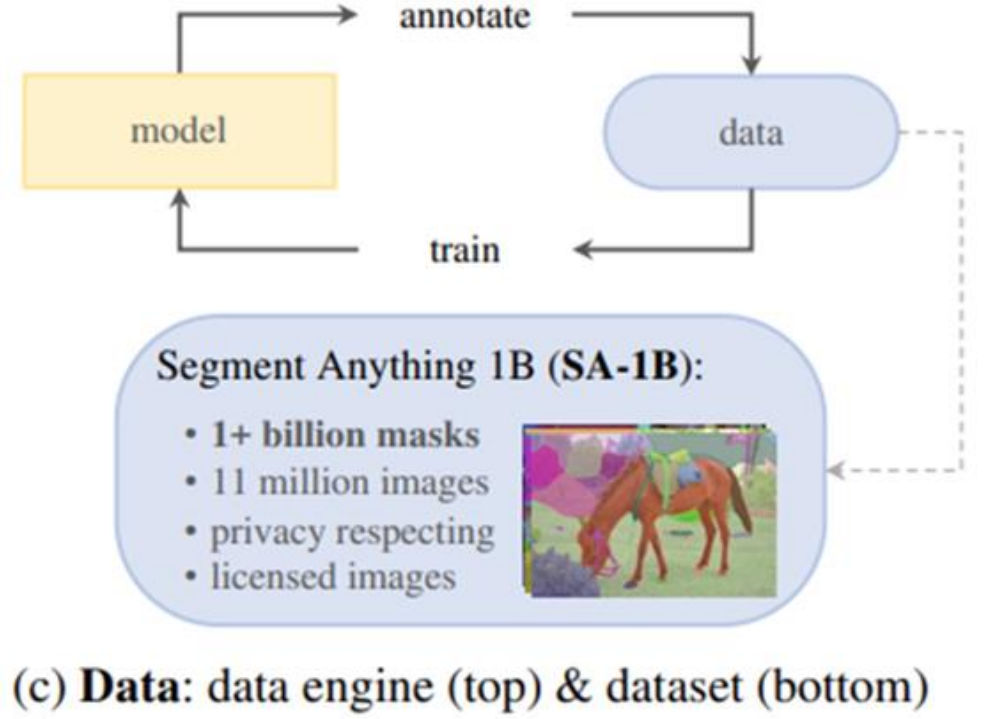


Segment Anything Model:

- A heavyweight image encoder outputs an image embedding.
- A lightweight prompt encoder embeds prompts and efficiently queries the image embedding.
- A lightweight mask decoder produces object masks and confidence scores by combining the two information sources.

SAM **supports flexible prompts**, works in real-time to allow interactive use, and is ambiguity-aware (it is designed to predict multiple masks for a single prompt, so to naturally handles ambiguity, such as the shirt vs. person).

Promptable Segmentation



(c) **Data**: data engine (top) & dataset (bottom)

SA-1B dataset:
To achieve strong generalization to new data distributions, it is necessary to train SAM on a large and diverse set of masks, beyond any segmentation dataset that was already exists.

The authors built a dataset that includes more than 1B high-resolution (3300×4950 pixels on average) masks from 11M licensed and privacy-preserving images.
All masks were generated fully automatically with a model-in-the-loop strategy.

Promptable Segmentation

Try the demo:
https://segment-anything.com/

Reach out to us @ *info@minerva4ai.eu*

Thank you

This project has received funding from the European High-Performance Computing Joint Undertaking (JU) under grant agreement No 101182737. The JU receives support from the Digital Europe Programme.

