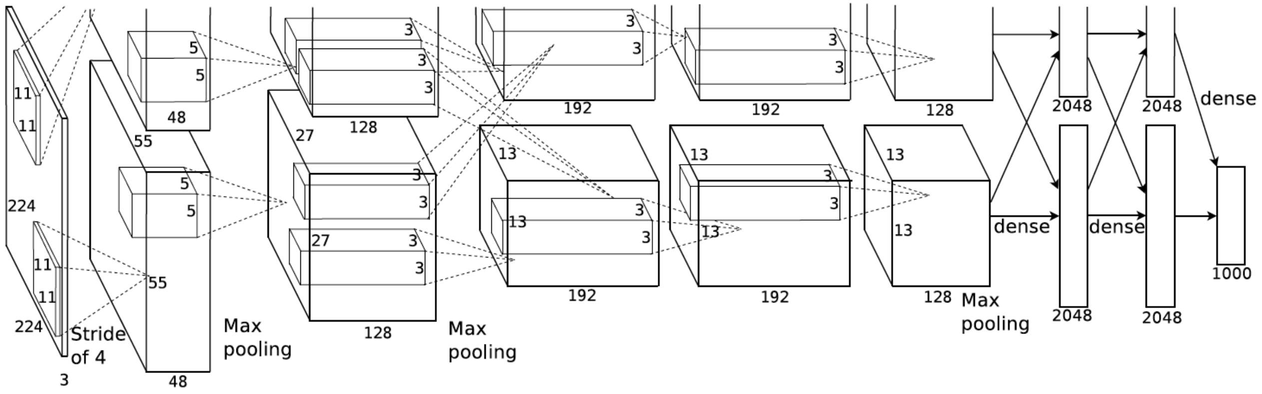
Object detection and segmentation in computer vision

12/04/2022

Colin Decourt (ANITI)

ANITI
ARTIFICIAL & NATURAL INTELLIGENCE TOULOUSE INSTITUTE
Université Fédérale Toulouse Midi-Pyrénées

From classification...



Feature vector

Fully connected

Class scores:

Cat: 0.85

Owl: 0.1

Dog: 0.03

...

Source: Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. 2017. ImageNet classification with deep convolutional neural networks. *Commun. ACM* 60, 6 (June 2017), 84–90. DOI:https://doi.org/10.1145/3065386

… to object detection and segmentation

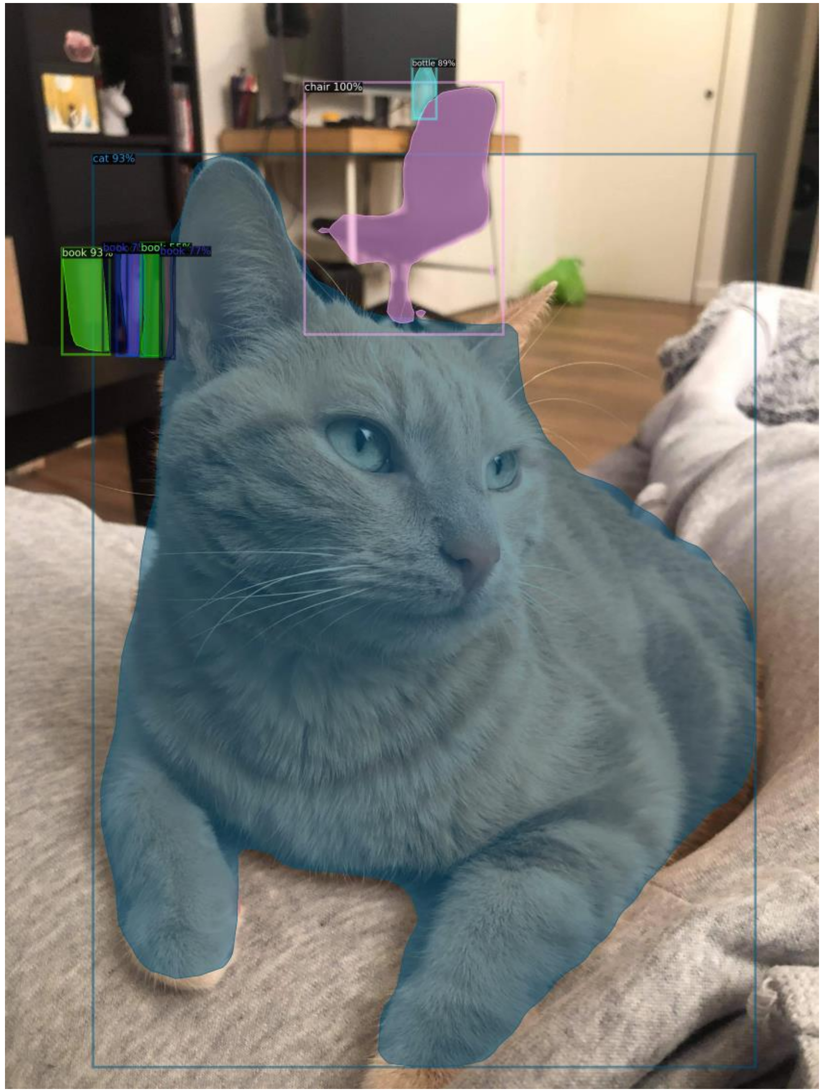
- Classification and localization: classify and localize **ONE** object in an image

- Object detection: draw bounding boxes around **multiple objects** of different classes in an image (find the edge contour of the object of interest)

- Instance segmentation: pixel level colouring of **multiple objects** of different classes in an image

- Semantic segmentation: assign each pixel in the image to a category label (cars, buildings, ground, sky, etc.)

- Panoptic segmentation: combination of semantic segmentation and instance segmentation



What is object detection?

- The task of assigning a label and a bounding box to all objects in the image

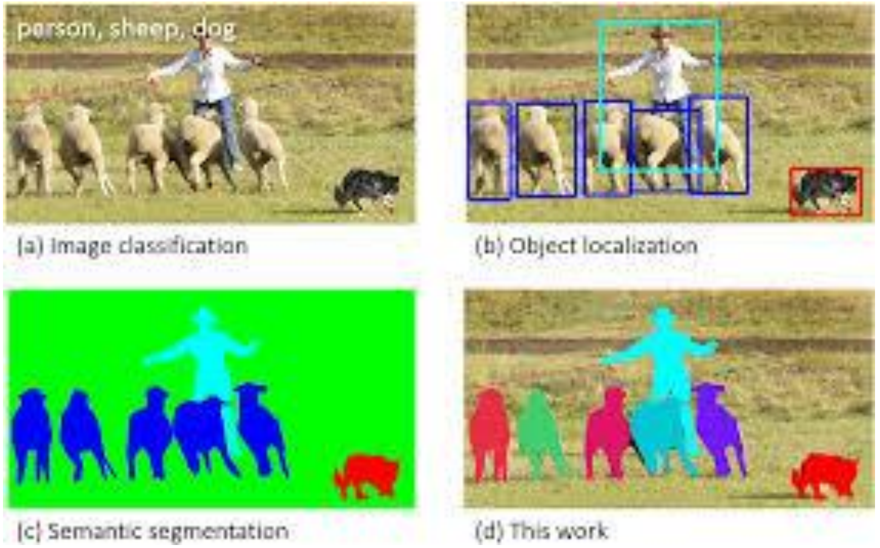
- Input: an RGB image

- Output for each object predict:
  - Category label
  - Bounding box: $(x, y, width, height)$

Object detection and segmentation datasets

- Pascal VOC dataset:
  - Detection, classification, segmentation
  - 10000 images with 20 categories

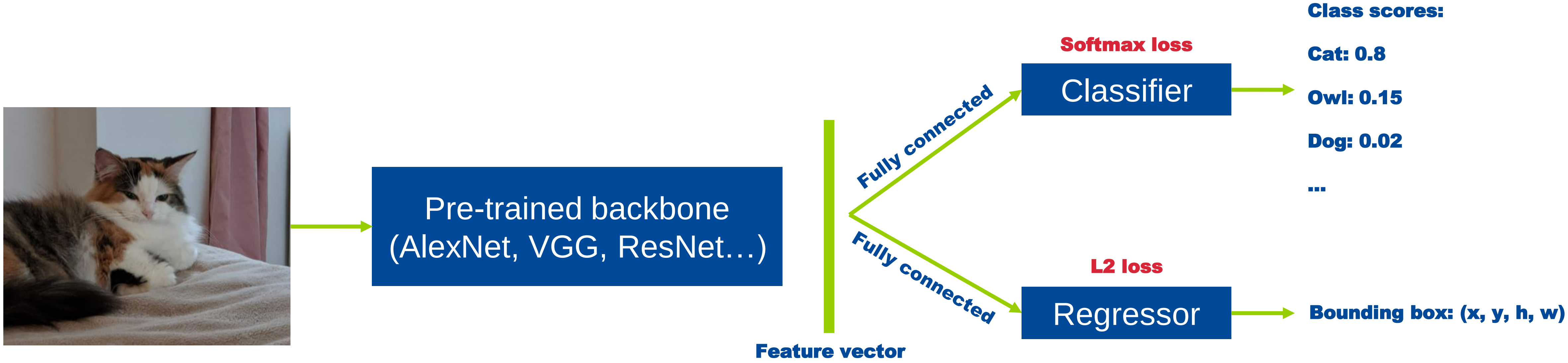
- COCO dataset:
  - Caption generation, object detection, key point detection and object segmentation
  - 120000 images for training / 40000 for validation with 80 categories



- KITTI autonomous driving dataset:
  - Detection, classification, segmentation, tracking…

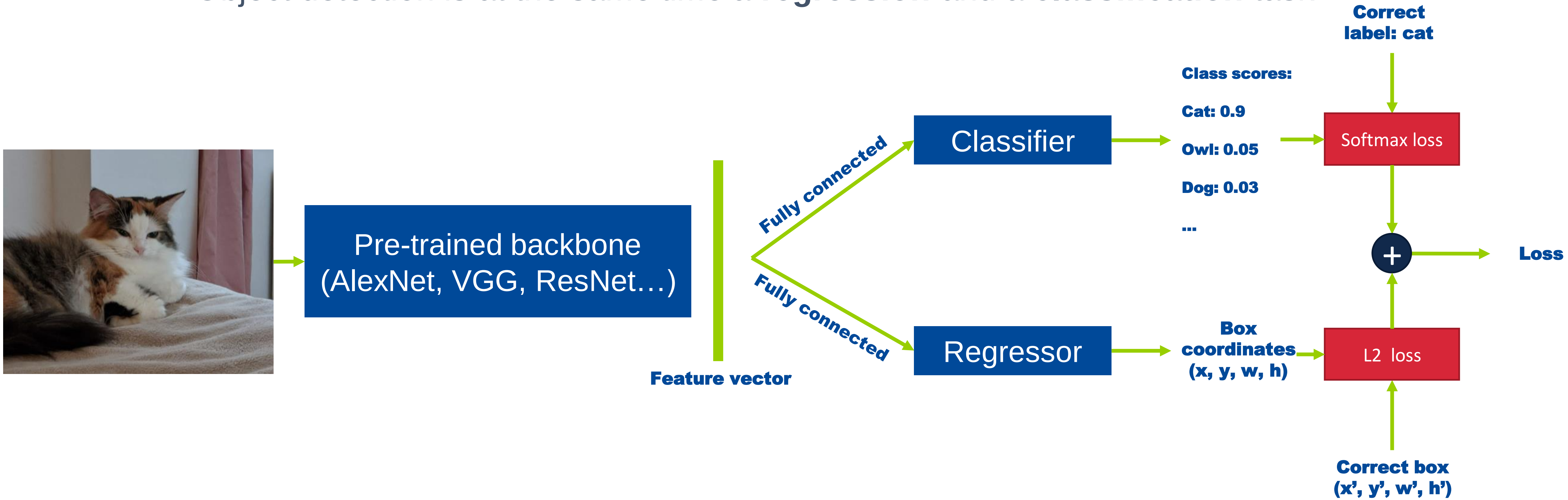
An easy case… Detecting single objects

- Object detection is at the same time a **regression** and a **classification** task



An easy case... Detecting single objects - The multitask loss

- Object detection is at the same time a **regression** and a **classification** task

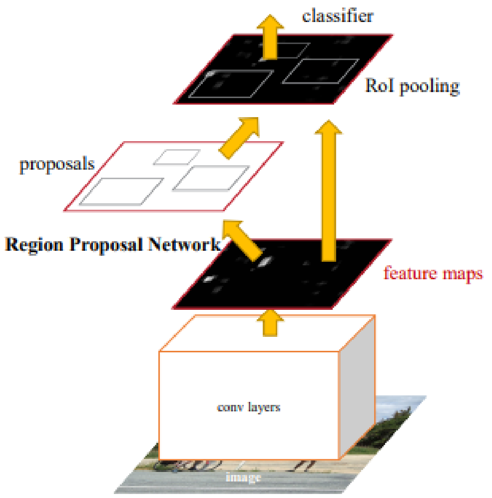


Challenges in object detection

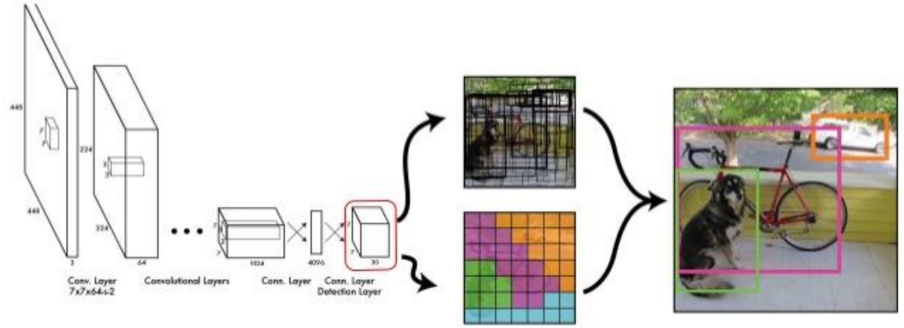
- Multiple outputs: variable number of objects per image ⇨ need different numbers of outputs per image!
- Multiple **types** of output: category label + bounding boxes
- Large images: ~224x224 for ImageNet vs. ~800x600 for MS-COCO

Object detector categorization

- Region proposal based framework: similar to the attentional mechanism of human brain
    1. Find objects in the image (sliding windows, region proposal methods)
    2. Classify the objects

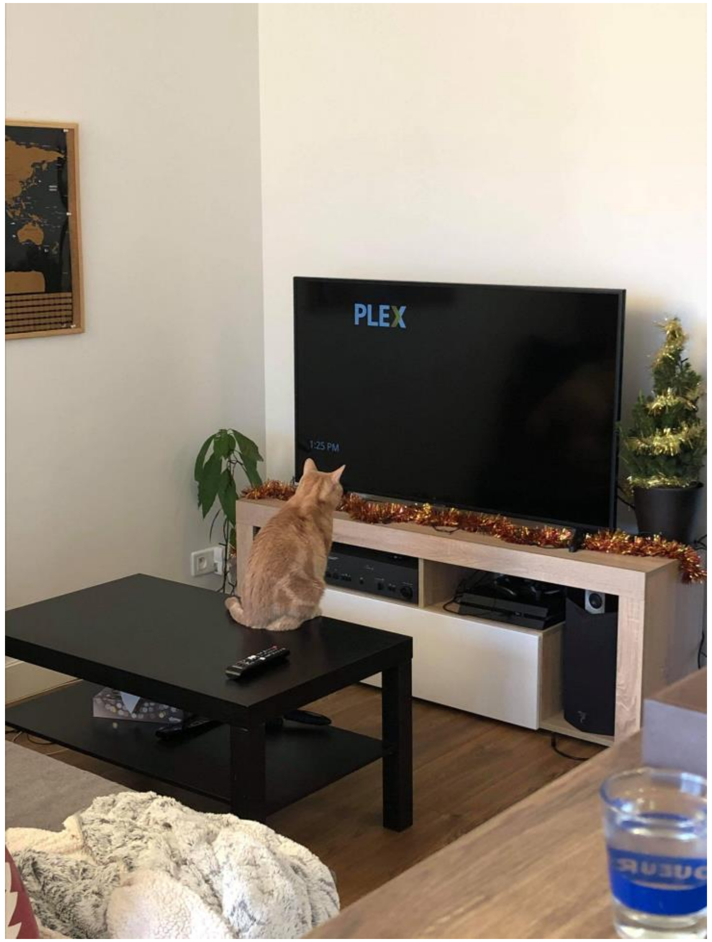


- Single-stage detectors:
    - Map image pixels to bounding box coordinates and class probabilities



The sliding window, a naive solution to multiple object detection

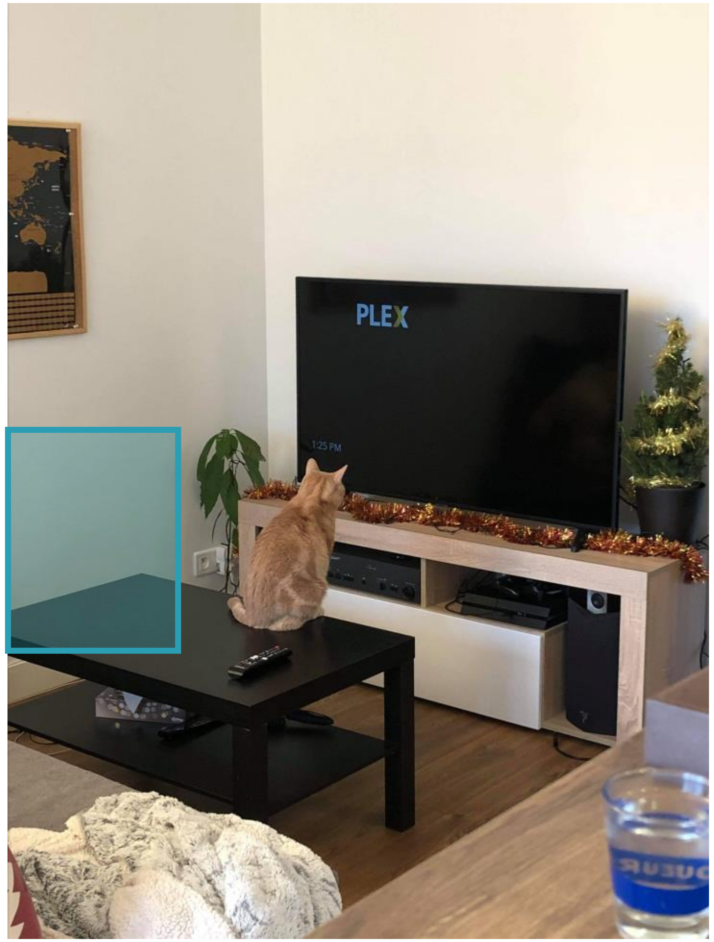
- Idea of the sliding window:
  - Apply a CNN to many different crops of the image and classify each crop as object or background



Adapted from: https://web.eecs.umich.edu/~justincj/slides/eecs498/498_FA2019_lecture15.pdf

The sliding window, a naive solution to multiple object detection

- Idea of the sliding window:
  - Apply a CNN to many different crops of the image and classify each crop as object or background



Adapted from: https://web.eecs.umich.edu/~justincj/slides/eecs498/498_FA2019_lecture15.pdf

The sliding window, a naive solution to multiple object detection

- Idea of the sliding window:
    - Apply a CNN to many different crops of the image and classify each crop as object or background

Adapted from: https://web.eecs.umich.edu/~justincj/slides/eecs498/498_FA2019_lecture15.pdf

The sliding window, a naive solution to multiple object detection

- Idea of the sliding window:
  - Apply a CNN to many different crops of the image and classify each crop as object or background

Adapted from: https://web.eecs.umich.edu/~justincj/slides/eecs498/498_FA2019_lecture15.pdf

The sliding window, a naive solution to multiple object detection

- Idea of the sliding window:
    - Apply a CNN to many different crops of the image and classify each crop as object or background

Adapted from: https://web.eecs.umich.edu/~justincj/slides/eecs498/498_FA2019_lecture15.pdf

The sliding window, a naive solution to multiple object detection

- Idea of the sliding window:
  - Apply a CNN to many different crops of the image and classify each crop as object or background
- Question:
  - How many possible boxes are there in an image of size HxW?
- Total possible boxes:

$$\sum_{h=1}^{H}\sum_{w=1}^{W}(W-w+1)(H-h+1) = \frac{H(H+1)W(W+1)}{2}$$

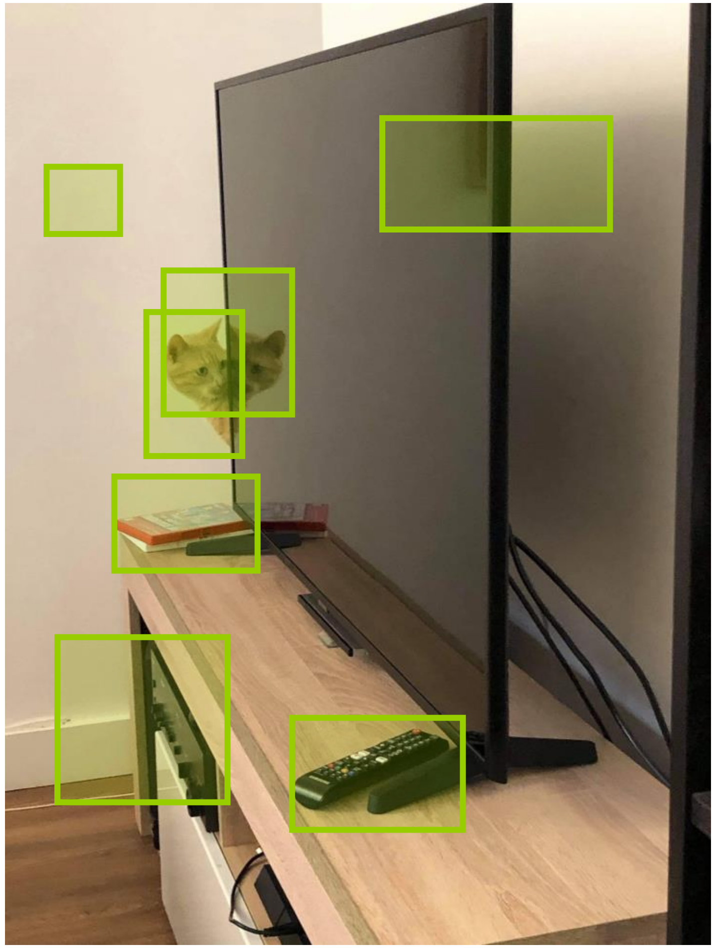
- For a 800x600 image: 58M boxes

Adapted from: https://web.eecs.umich.edu/~justincj/slides/eecs498/498_FA2019_lecture15.pdf

R-CNN (Girshick et al., 2013) and the region proposals

- Region proposals:
  - Find a small set of boxes that are likely to cover all objects
  - Often based on heuristics: look for "blob-like" image regions
  - Relatively fast to run: Selective Search algorithm gives 2000 region proposals in a few seconds on CPU

- R-CNN (R. Girshick et al., 2013):
  - First model to use region search and then perform the classification
  - Use the Selective Search algorithm to propose regions



Selective search algorithm

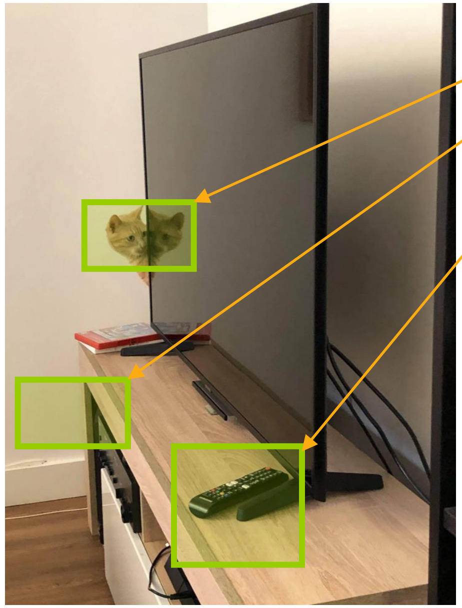
1. Generate initial sub-segmentation, many candidate regions generation

2. Use greedy algorithm to recursively combine similar region into larger ones
    1. From set of regions, choose two that are most similar.
    2. Combine them into a single, larger region.
    3. Repeat the above steps for multiple iterations.

3. Use the generated regions to produce the final candidate region proposals

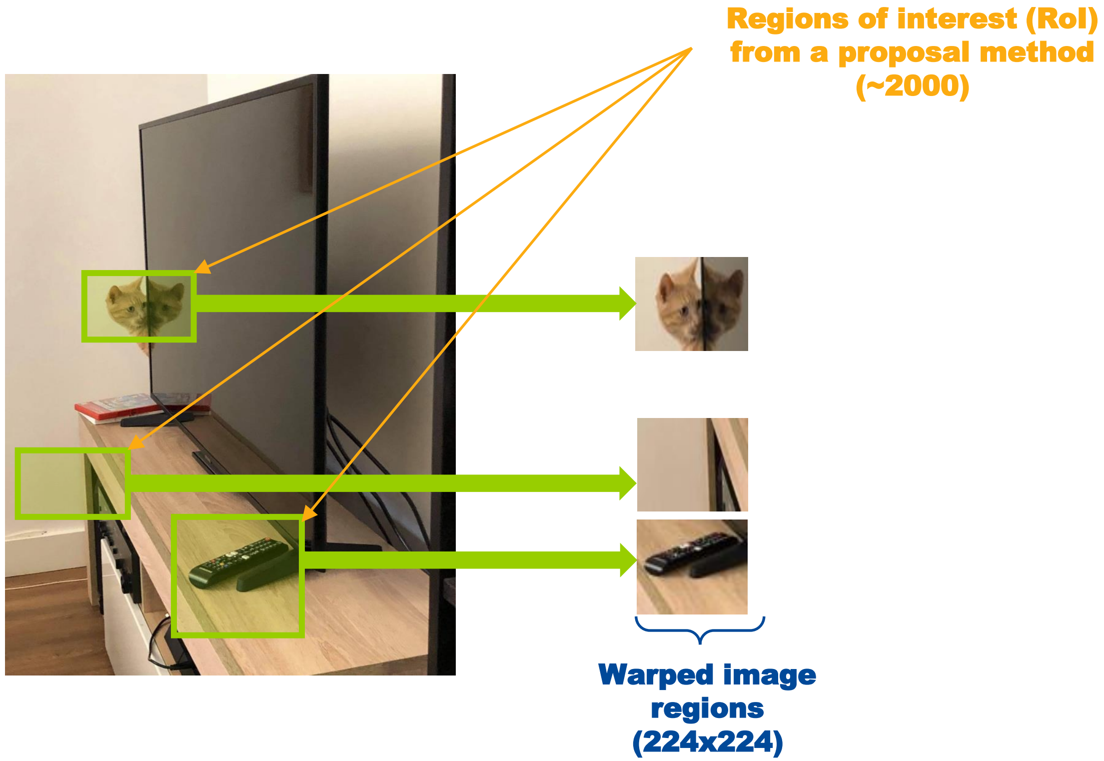
R-CNN (Girshick et al., 2013) and the region proposals

Regions of interest (RoI)
from a proposal method
(~2000)



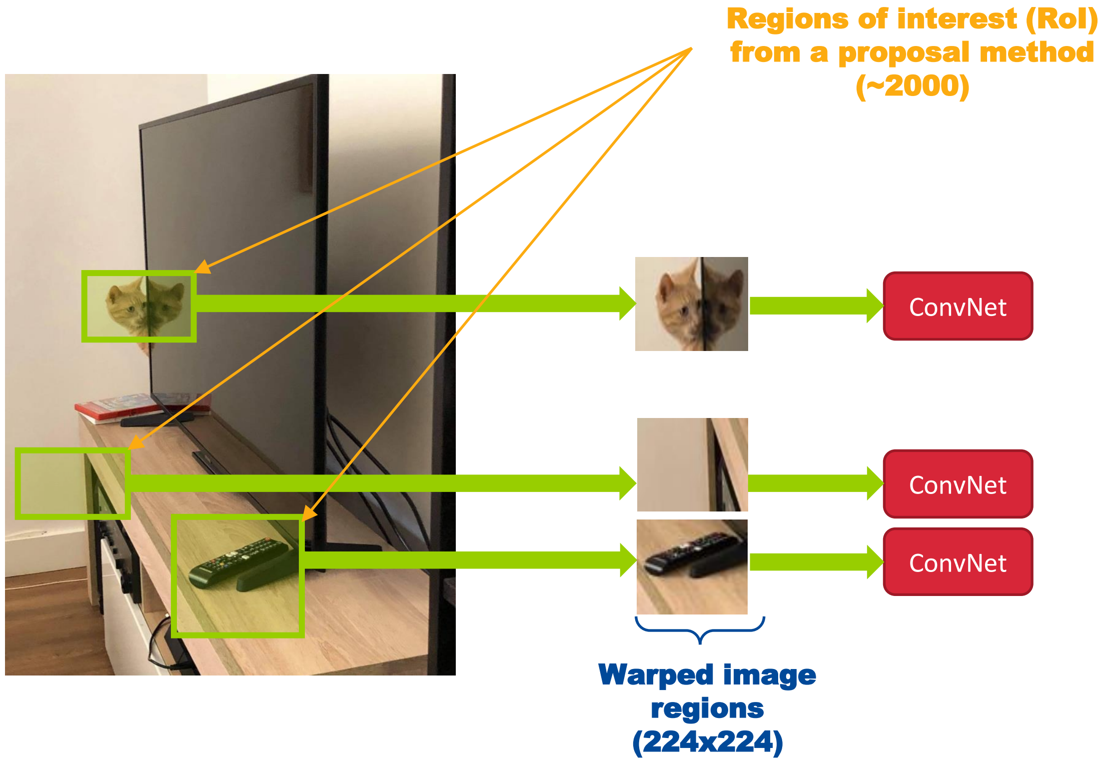
Adapted from: https://web.eecs.umich.edu/~justincj/slides/eecs498/498_FA2019_lecture15.pdf

R-CNN (Girshick et al., 2013) and the region proposals



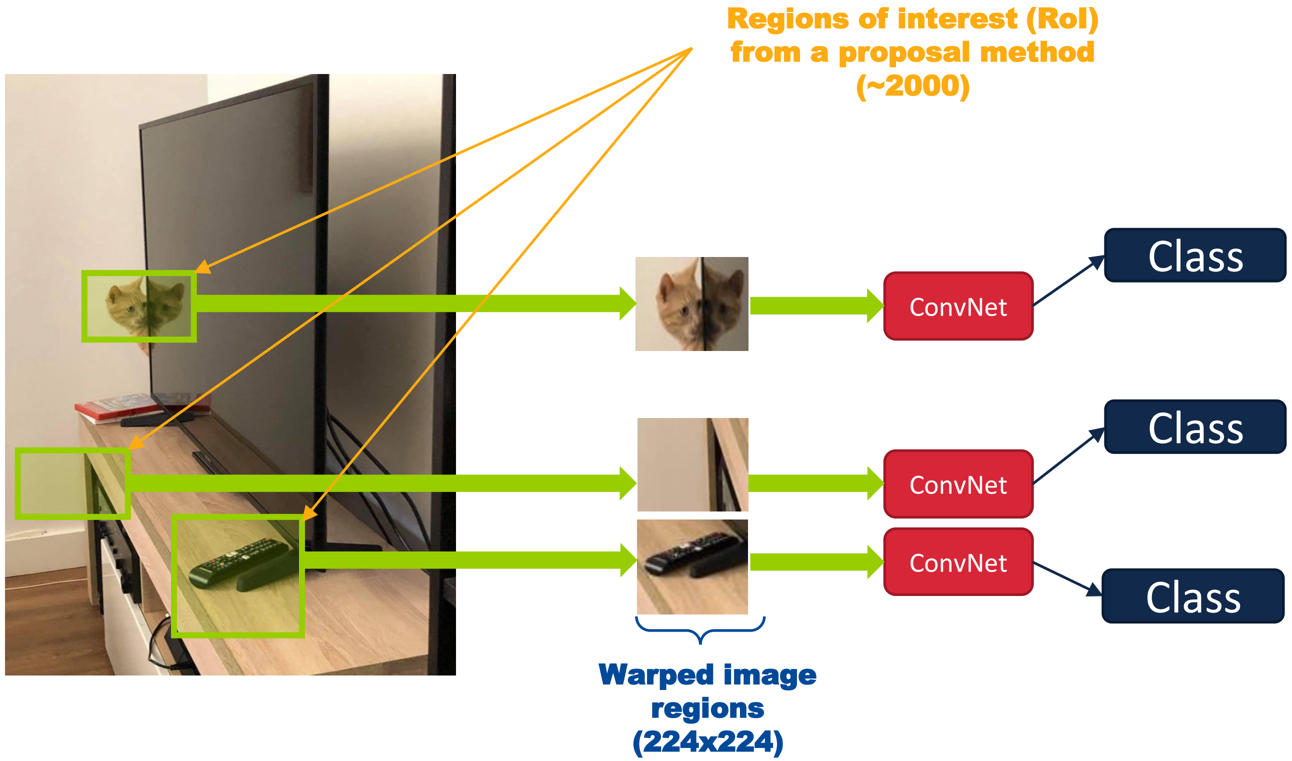
Adapted from: https://web.eecs.umich.edu/~justincj/slides/eecs498/498_FA2019_lecture15.pdf

R-CNN (Girshick et al., 2013) and the region proposals



Adapted from: https://web.eecs.umich.edu/~justincj/slides/eecs498/498_FA2019_lecture15.pdf

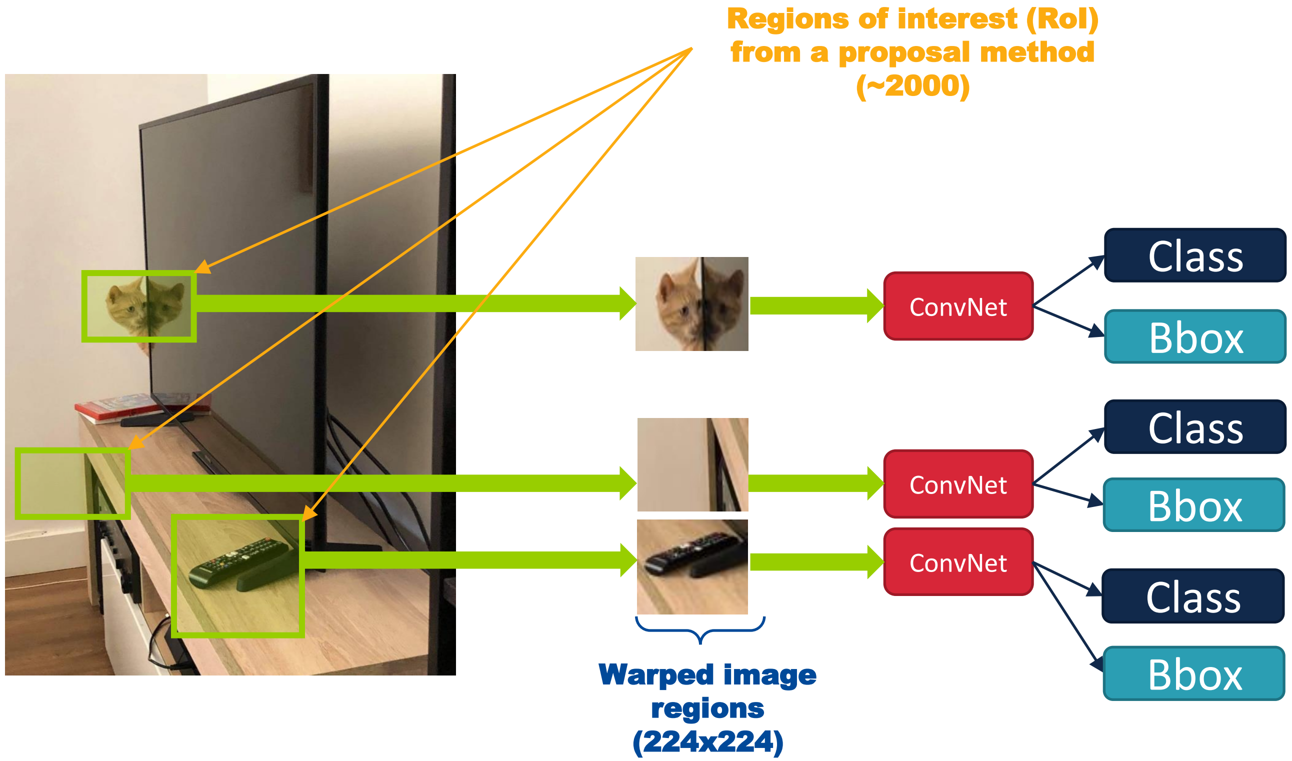
R-CNN (Girshick et al., 2013) and the region proposals



- Classify EACH proposed regions (SVM)

Adapted from: https://web.eecs.umich.edu/~justincj/slides/eecs498/498_FA2019_lecture15.pdf

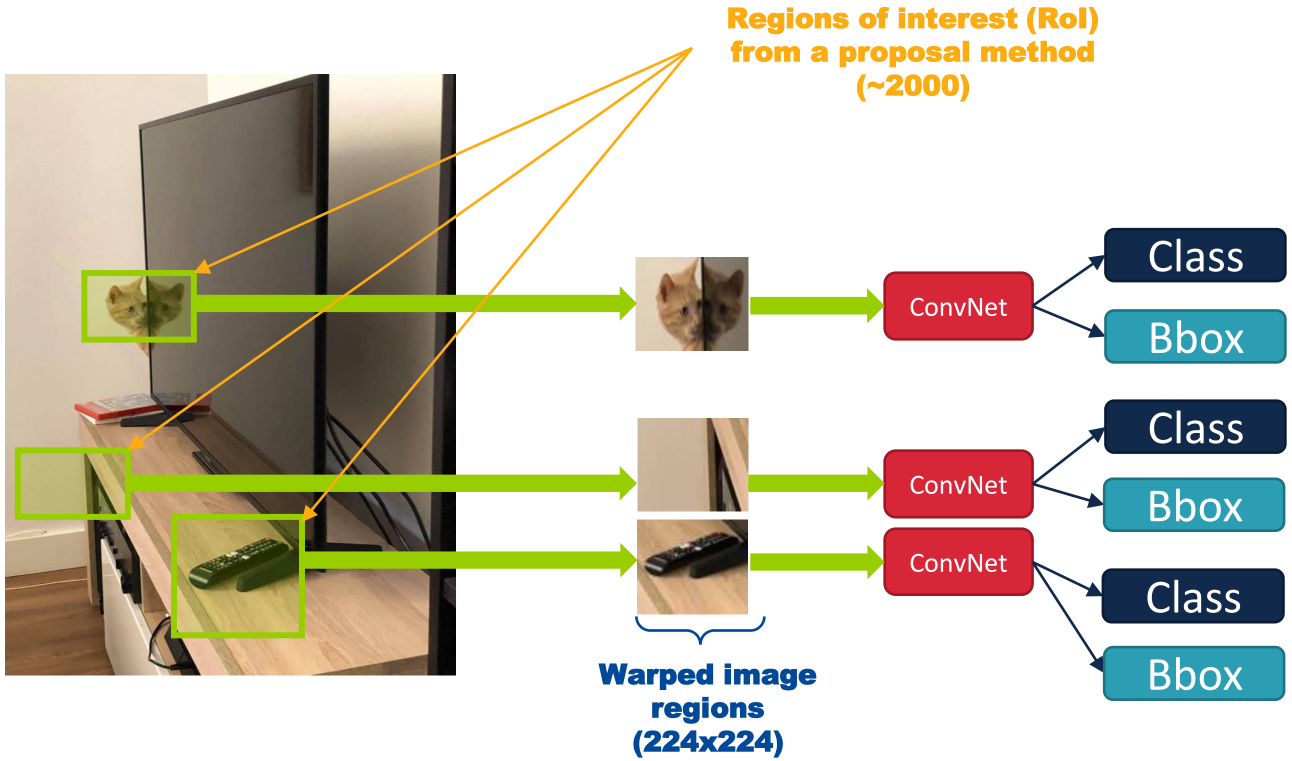
R-CNN (Girshick et al., 2013) and the region proposals



- Classify EACH proposed regions (SVM)
- Bounding box regression:
  - Predict "transform" to correct the proposed RoI
  - 4 numbers: (t_x, t_y, t_h, t_w)

Adapted from: https://web.eecs.umich.edu/~justincj/slides/eecs498/498_FA2019_lecture15.pdf

R-CNN (Girshick et al., 2013) and the region proposals



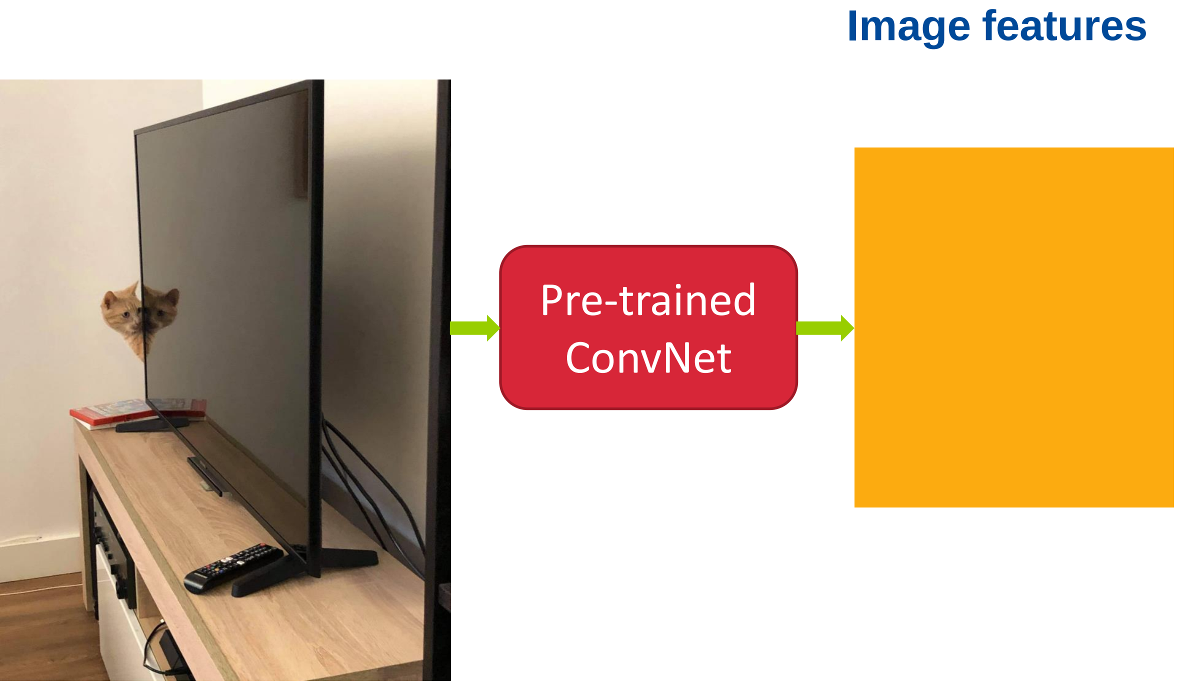
- Classify EACH proposed regions (SVM)
- Bounding box regression:
  - Predict “transform” to correct the proposed RoI
  - 4 numbers: (t_x, t_y, t_h, t_w)
- Final output:
  - Proposal: (p_x, p_y, p_h, p_w)
  - Transform: (t_x, t_y, t_h, t_w)
  - Output box: (b_x, b_y, b_h, b_w)
    - $b_x = p_x + p_w t_x$ and $b_y = p_y + p_h t_y$
    - $b_w = p_w e^{t_w}$ and $b_h = p_h e^{t_h}$

Adapted from: https://web.eecs.umich.edu/~justincj/slides/eecs498/498_FA2019_lecture15.pdf

RCNN drawbacks

- Takes a huge amount of time to train networks (2000 RoIs/image)
- Cannot be implemented in real time (47s/image in inference)
- Selective Search algorithm can lead to poor region candidates

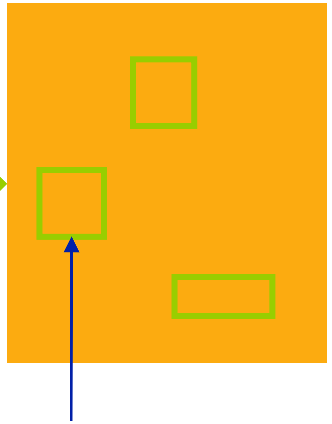
Fast R-CNN (Girshick, 2015) – A slight improvement



Fast R-CNN (Girshick, 2015) – A slight improvement

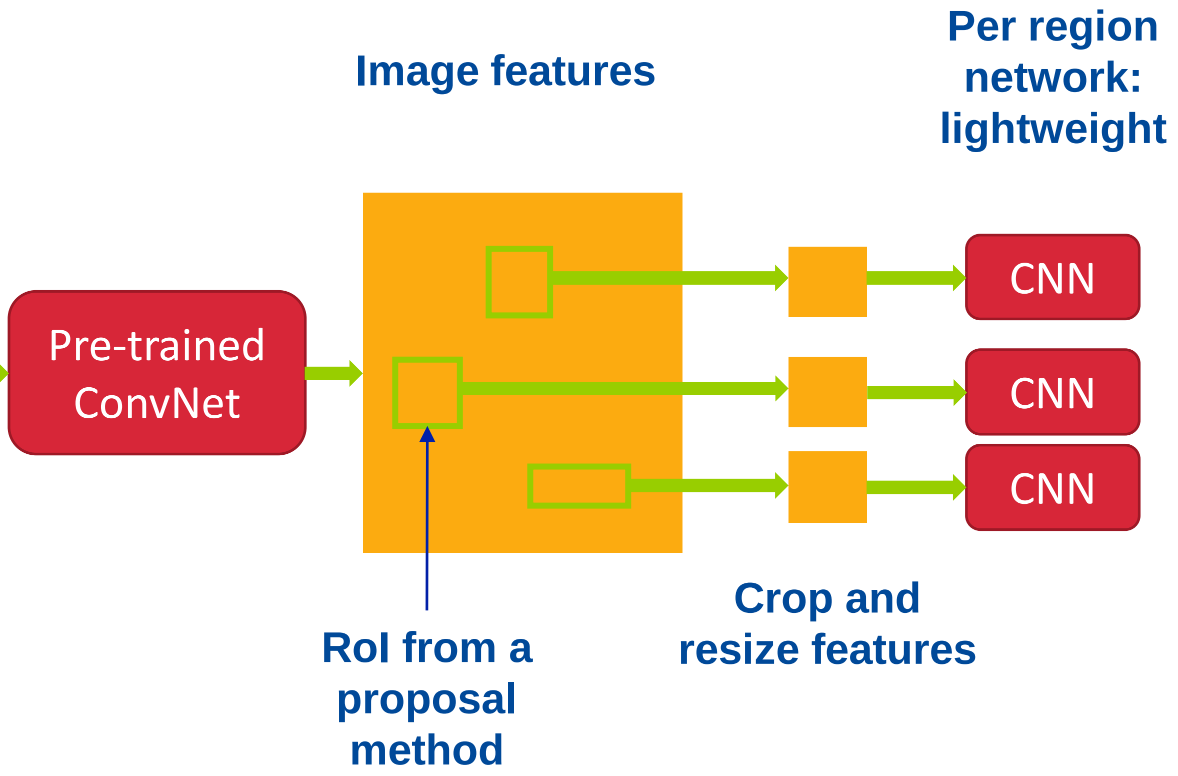
Pre-trained ConvNet

Image features

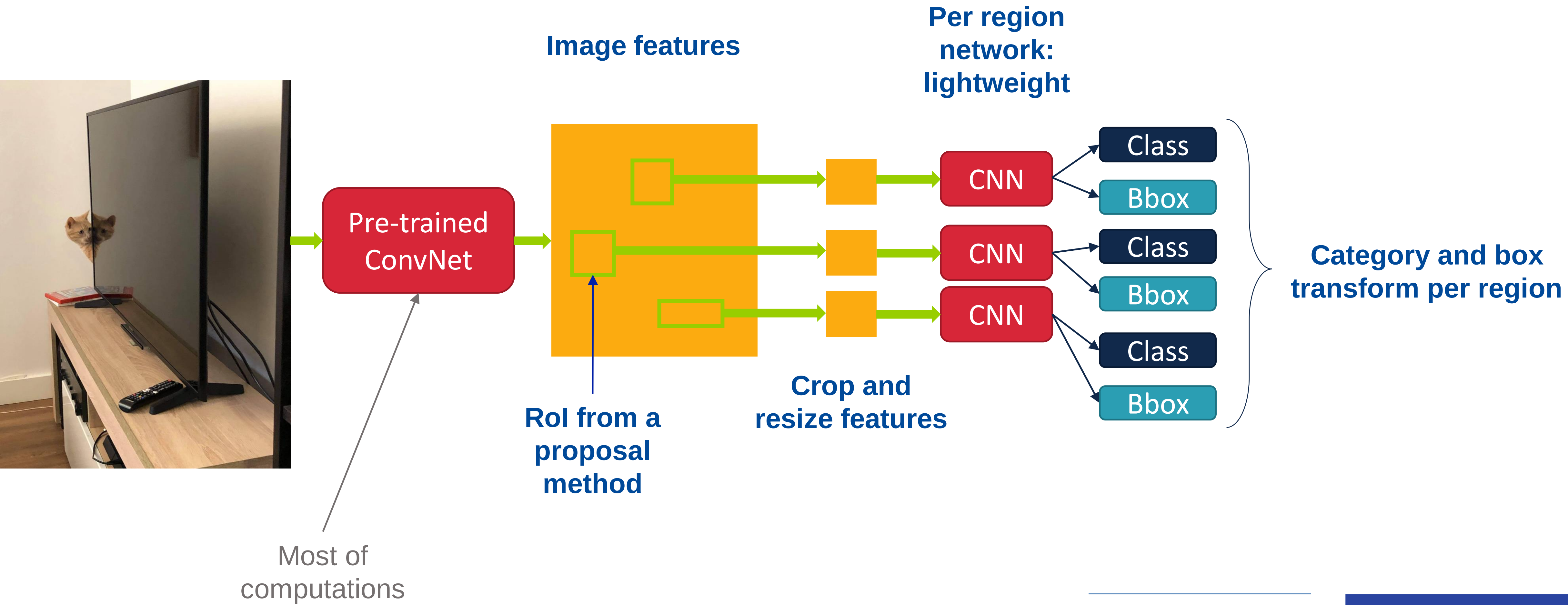


RoI from a proposal method

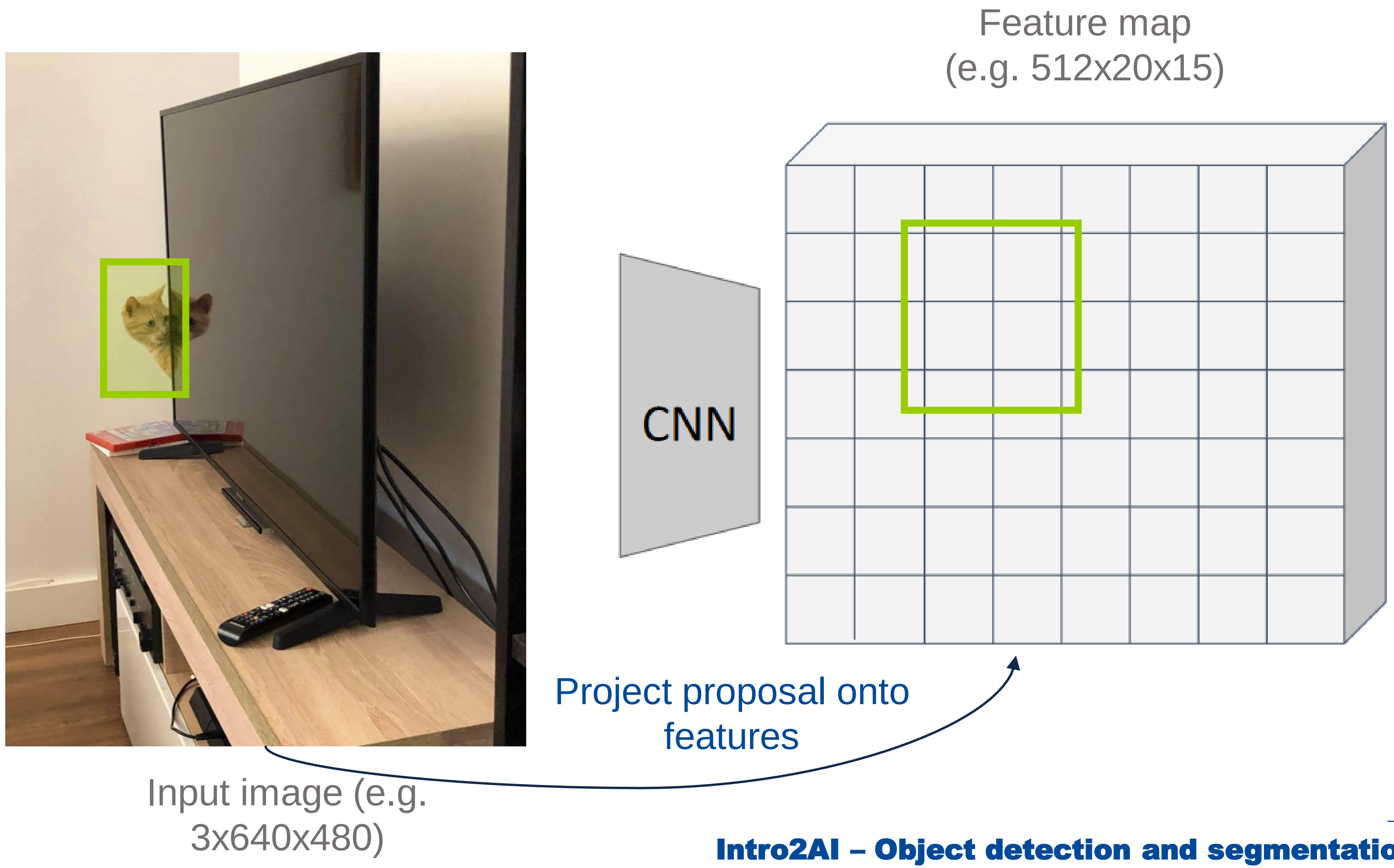
Fast R-CNN (Girshick, 2015) – A slight improvement



Fast R-CNN (Girshick, 2015) – A slight improvement

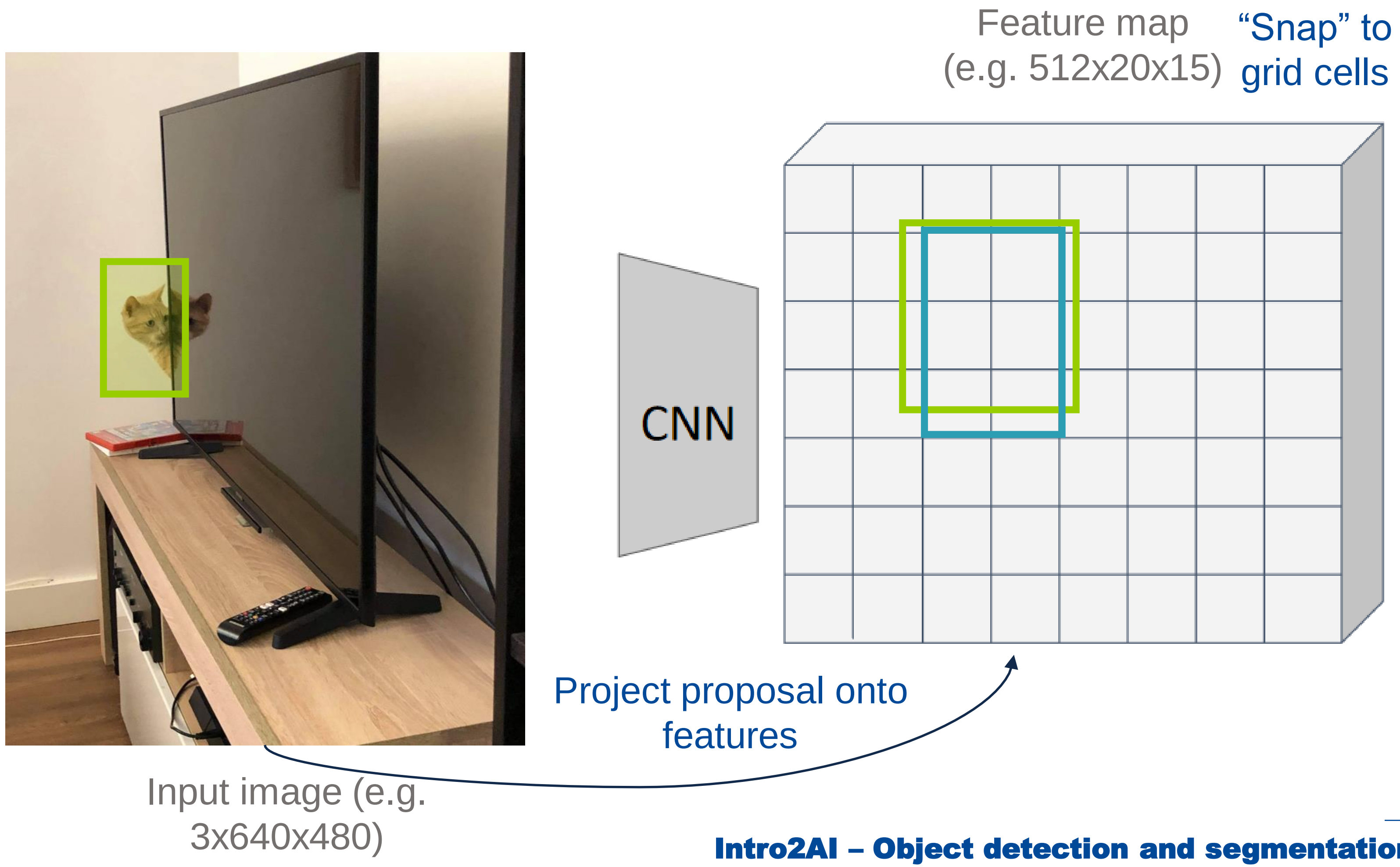


Cropping features – ROI pooling



From: https://web.eecs.umich.edu/~justincj/slides/eecs498/498_FA2019_lecture15.pdf

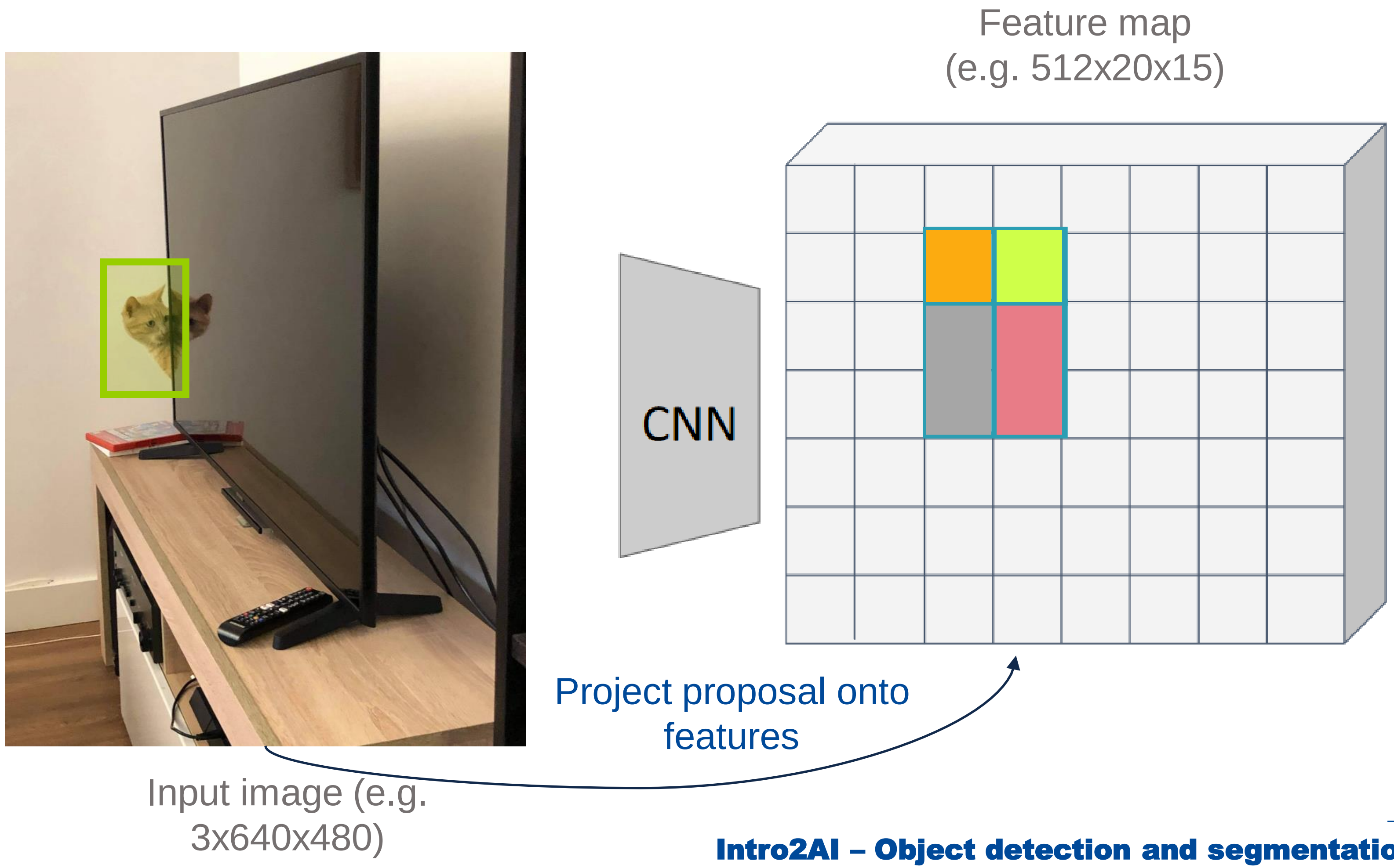
Cropping features – ROI pooling



From: https://web.eecs.umich.edu/~justincj/slides/eecs498/498_FA2019_lecture15.pdf

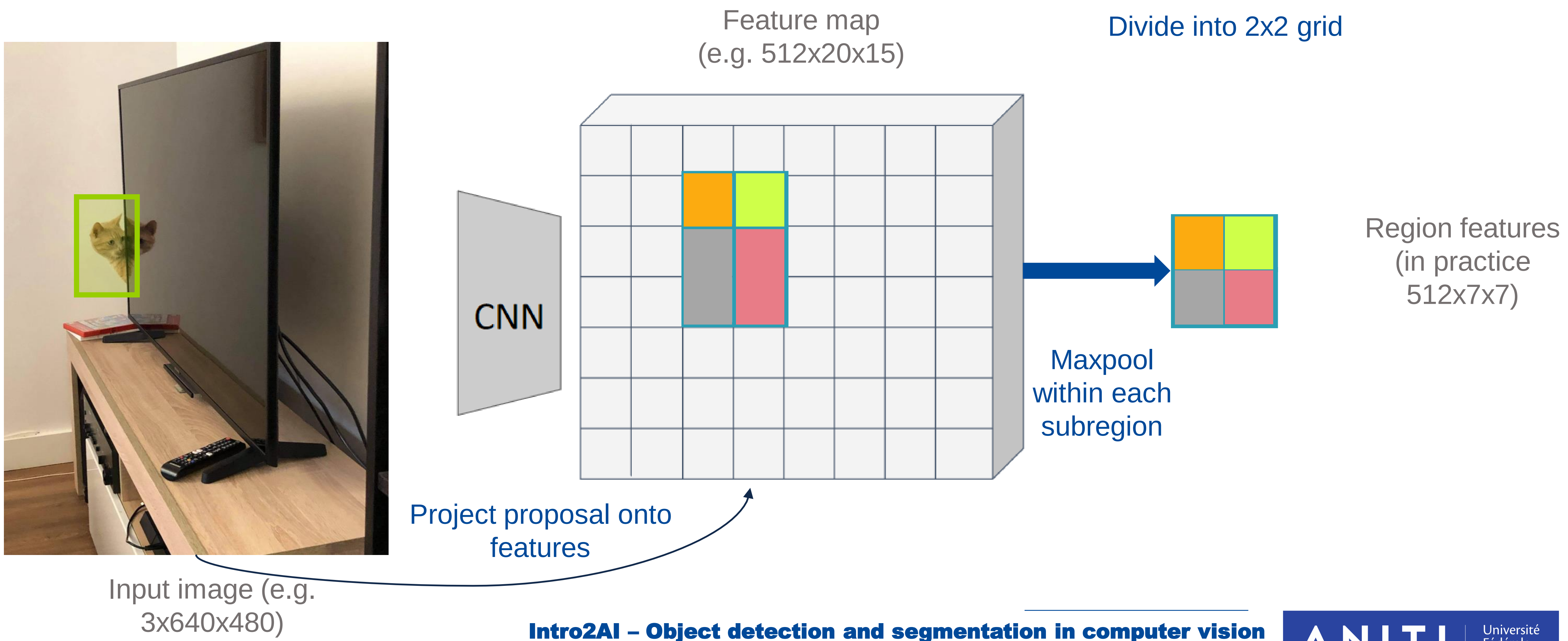
Cropping features – ROI pooling

Divide into 2x2 grid



From: https://web.eecs.umich.edu/~justincj/slides/eecs498/498_FA2019_lecture15.pdf

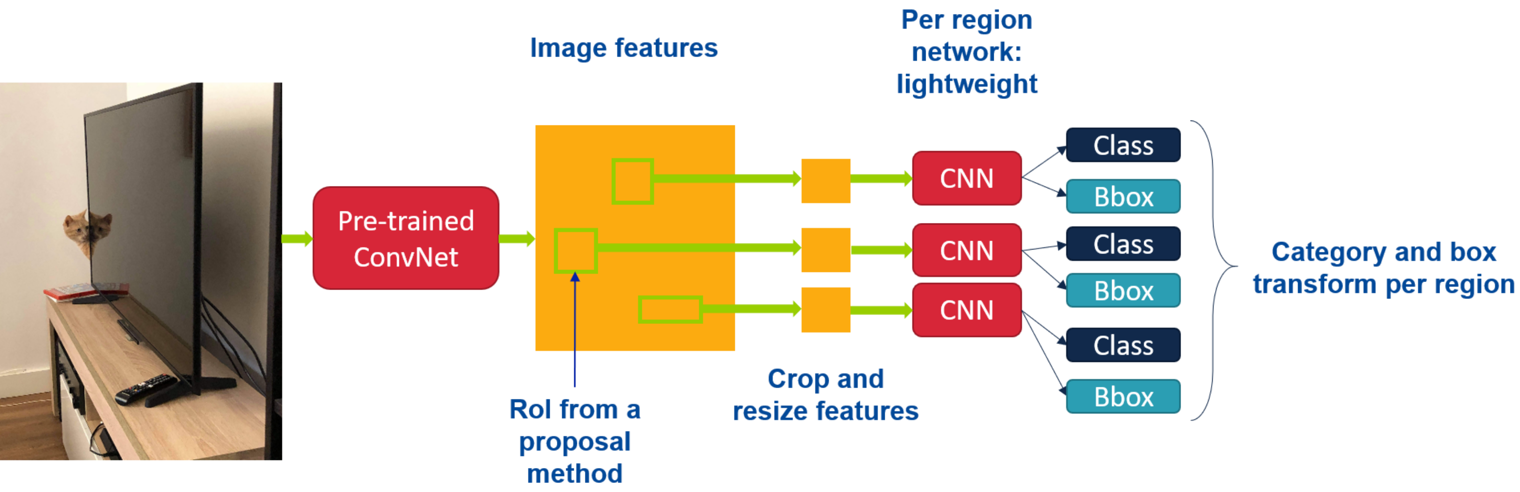
Cropping features – ROI pooling



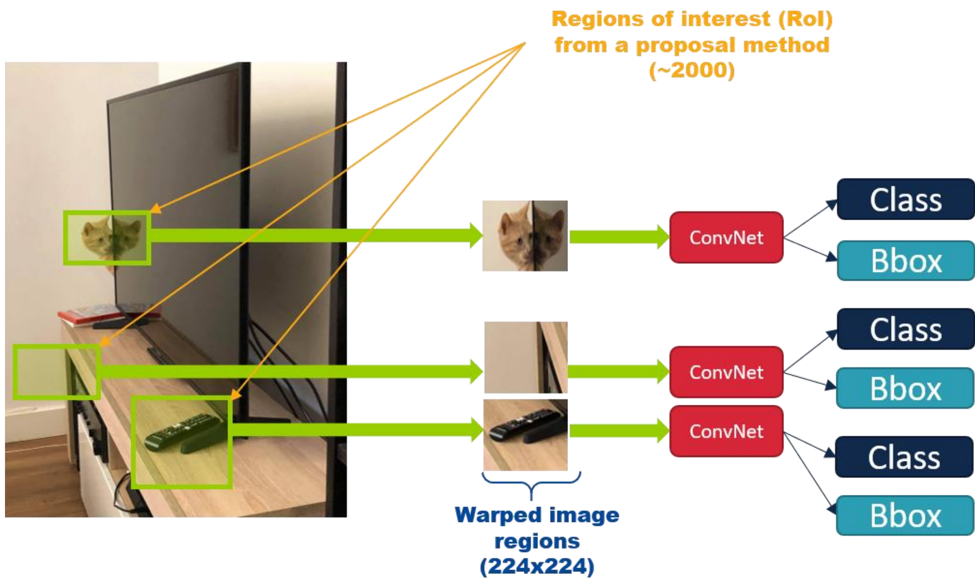
From: https://web.eecs.umich.edu/~justincj/slides/eecs498/498_FA2019_lecture15.pdf

Fast R-CNN vs. “slow” R-CNN

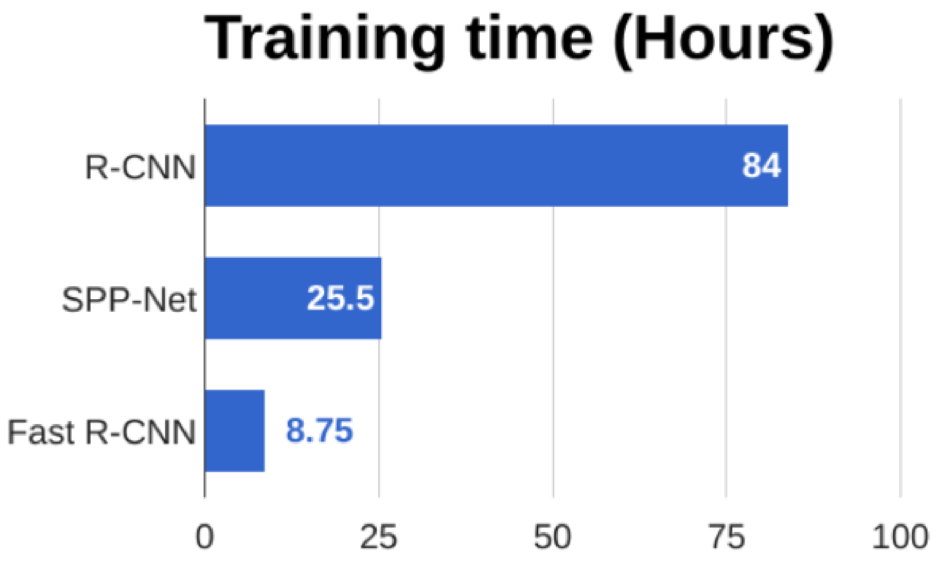
Fast RCNN

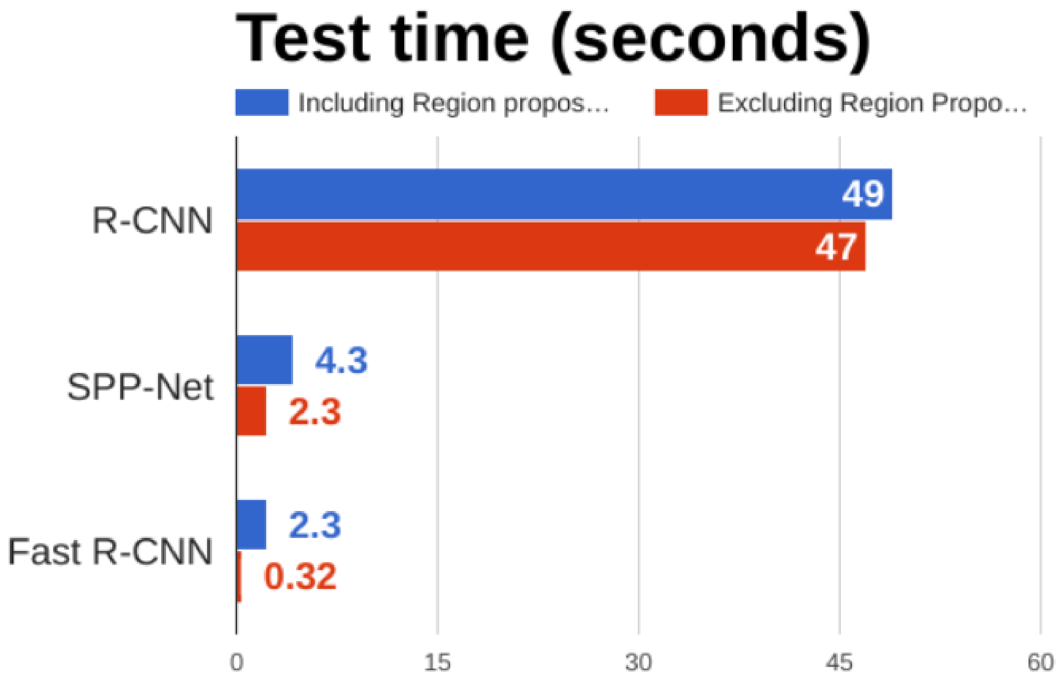


“Slow” R-CNN



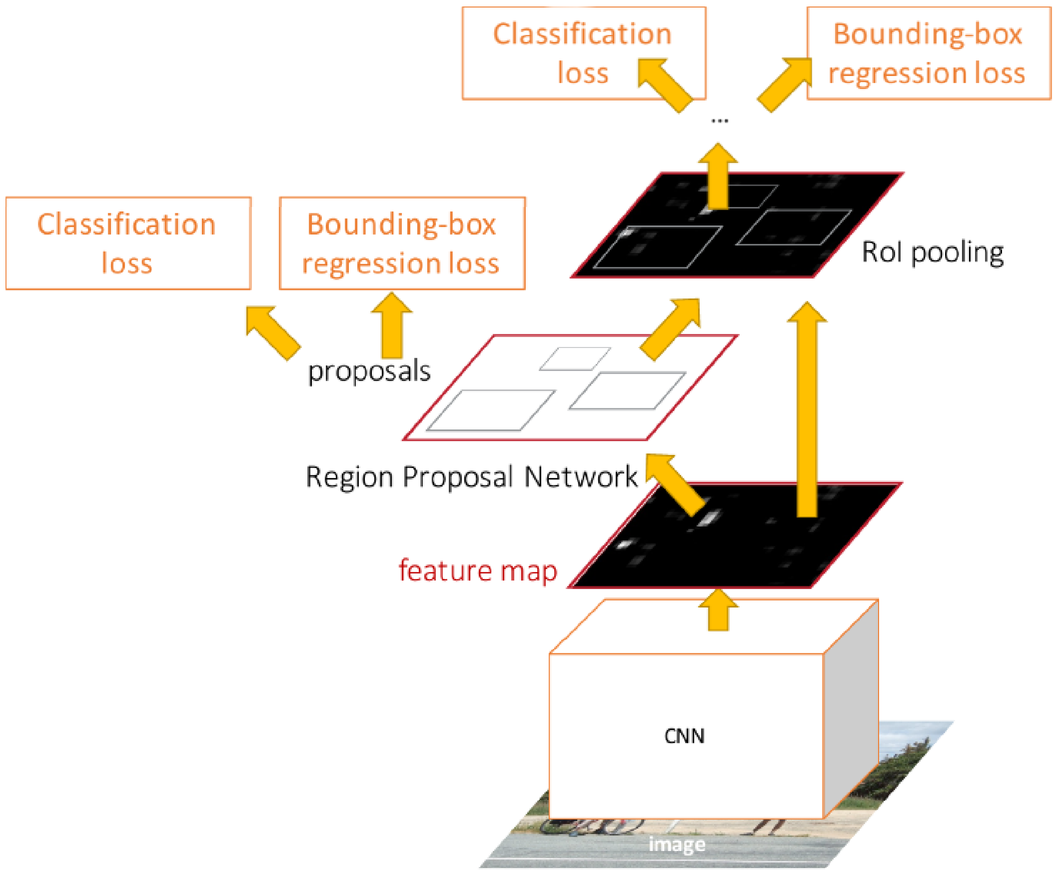
Fast R-CNN vs. “Slow” R-CNN





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

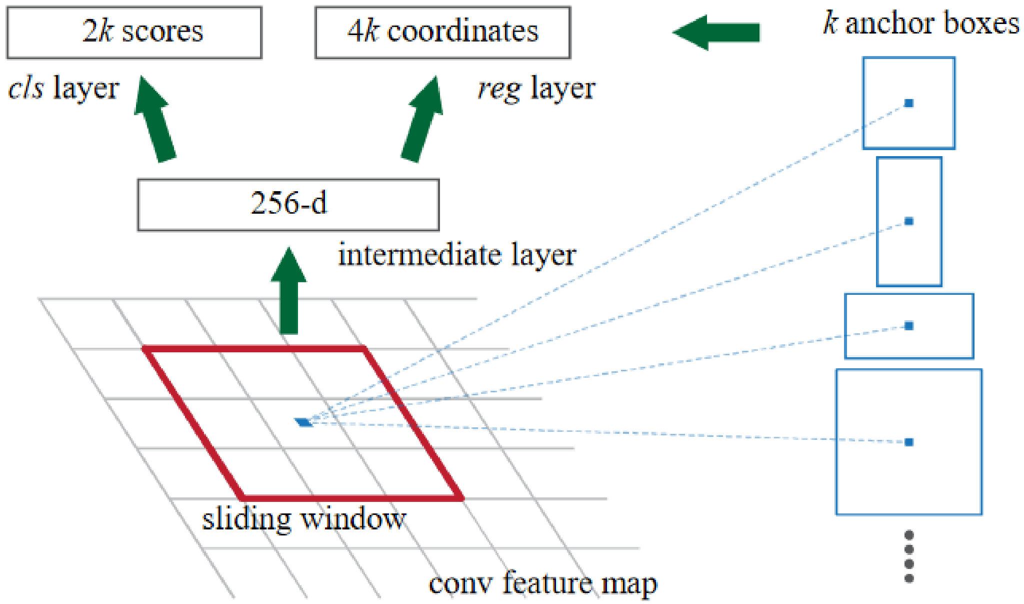
Learning to propose regions – Faster RCNN (Ren et al., 2015)



- Idea: insert a **Region Proposal Network (RPN)** to predict proposals from features
- Otherwise same as Fast R-CNN: crop features for each proposal, classify each one

Learning to propose regions – Faster RCNN (Ren et al., 2015)

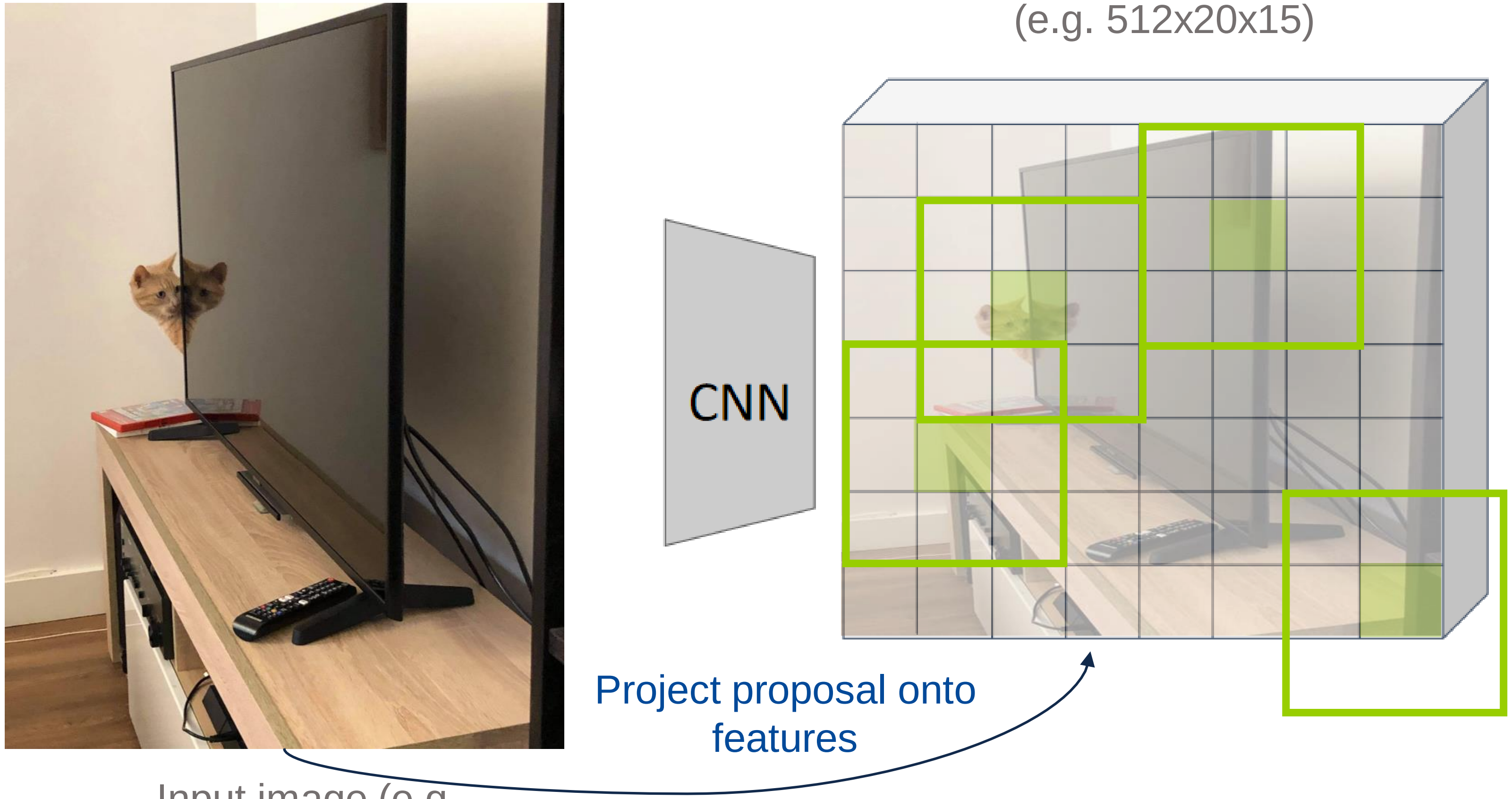
- A 3x3 sliding window is run spatially on the feature maps
- For each position in the feature map, use a predefined set of k anchor boxes
- Anchors correspond to a region in the original image
- Each sliding windows output a feature which is fed to:
  - A foreground/background classifier gives the probability that each proposed RoI shows an object
  - A box regression layer gives offsets from anchor boxes to proposed RoI



Region proposal network

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

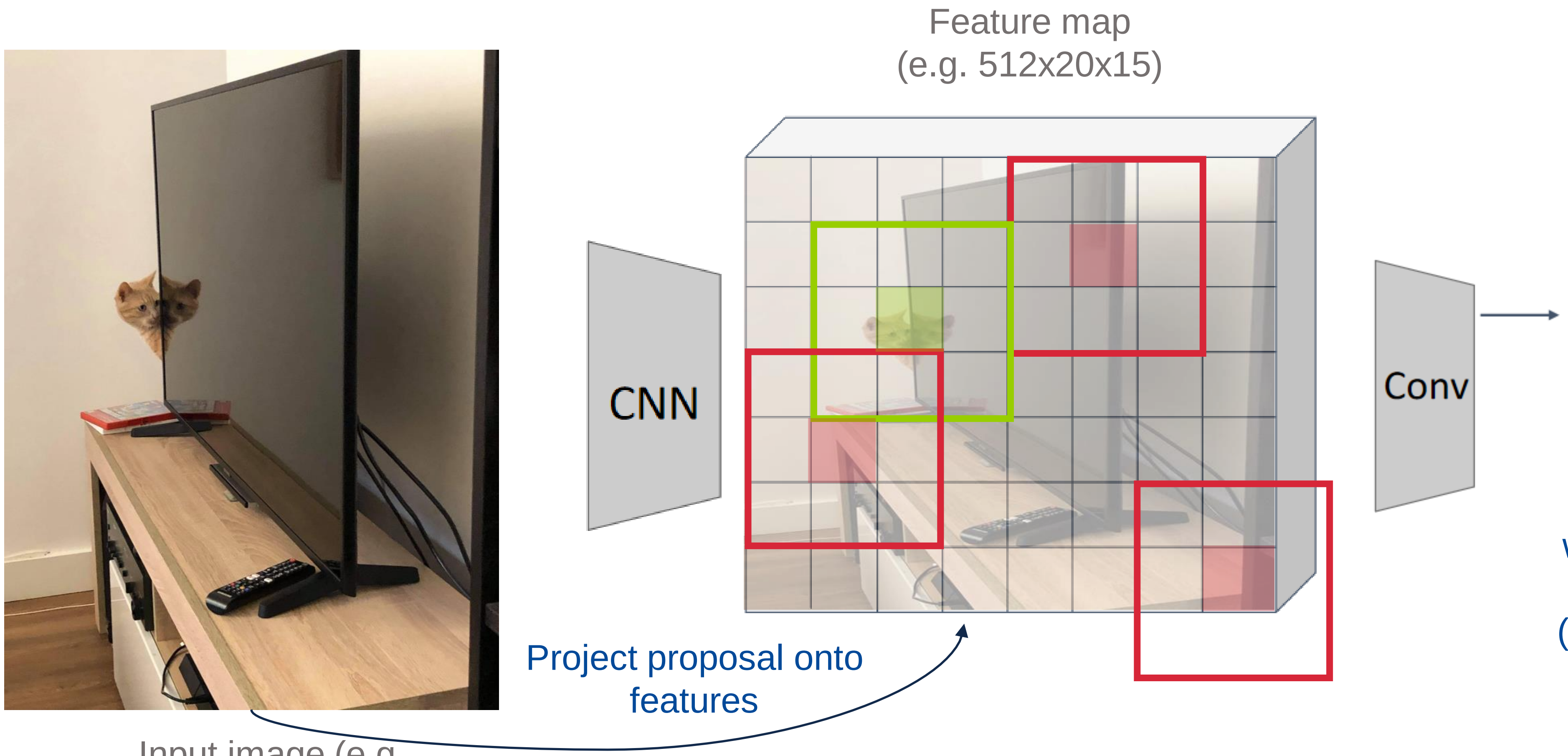
Feature map
(e.g. 512x20x15)



Input image (e.g. 3x640x480)

Region proposal network

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



Input image (e.g. 3x640x480)

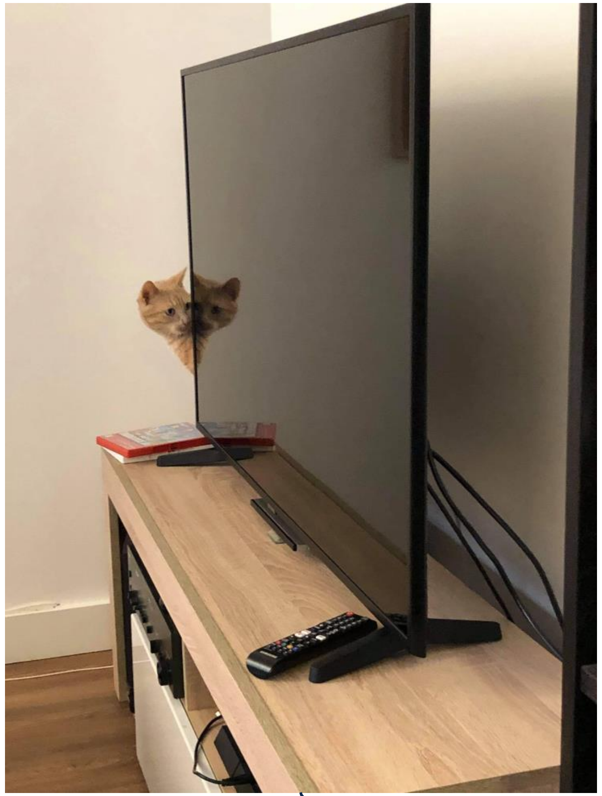
Anchor is an object?

At each point, predict whether the corresponding anchor contains an object (per-cell logistic regression, predict scores with conv layer)

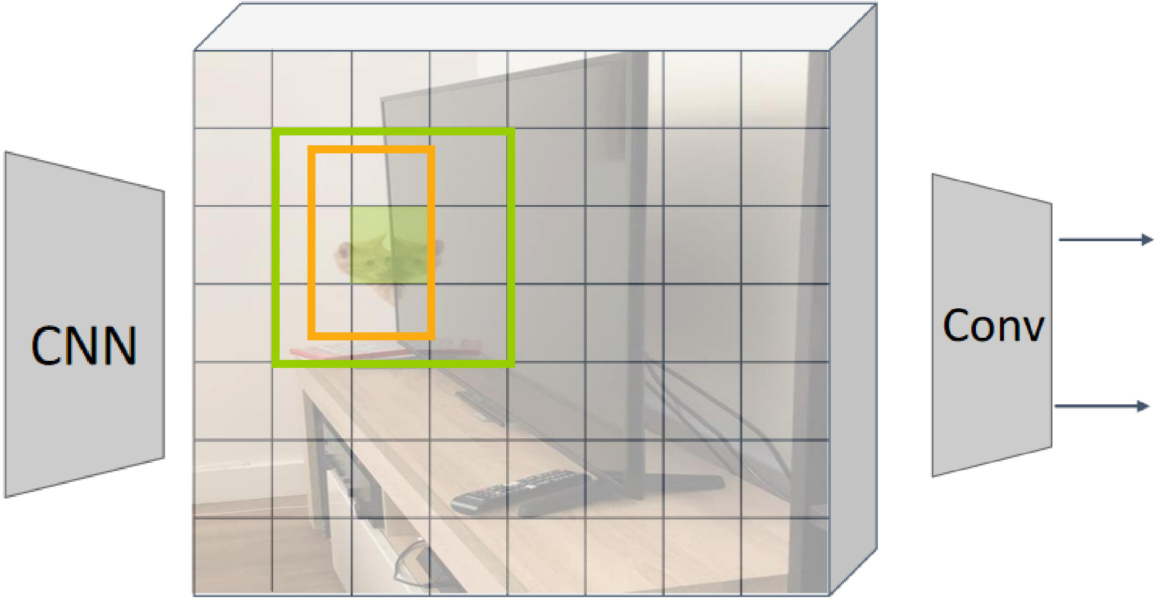
Region proposal network

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

Feature map
(e.g. 512x20x15)



Input image (e.g. 3x640x480)



Project proposal onto features

Anchor is an object?

1x20x15

Box transforms

4x20x15

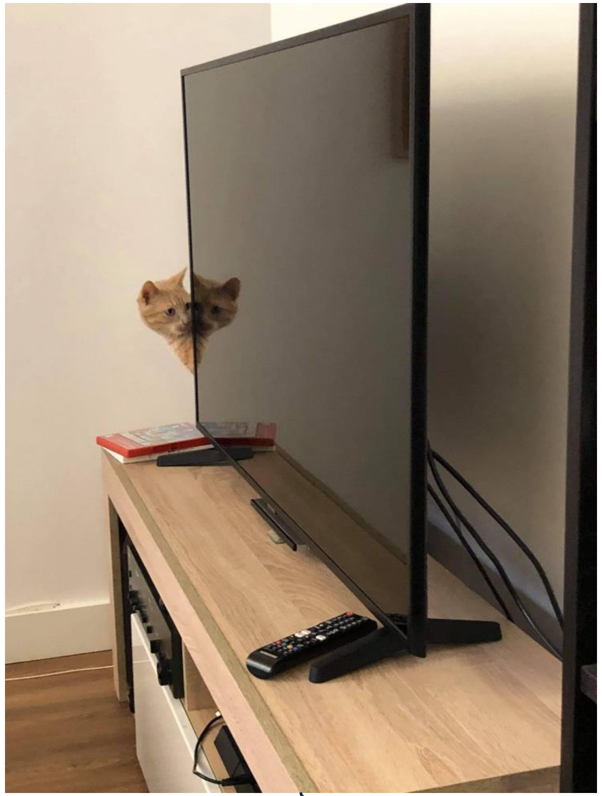
For positive boxes, also predict a box transform to regress from **anchor box** to **object box**

Region proposal network

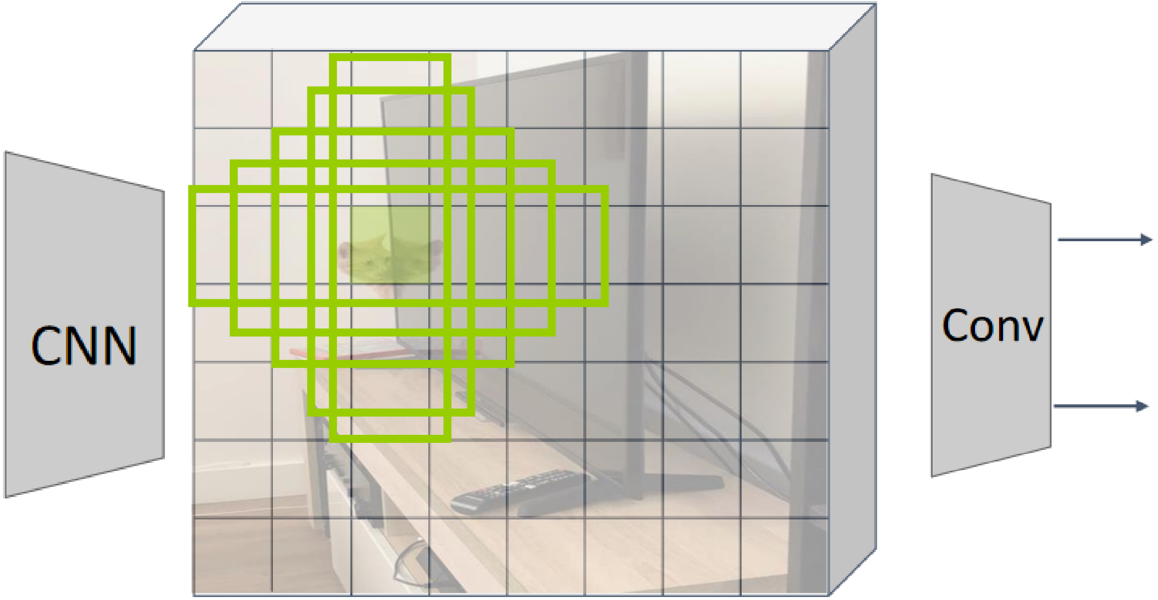
Problem: Anchor box may have the wrong size/shape

Solution: Use K different anchor boxes at each point!

Feature map
(e.g. 512x20x15)



Input image (e.g. 3x640x480)



Project proposal onto features

Anchor is an object?

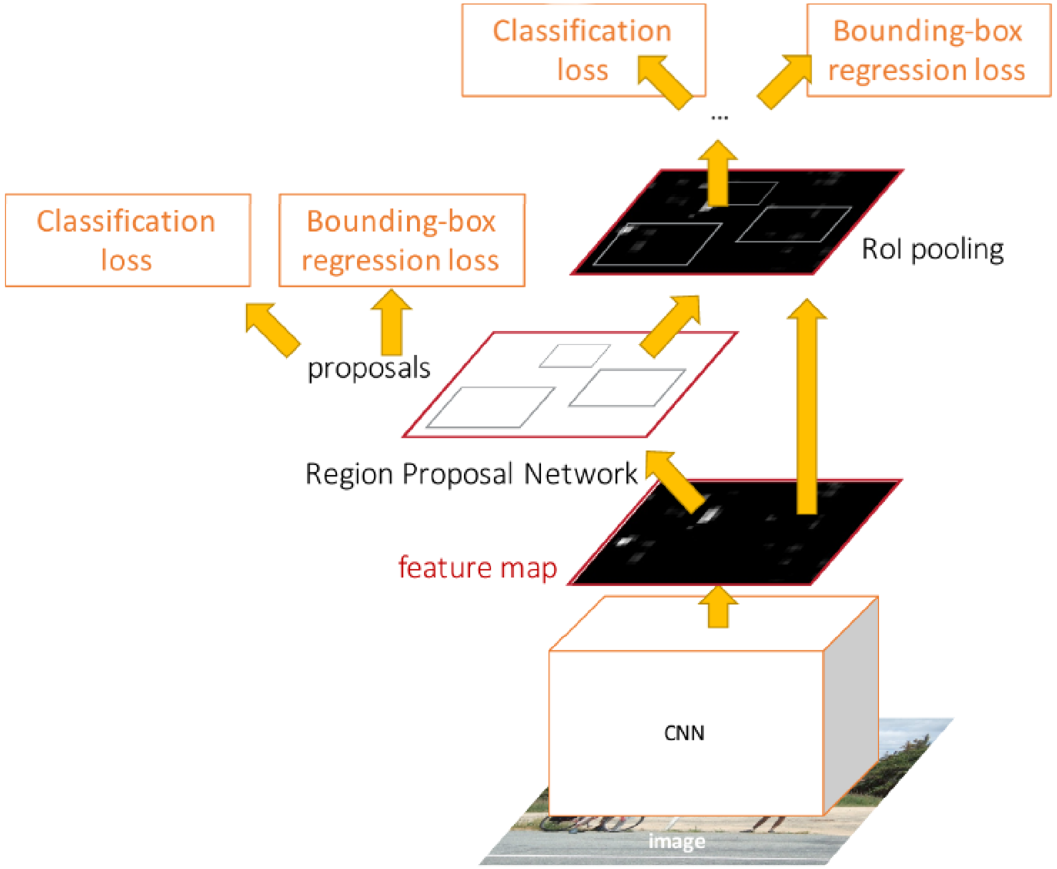
Kx20x15

Box transforms

4**K**x20x15

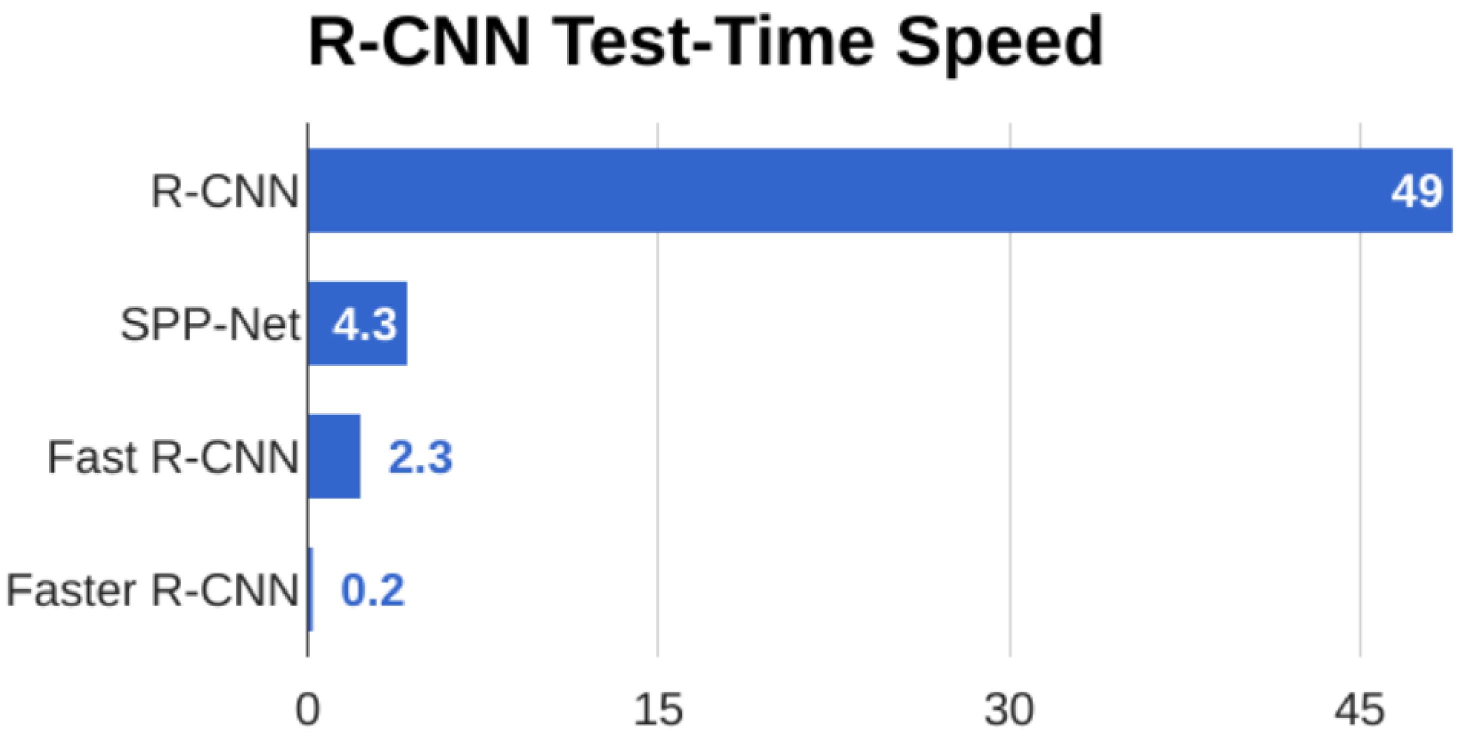
At test time: sort all Kx20x15 boxes by their score, and take the top ~300 as our region proposals

Learning to propose regions – Faster RCNN (Ren et al., 2015)



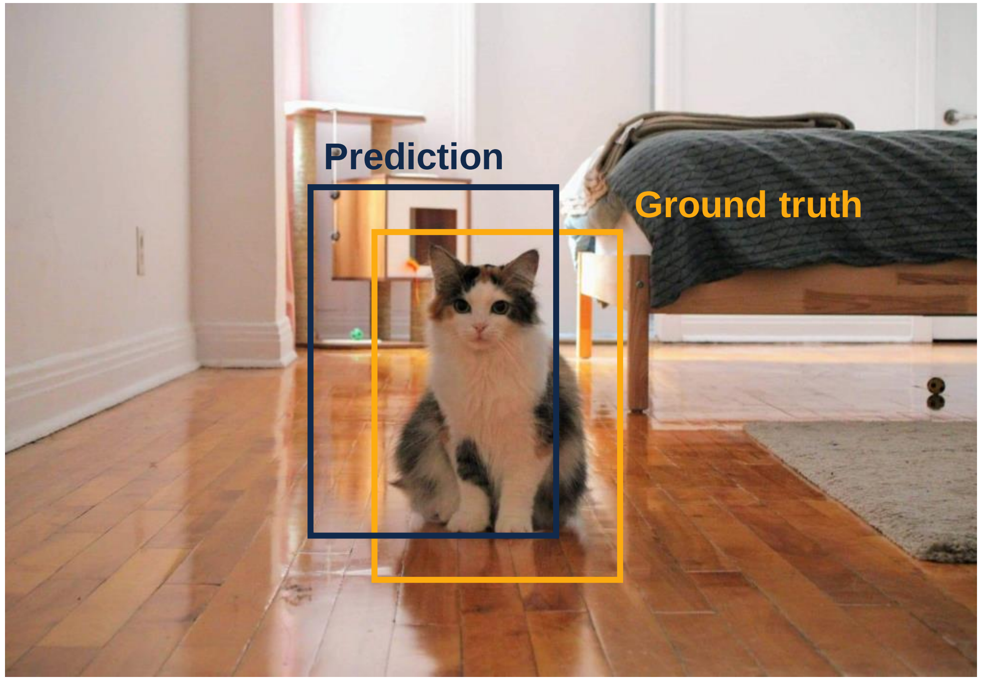
- Jointly train with 4 losses:
  1. **RPN classification**: anchor box is object / not an object
  2. **RPN regression**: predict transform from anchor box to proposal box
  3. **Object classification**: classify proposals as background / object class
  4. **Object regression**: predict transform from proposal to object box

Performances improvement



Object detectors evaluation – Comparing boxes

- How can we compare the prediction and the bounding boxes?

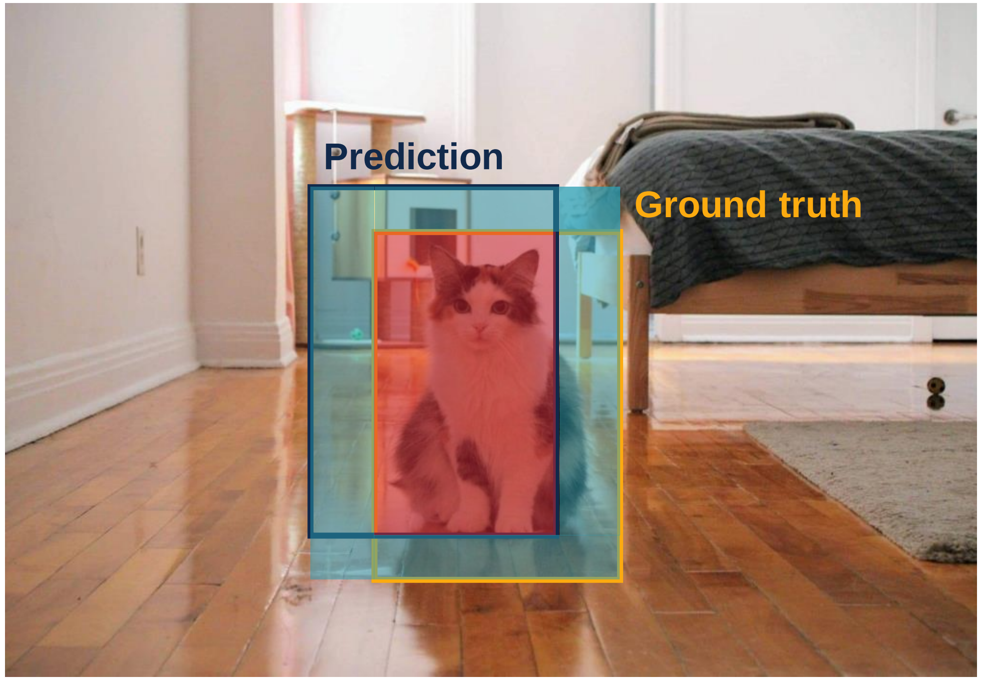


From: https://web.eecs.umich.edu/~justincj/slides/eecs498/498_FA2019_lecture15.pdf

Object detectors evaluation – Comparing boxes

- How can we compare the prediction and the bounding boxes?
- Use the **Intersection over Union (IoU)**:

$$IoU = \frac{Area\ of\ Insersection}{Area\ of\ Union}$$



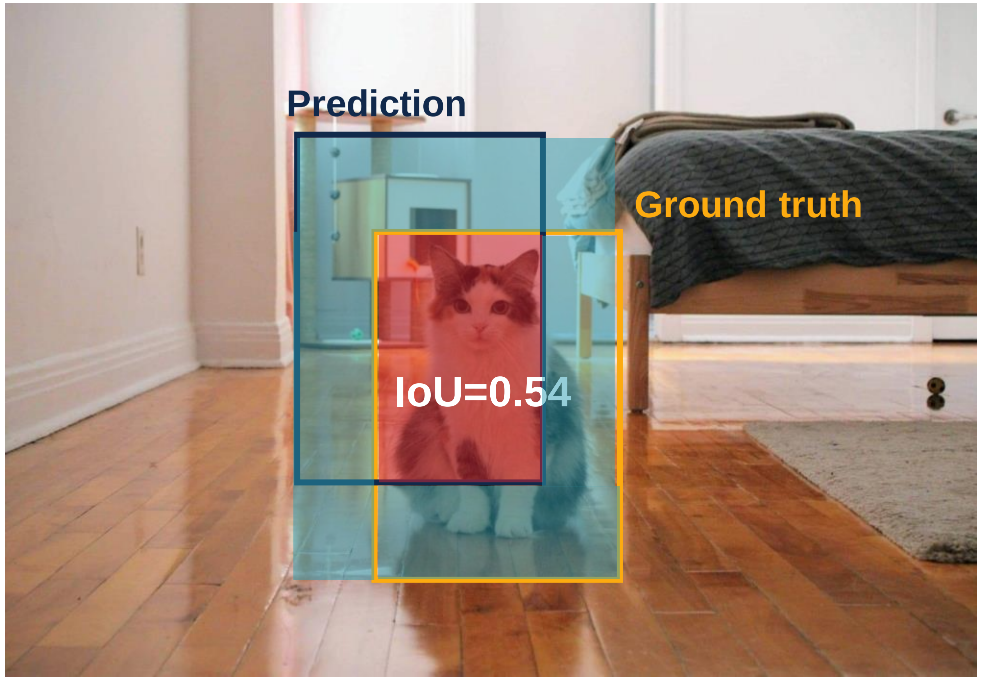
From: https://web.eecs.umich.edu/~justincj/slides/eecs498/498_FA2019_lecture15.pdf

Object detectors evaluation – Comparing boxes

- How can we compare the prediction and the bounding boxes?
- Use the **Intersection over Union (IoU)**:

$$IoU = \frac{Area\ of\ Insersection}{Area\ of\ Union}$$

- IoU > 0.5 is "decent"



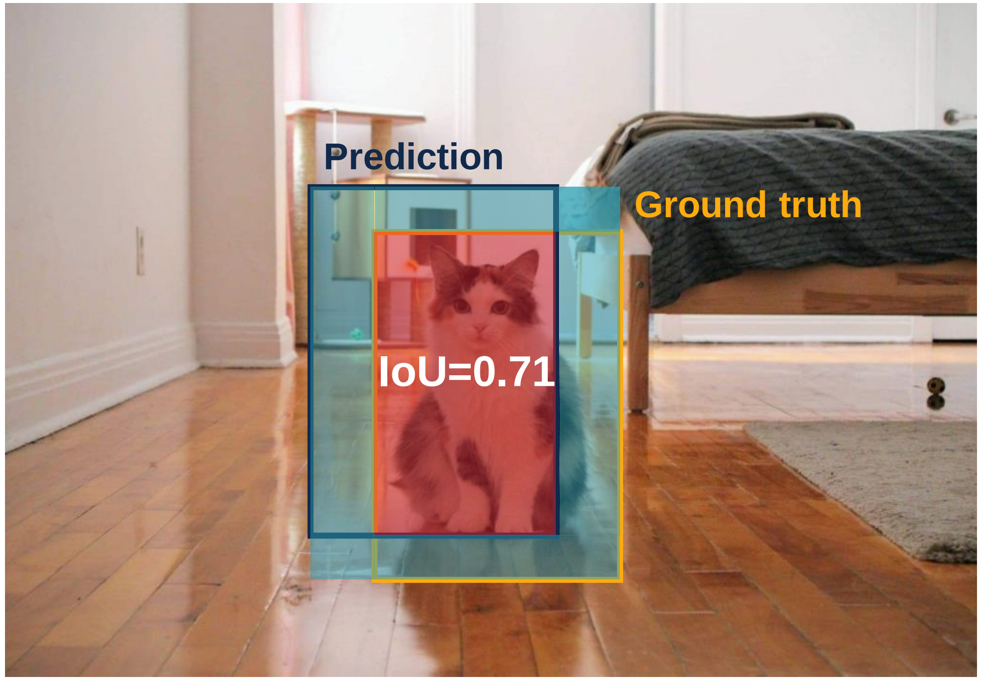
From: https://web.eecs.umich.edu/~justincj/slides/eecs498/498_FA2019_lecture15.pdf

Object detectors evaluation – Comparing boxes

- How can we compare the prediction and the bounding boxes?
- Use the **Intersection over Union (IoU)**:

$$IoU = \frac{Area\ of\ Insersection}{Area\ of\ Union}$$

- IoU > 0.5 is “decent”
- IoU > 0.7 is “pretty good”



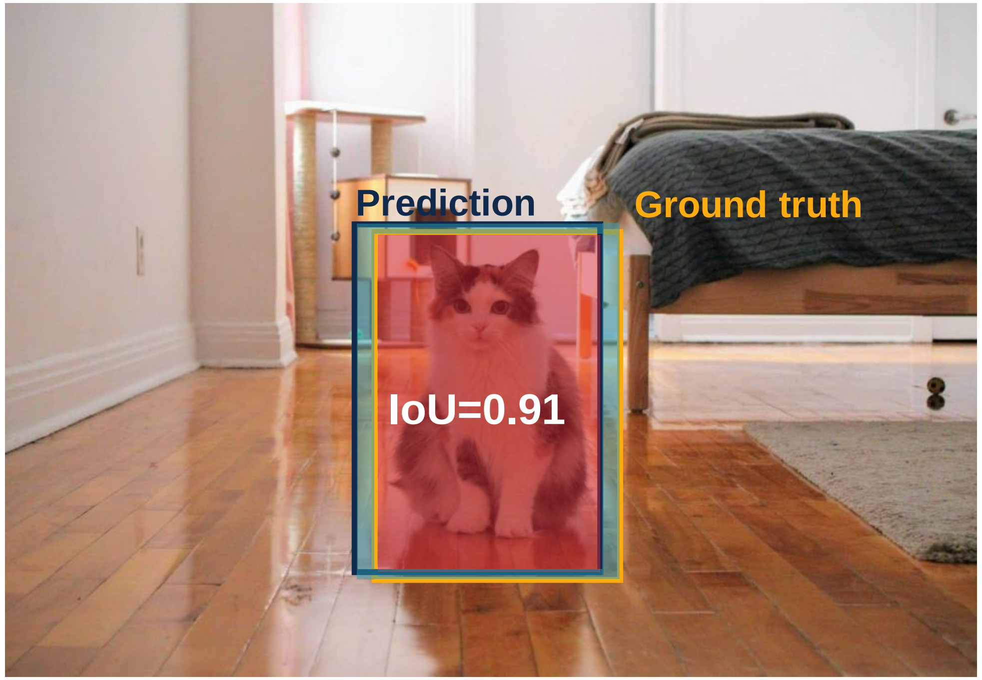
From: https://web.eecs.umich.edu/~justincj/slides/eecs498/498_FA2019_lecture15.pdf

Object detectors evaluation – Comparing boxes

- How can we compare the prediction and the bounding boxes?
- Use the **Intersection over Union (IoU)**:

$$IoU = \frac{Area\ of\ Insersection}{Area\ of\ Union}$$

- IoU > 0.5 is “decent”
- IoU > 0.7 is “pretty good”
- IoU > 0.9 is “almost perfect”



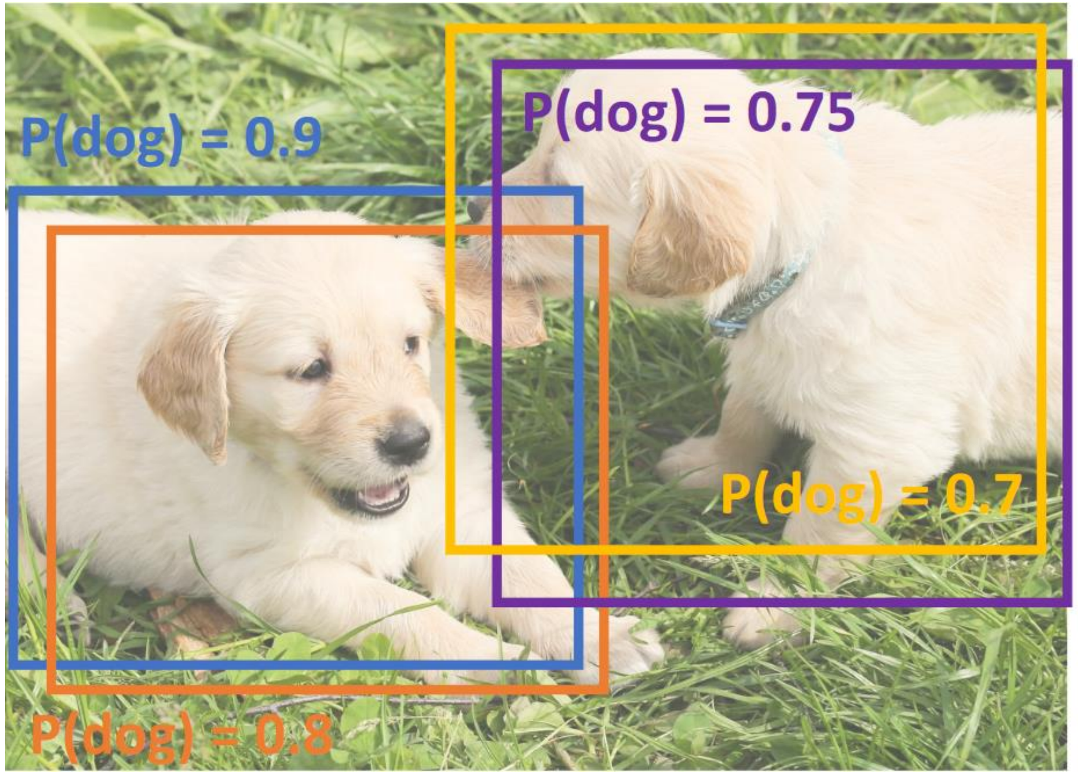
From: https://web.eecs.umich.edu/~justincj/slides/eecs498/498_FA2019_lecture15.pdf

Non-Max Suppression (NMS) – Deal with overlapping boxes

- **Problem**: object detectors often output many overlapping detection (due to multiple anchors per pixel)

- **Solution**: post-process raw detections using **Non-Max Suppression (NMS)**

- Algorithm:
  1. Select highest-scoring box
  2. Eliminate lower-scoring boxes with IoU > threshold (e.g. 0.7)
  3. If any boxes remain, GOTO 1



Puppy image is CC0 Public Domain

From: https://web.eecs.umich.edu/~justincj/slides/eecs498/498_FA2019_lecture15.pdf

Non-Max Suppression (NMS) – Deal with overlapping boxes

- **Problem**: object detectors often output many overlapping detection (due to multiple anchors per pixel)

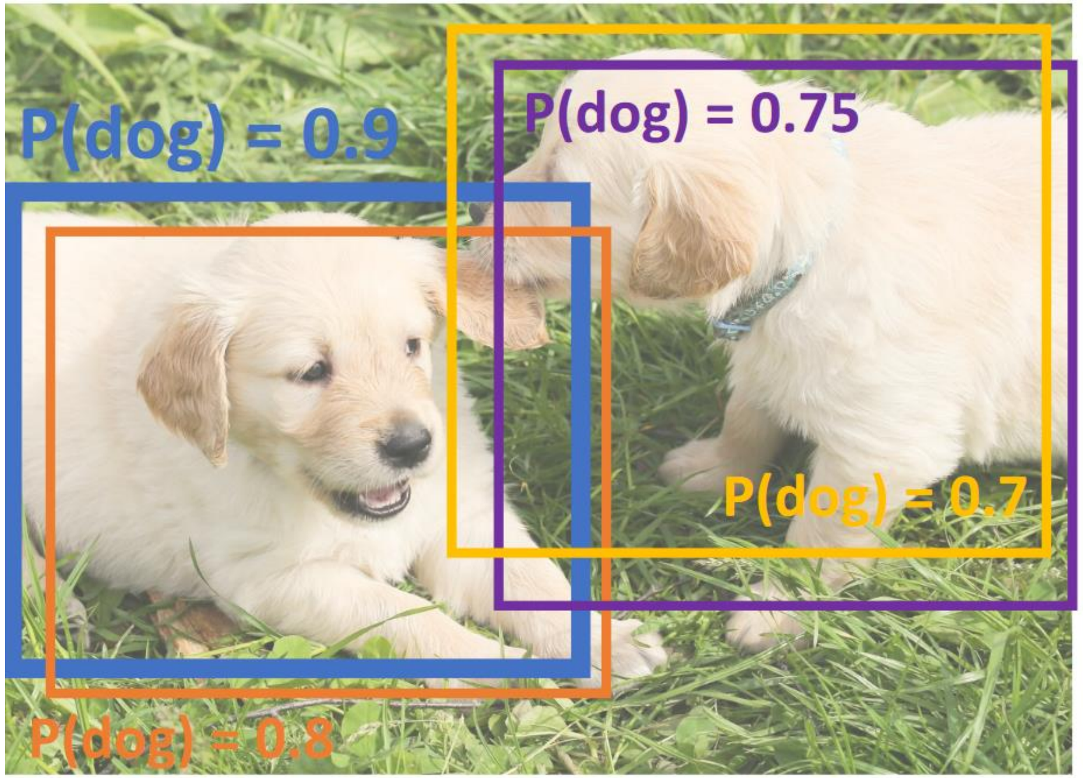
- **Solution**: post-process raw detections using **Non-Max Suppression (NMS)**

- <u>Algorithm:</u>
  1. Select highest-scoring box
  2. Eliminate lower-scoring boxes with IoU > threshold (e.g. 0.7)
  3. If any boxes remain, GOTO 1

IoU(■ , ■) = 0.78

IoU(■ , ■) = 0.05

IoU(■ , ■) = 0.07



Puppy image is CC0 Public Domain

From: https://web.eecs.umich.edu/~justincj/slides/eecs498/498_FA2019_lecture15.pdf

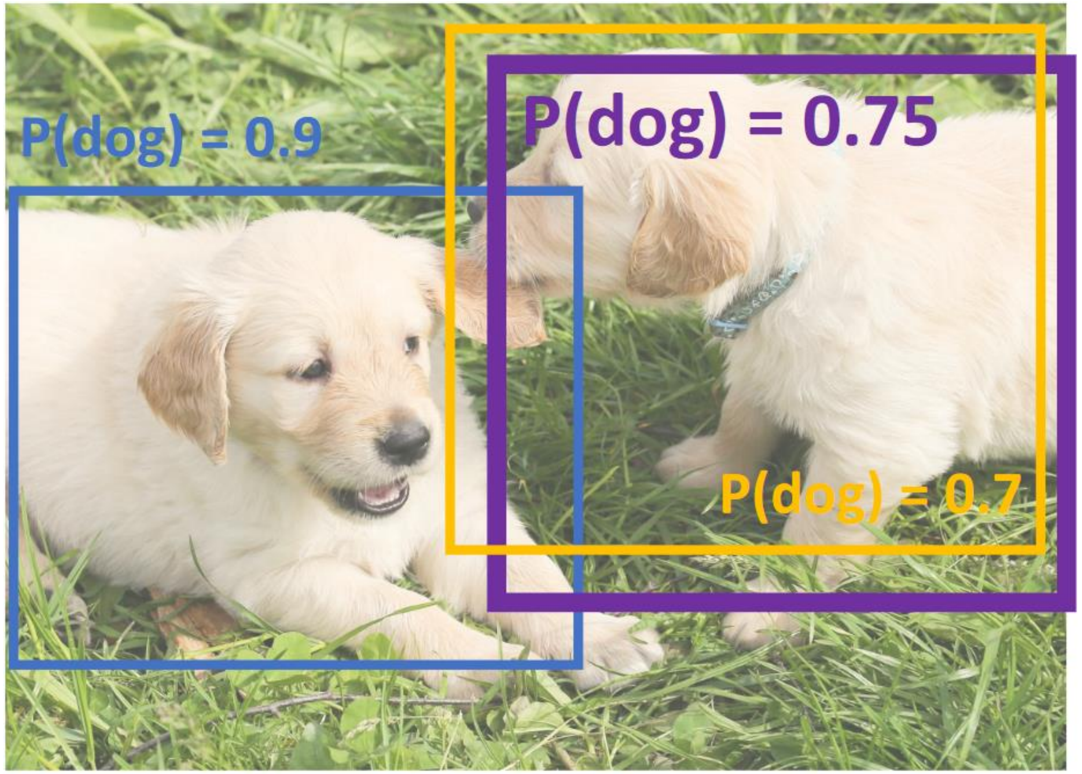
Non-Max Suppression (NMS) – Deal with overlapping boxes

- **Problem**: object detectors often output many overlapping detection (due to multiple anchors per pixel)

- **Solution**: post-process raw detections using **Non-Max Suppression (NMS)**

- Algorithm:
  1. Select highest-scoring box
  2. Eliminate lower-scoring boxes with IoU > threshold (e.g. 0.7)
  3. If any boxes remain, GOTO 1

IoU(■ , ■) = 0.74



Puppy image is CC0 Public Domain

From: https://web.eecs.umich.edu/~justincj/slides/eecs498/498_FA2019_lecture15.pdf

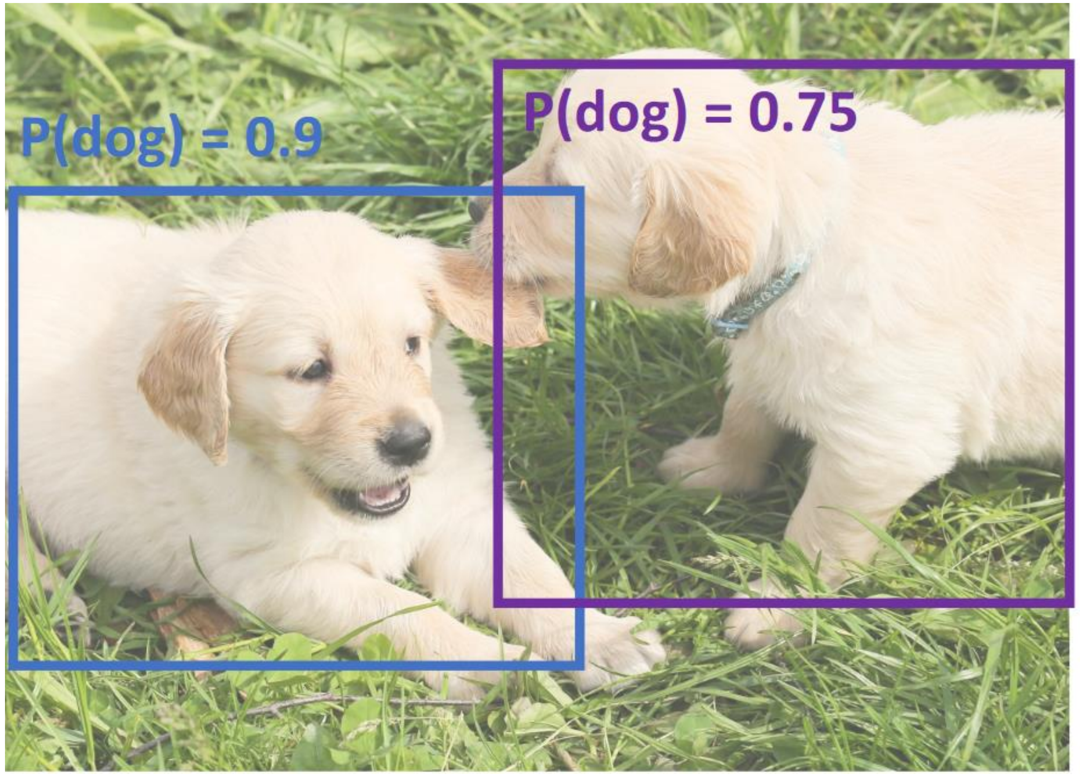
Non-Max Suppression (NMS) – Deal with overlapping boxes

- **Problem**: object detectors often output many overlapping detection (due to multiple anchors per pixel)

- **Solution**: post-process raw detections using **Non-Max Suppression (NMS)**

- <u>Algorithm:</u>
  1. Select highest-scoring box
  2. Eliminate lower-scoring boxes with IoU > threshold (e.g. 0.7)
  3. If any boxes remain, GOTO 1

- **Problem**: NMS may eliminate "good" boxes when objects are highly overlapping… NO GOOD SOLUTION

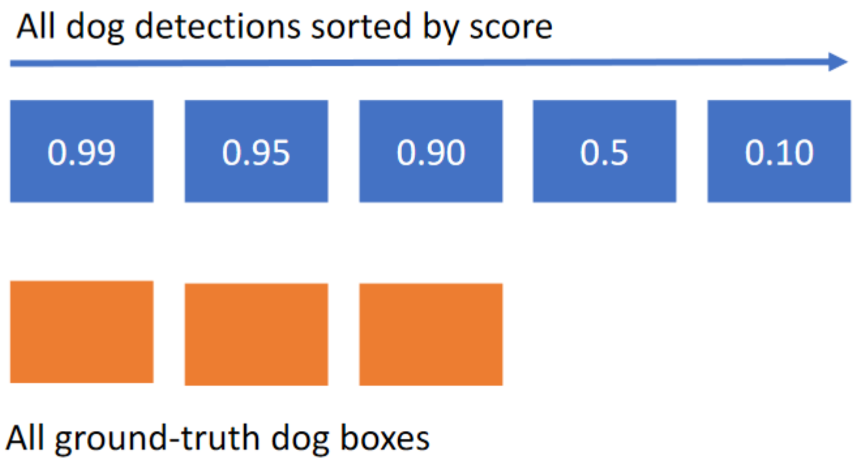


Puppy image is CC0 Public Domain

From: https://web.eecs.umich.edu/~justincj/slides/eecs498/498_FA2019_lecture15.pdf

Evaluating object detectors – Mean Average Precision (mAP)

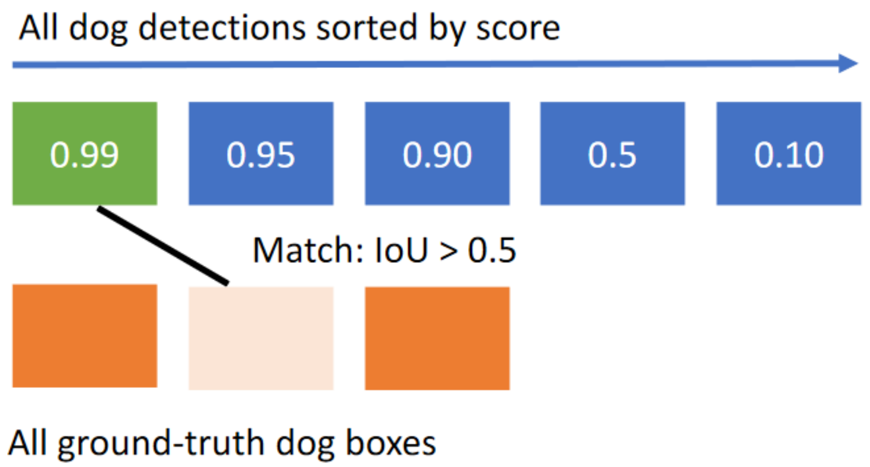
1. Run object detector on all test images (with NMS)
2. For each category, compute Average Precision (AP) = Area under Precision vs Recall Curve
    1. For each detection (highest score to lowest score)



From: https://web.eecs.umich.edu/~justincj/slides/eecs498/498_FA2019_lecture15.pdf

Evaluating object detectors – Mean Average Precision (mAP)

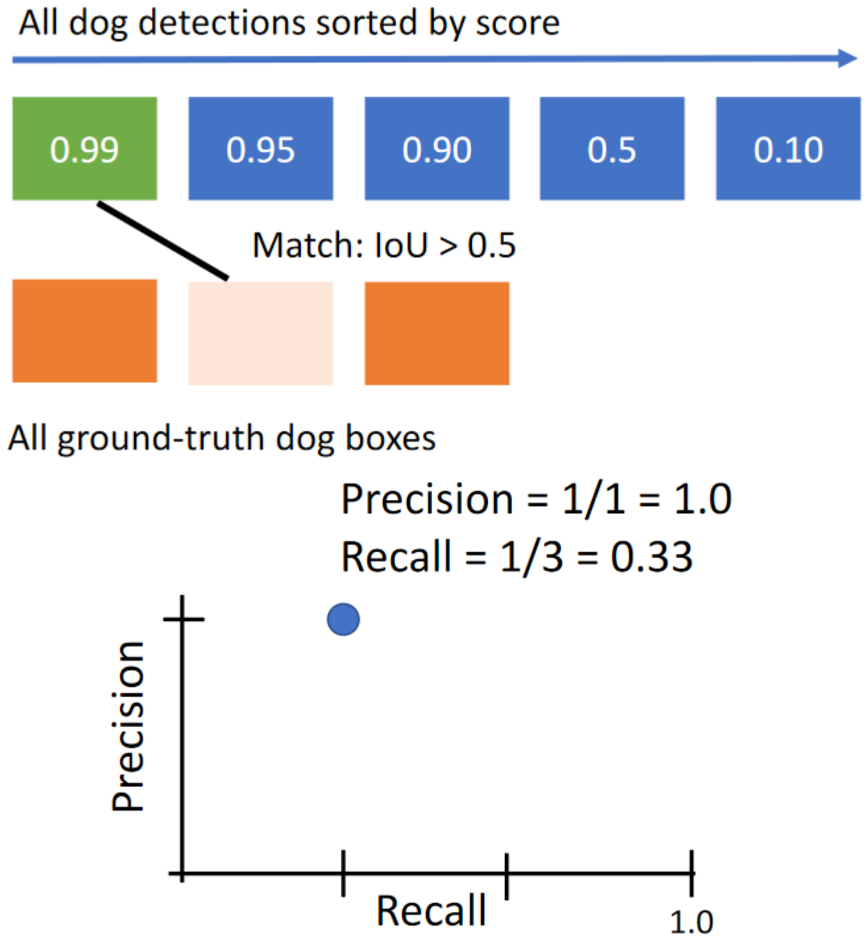
1. Run object detector on all test images (with NMS)
2. For each category, compute Average Precision (AP) = Area under Precision vs Recall Curve
    1. For each detection (highest score to lowest score)
        1. If it matches some GT box with IoU > 0.5, mark it as positive and eliminate the GT
        2. Otherwise mark it as negative



From: https://web.eecs.umich.edu/~justincj/slides/eecs498/498_FA2019_lecture15.pdf

Evaluating object detectors – Mean Average Precision (mAP)

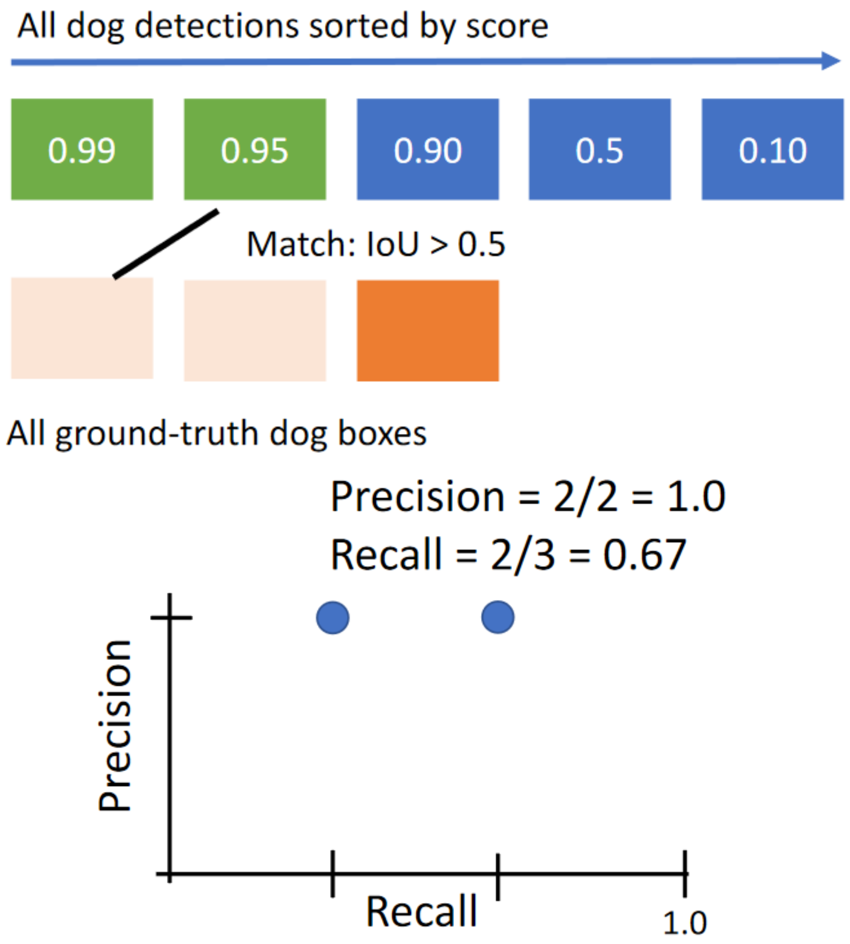
1. Run object detector on all test images (with NMS)
2. For each category, compute Average Precision (AP) = Area under Precision vs Recall Curve
    1. For each detection (highest score to lowest score)
        1. If it matches some GT box with IoU > 0.5, mark it as positive and eliminate the GT
   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



From: https://web.eecs.umich.edu/~justincj/slides/eecs498/498_FA2019_lecture15.pdf

Evaluating object detectors – Mean Average Precision (mAP)

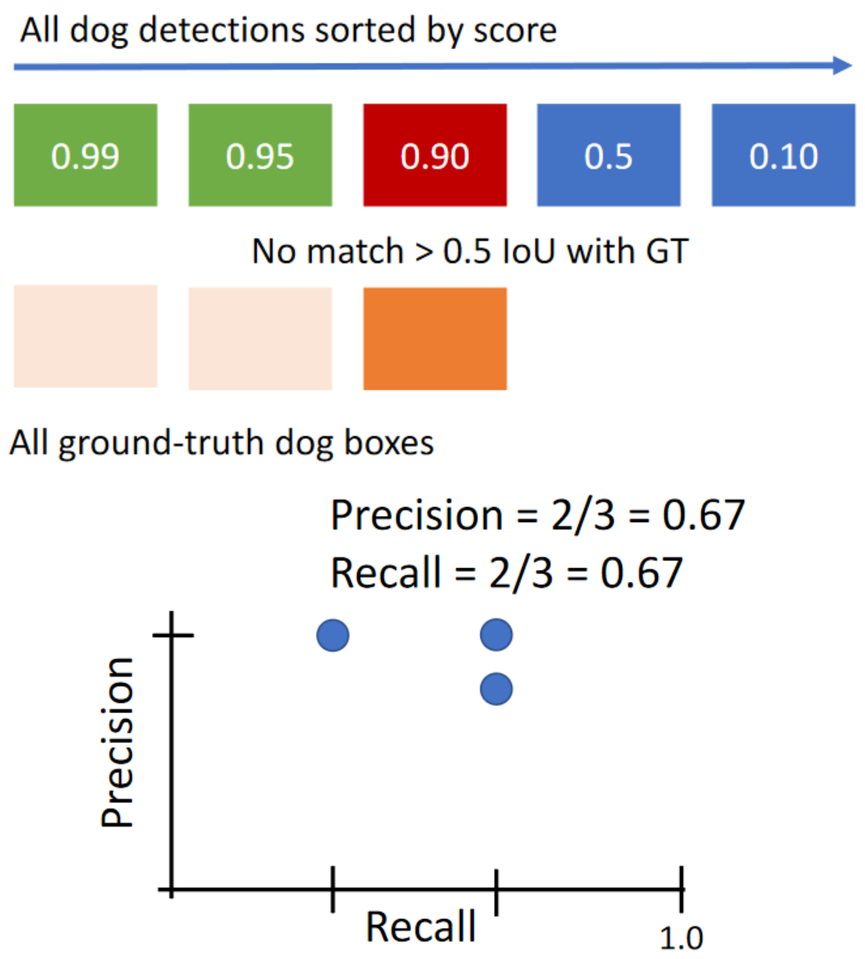
1. Run object detector on all test images (with NMS)
2. For each category, compute Average Precision (AP) = Area under Precision vs Recall Curve
    1. For each detection (highest score to lowest score)
        1. If it matches some GT box with IoU > 0.5, mark it as positive and eliminate the GT
   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



From: https://web.eecs.umich.edu/~justincj/slides/eecs498/498_FA2019_lecture15.pdf

Evaluating object detectors – Mean Average Precision (mAP)

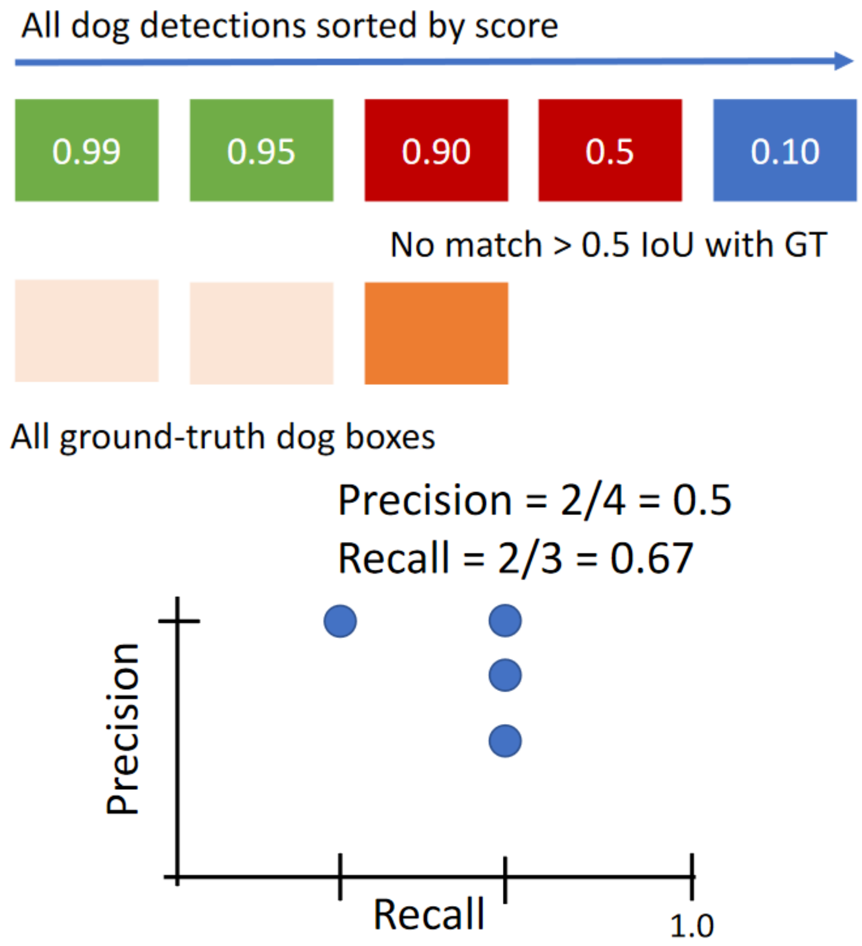
1. Run object detector on all test images (with NMS)
2. For each category, compute Average Precision (AP) = Area under Precision vs Recall Curve
    1. For each detection (highest score to lowest score)
        1. If it matches some GT box with IoU > 0.5, mark it as positive and eliminate the GT
   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



From: https://web.eecs.umich.edu/~justincj/slides/eecs498/498_FA2019_lecture15.pdf

Evaluating object detectors – Mean Average Precision (mAP)

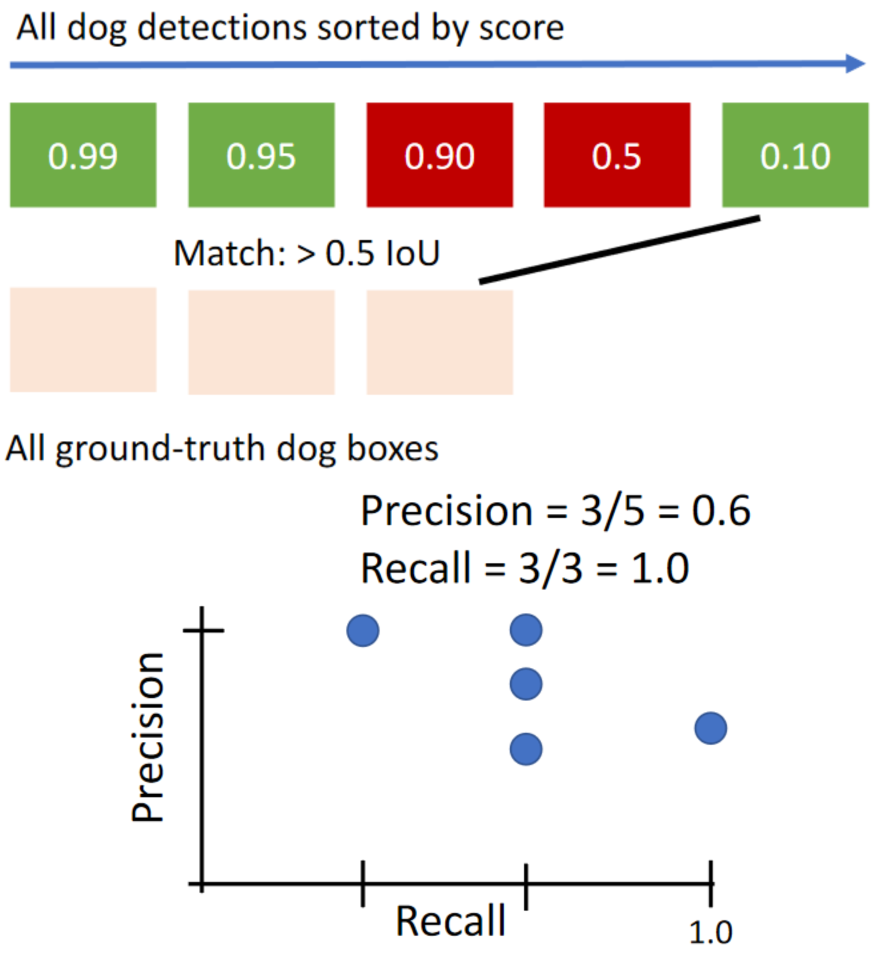
1. Run object detector on all test images (with NMS)
2. For each category, compute Average Precision (AP) = Area under Precision vs Recall Curve
   1. For each detection (highest score to lowest score)
      1. If it matches some GT box with IoU > 0.5, mark it as positive and eliminate the GT
      2.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



From: https://web.eecs.umich.edu/~justincj/slides/eecs498/498_FA2019_lecture15.pdf

Evaluating object detectors – Mean Average Precision (mAP)

1. Run object detector on all test images (with NMS)
2. 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



From: https://web.eecs.umich.edu/~justincj/slides/eecs498/498_FA2019_lecture15.pdf

Evaluating object detectors – Mean Average Precision (mAP)

1. Run object detector on all test images (with NMS)
2. 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. Mean Average Precision (mAP) = average of AP for each category

CarAP = 0.65

Cat AP = 0.80

Dog AP = 0.86

mAP@0.5 = 0.77

From: https://web.eecs.umich.edu/~justincj/slides/eecs498/498_FA2019_lecture15.pdf

Evaluating object detectors – Mean Average Precision (mAP)

1. Run object detector on all test images (with NMS)
2. 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. Mean Average Precision (mAP) = average of AP for each category
4. For “COCO mAP”: Compute mAP@thresh for each IoU threshold (0.5, 0.55, 0.6, …, 0.95) and take average

mAP@0.50 = 0.77

mAP@0.55 = 0.71

mAP@0.60 = 0.65

…

mAP@0.95 = 0.2

COCO mAP =0.4

From: https://web.eecs.umich.edu/~justincj/slides/eecs498/498_FA2019_lecture15.pdf

Do we really need two-stage detector?

1. Once per image:
    1. Features extraction (ResNet, MobileNet, VGG…)
    2. Region proposal network

2. Once per region:
    1. Crop features: RoI pooling
    2. Predict object class
    3. Predict bbox offset

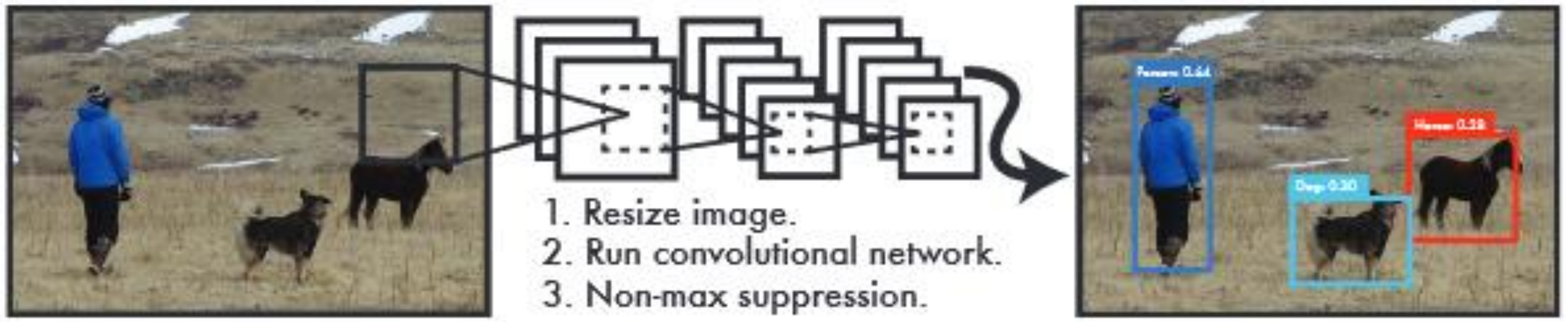
Use two-stage detector is computationally expensive and time consuming !

Single-stage object detectors

- Predict object class and location in **ONE** single step
- Similar to RPN of Faster R-CNN
- Predict the position of the box AND the class of the object in a given box

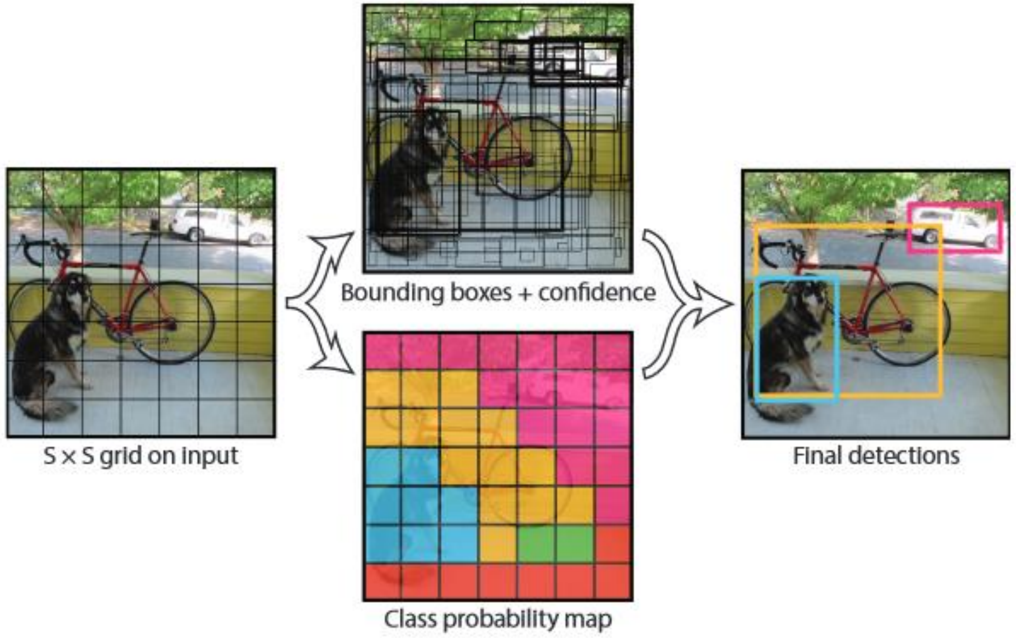
YOLO – You Only Look Once (Redmon et al., 2016)

- Instead of making predictions on many regions of an image, YOLO passes **entire** image at once into a CNN (much faster than two-stage detectors!)
- The CNN predicts the **labels, bounding boxes, and confidence probabilities** for objects in the image
- Perform non-max suppression



YOLO – You Only Look Once (Redmon et al., 2016) overview

- Divide the image into cells with an SxS grid
- Each cell produces class and bounding prediction for objects if the center of an object falls inside that cell
- Each cell contains:
  - Class probabilities
  - B bounding boxes with confidences:
    $$\mathrm{B} \times (x, y, w, h, c)$$

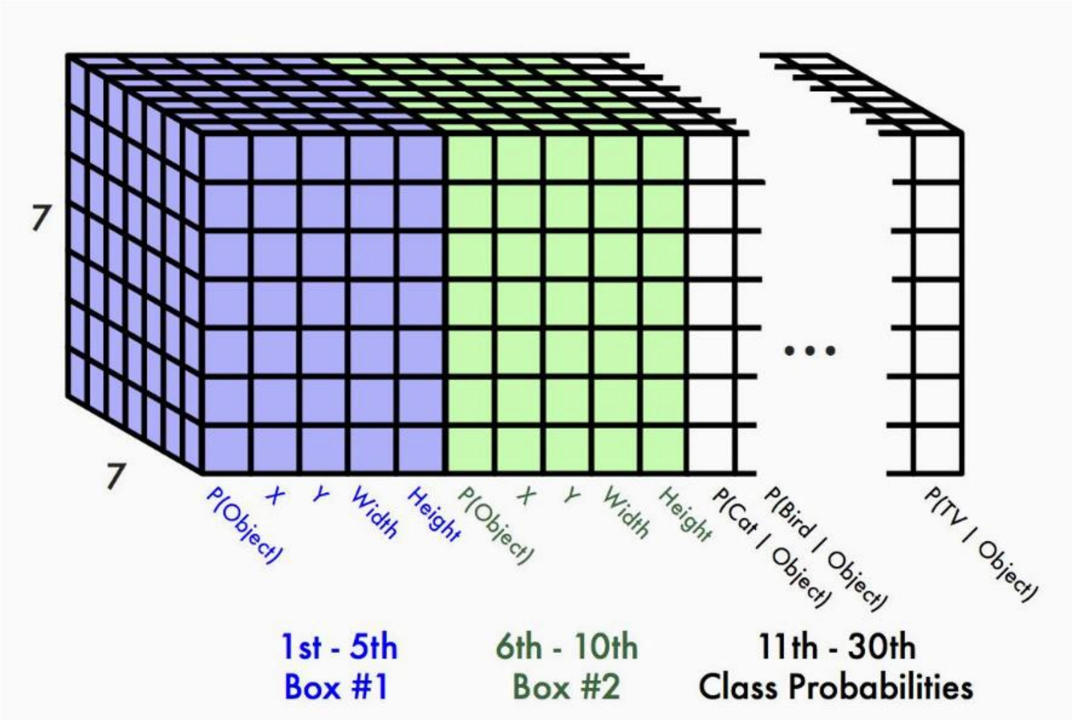


From: https://www.charles-deledalle.fr/pages/files/ucsd_ece285_mlip/5_detection.pdf

YOLO – You Only Look Once (Redmon et al., 2016) overview

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

- Common values:
  - 7x7 grid
  - 2 bounding boxes/cell
  - 20 classes
- This results to an 7x7x(C+Bx5) output



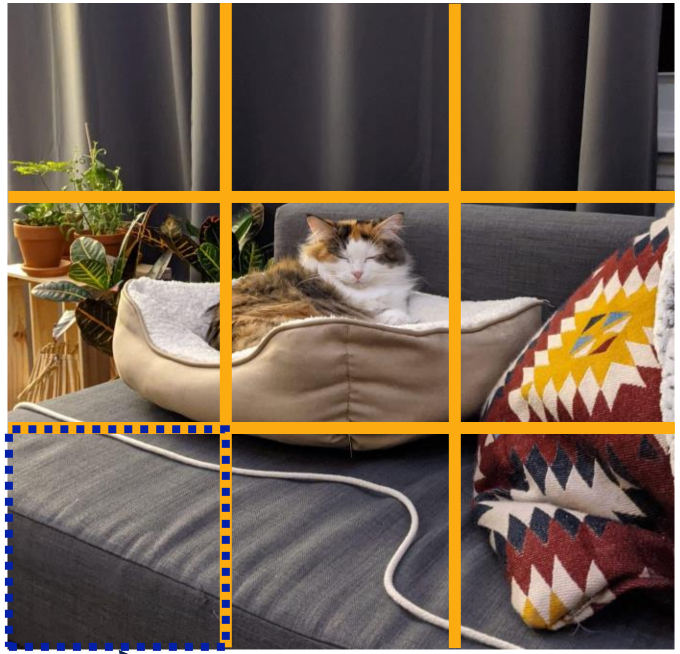
From: https://www.charles-deledalle.fr/pages/files/ucsd_ece285_mlip/5_detection.pdf

YOLO steps

1. Divide the image into cells with an SxS grid

YOLO steps

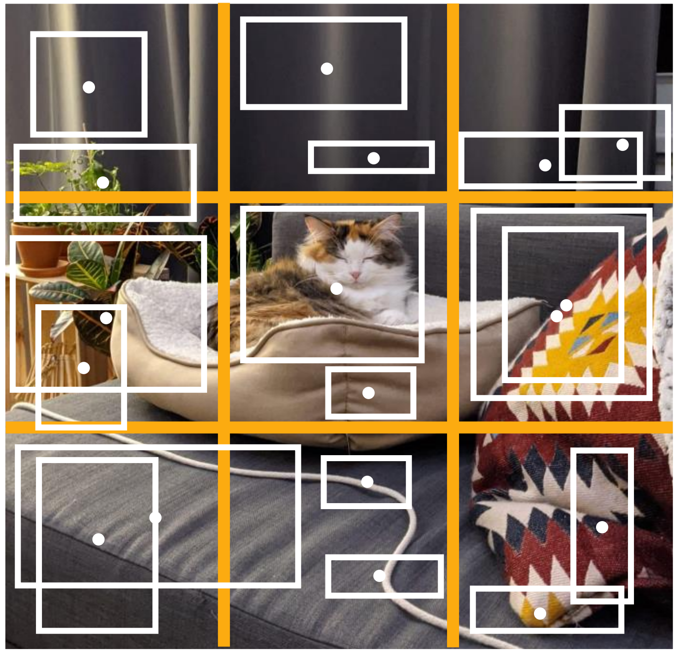
1. Divide the image into cells with an SxS grid



S=3

Cell

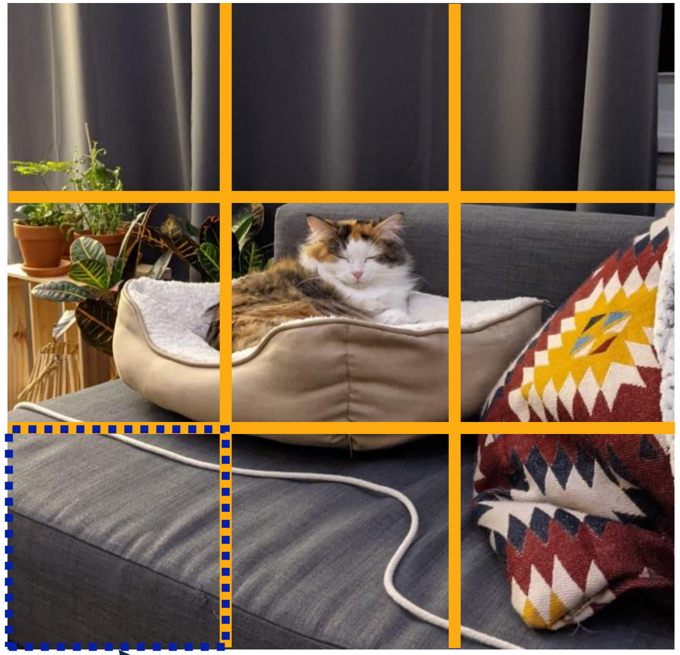
2. Each cell predicts B bounding boxes



B=2

YOLO steps

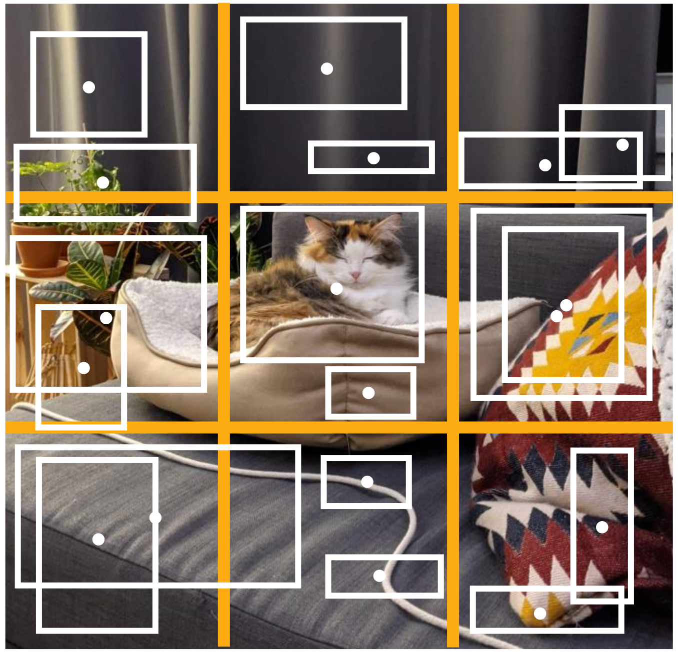
1. Divide the image into cells with an SxS grid



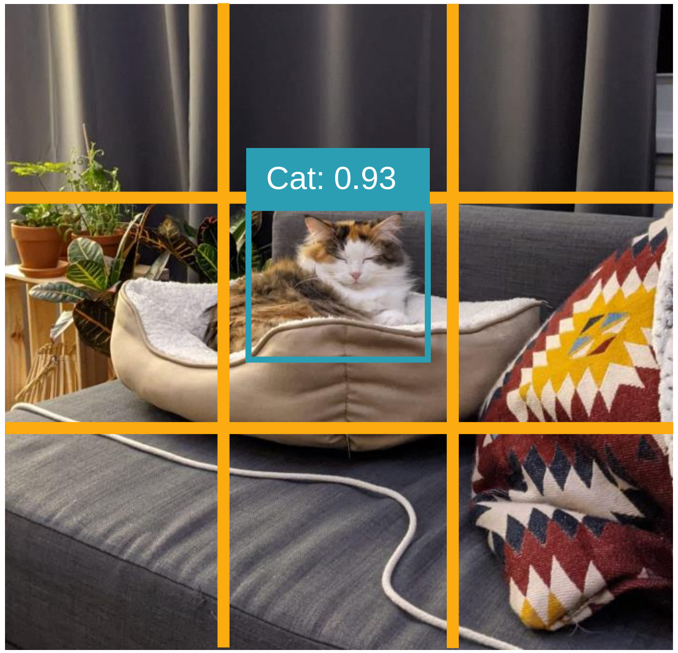
S=3

Cell

2. Each cell predicts B bounding boxes



3. Return bounding boxes above confidence threshold

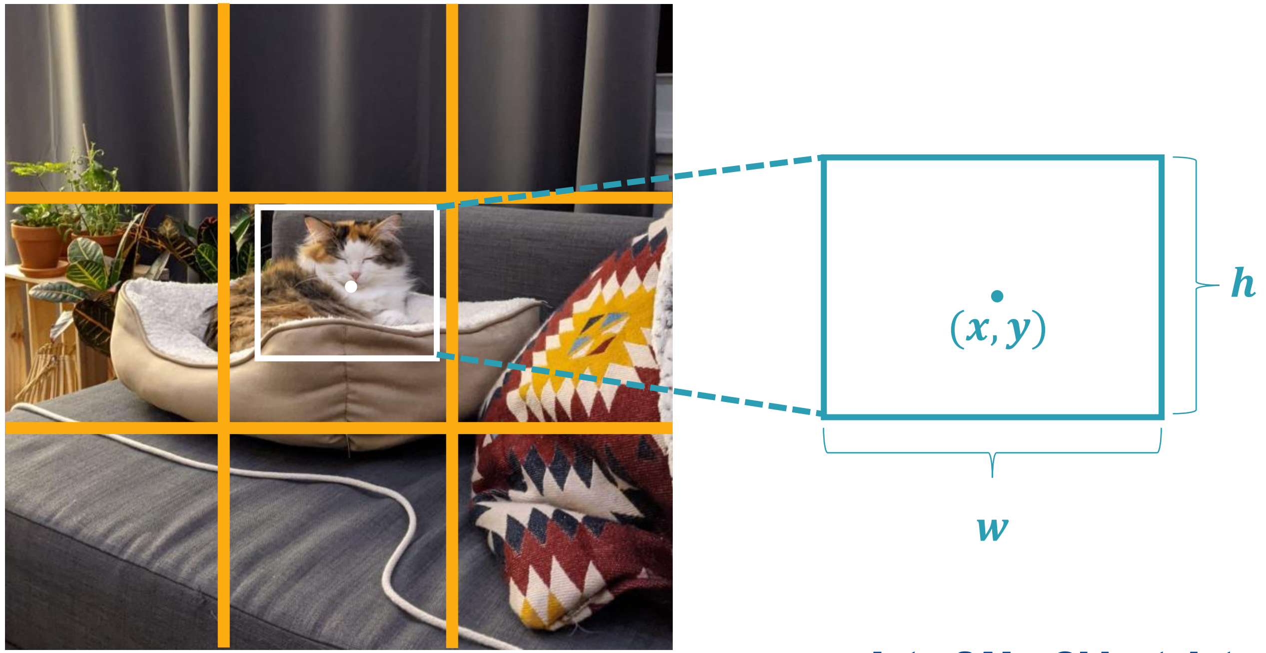


All other bounding boxes have a confidence probability less than the threshold (e.g 0.9) so they are suppressed

How are bounding boxes encoded?

Let's use the previous example where there a 3x3 grid (S=3), each cell predicts 1 bounding box (B=1) and objects are either cat = 1 or human = 2.

For each cell, the CNN predicts a vector y:



$$\boldsymbol{y} = \begin{bmatrix} p_c \\ x \\ y \\ h \\ w \\ c_1 \\ c_2 \end{bmatrix}$$

- p_c: Probability the bounding box contains an object
- x, y: Coordinates of the bounding box's center
- h, w: Width (height) of bounding box as a percent of the cell's width or (height)
- c_1, c_2: Probability the cell contains an object that belongs to class 1 (or 2) given the bounding box contains an object

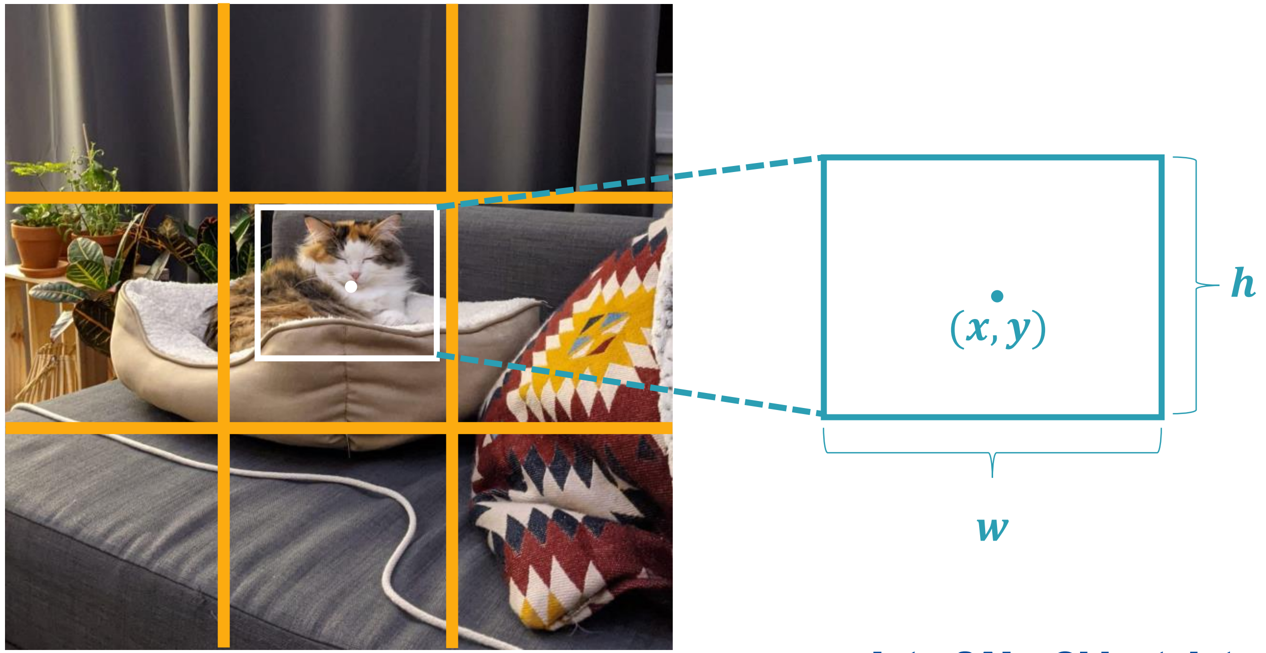
Example:

$$\boldsymbol{y} = \begin{bmatrix} ? \\ ? \\ ? \\ ? \\ ? \\ ? \\ ? \end{bmatrix}$$

How are bounding boxes encoded?

Let's use the previous example where there a 3x3 grid (S=3), each cell predicts 1 bounding box (B=1) and objects are either cat = 1 or human = 2.

For each cell, the CNN predicts a vector y:



$$\boldsymbol{y} = \begin{bmatrix} p_c \\ x \\ y \\ h \\ w \\ c_1 \\ c_2 \end{bmatrix}$$

- p_c: Probability the bounding box contains an object
- x, y: Coordinates of the bounding box's center
- h, w: Width (height) of bounding box as a percent of the cell's width or (height)
- c_1, c_2: Probability the cell contains an object that belongs to class 1 (or 2) given the bounding box contains an object

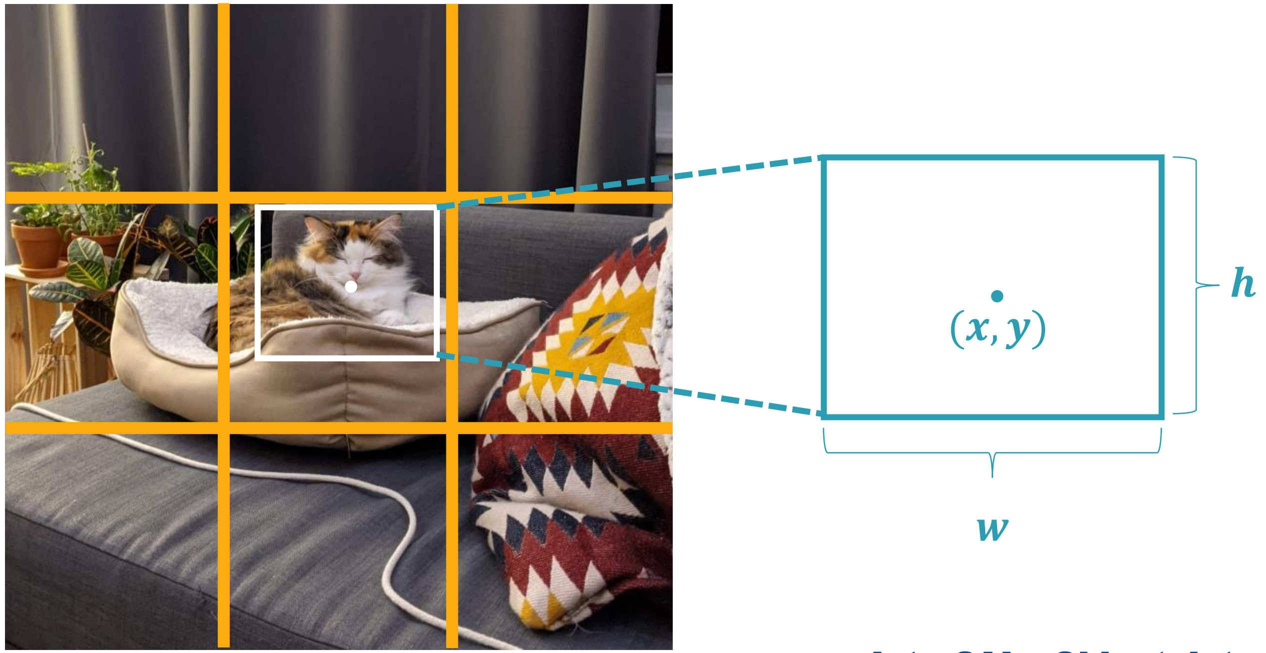
Example:

$$\boldsymbol{y} = \begin{bmatrix} 1 \\ ? \\ ? \\ ? \\ ? \\ ? \\ ? \end{bmatrix}$$

How are bounding boxes encoded?

Let’s use the previous example where there a 3x3 grid (S=3), each cell predicts 1 bounding box (B=1) and objects are either cat = 1 or human = 2.

For each cell, the CNN predicts a vector y:



$$\boldsymbol{y} = \begin{bmatrix} p_c \\ x \\ y \\ h \\ w \\ c_1 \\ c_2 \end{bmatrix}$$

- p_c: Probability the bounding box contains an object
- x, y: Coordinates of the bounding box’s center
- h, w: Width (height) of bounding box as a percent of the cell’s width or (height)
- c_1, c_2: Probability the cell contains an object that belongs to class 1 (or 2) given the bounding box contains an object

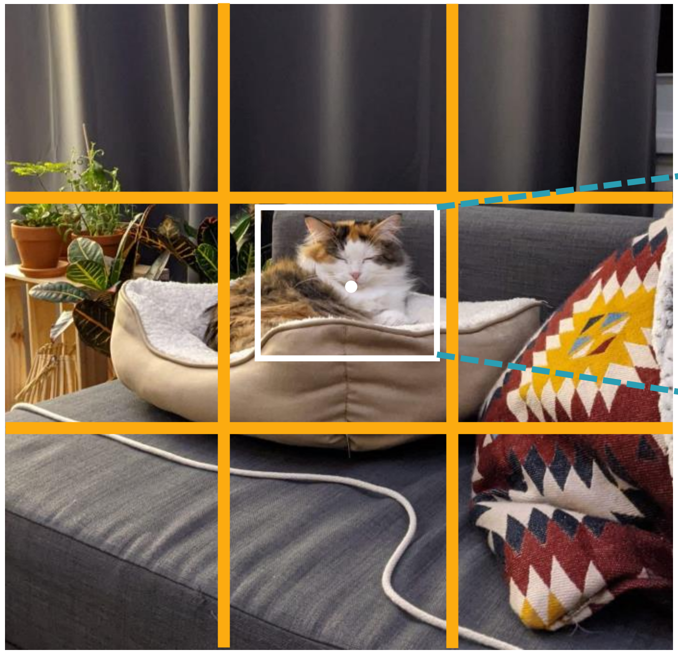
Example:

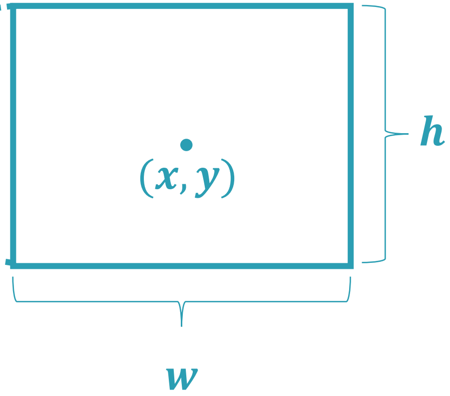
$$\boldsymbol{y} = \begin{bmatrix} 1 \\ x \\ y \\ h \\ w \\ ? \\ ? \end{bmatrix}$$

How are bounding boxes encoded?

Let’s use the previous example where there a 3x3 grid (S=3), each cell predicts 1 bounding box (B=1) and objects are either cat = 1 or human = 2.

For each cell, the CNN predicts a vector y:





$$\boldsymbol{y} = \begin{bmatrix} p_c \\ x \\ y \\ h \\ w \\ c_1 \\ c_2 \end{bmatrix}$$

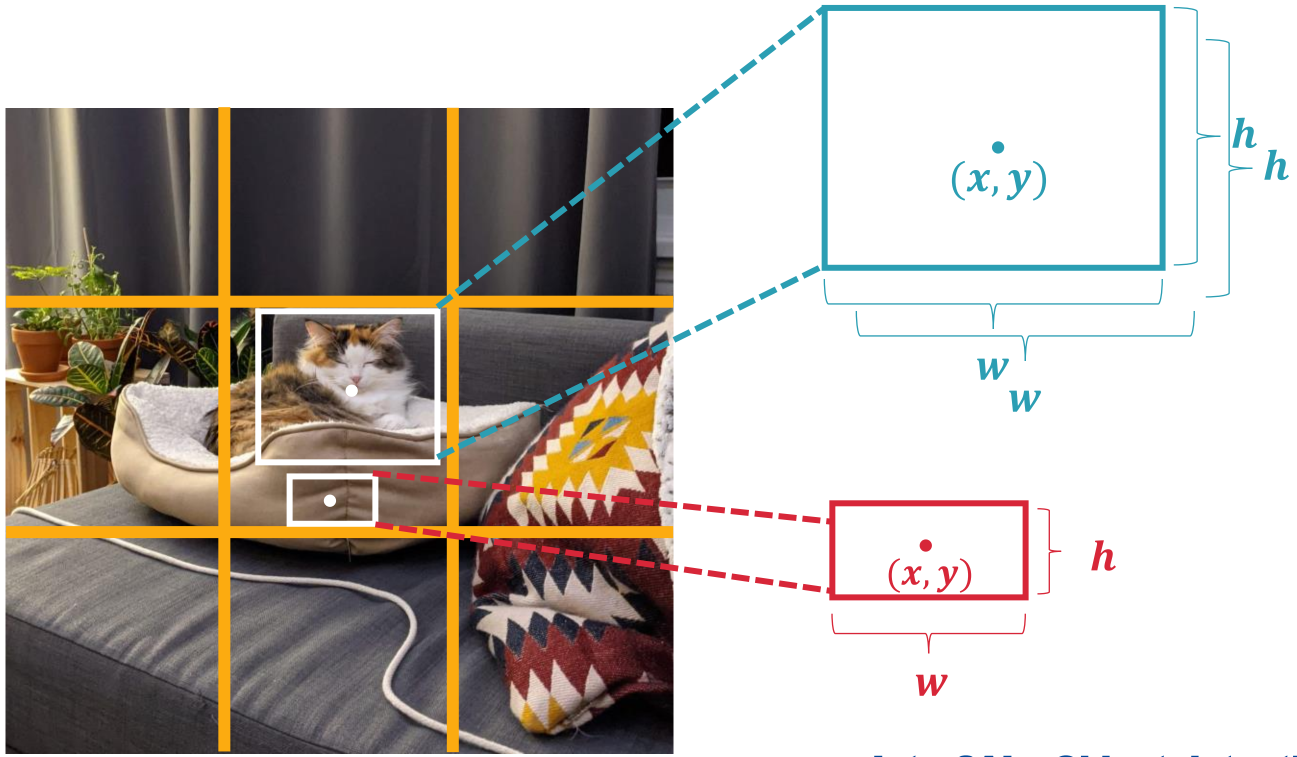
- p_c: Probability the bounding box contains an object
- x, y: Coordinates of the bounding box’s center
- h, w: Width (height) of bounding box as a percent of the cell’s width or (height)
- c_1, c_2: Probability the cell contains an object that belongs to class 1 (or 2) given the bounding box contains an object

Example:

$$\boldsymbol{y} = \begin{bmatrix} 1 \\ x \\ y \\ h \\ w \\ 1 \\ 0 \end{bmatrix}$$

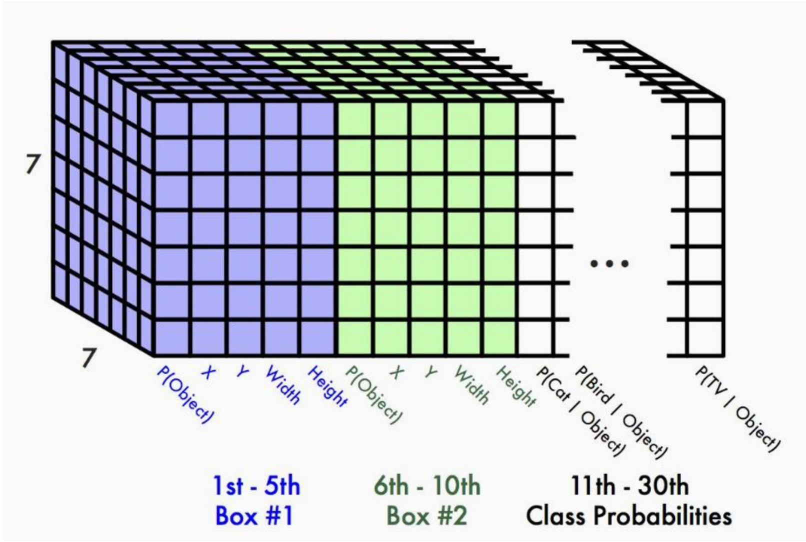
Encoding multiple bounding boxes

What happens if we predict multiple bounding boxes per cell (B>1) ? We simply augment y:



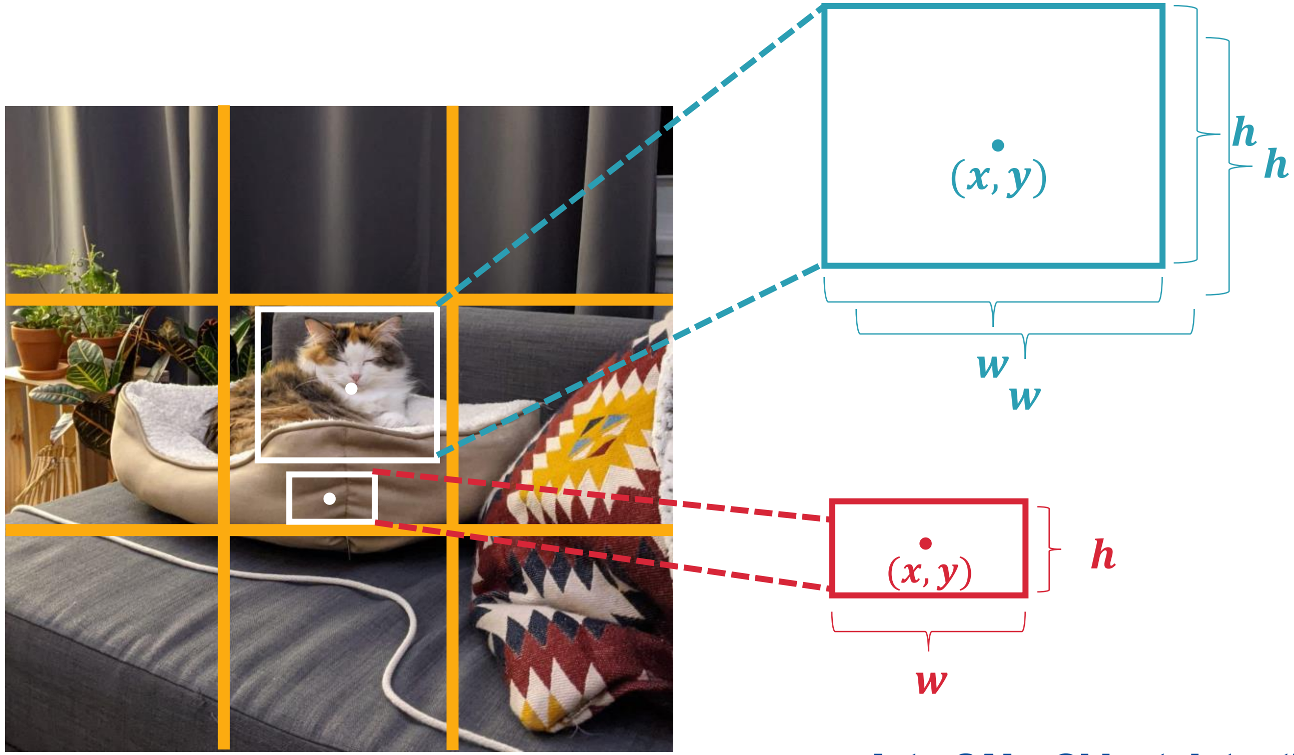
$$y = \begin{bmatrix} p_c \\ x \\ y \\ h \\ w \\ p_c \\ x \\ y \\ h \\ w \\ c_1 \\ c_2 \end{bmatrix}$$

The CNN will predict y for each cell, so the size of the output tensor should be: SxSx(5B+5)



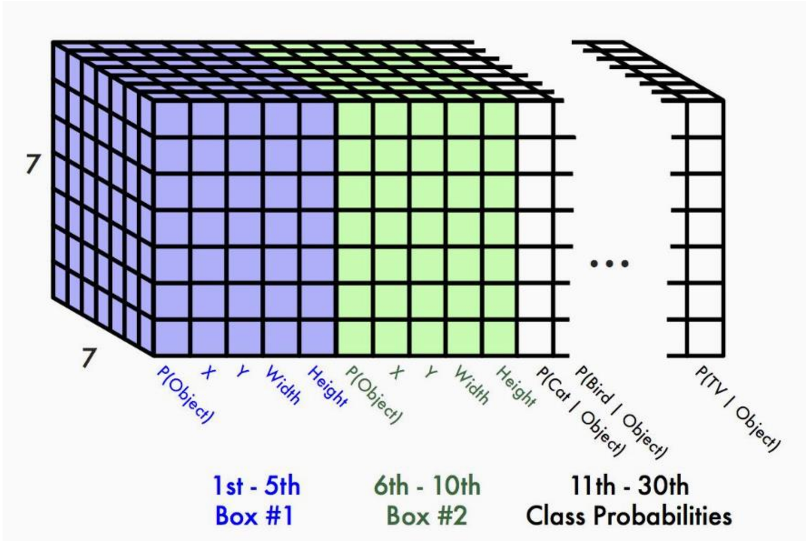
Encoding multiple bounding boxes

What happens if we predict multiple bounding boxes per cell (B>1) ? We simply augment y:



$$y = \begin{bmatrix} 1 \\ x \\ y \\ h \\ w \\ 0 \\ x \\ y \\ h \\ w \\ 1 \\ 0 \end{bmatrix}$$

The CNN will predict y for each cell, so the size of the output tensor should be: SxSx(5B+5)



Training YOLO

- Calculates the sum-squared loss of between the predicted coordinates and the ground truth coordinates
- Calculates the sum-squared loss of the predicted height and width

- Calculates the sum-squared error between the predicted confidence score and the ground truth for each bounding box in each cell
- Calculates the sum-squared error of the cells which do not contain any objects.
- Computes the same loss for the class probabilities

Localization error

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

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

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

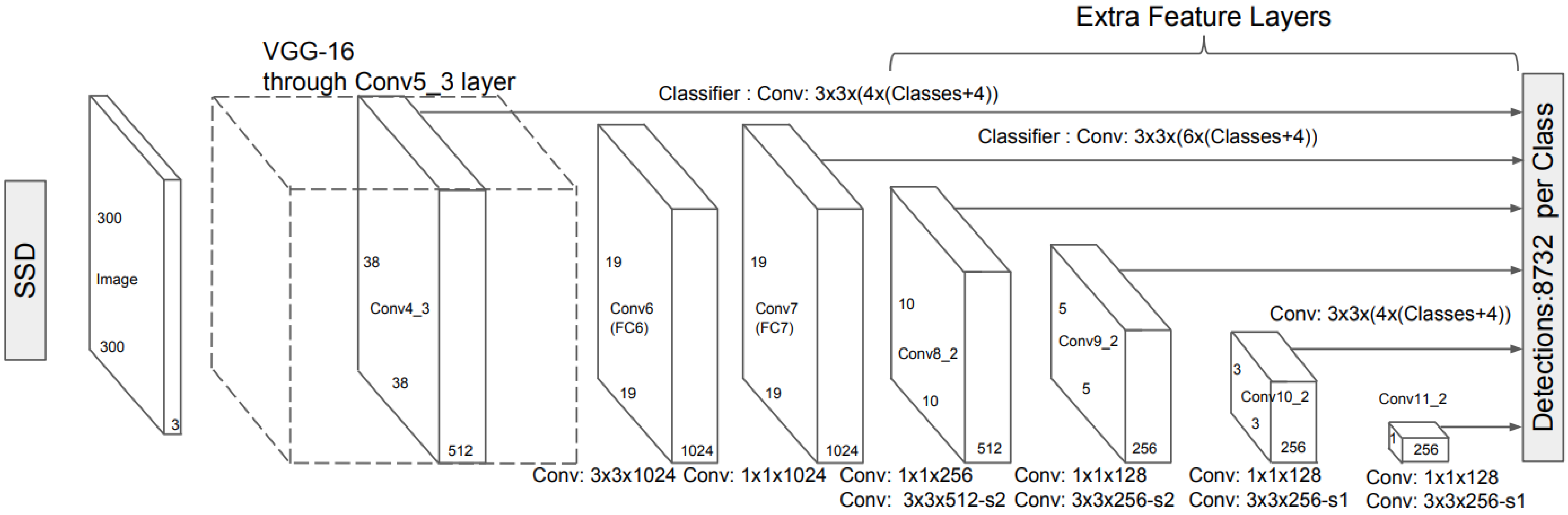
$$+ \lambda_{\text{noobj}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{noobj}} \left(C_i - \hat{C}_i \right)^2$$

$$+ \sum_{i=0}^{S^2} \mathbb{1}_{i}^{\text{obj}} \sum_{c \in \text{classes}} (p_i(c) - \hat{p}_i(c))^2 \qquad (3)$$

Classification error

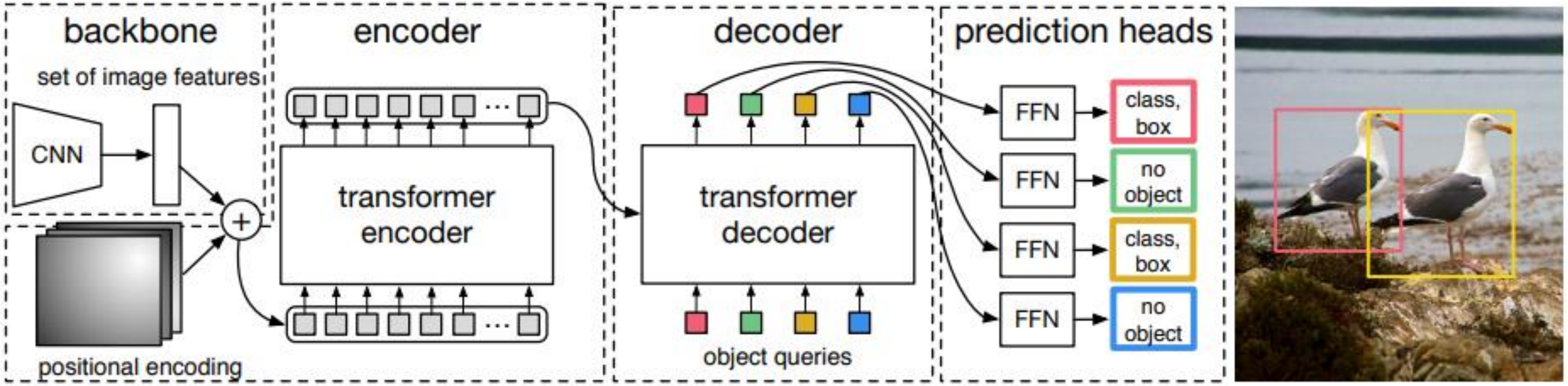
From: https://www.charles-deledalle.fr/pages/files/ucsd_ece285_mlip/5_detection.pdf

Appendix
Single Shot Detector model (SSD)



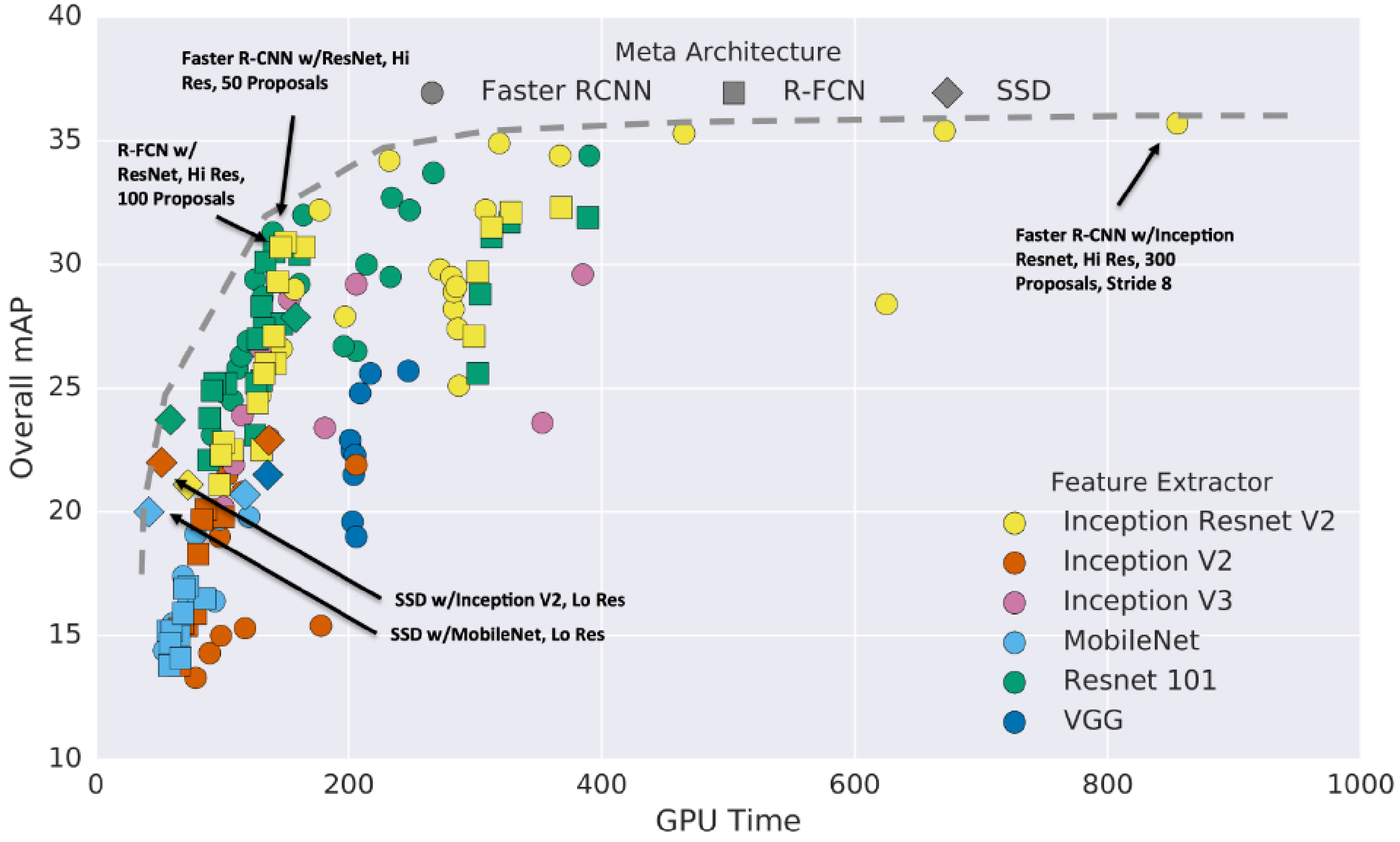
Liu, W., Anguelov, D., Erhan, D., Szegedy, C., Reed, S., Fu, C. Y., & Berg, A. C. (2016, October). Ssd: Single shot multibox detector. In *European conference on computer vision* (pp. 21-37). Springer, Cham.

Appendix
Object detection using transformers



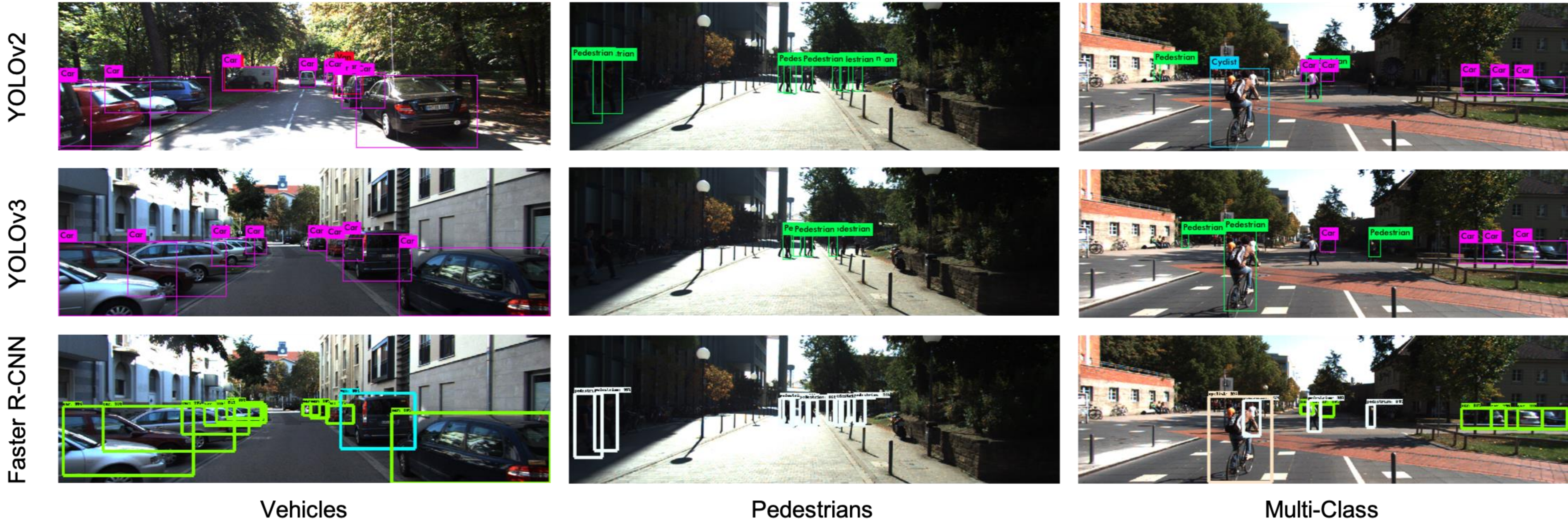
Carion, N., Massa, F., Synnaeve, G., Usunier, N., Kirillov, A., & Zagoruyko, S. (2020, August). End-to-end object detection with transformers. In *European Conference on Computer Vision* (pp. 213-229). Springer, Cham.

Single stage detectors or two-stage detectors?



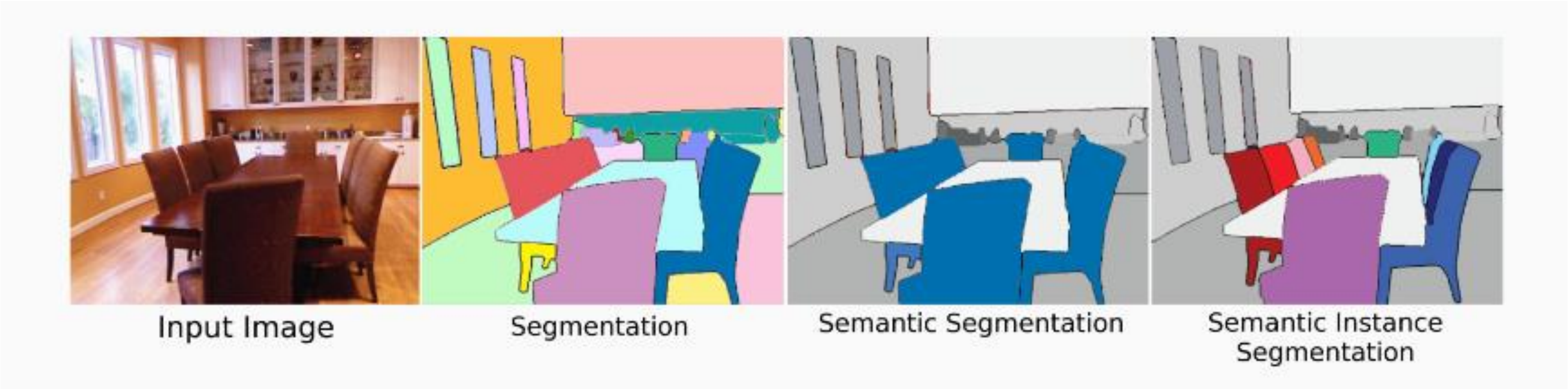
Huang et al, "Speed/accuracy trade-offs for modern convolutional object detectors", CVPR 2017

Object detection application – Autonomous driving



What is segmentation?

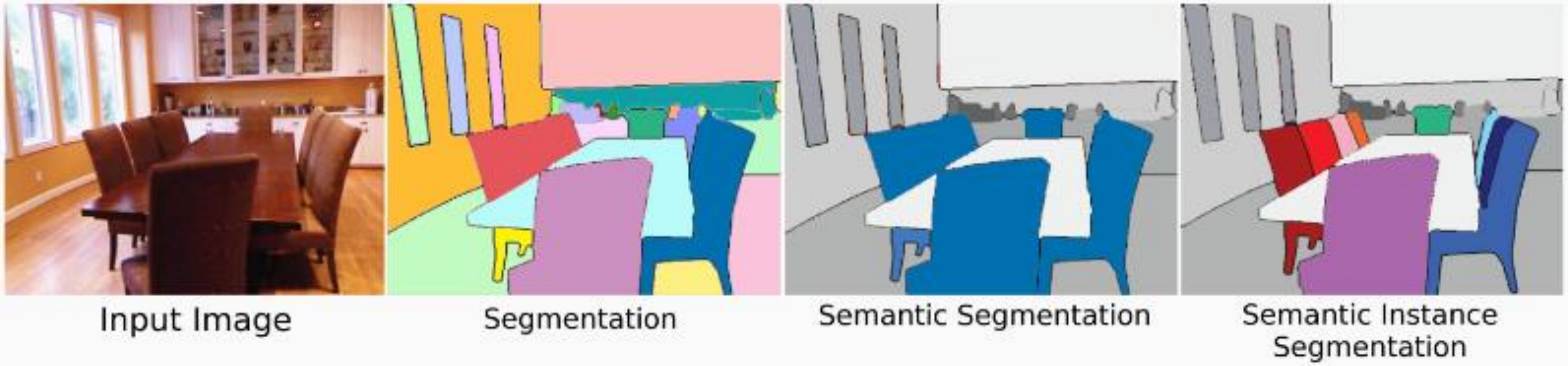
- Partition of an image into several ”coherent” parts/segments
- Without any attempt at understanding what these parts represent
- Typically based on color, textures, smoothness of boundaries
- Also referred to as super-pixel segmentation



From: https://www.charles-deledalle.fr/pages/files/ucsd_ece285_mlip/5_detection.pdf

What is segmentation?

- Semantic segmentation
  - Each segment corresponds to a class label (objects + background)
  - Also referred to as scene parsing or scene labeling
- Instance segmentation:
  - Find object boundaries between objects, including delineations between instances of the same object.
- Semantic instance segmentation or **panoptic segmentation**: find object boundaries + labels.



From: https://www.charles-deledalle.fr/pages/files/ucsd_ece285_mlip/5_detection.pdf

Image segmentation - Overview

- Object classification and detection:
  1. Convolutional backbone for features extraction → low-resolution
  2. Use the feature map to regress bounding boxes and assign a class to each pixel of the feature map
- Object segmentation:
  - Assign each pixel to a class in the input dimension → high-resolution output
  - Project low-resolution feature onto the pixel space
  - Learn high-level and low-level features

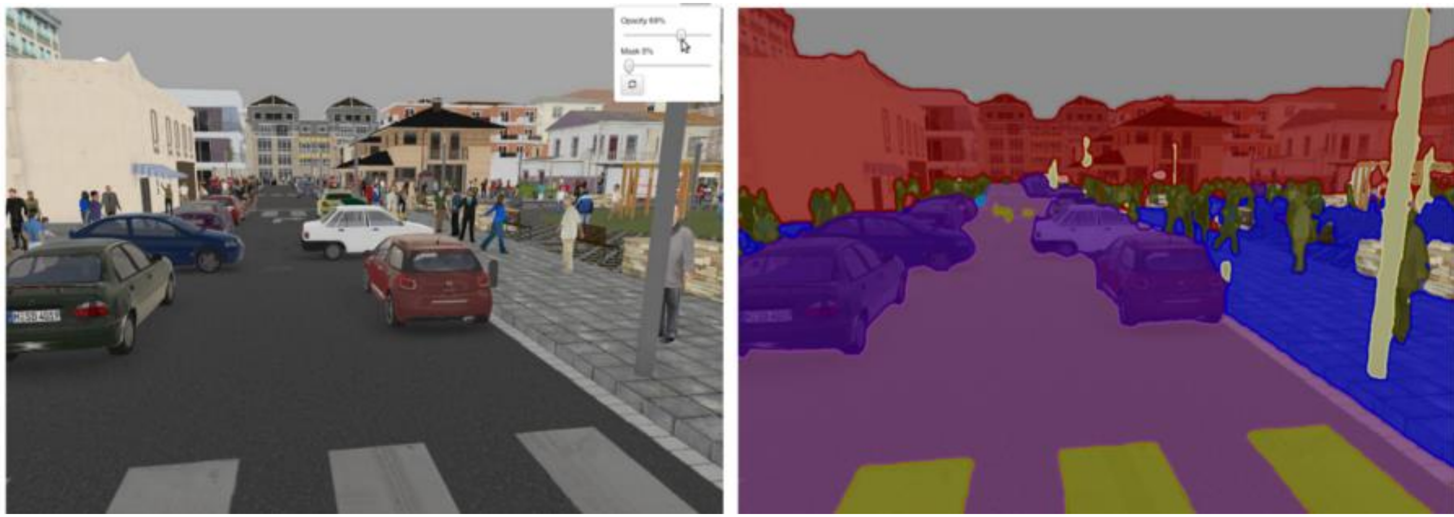
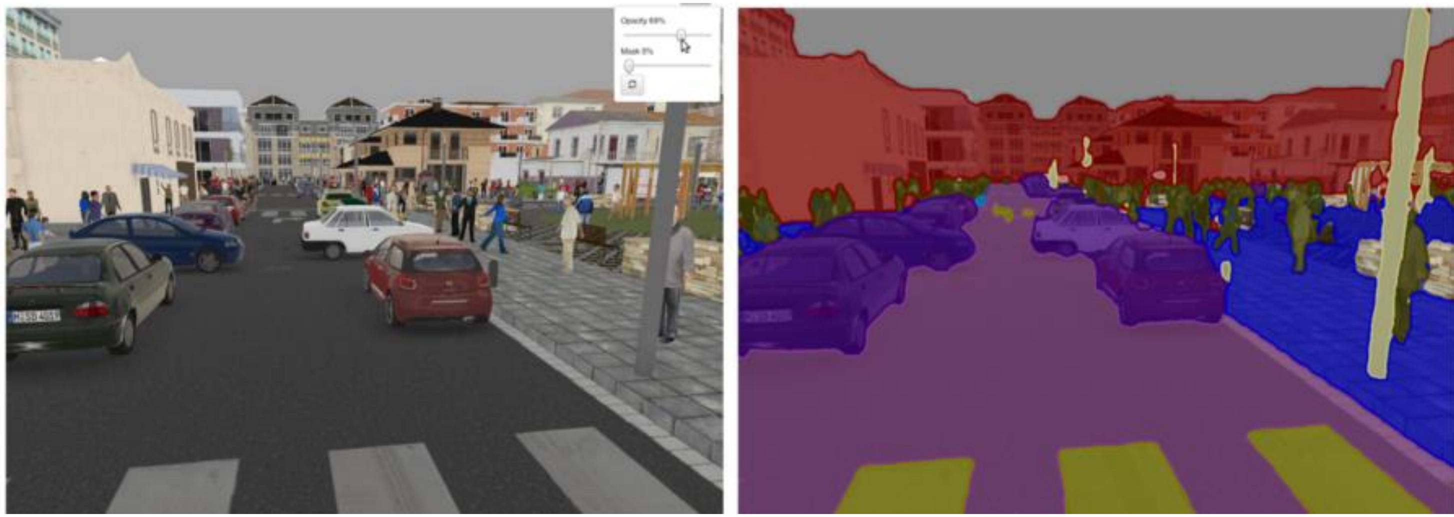


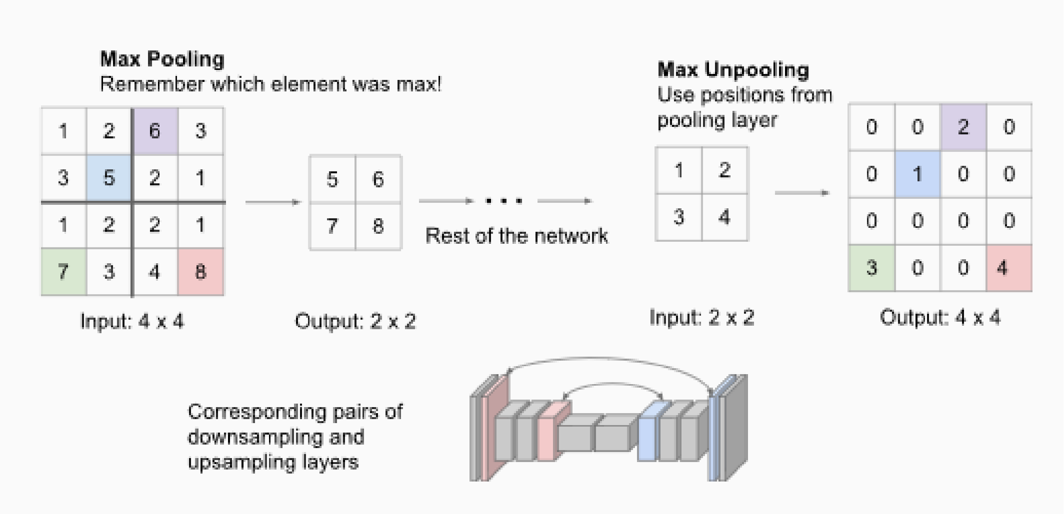
Image segmentation - Overview

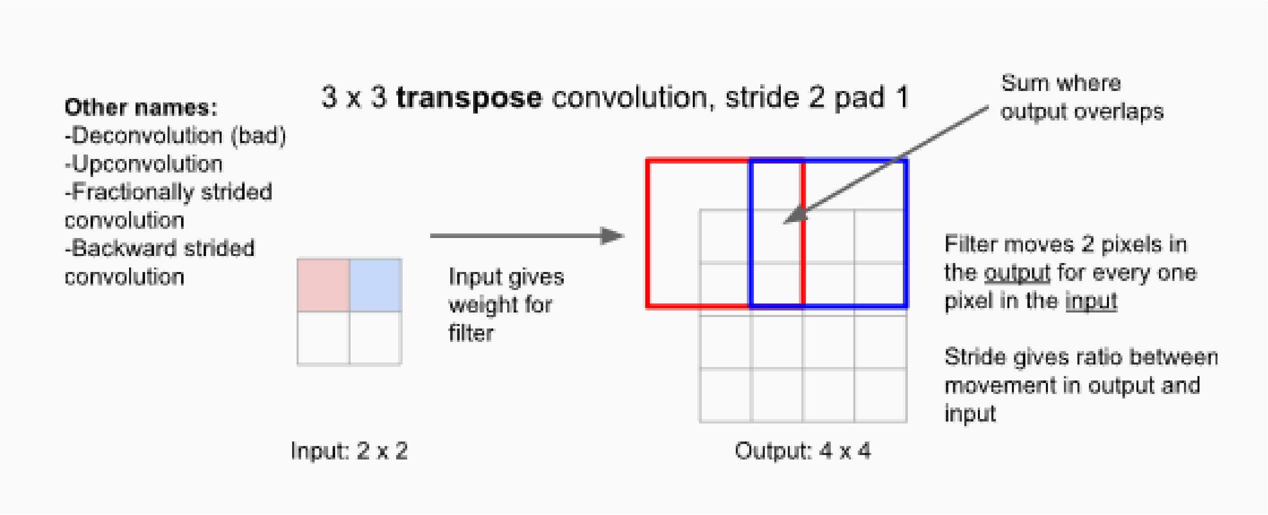
- Object classification and detection:
  1. Convolutional backbone for features extraction → low-resolution
  2. Use the feature map to regress bounding boxes and assign a class to each pixel of the feature map
- Object segmentation:
  - Assign each pixel to a class in the input dimension → high-resolution output
  - Project low-resolution feature onto the pixel space: HOW?
  - Learn high-level and low-level features



The upsampling operation

- Idea: restore the condensed feature map to the original size of the input image → expand the feature dimension
- How?
  - Unpooling
  - Transposed convolution





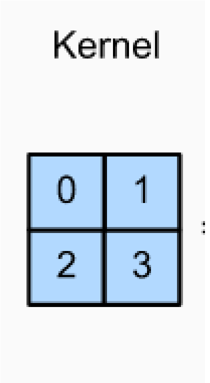
From: https://www.charles-deledalle.fr/pages/files/ucsd_ece285_mlip/5_detection.pdf

Transpose convolution

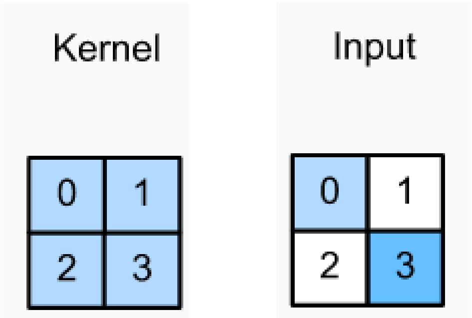
- Transposed convolutions are used to upsample the input feature map to a desired output feature map using some learnable parameters.
- Consider a 2x2 encoded feature map which needs to be upsampled to a 3x3 feature map:

Transpose convolution

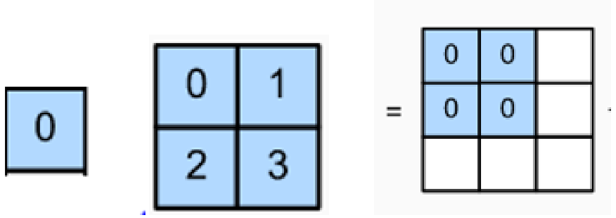
- We take a kernel of size 2x2 with a stride of 1 and zero padding:



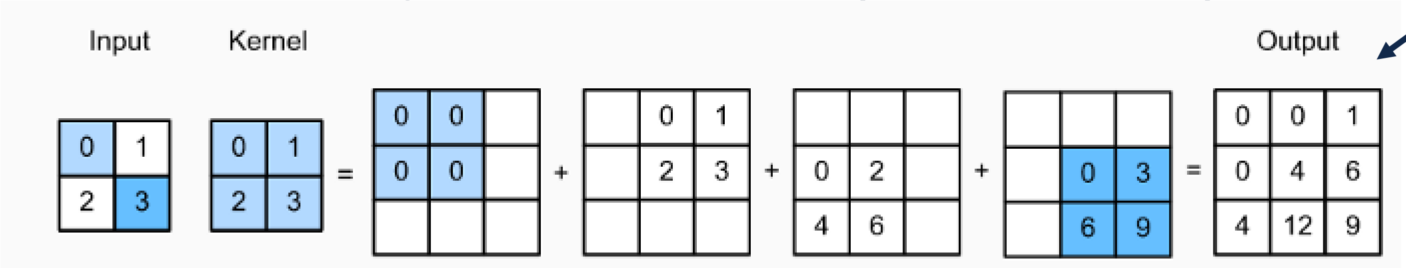
Transpose convolution



- We take a kernel of size 2x2 with a stride of 1 and zero padding:
- Now we take the upper left element of the input feature map and multiply it with every element of the kernel:



- Similarly for all the remaining elements of the input feature map:

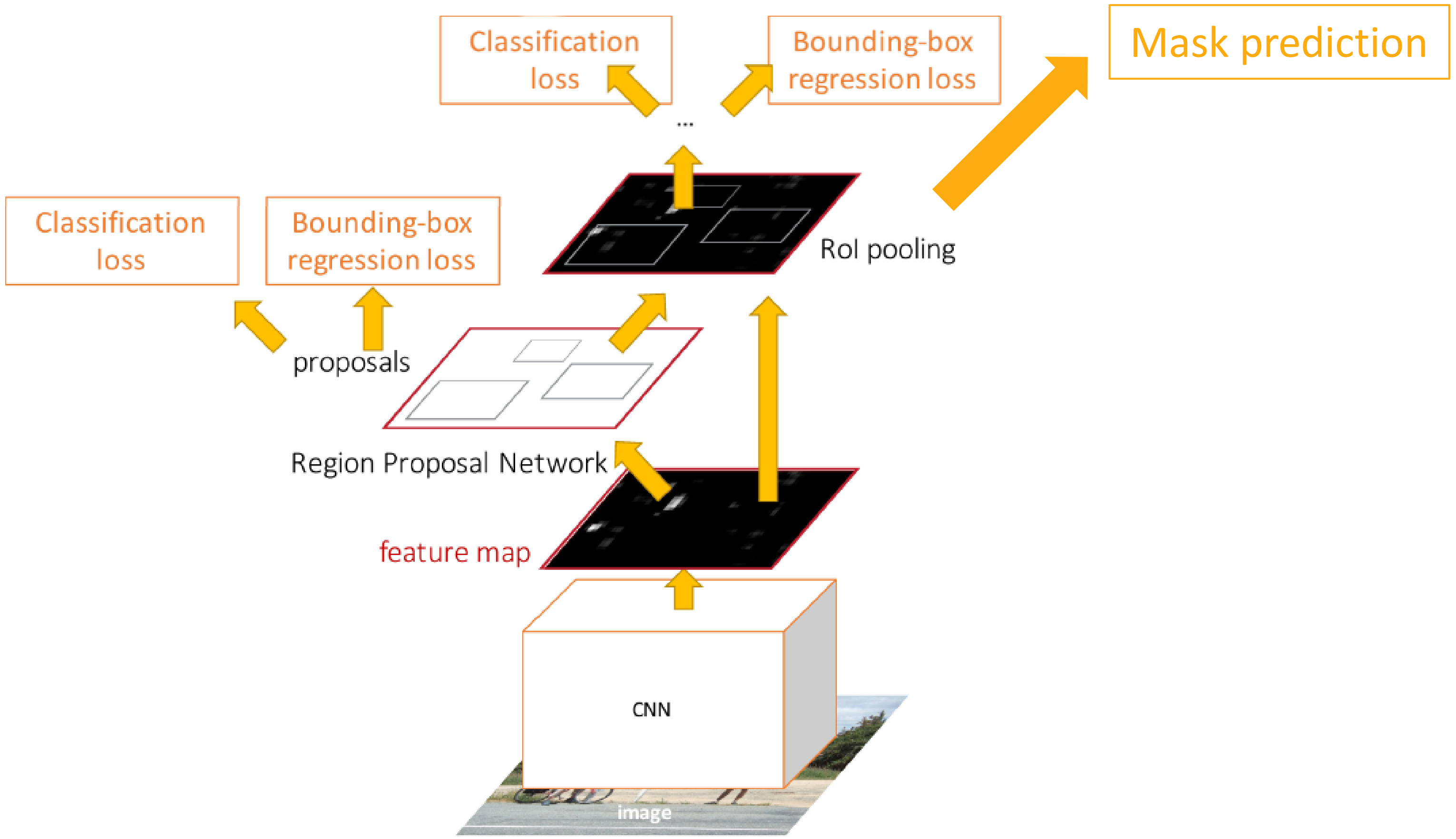


The resulting output will be the final upsampled feature map having the required spatial dimensions of 3x3

Object segmentation approaches

- Region based segmentation approaches: Mask R-CNN, SDS…
- Fully Convolutional Network (FCNs) approaches: FCN, DeepLab…
- Transformer based method: SegFormer, SOTR…

Region based approaches
From Faster R-CNN to Mask R-CNN (He et al., 2017)



Region based approaches
From Faster R-CNN to Mask R-CNN (He et al., 2017)

- Generate a mask for each proposals of the RPN
- Drawbacks:
  - Time consuming (one mask for each proposals ~ 2000)
  - The feature does not contain enough spatial information for precise boundary generation
  - The feature is not compatible with the segmentation task.

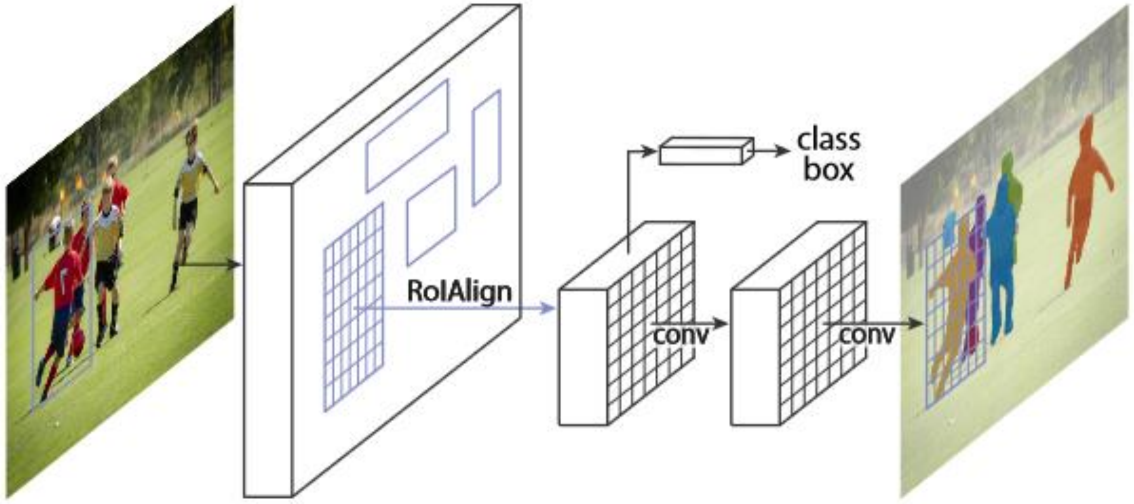
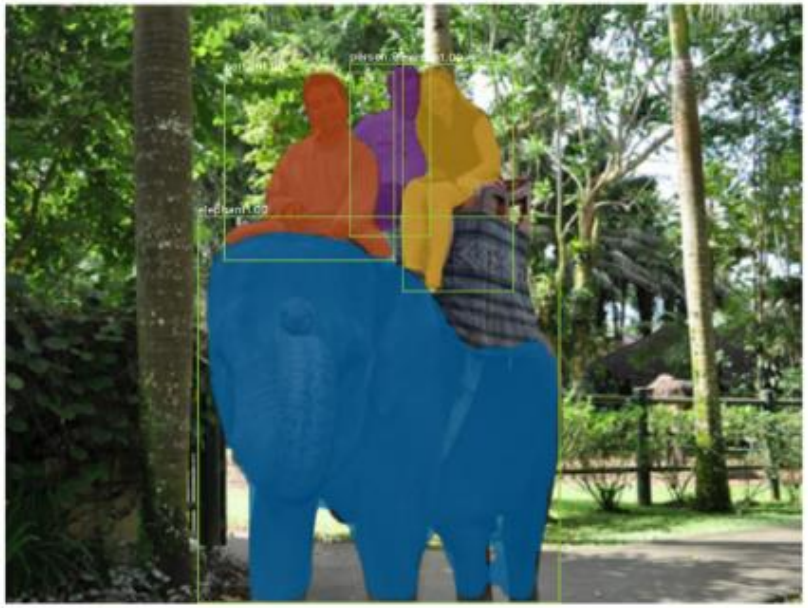
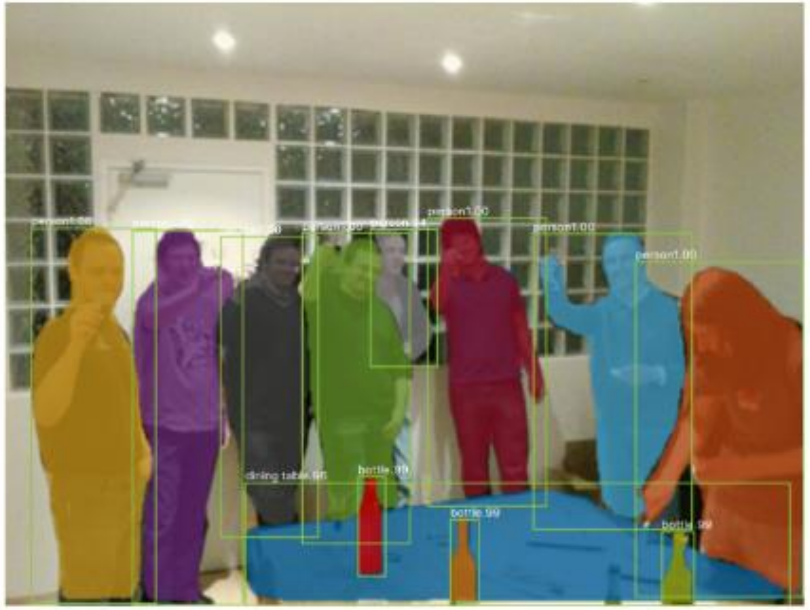


Figure 1. The **Mask R-CNN** framework for instance segmentation.

Region based approaches
From Faster R-CNN to Mask R-CNN (He et al., 2017)



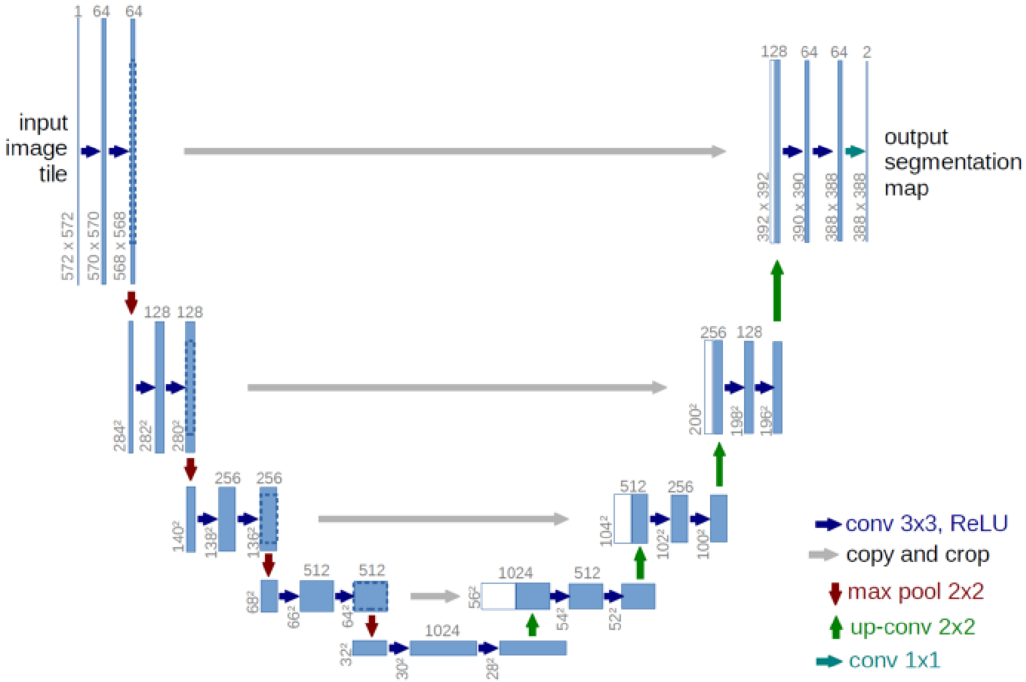


FCN based approaches
Unet (Ronneberger et al., 2015)

- Semantic segmentation requires a mechanism to project the discriminative features learn at different stages of the encoder onto the pixel space

- Originally invented and first used for biomedical image segmentation

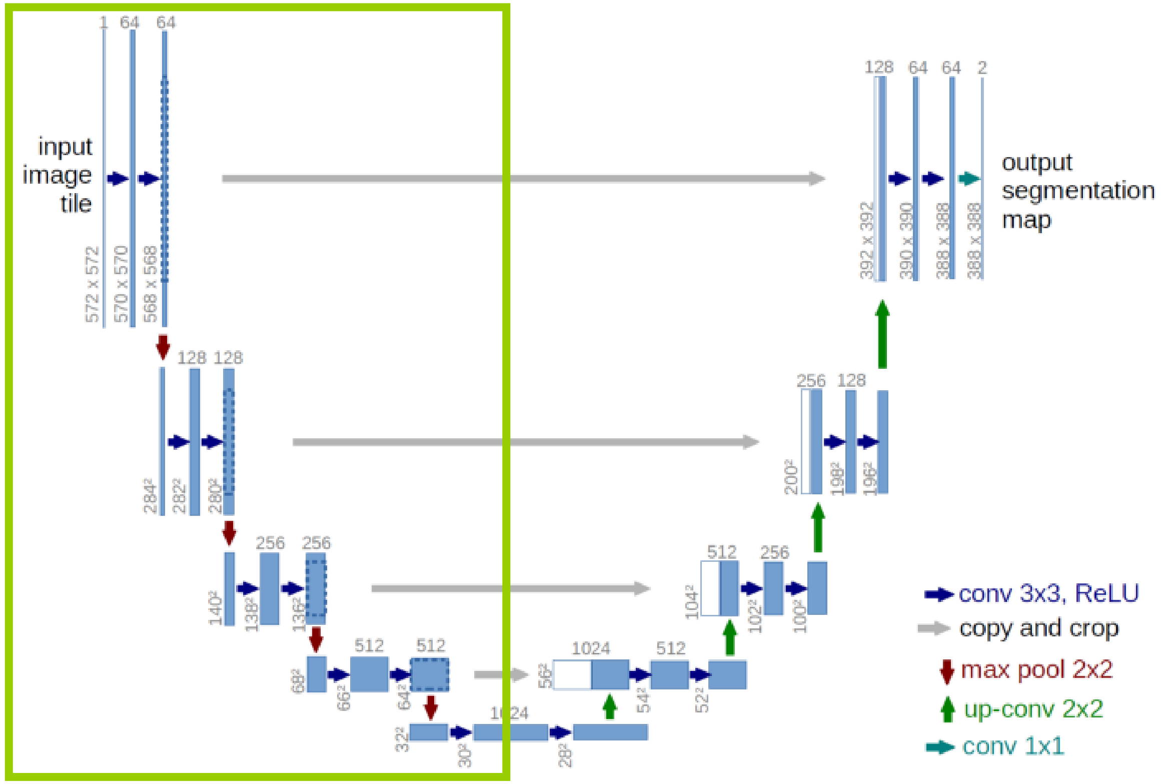
- Encoder-decoder architecture



Ronneberger O., Fischer P., Brox T. (2015) U-Net: Convolutional Networks for Biomedical Image Segmentation. https://doi.org/10.1007/978-3-319-24574-4_28

FCN based approaches
Unet - Encoder

- Usually a pre-trained classification network like VGG or ResNet
- Encode the input image into feature representations at multiple different levels
- Done with convolution blocks followed by maxpooling layers



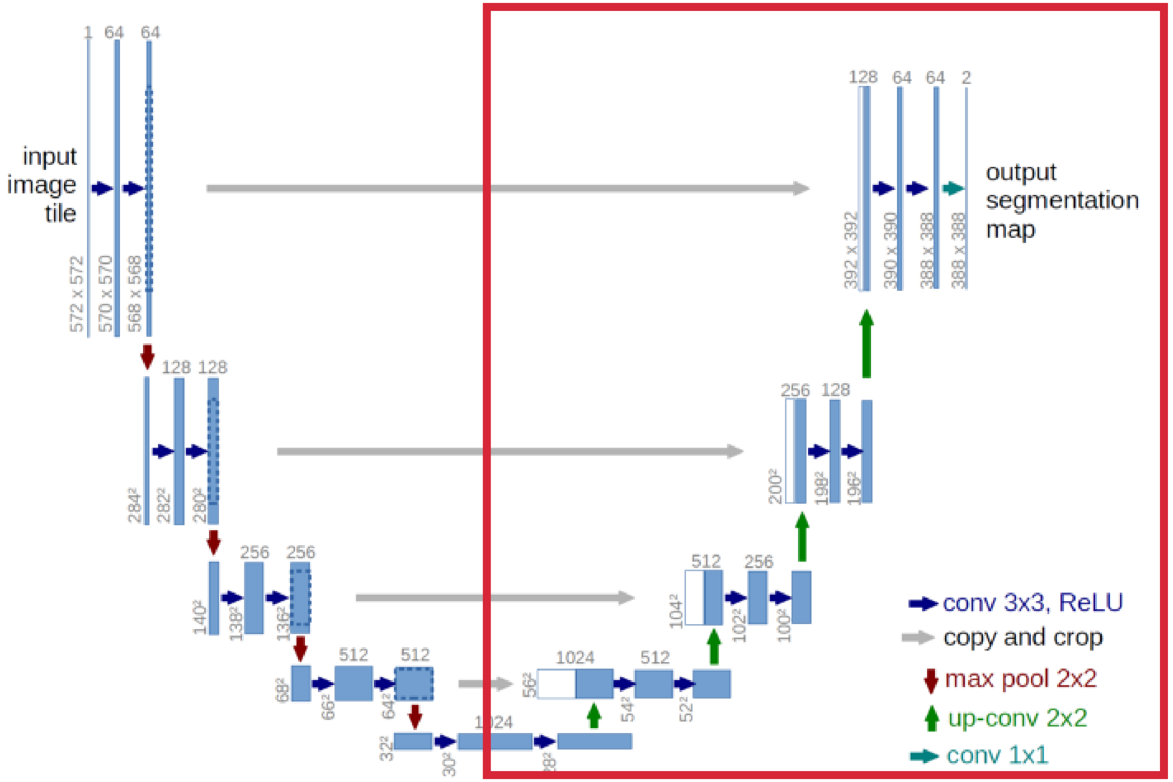
Ronneberger O., Fischer P., Brox T. (2015) U-Net: Convolutional Networks for Biomedical Image Segmentation. https://doi.org/10.1007/978-3-319-24574-4_28

FCN based approaches
Unet - Decoder

- Goal: project the features learnt by the encoder onto the pixel space (higher resolution) to get a dense classification

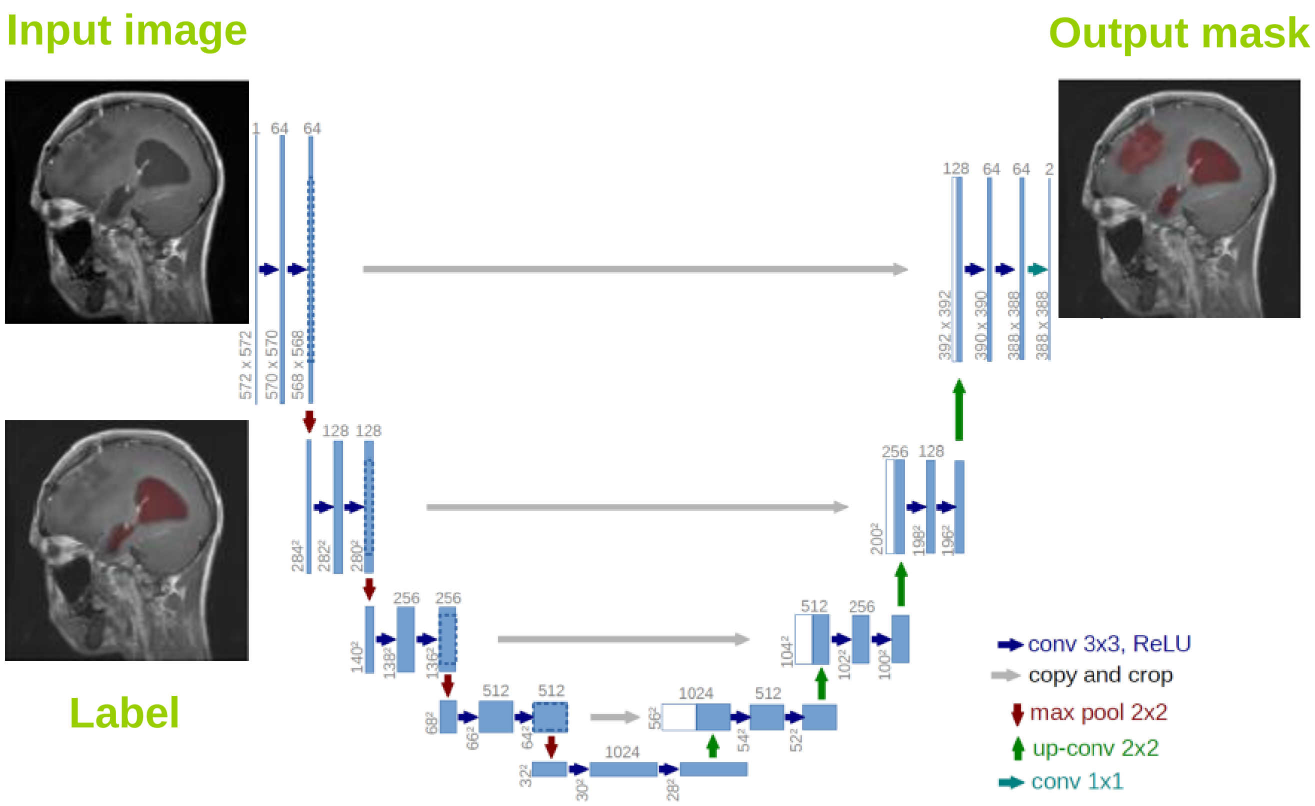
- Layers of the decoder:
  - Up sampling
  - Concatenation
  - Convolutions

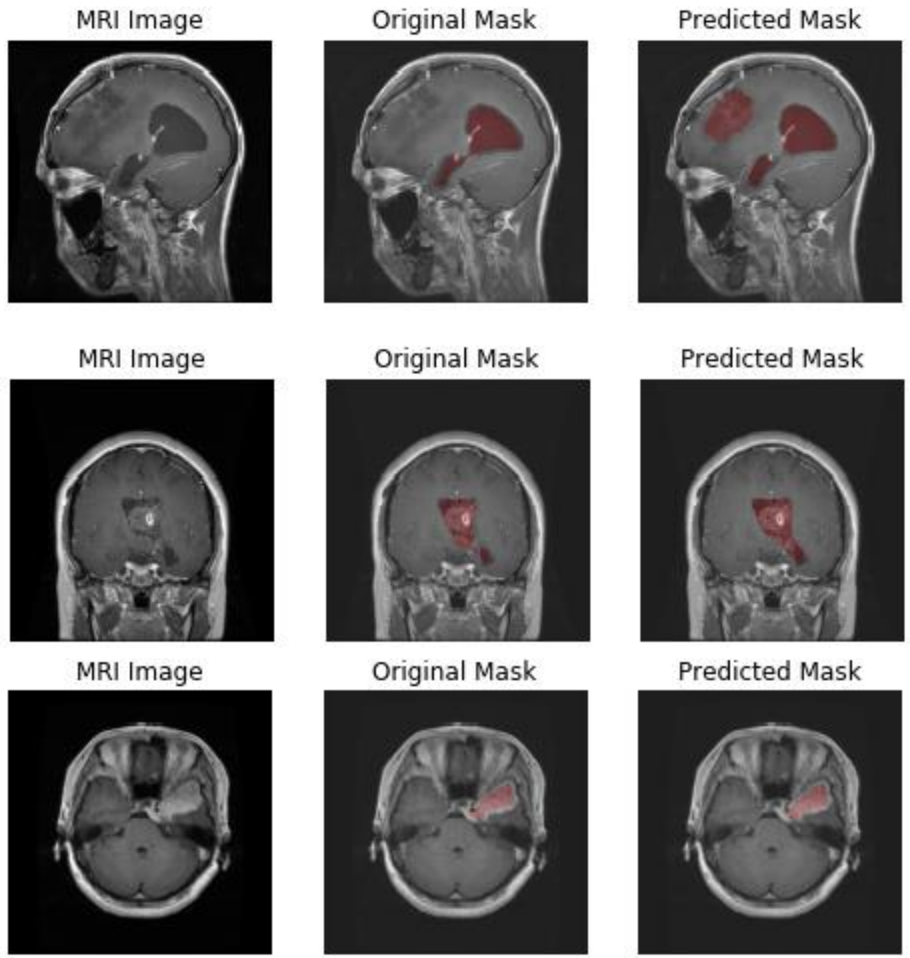
- Concatenation of higher resolution feature maps with up sampled features:
  - Better learning of representations
  - Counterbalance up sampling's information loss



Ronneberger O., Fischer P., Brox T. (2015) U-Net: Convolutional Networks for Biomedical Image Segmentation. https://doi.org/10.1007/978-3-319-24574-4_28

Application of semantic segmentation – Brain tumour segmentation





Data from: https://github.com/adityajn105/brain-tumor-segmentation-unet

Practical session

- Speed differences between R-CNN, Fast R-CNN, Faster R-CNN
- How does anchors work?
- Mask R-CNN
- Evaluate models (Intersection over Union, mAP)
- Left ventricle segmentation with U-Net