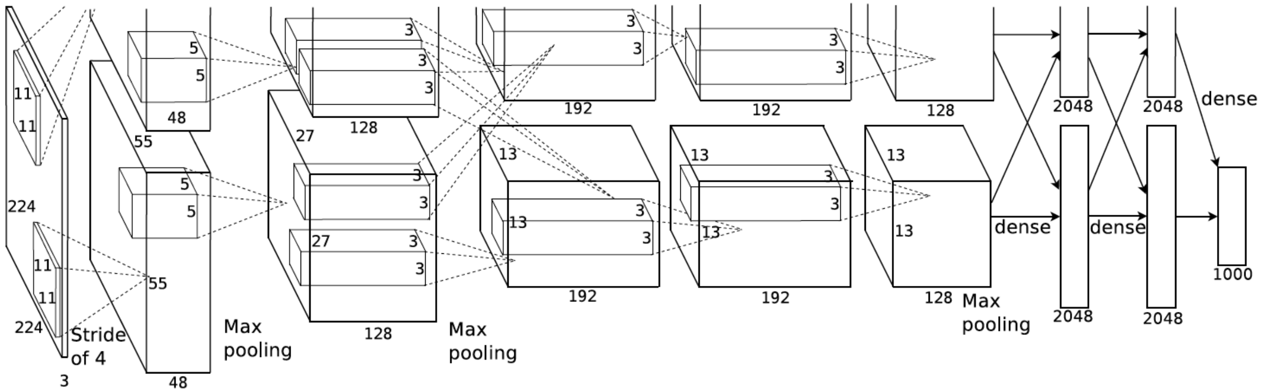
Object detection and segmentation in deep learning

Colin Decourt – ANITI / NXP Semiconductors

08/02/2023

ANITI Université Fédérale Toulouse Midi-Pyrénées

From classification...



Feature vector

Fully connected

Class scores:

Cat: 0.85

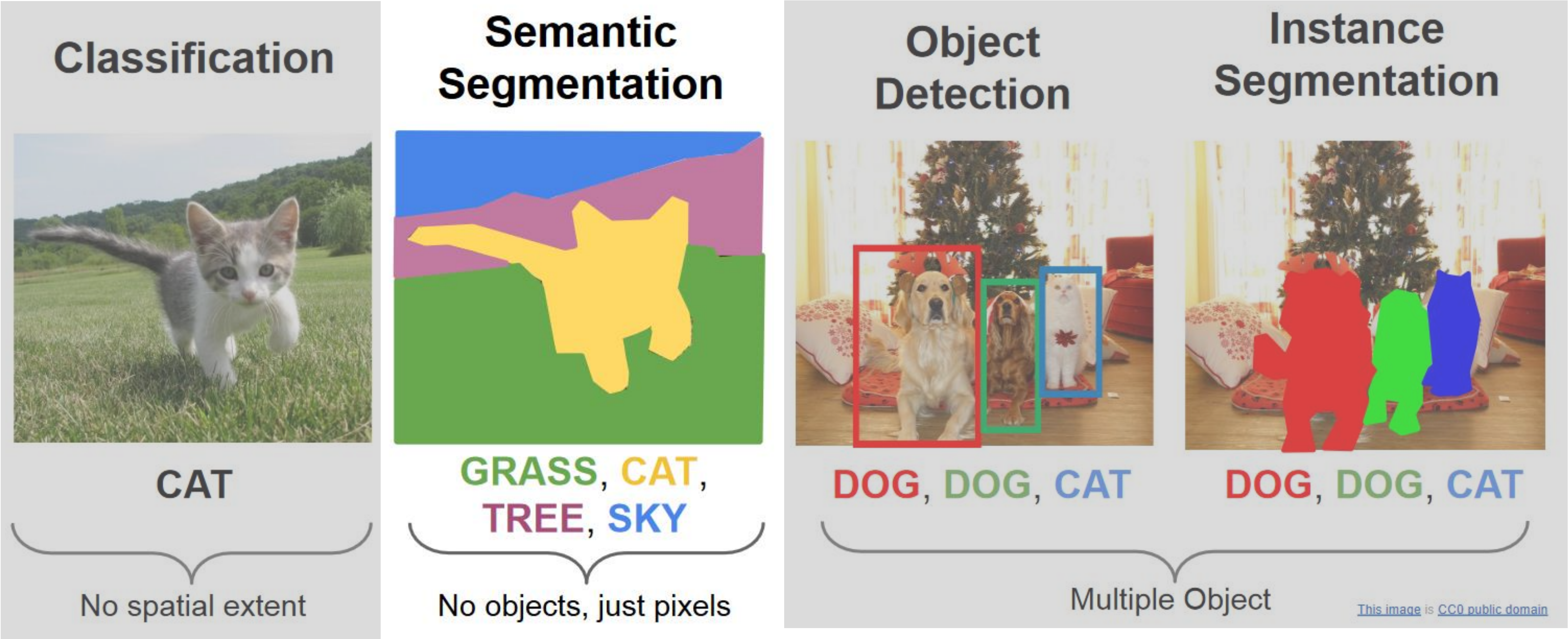
Owl: 0.1

Dog: 0.03

...

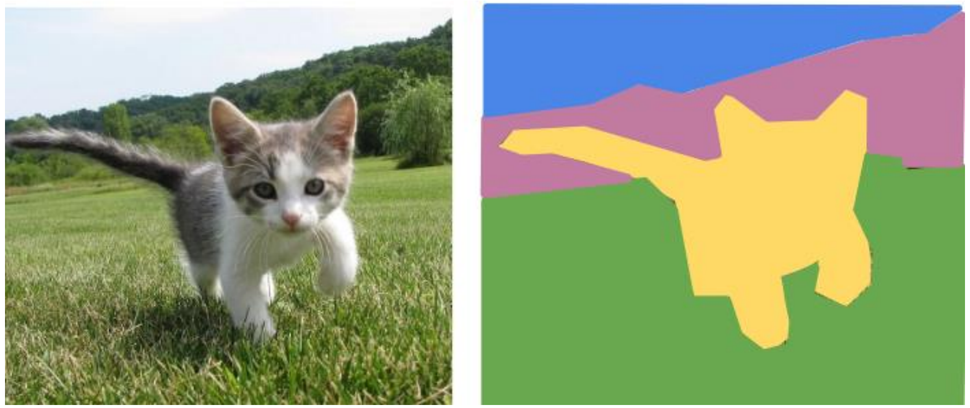
Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. 2017. ImageNet classification with deep convolutional neural networks. *Commun. ACM* 60, 6 (June 2017), 84–90. DOI:https://doi.org/10.1145/3065386

… to object detection and segmentation



Semantic segmentation: what is it?

- Assign each pixel in the image to a category label (grass, cat, tree, sky…)
- Paired training data: for each training image, each pixel is labelled with a *semantic* category
- At test time, classify each pixel of a new image



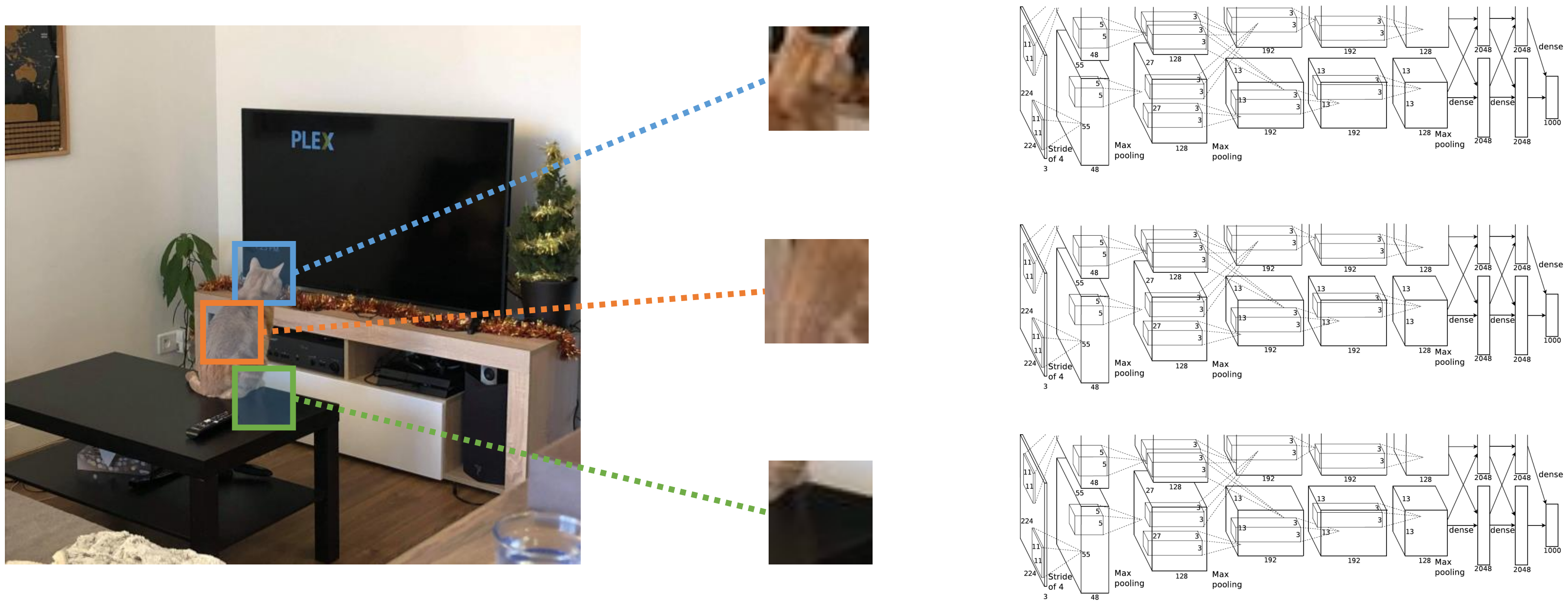
Adapted from http://cs231n.stanford.edu/slides/2022/lecture_9_jiajun.pdf

Semantic segmentation idea: sliding window

- Classify many crops of the image using CNN (ResNet, VGG…)
- How to classify without context?
- How do we include context?

Extract patches

Classify centre pixel with CNN



Cat

Cat

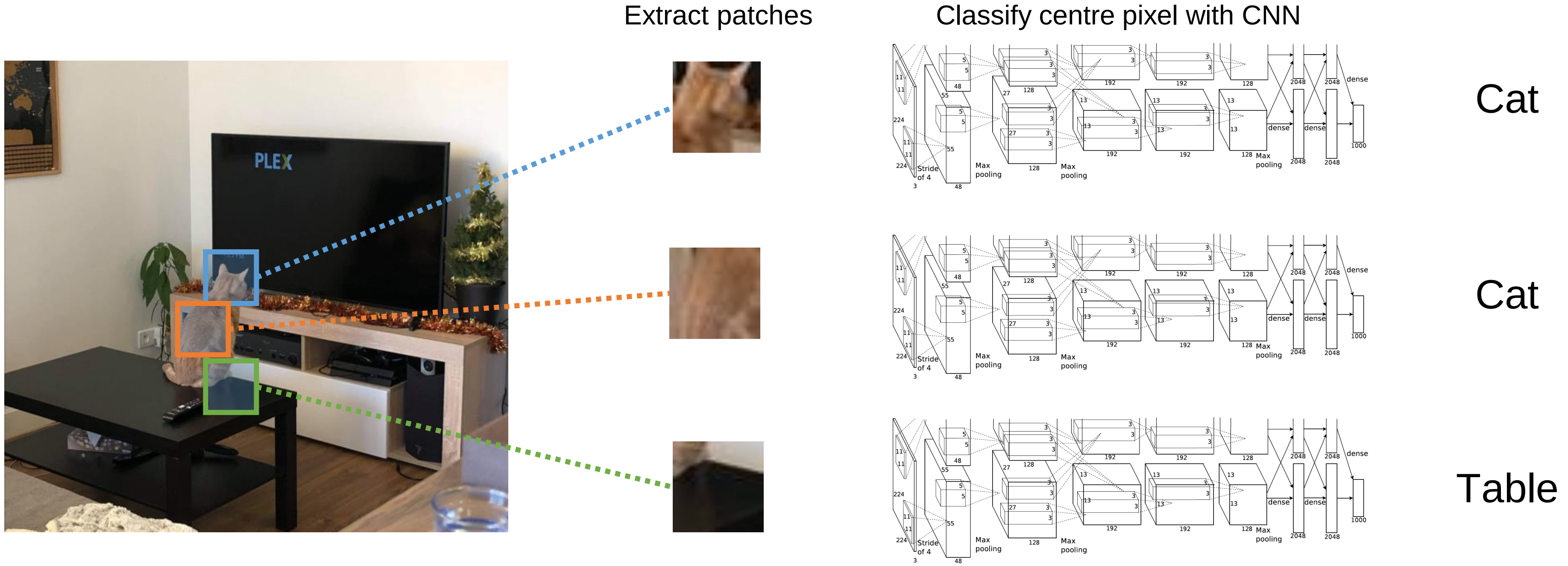
Table

Adapted from http://cs231n.stanford.edu/slides/2022/lecture_9_jiajun.pdf

Semantic segmentation idea: sliding window

Very inefficient! For a 800x600 images ~ **58M boxes**
Not reusing shared features between overlapping patches

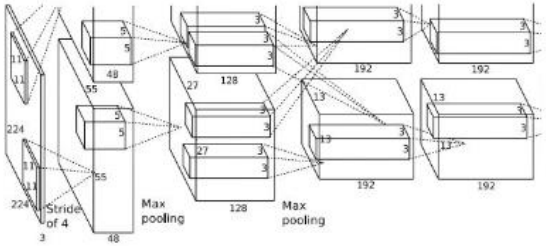
- Classify many crops of the image using CNN (ResNet, VGG…)
- How to classify without context?
- How do we include context?



Adapted from http://cs231n.stanford.edu/slides/2022/lecture_9_jiajun.pdf

Semantic segmentation idea: convolution

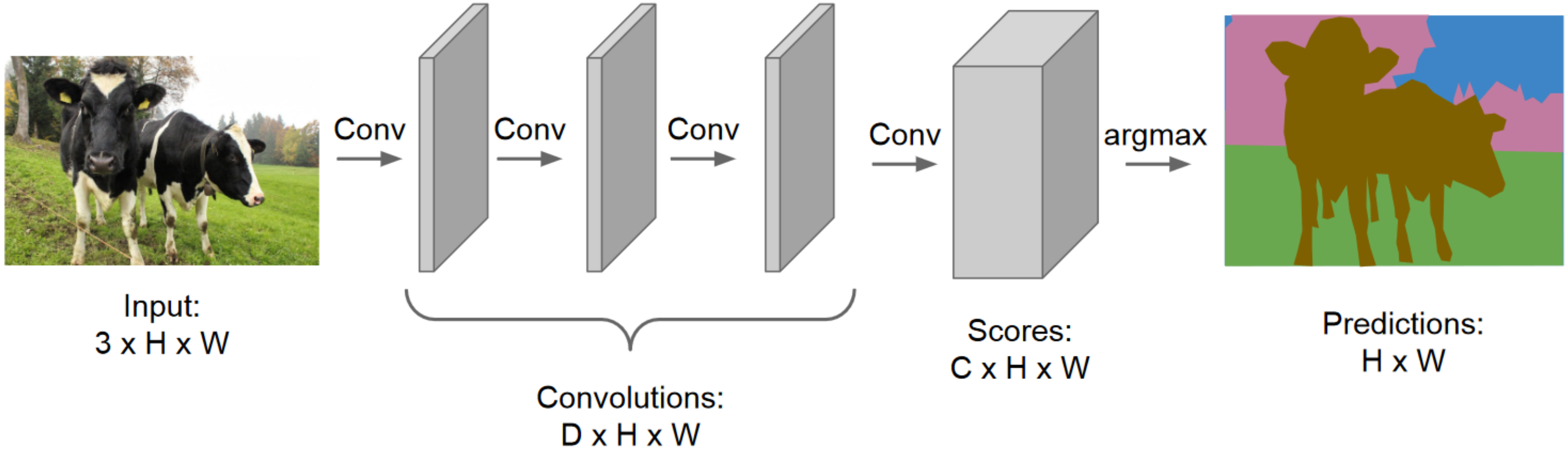
- Idea: encode image features with ConvNets, and perform semantic segmentation on top
- Classification architectures reduce spatial sizes to go deeper, but semantic segmentation requires the output size to be the same as the input size



From http://cs231n.stanford.edu/slides/2022/lecture_9_jiajun.pdf

Semantic segmentation idea: fully convolutional

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

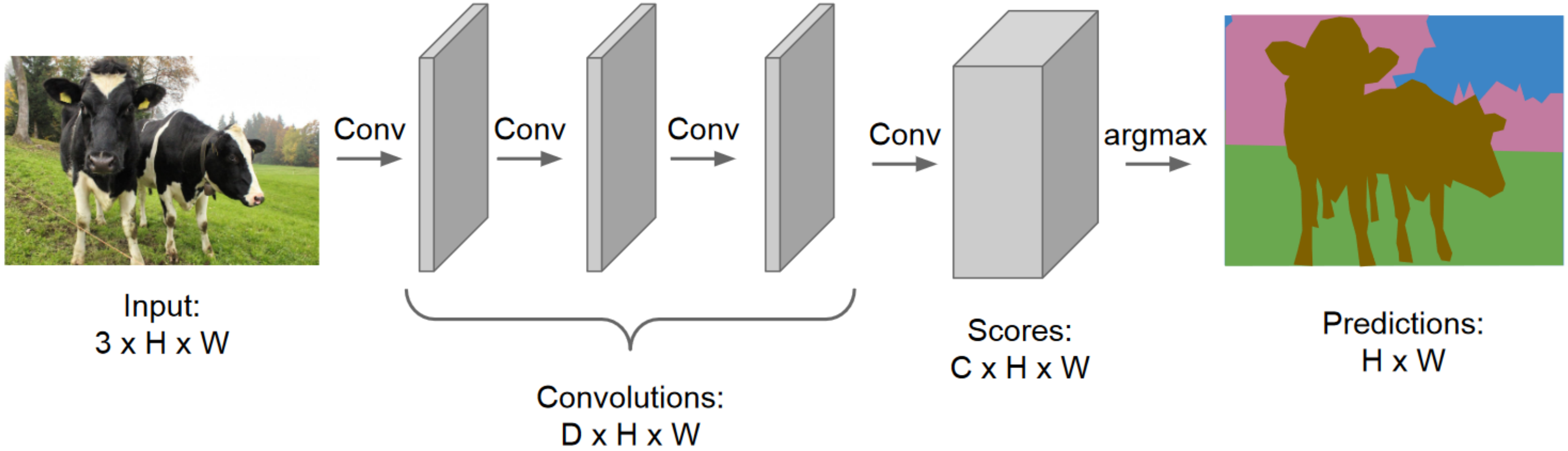


From http://cs231n.stanford.edu/slides/2022/lecture_9_jiajun.pdf

Very expensive as we don’t decrease input size!

Semantic segmentation idea: fully convolutional

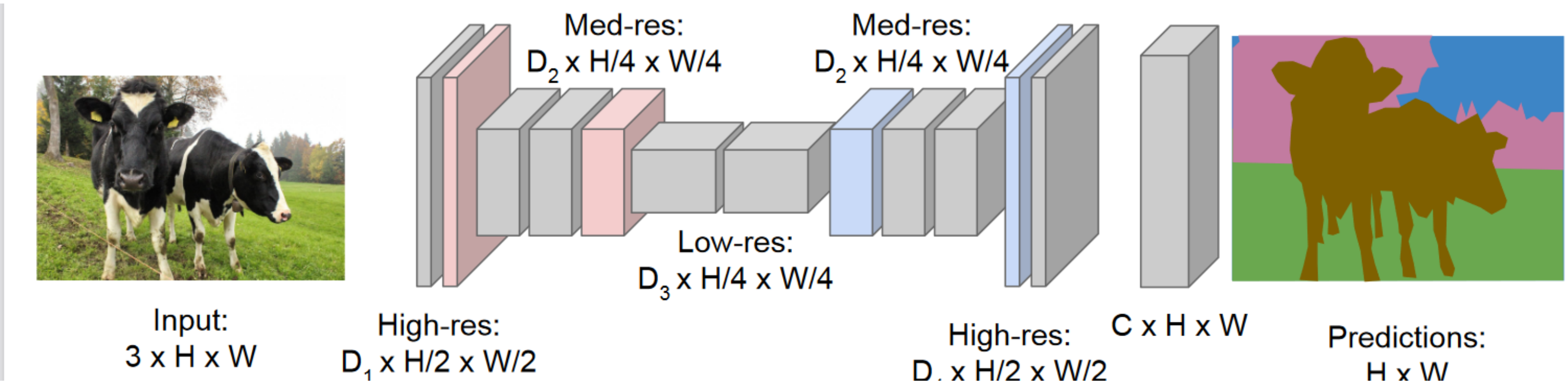
Design a network with only convolutional layers without downsampling operators to make predictions for pixels all at ones



From http://cs231n.stanford.edu/slides/2022/lecture_9_jiajun.pdf

Semantic segmentation idea: fully convolutional

- Design the network as a bunch of convolutional layers, with downsampling and upsampling inside the network!
- Downsampling can be pooling, strided convolution…
- Upsampling can be nearest neighbour interpolation, unpooling, transposed convolution

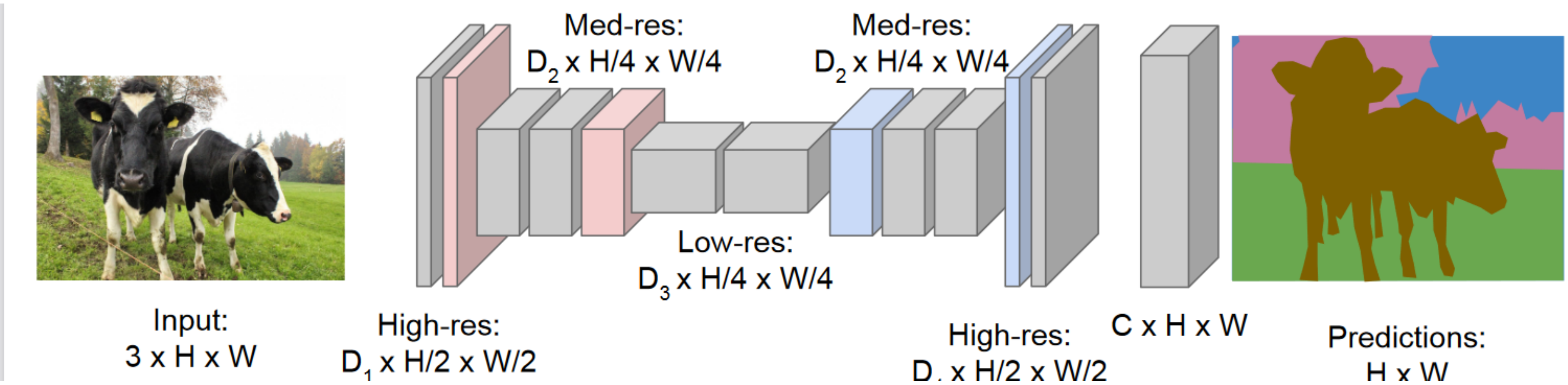


From http://cs231n.stanford.edu/slides/2022/lecture_9_jiajun.pdf

Semantic segmentation idea: fully convolutional

Idea: Learn to project low-resolution features onto pixel level (high-resolution)

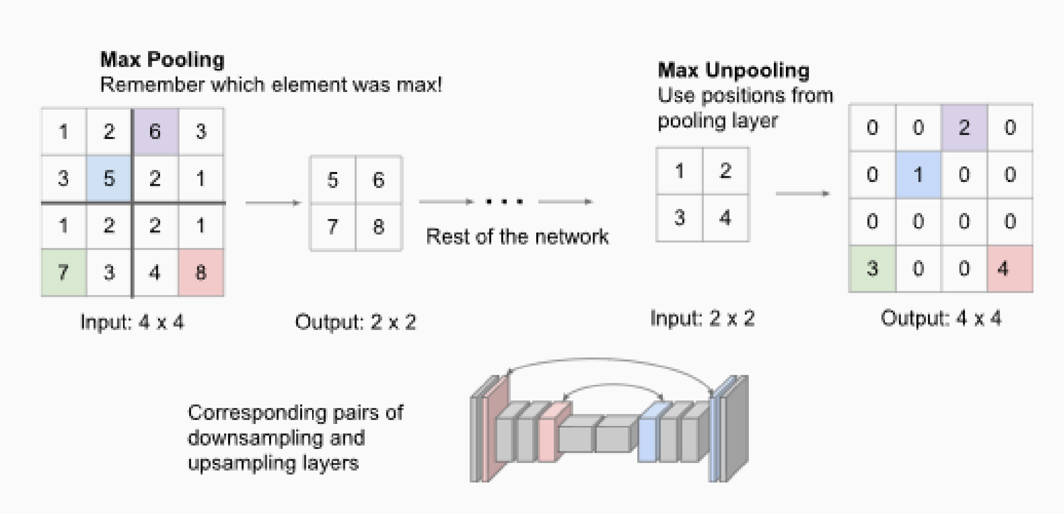
- Design the network as a bunch of convolutional layers, with downsampling and upsampling inside the network!
- Downsampling can be pooling, strided convolution…
- Upsampling can be nearest neighbour interpolation, unpooling, transposed convolution



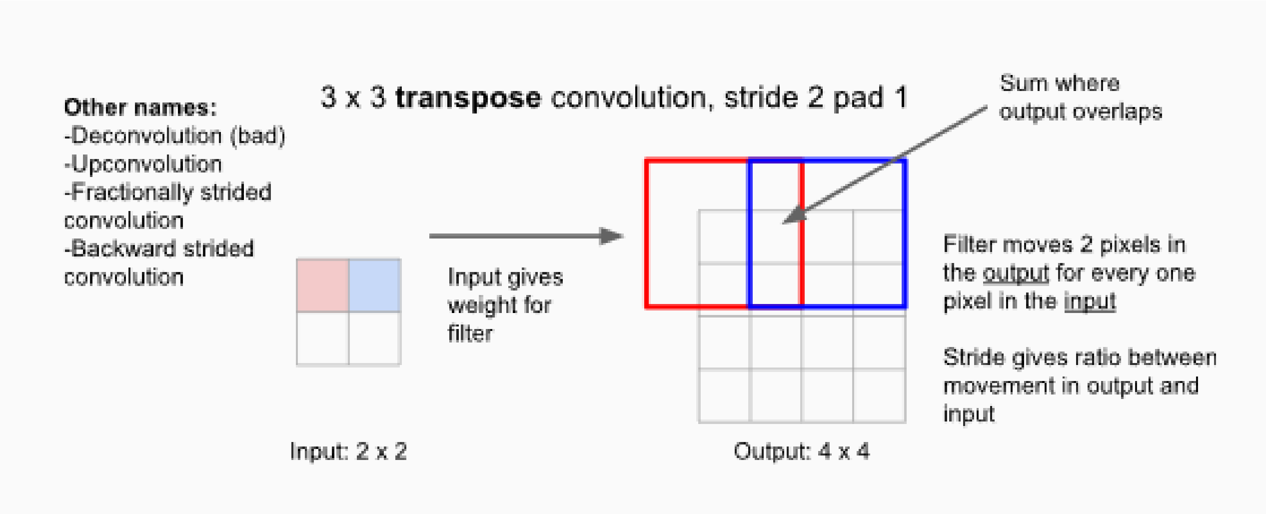
From http://cs231n.stanford.edu/slides/2022/lecture_9_jiajun.pdf

Upsampling low-resolution features

Restore the condensed feature map to the original size of the input image by expanding the feature dimension



1. Unpooling



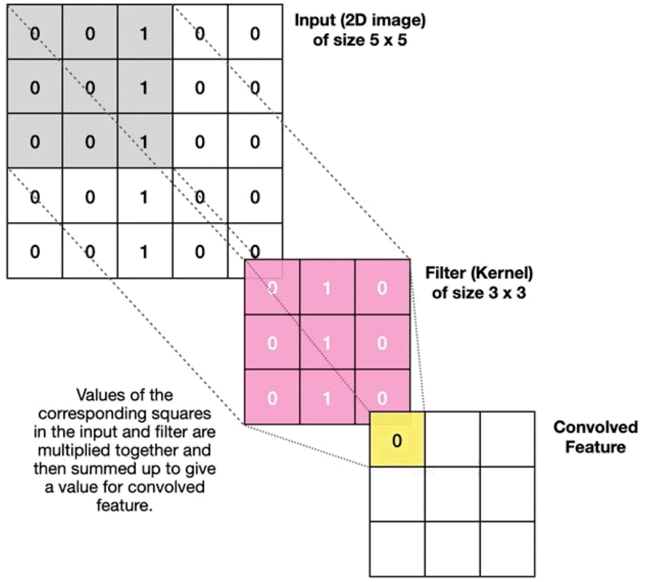
2. Transposed convolution

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

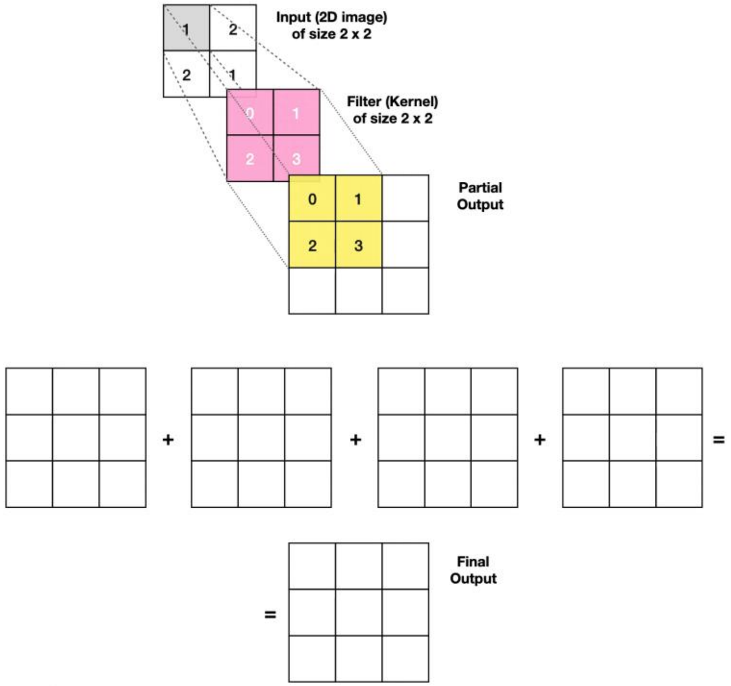
Transposed convolution

Learn to upsample the input feature map to the desired size using **learnable parameters**

Classic convolution

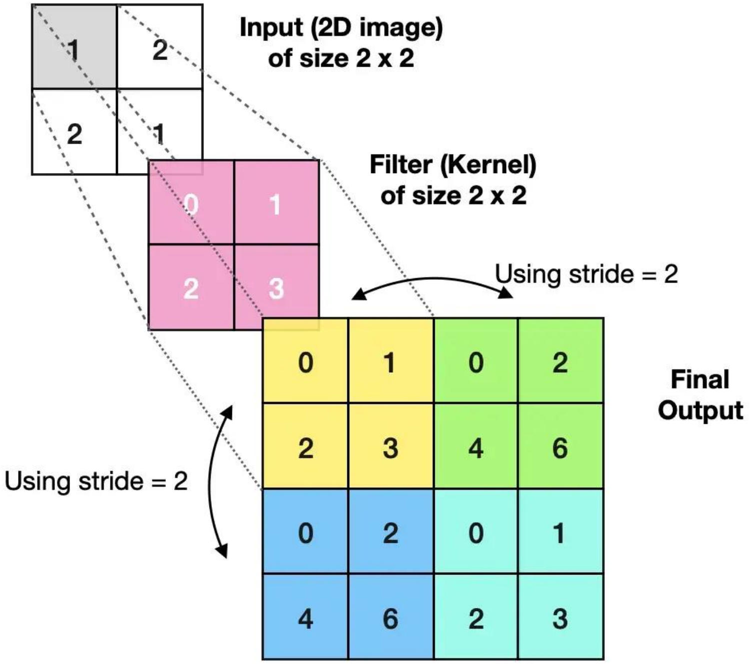


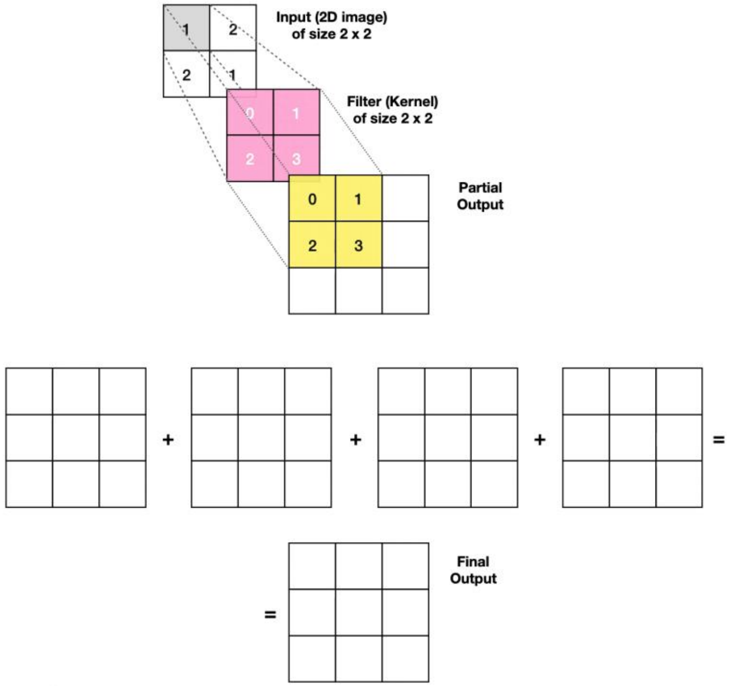
Transposed convolution



Transposed convolution

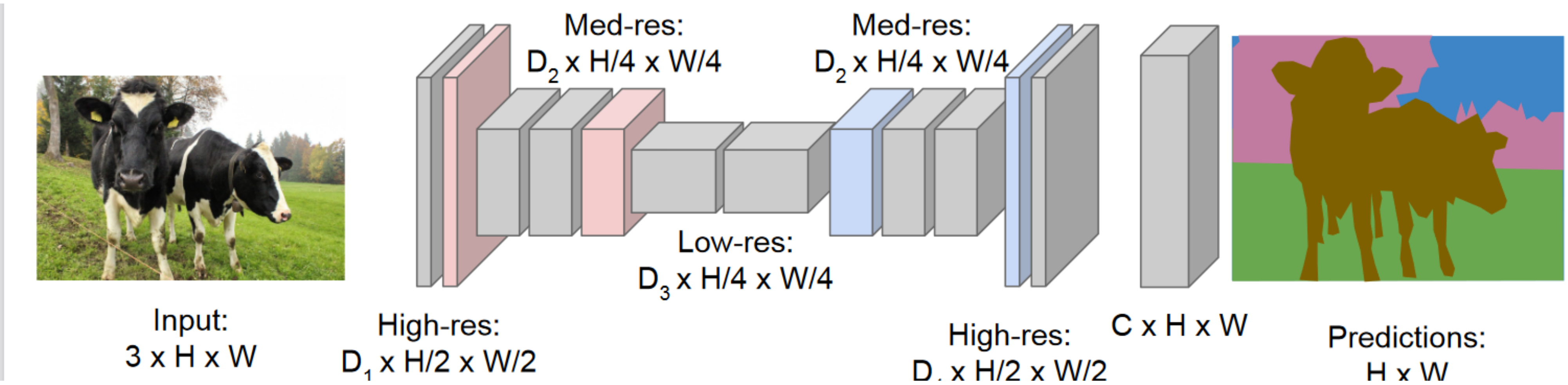
Learn to upsample the input feature map to the desired size using **learnable parameters**





Semantic segmentation: fully convolutional

- Design the network as a bunch of convolutional layers, with downsampling and upsampling inside the network!
- Downsampling can be pooling, strided convolution…
- Upsampling can be nearest neighbour interpolation, unpooling, transposed convolution

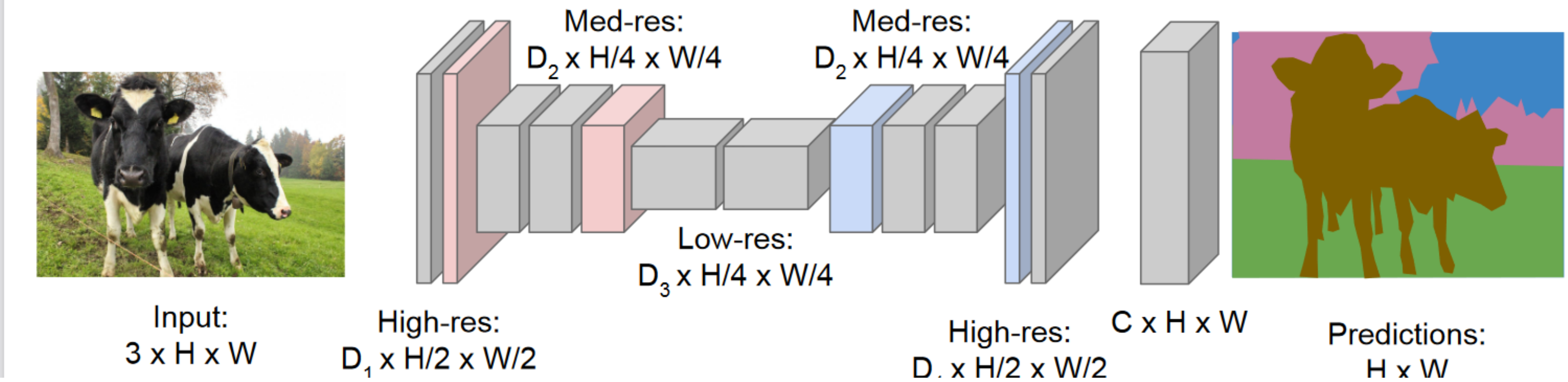


From http://cs231n.stanford.edu/slides/2022/lecture_9_jiajun.pdf

Don't differentiate instances (yet), only care about pixels

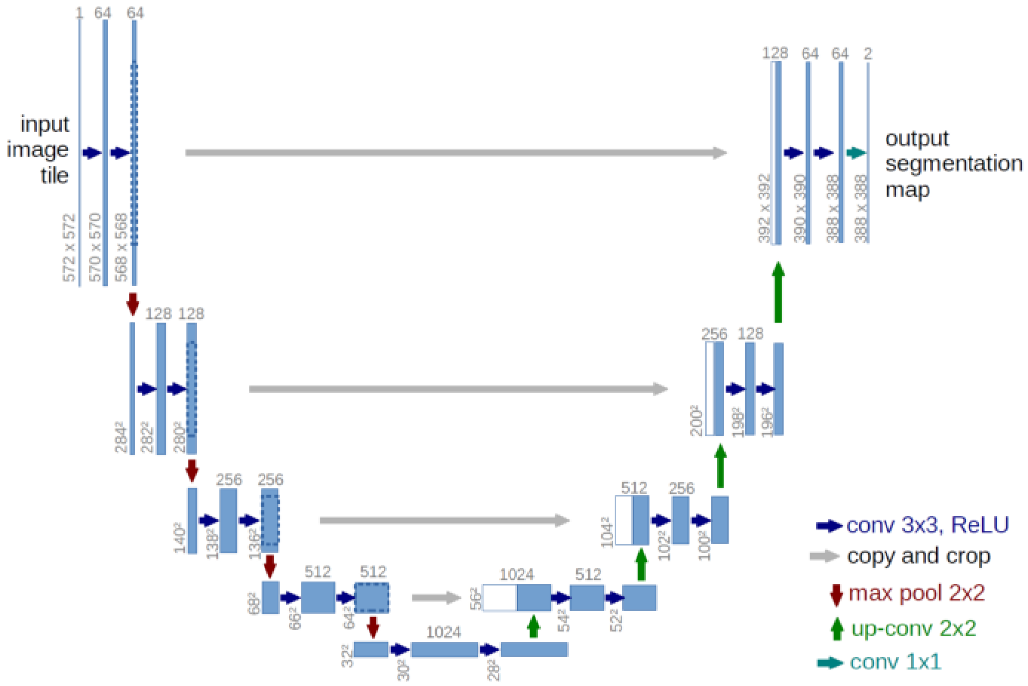
Semantic segmentation: fully convolutional

- Design the network as a bunch of convolutional layers, with downsampling and upsampling inside the network!
- Downsampling can be pooling, strided convolution…
- Upsampling can be nearest neighbour interpolation, unpooling, transposed convolution

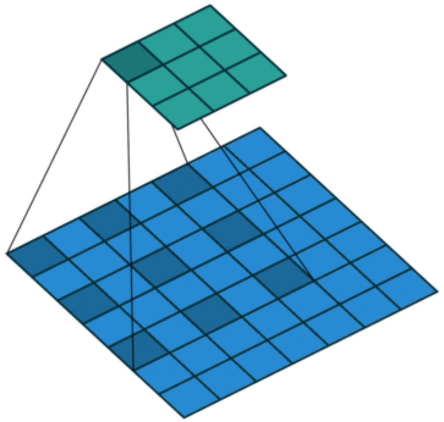


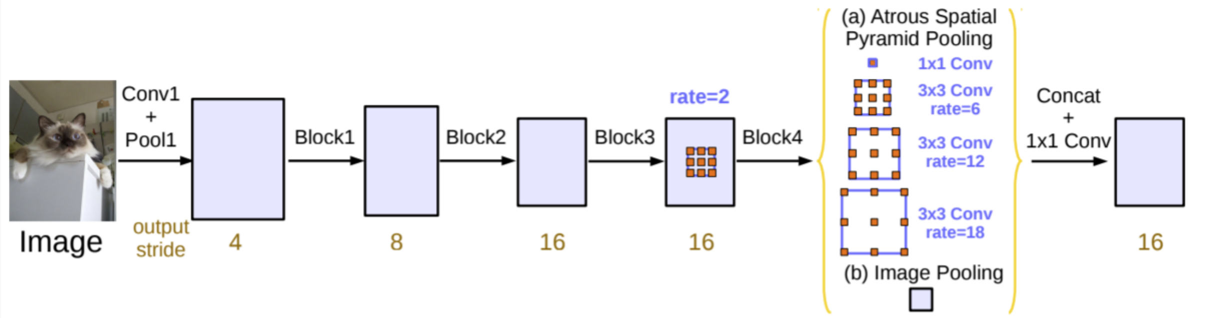
From http://cs231n.stanford.edu/slides/2022/lecture_9_jiajun.pdf

Semantic segmentation: some models…



U-Net

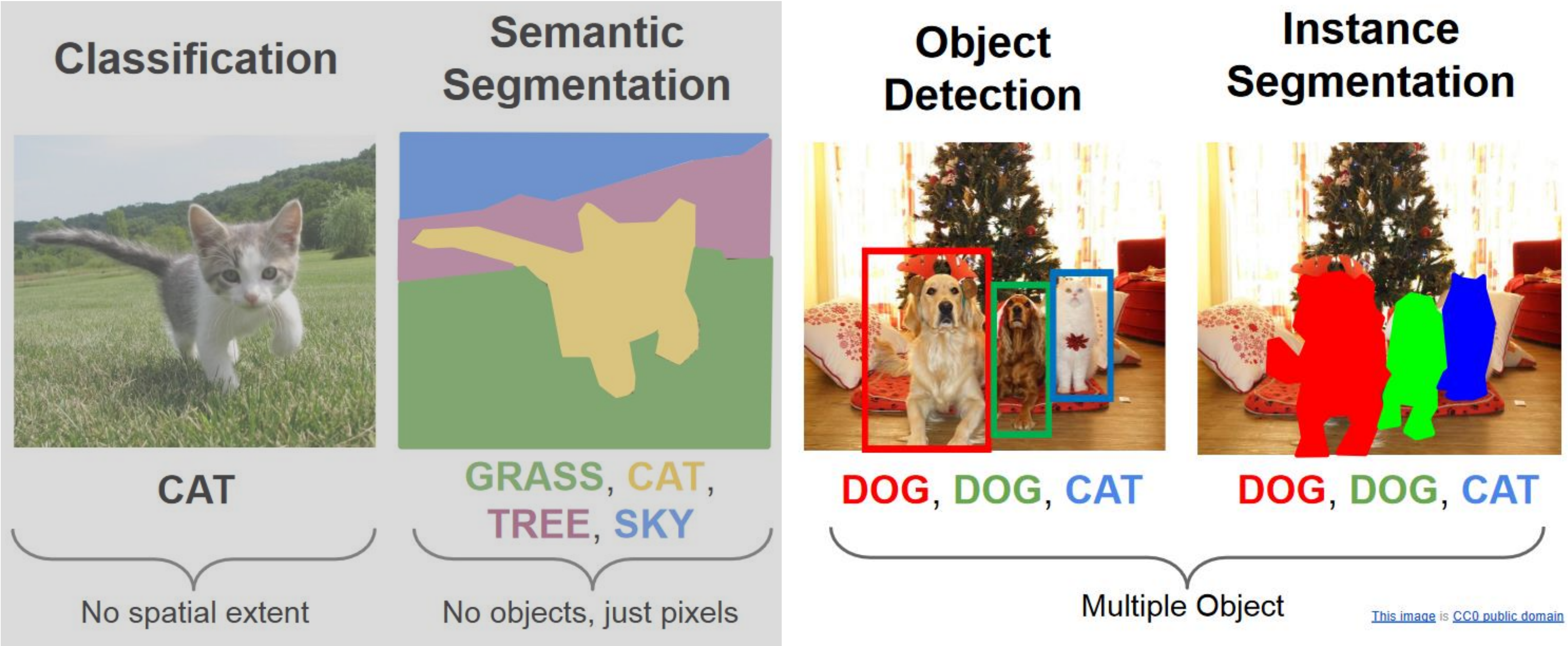




DeepLabv3

1. 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
2. L Chen, G Papandreou, F Schroff, H Adam Rethinking Atrous Convolution for Semantic Image Segmentation arXiv:1706.05587 2017

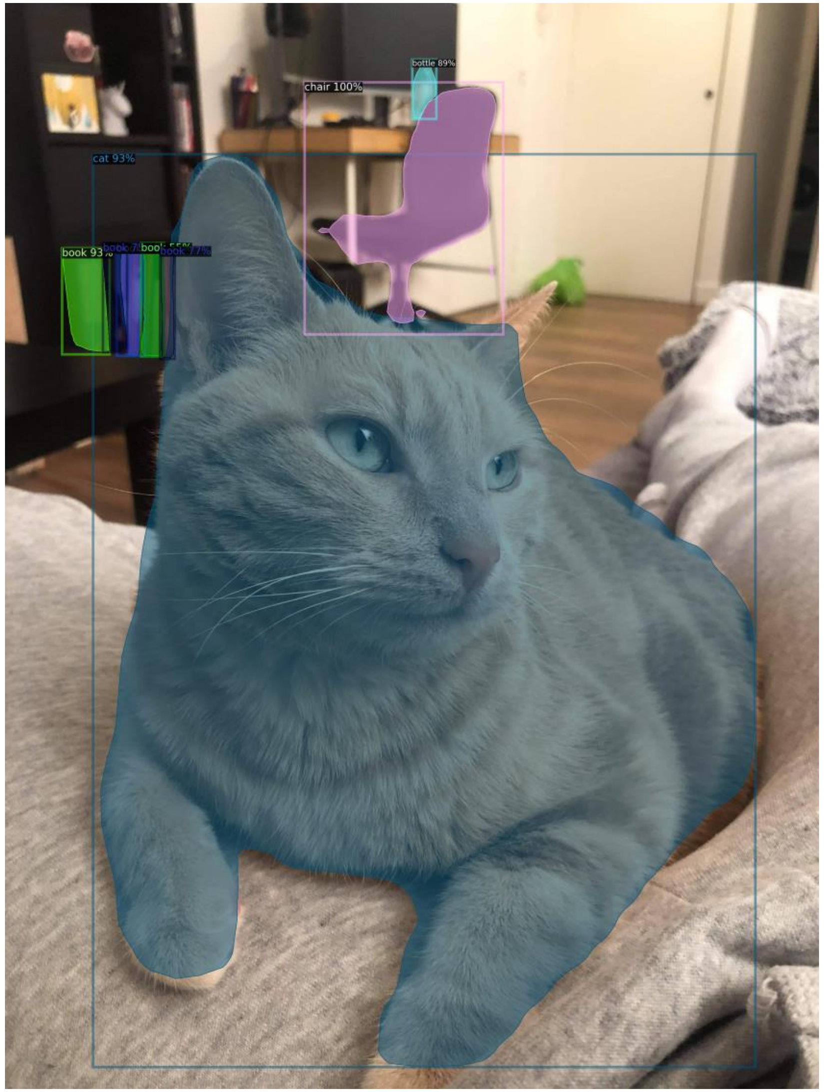
From classification to object detection and segmentation



Object detection: what is it?

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

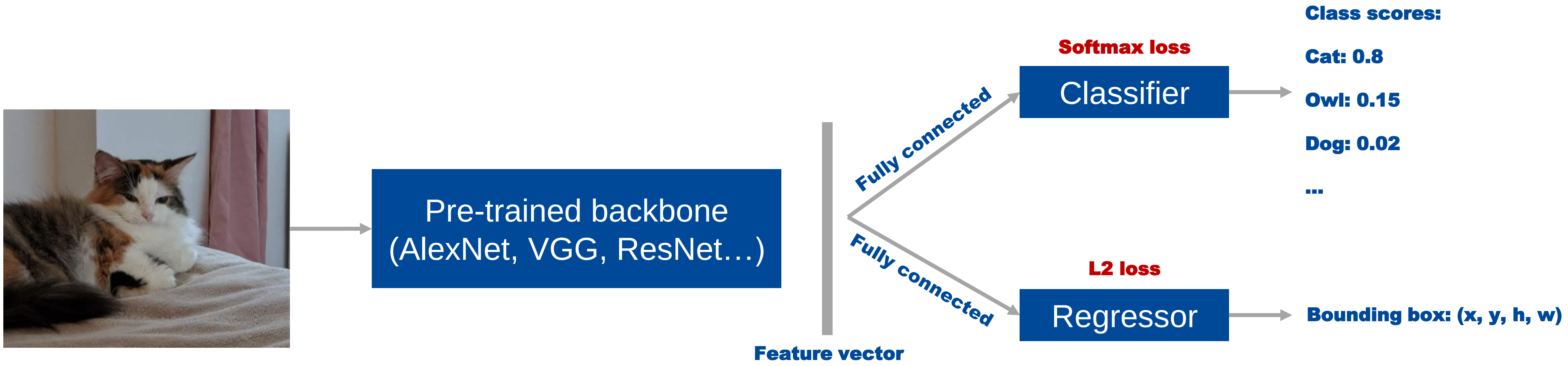
- Semantic segmentation:
  - No objects, just pixels
  - Output a segmentation mask with the same size as the input
- Object detection: for each object output **a label** and a box (or mask)



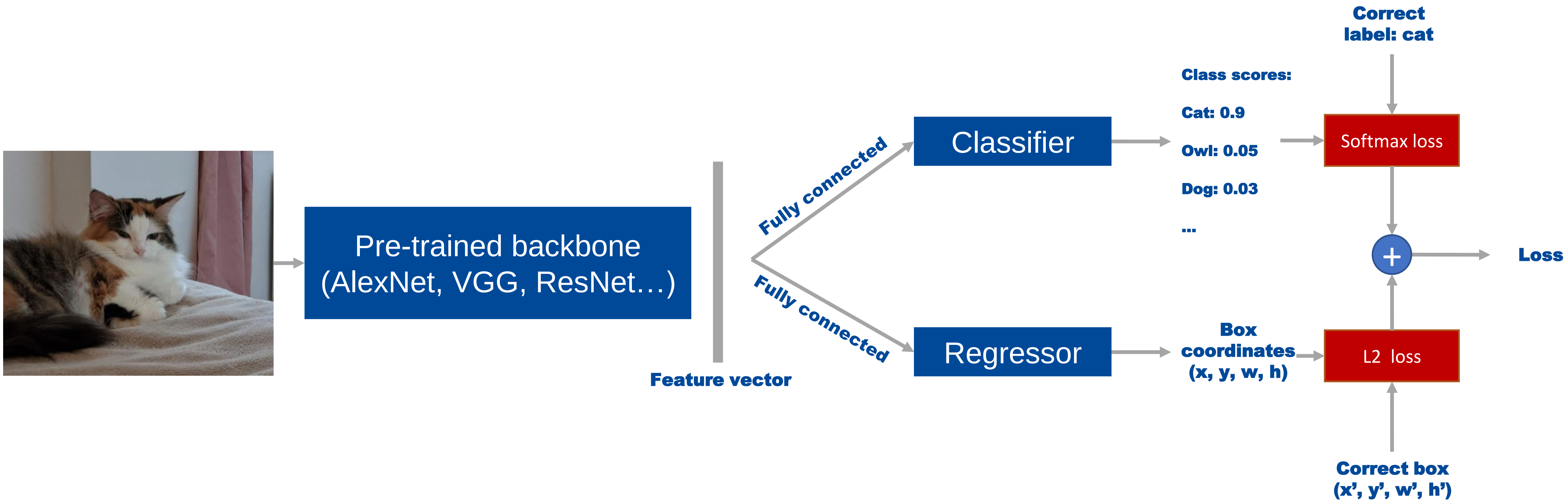
ANITI Université Fédérale Toulouse Midi-Pyrénées

An easy case… Detecting single objects

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

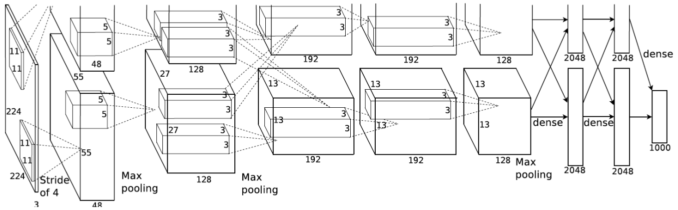


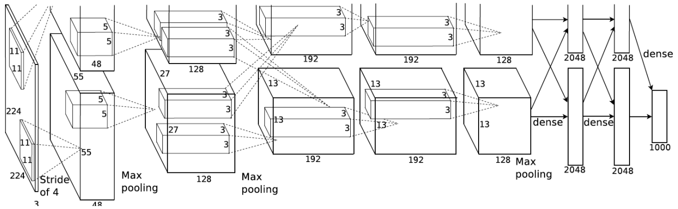
Detecting single objects: multitask loss

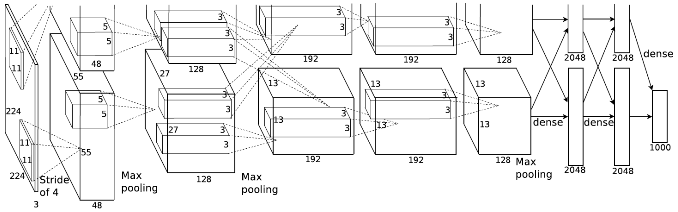


Object detection: challenges

- Multiple outputs: **variable number of objects per image**
- Multiple types of output: category label, bounding boxes (orientation, velocity…)



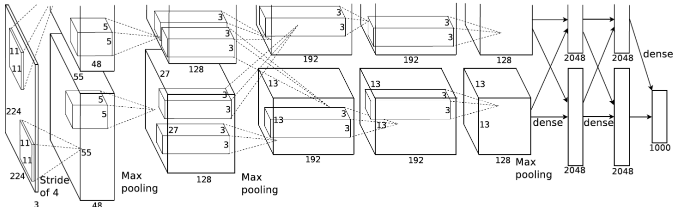




Adapted from http://cs231n.stanford.edu/slides/2022/lecture_9_jiajun.pdf

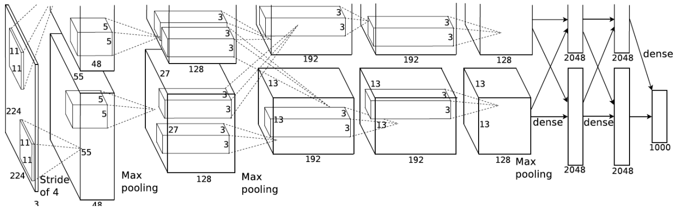
Object detection: challenges

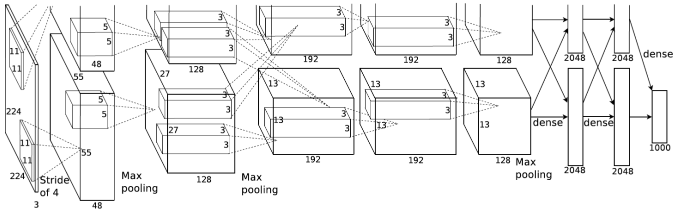
- Multiple outputs: **variable number of objects per image**
- Multiple types of output: category label, bounding boxes (orientation, velocity…)



Cat + (x, y, w, h)

4 numbers to predict

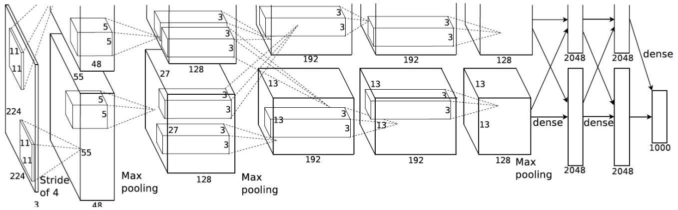




Adapted from http://cs231n.stanford.edu/slides/2022/lecture_9_jiajun.pdf

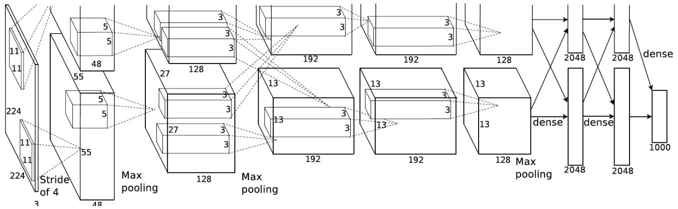
Object detection: challenges

- Multiple outputs: **variable number of objects per image**
- Multiple types of output: category label, bounding boxes (orientation, velocity…)



Cat + (x, y, w, h)

4 numbers to predict

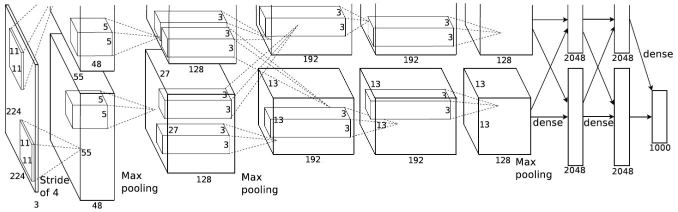


Dog + (x, y, w, h)

Dog + (x, y, w, h)

Cat + (x, y, w, h)

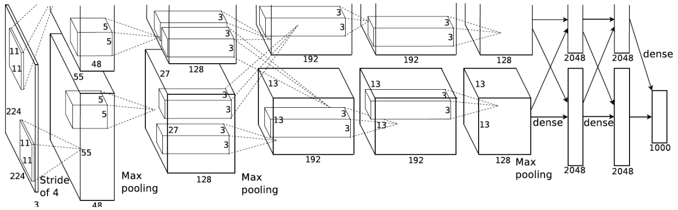
12 numbers to predict



Adapted from http://cs231n.stanford.edu/slides/2022/lecture_9_jiajun.pdf

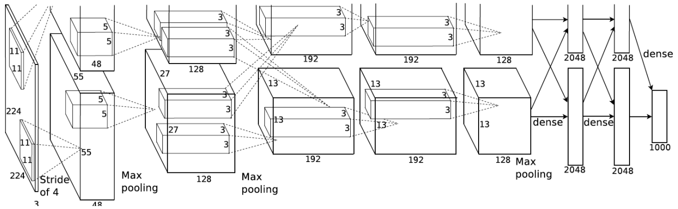
Object detection: challenges

- Multiple outputs: **variable number of objects per image**
- Multiple types of output: category label, bounding boxes (orientation, velocity…)



Cat + (x, y, w, h)

4 numbers to predict

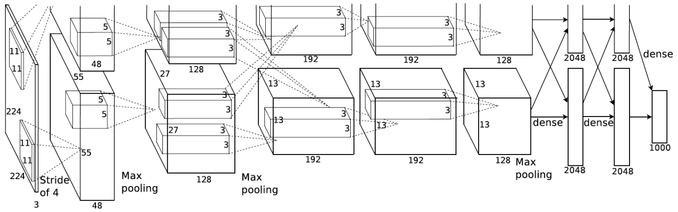


Dog + (x, y, w, h)

Dog + (x, y, w, h)

Cat + (x, y, w, h)

12 numbers to predict



Duck + (x, y, w, h)

Duck + (x, y, w, h)

…

Many numbers to predict!!

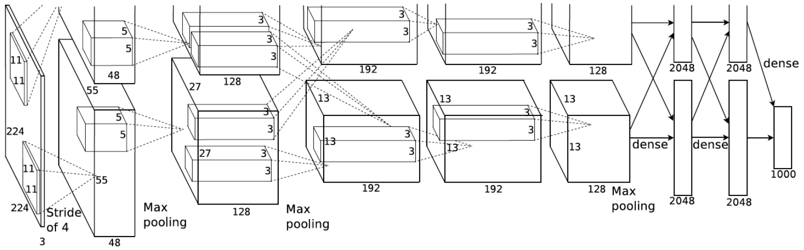
Adapted from http://cs231n.stanford.edu/slides/2022/lecture_9_jiajun.pdf

Remember... The sliding window idea

Very inefficient! For a 800x600 images ~ **58M boxes**
Need to apply CNN to a huge number of locations!

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

Extract patches



Cat? Yes
Table? No
Background? No

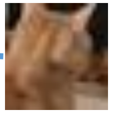
Adapted from http://cs231n.stanford.edu/slides/2022/lecture_9_jiajun.pdf

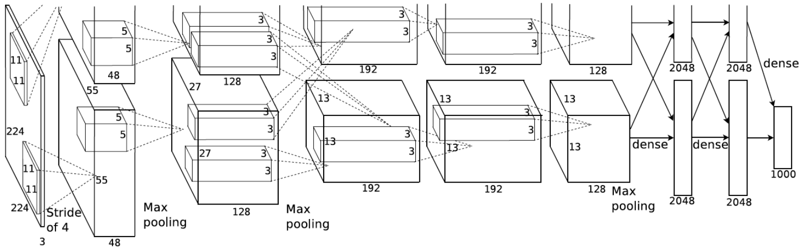
Remember… The sliding window idea

Very inefficient! For a 800x600 images ~ **58M boxes**
Need to apply CNN to a huge number of locations!

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

Extract patches





Cat? Yes
Table? No
Background? No

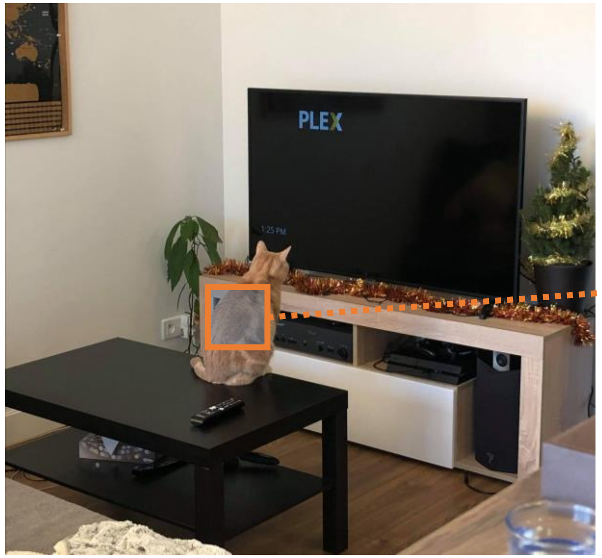
Adapted from http://cs231n.stanford.edu/slides/2022/lecture_9_jiajun.pdf

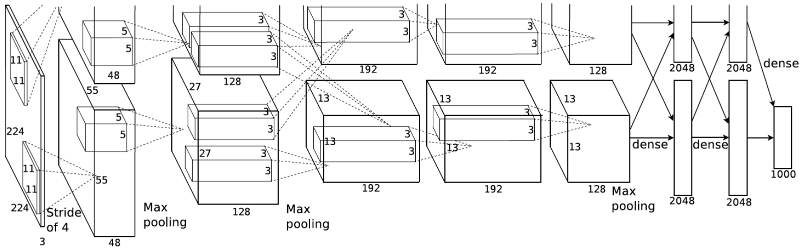
Very inefficient! For a 800x600 images ~ **58M boxes**
Need to apply CNN to a huge number of locations!

Remember… The sliding window idea

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

Extract patches





Cat? Yes
Table? No
Background? No

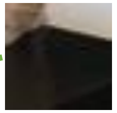
Adapted from http://cs231n.stanford.edu/slides/2022/lecture_9_jiajun.pdf

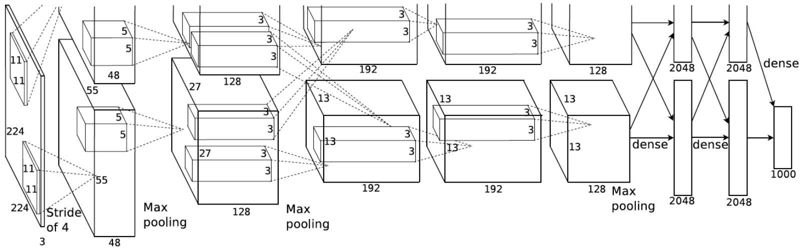
Remember… The sliding window idea

Very inefficient! For a 800x600 images ~ **58M boxes**
Need to apply CNN to a huge number of locations!

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

Extract patches





Cat? No
Table? Yes
Background? No

Adapted from http://cs231n.stanford.edu/slides/2022/lecture_9_jiajun.pdf

Object detection: region proposals

<u>Selective search algorithm :</u>

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.

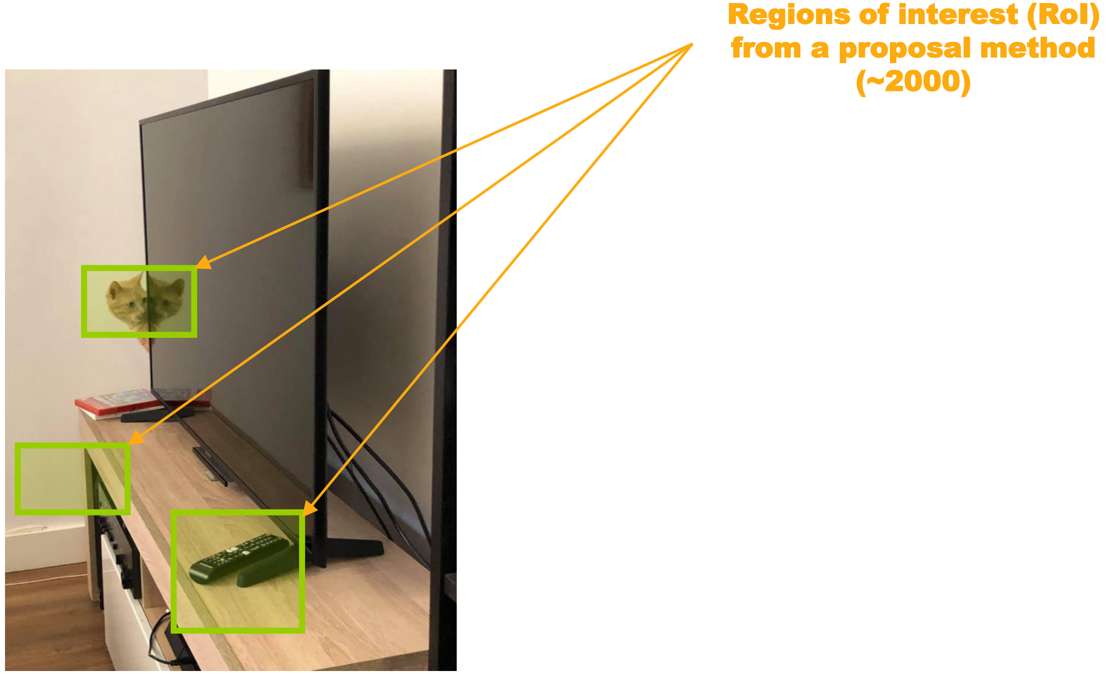
3. Use the generated regions to produce the final candidate region proposals

Object detection: R-CNN (Girshick et al., 2013)

- Classify EACH proposed region (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) (a "*correction*")
  - 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 $b_h = p_h e^{t_h}$

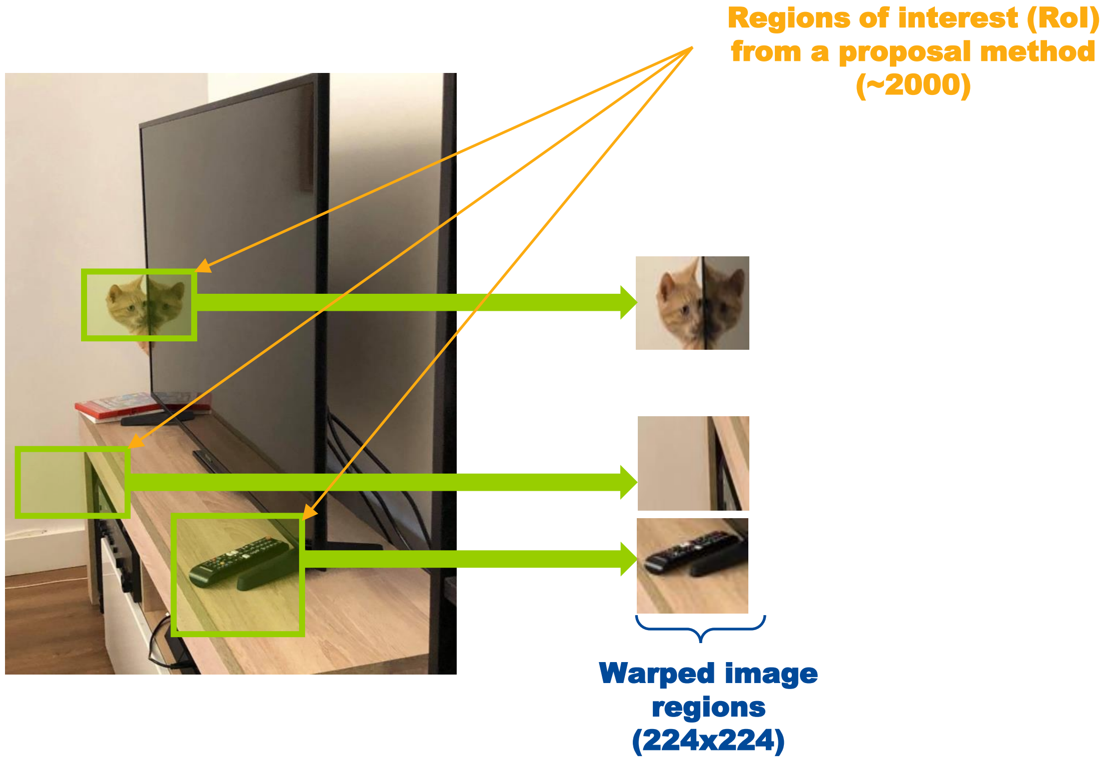
Object detection: R-CNN (Girshick et al., 2013)

- Classify EACH proposed region (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) (a "*correction*")
  - 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 $b_h = p_h e^{t_h}$



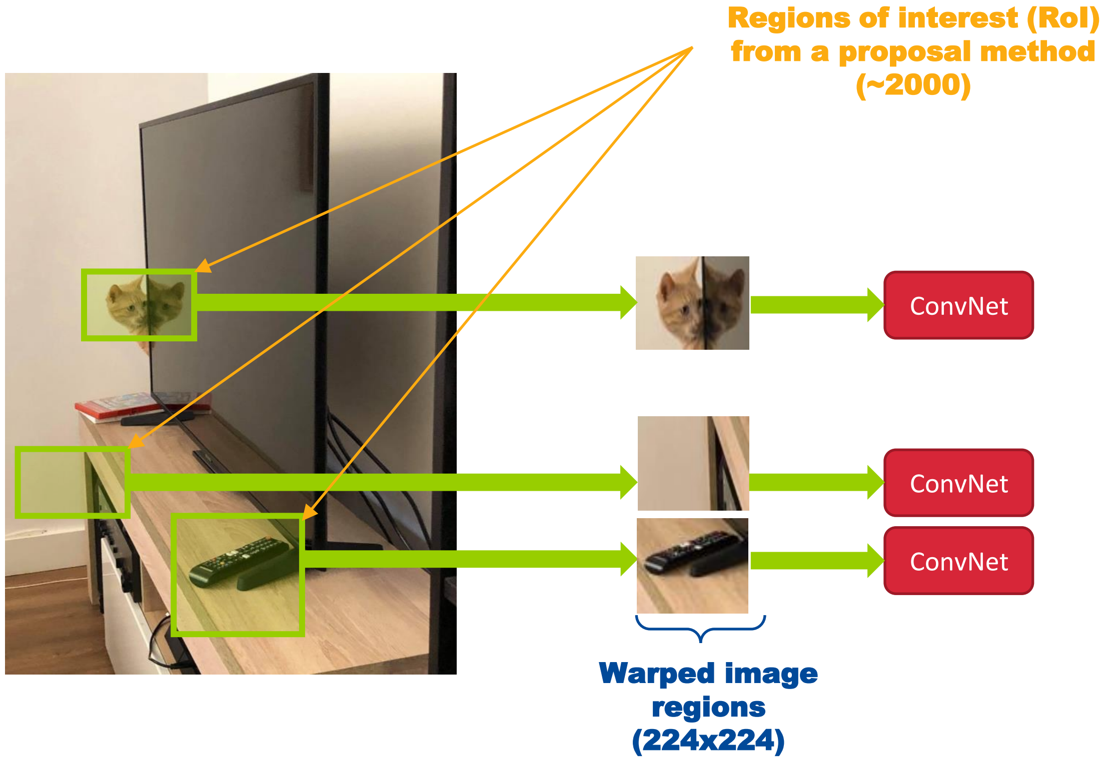
Object detection: R-CNN (Girshick et al., 2013)

- Classify EACH proposed region (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) (a *“correction”*)
  - 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 $b_h = p_h e^{t_h}$



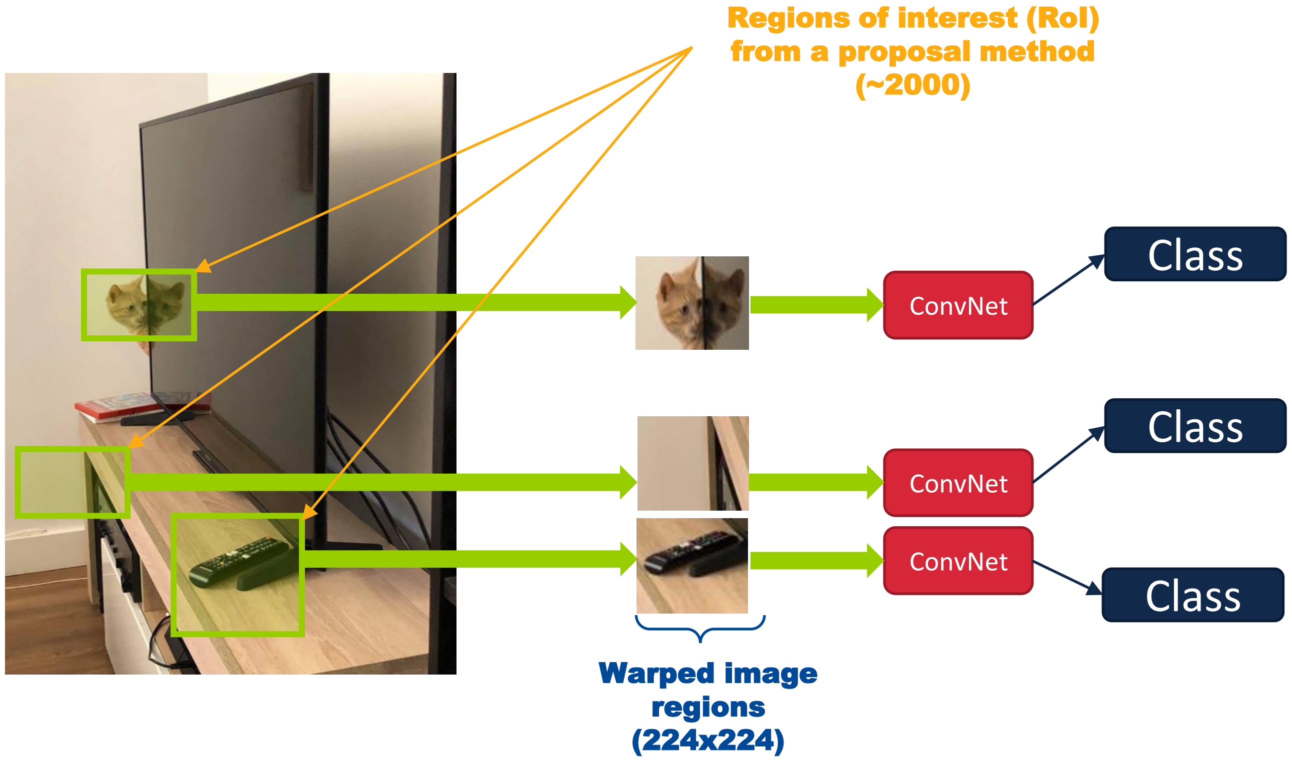
Object detection: R-CNN (Girshick et al., 2013)

- Classify EACH proposed region (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) (a "*correction*")
  - 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 $b_h = p_h e^{t_h}$



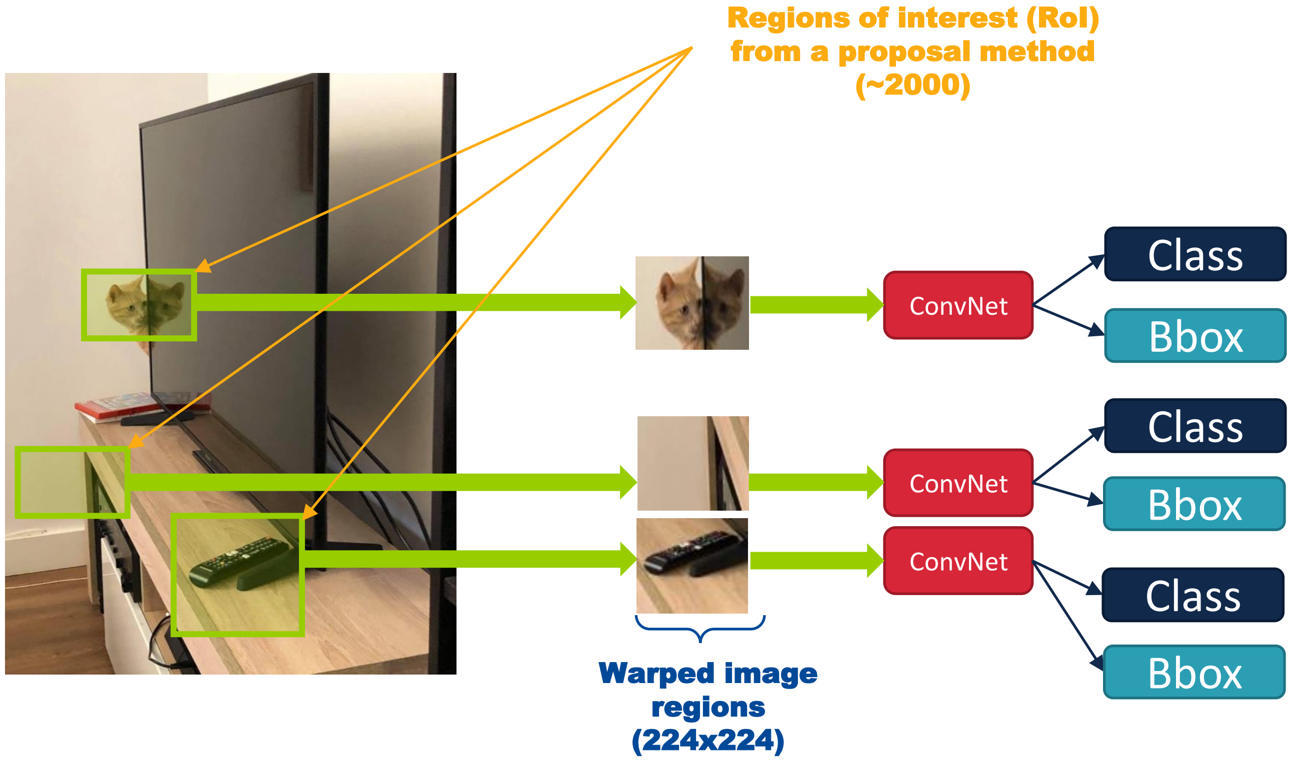
Object detection: R-CNN (Girshick et al., 2013)

- Classify EACH proposed region (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) (a "*correction*")
  - 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 $b_h = p_h e^{t_h}$



Object detection: R-CNN (Girshick et al., 2013)

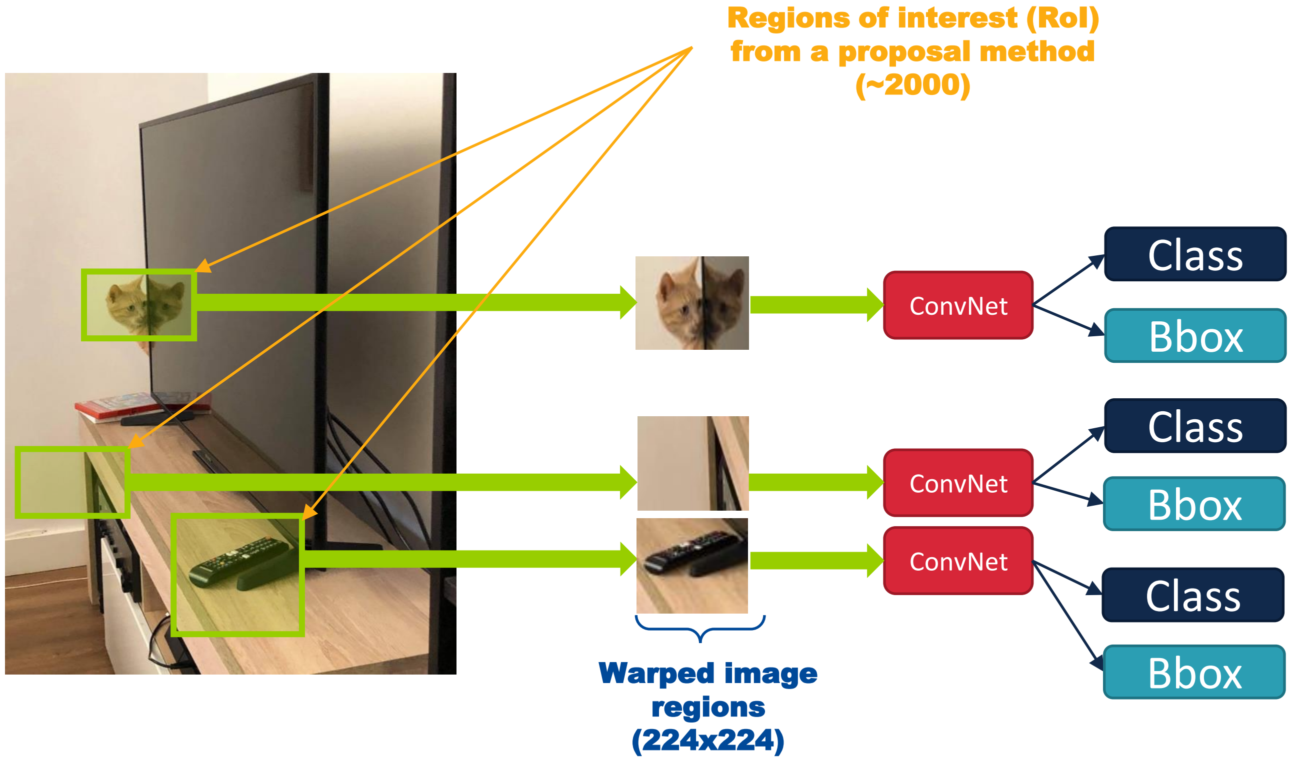
- Classify EACH proposed region (SVM)
- Bounding box regression:
  - Predict "transform" to correct the proposed RoI
  - 4 numbers: (t_x, t_y, t_h, t_w)
- Final output:
  - Proposal: (p_x, p_y, p_h, p_w)
  - Transform: (t_x, t_y, t_h, t_w) (a "*correction*")
  - Output box: (b_x, b_y, b_h, b_w)
    -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



Problem: very slow! Need 2000 independent forward pass for each image

Object detection: R-CNN (Girshick et al., 2013)

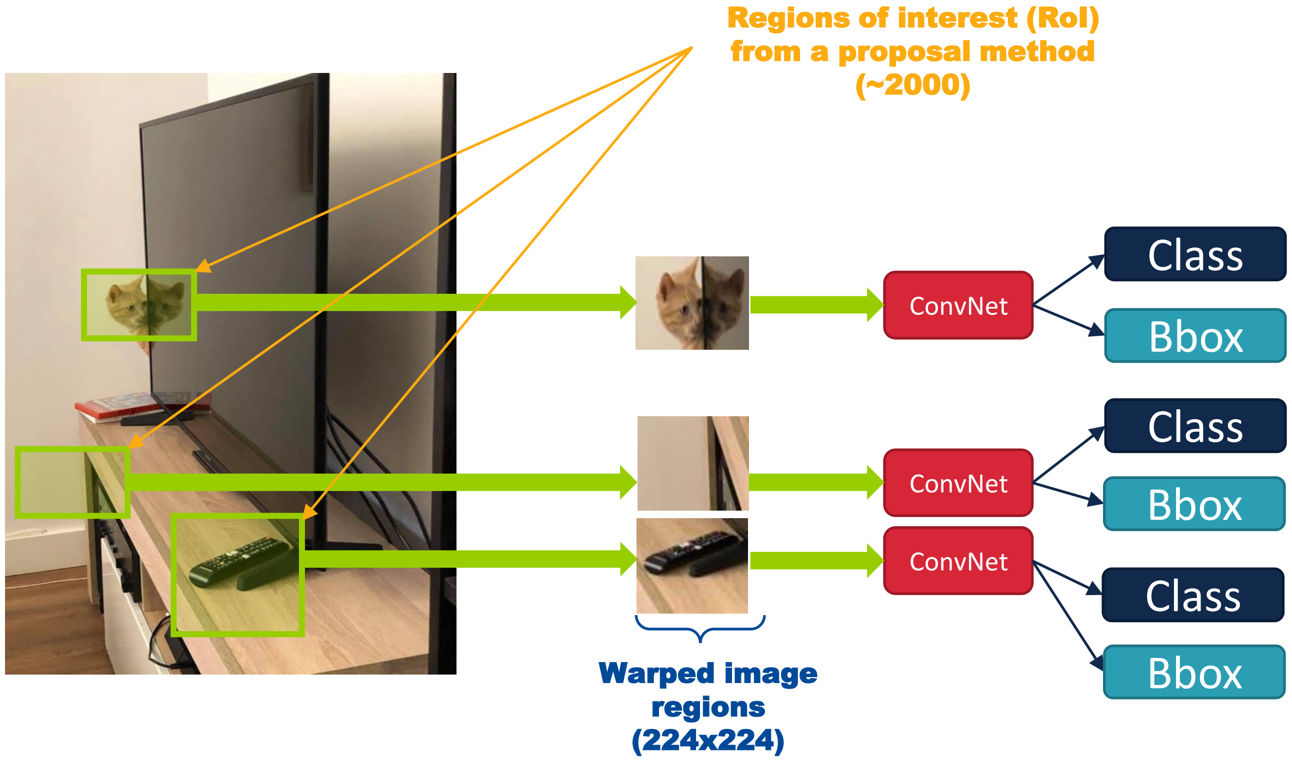
- Classify EACH proposed region (SVM)
- Bounding box regression:
  - Predict “transform” to correct the proposed RoI
  - 4 numbers: (t_x, t_y, t_h, t_w)
- Final output:
  - Proposal: (p_x, p_y, p_h, p_w)
  - Transform: (t_x, t_y, t_h, t_w) (a “*correction*”)
  - Output box: (b_x, b_y, b_h, b_w)
    -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



Object detection: "*Slow*" R-CNN (Girshick et al., 2013)

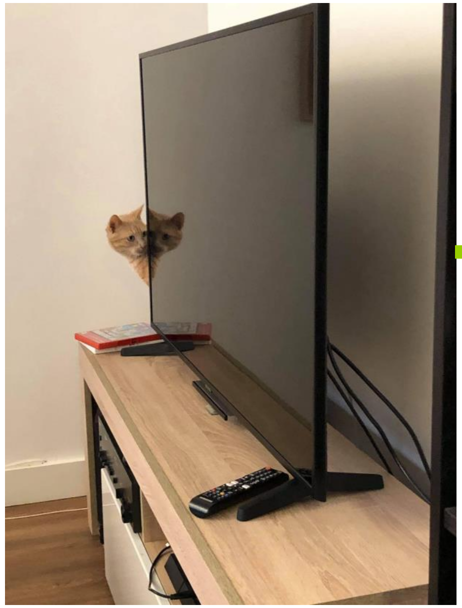
Problem: very slow! Need 2000 independent forward pass for each image

- Classify EACH proposed region (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) (a "*correction*")
  - 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 $b_h = p_h e^{t_h}$



Object detection: Fast-RCNN (Girshick et al., 2015)

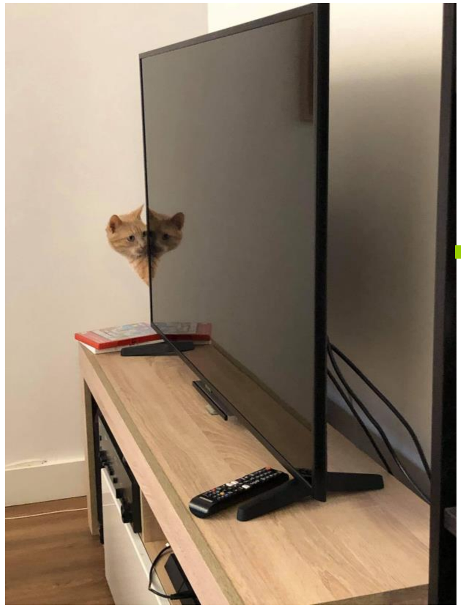
Object detection: Fast-RCNN (Girshick et al., 2015)

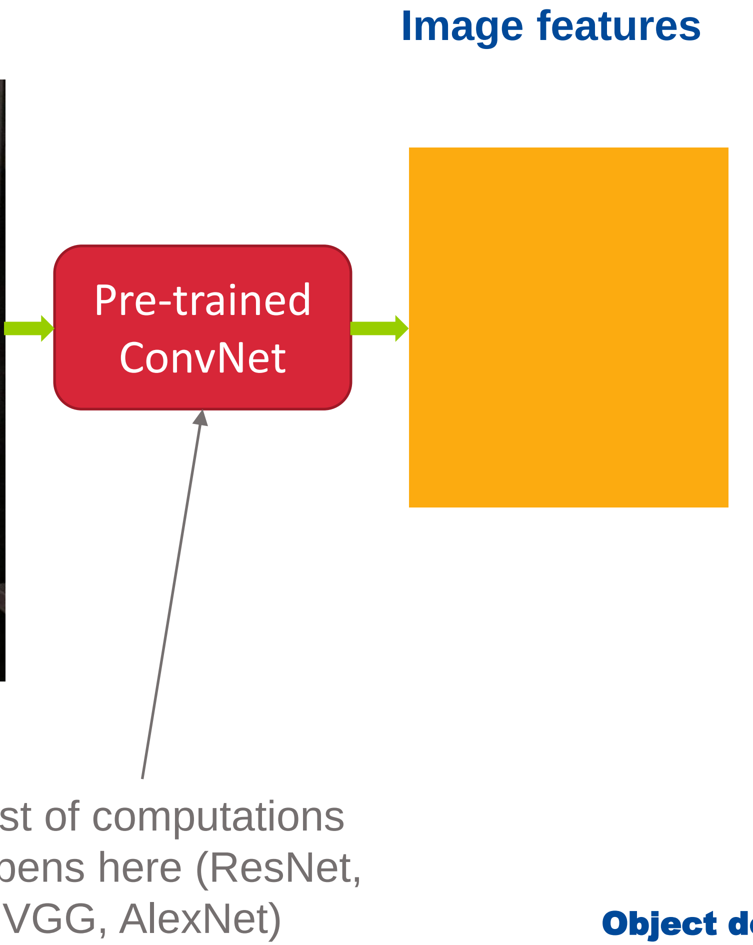


Pre-trained ConvNet

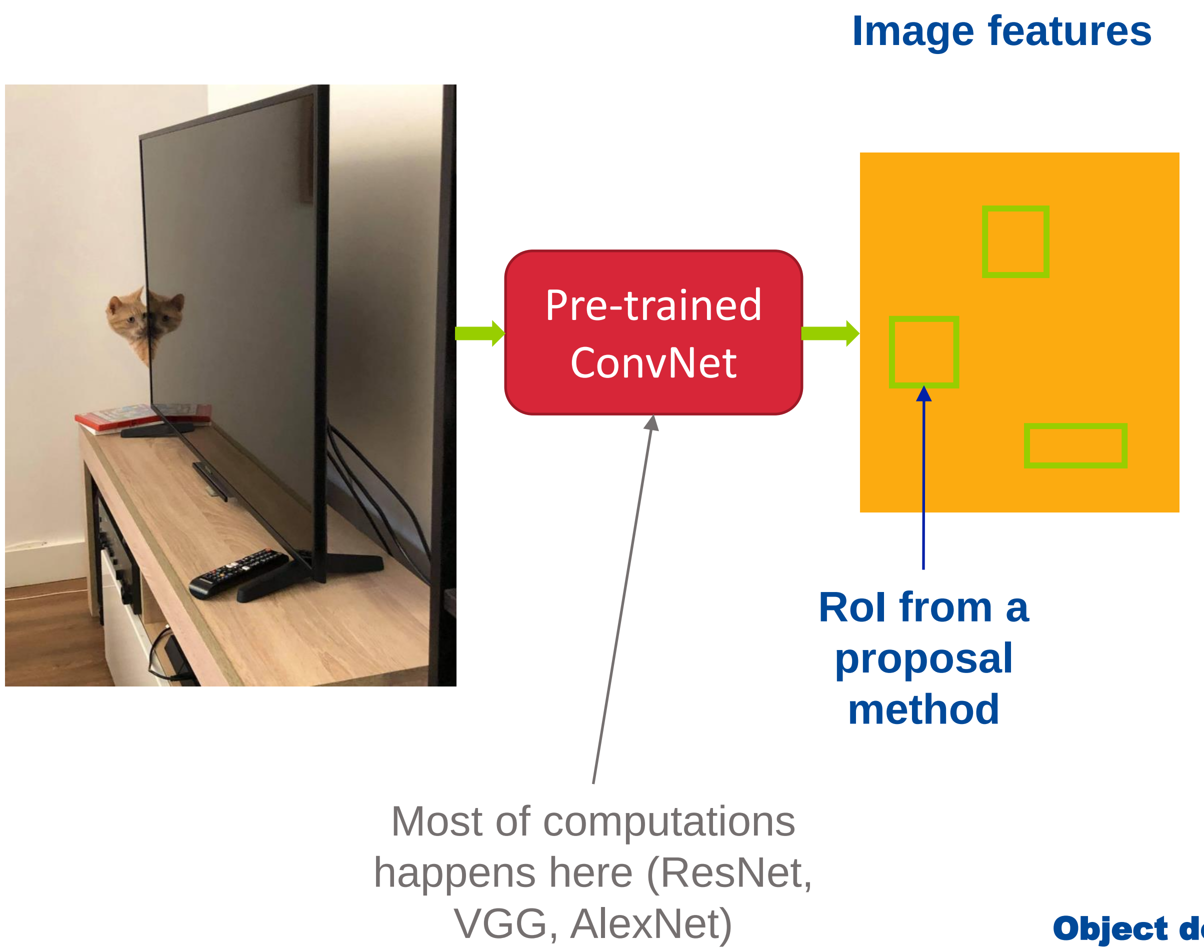
Most of computations happens here (ResNet, VGG, AlexNet)

Object detection: Fast-RCNN (Girshick et al., 2015)

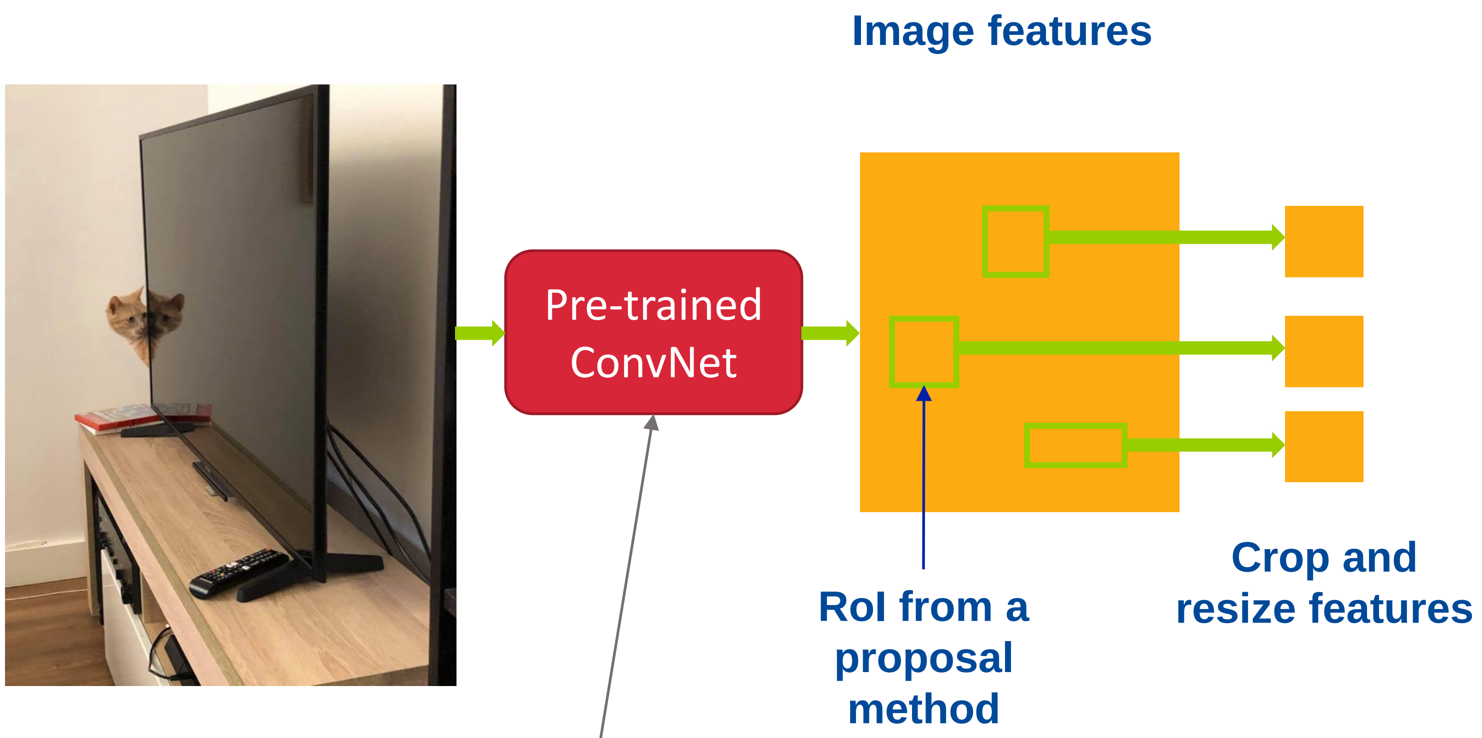




Object detection: Fast-RCNN (Girshick et al., 2015)

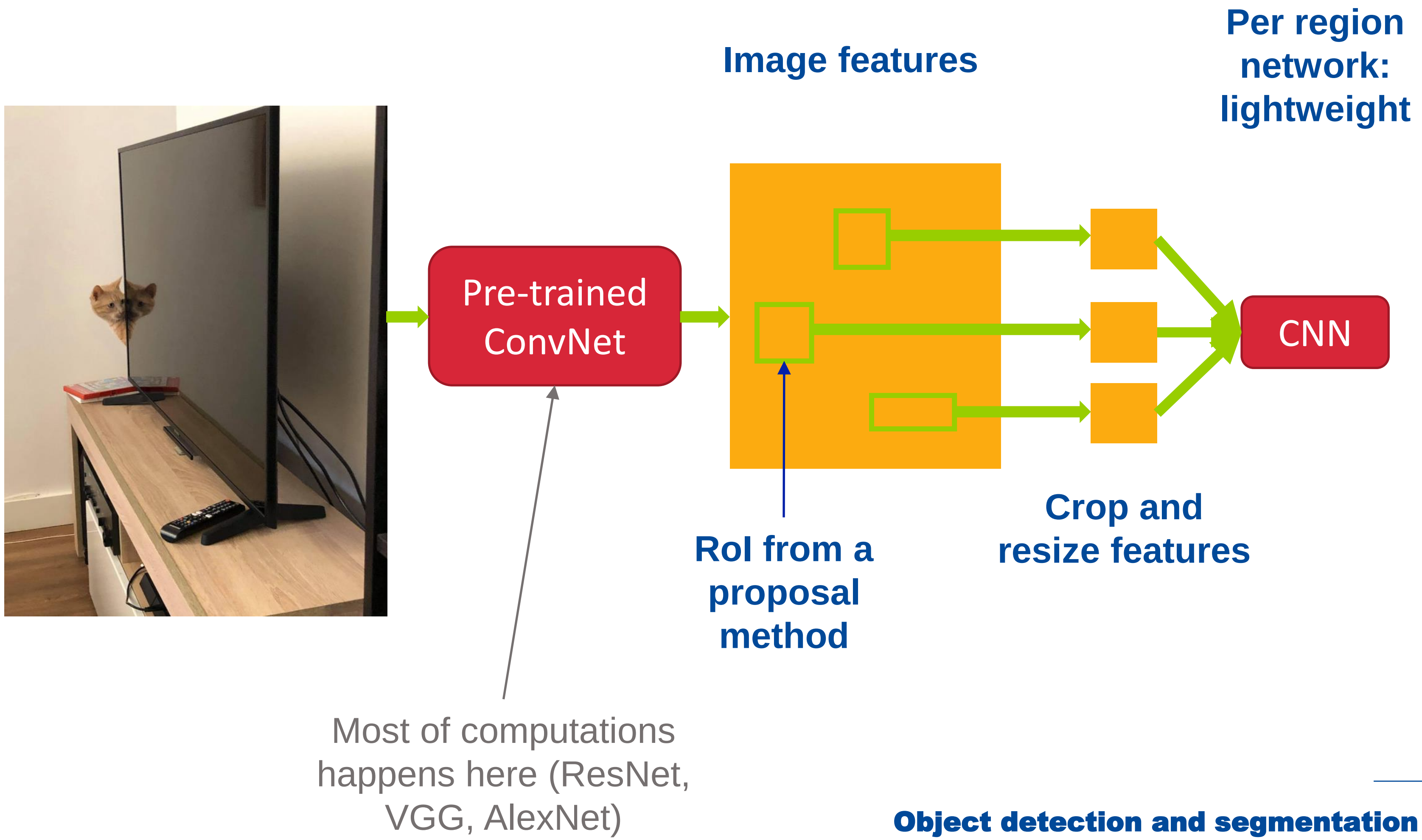


Object detection: Fast-RCNN (Girshick et al., 2015)



Most of computations happens here (ResNet, VGG, AlexNet)

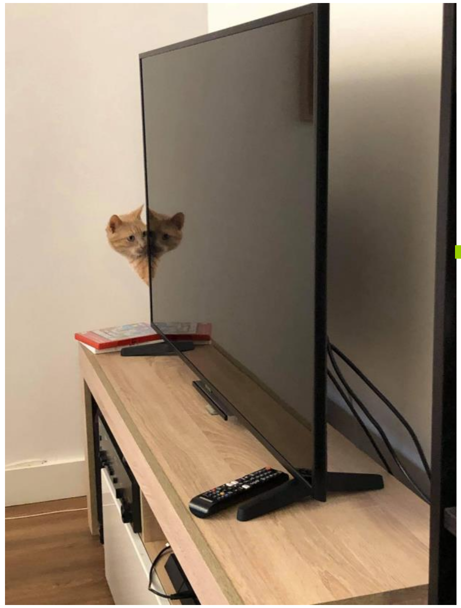
Object detection: Fast-RCNN (Girshick et al., 2015)

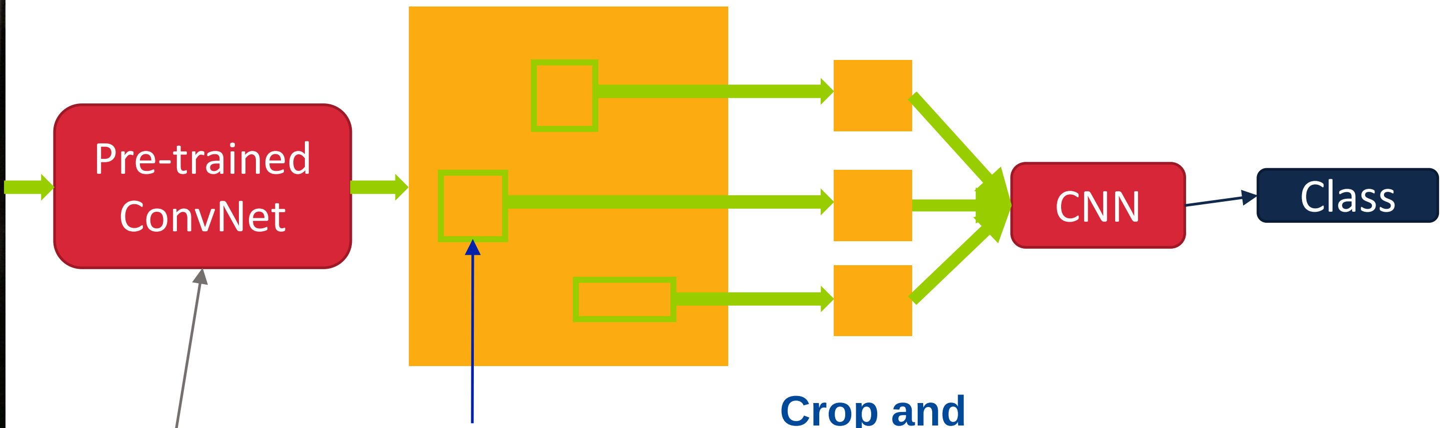


Object detection: Fast-RCNN (Girshick et al., 2015)

Image features

Per region network: lightweight





RoI from a proposal method

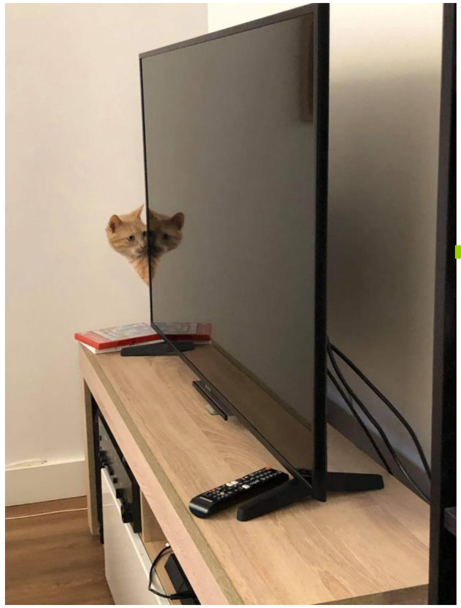
Crop and resize features

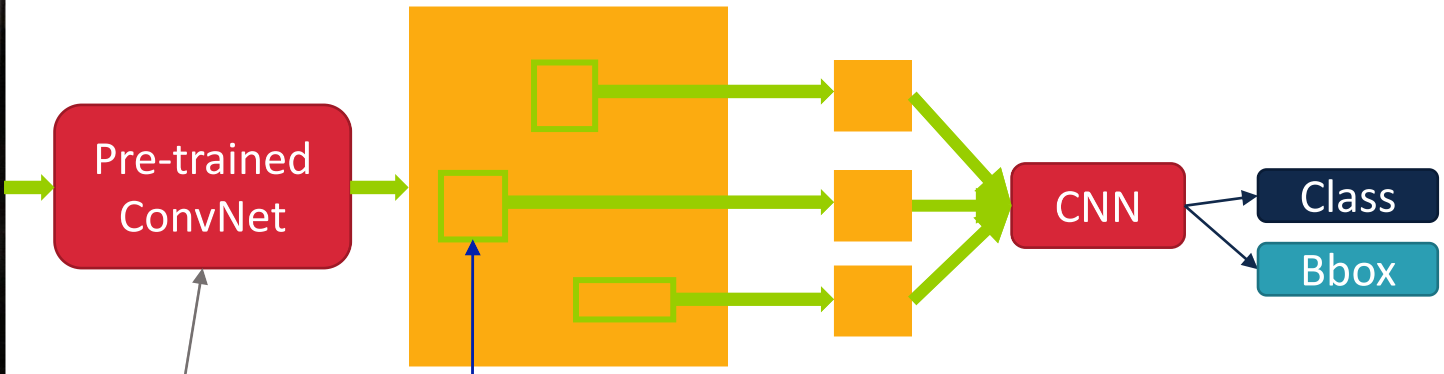
Most of computations happens here (ResNet, VGG, AlexNet)

Object detection: Fast-RCNN (Girshick et al., 2015)

Image features

Per region network: lightweight





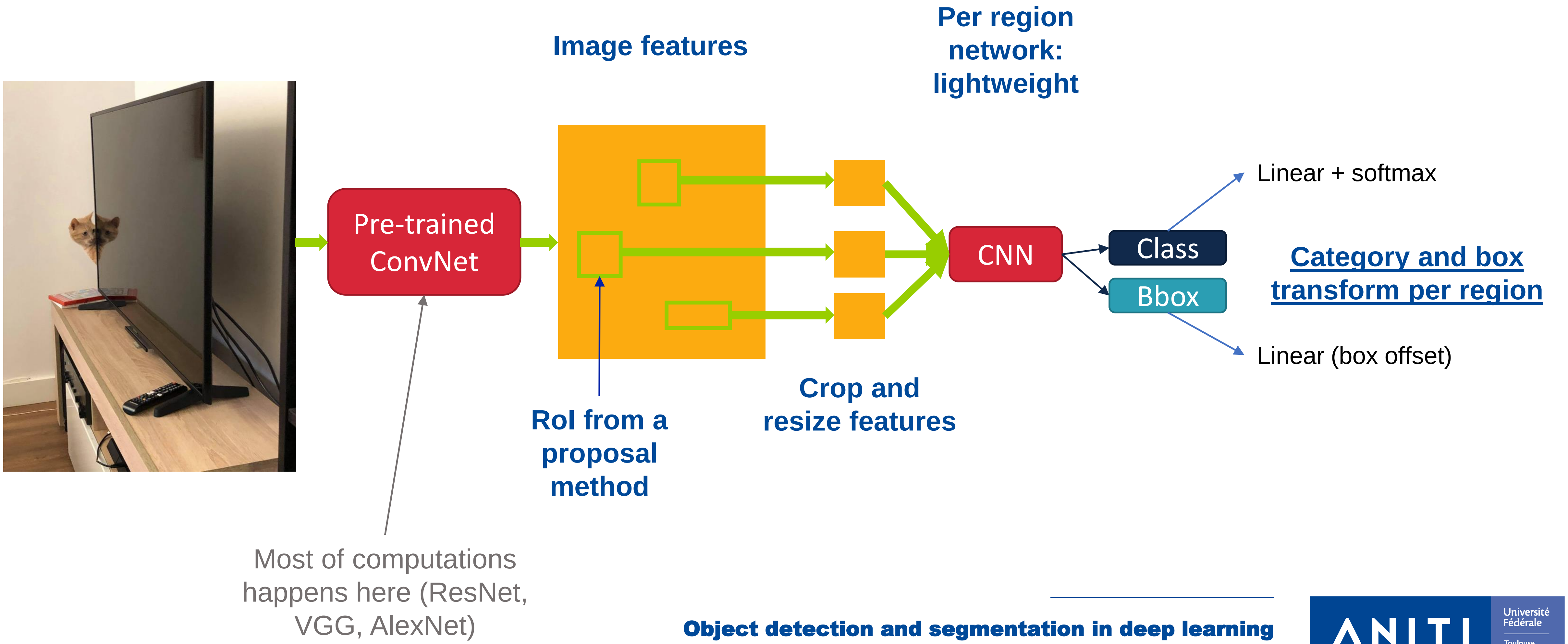
Category and box transform per region

RoI from a proposal method

Crop and resize features

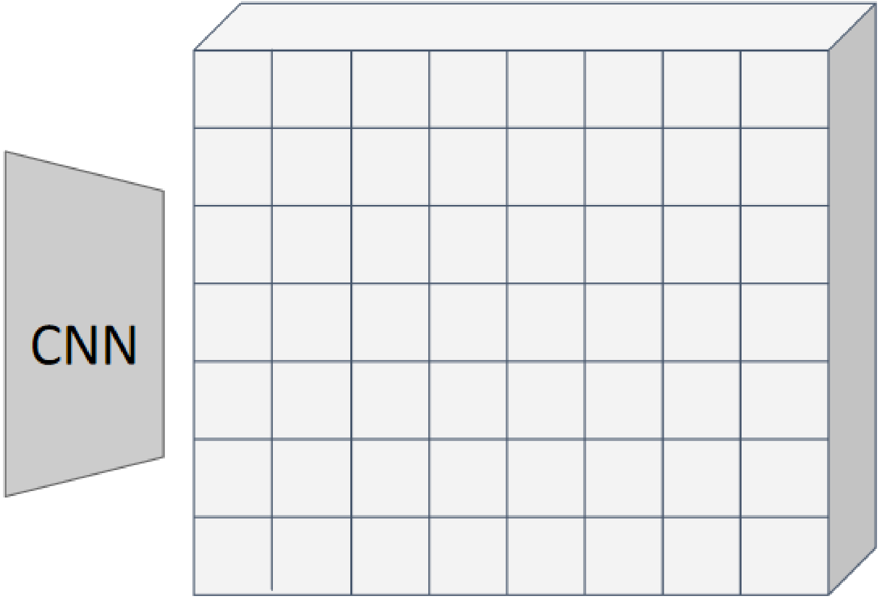
Most of computations happens here (ResNet, VGG, AlexNet)

Object detection: Fast-RCNN (Girshick et al., 2015)



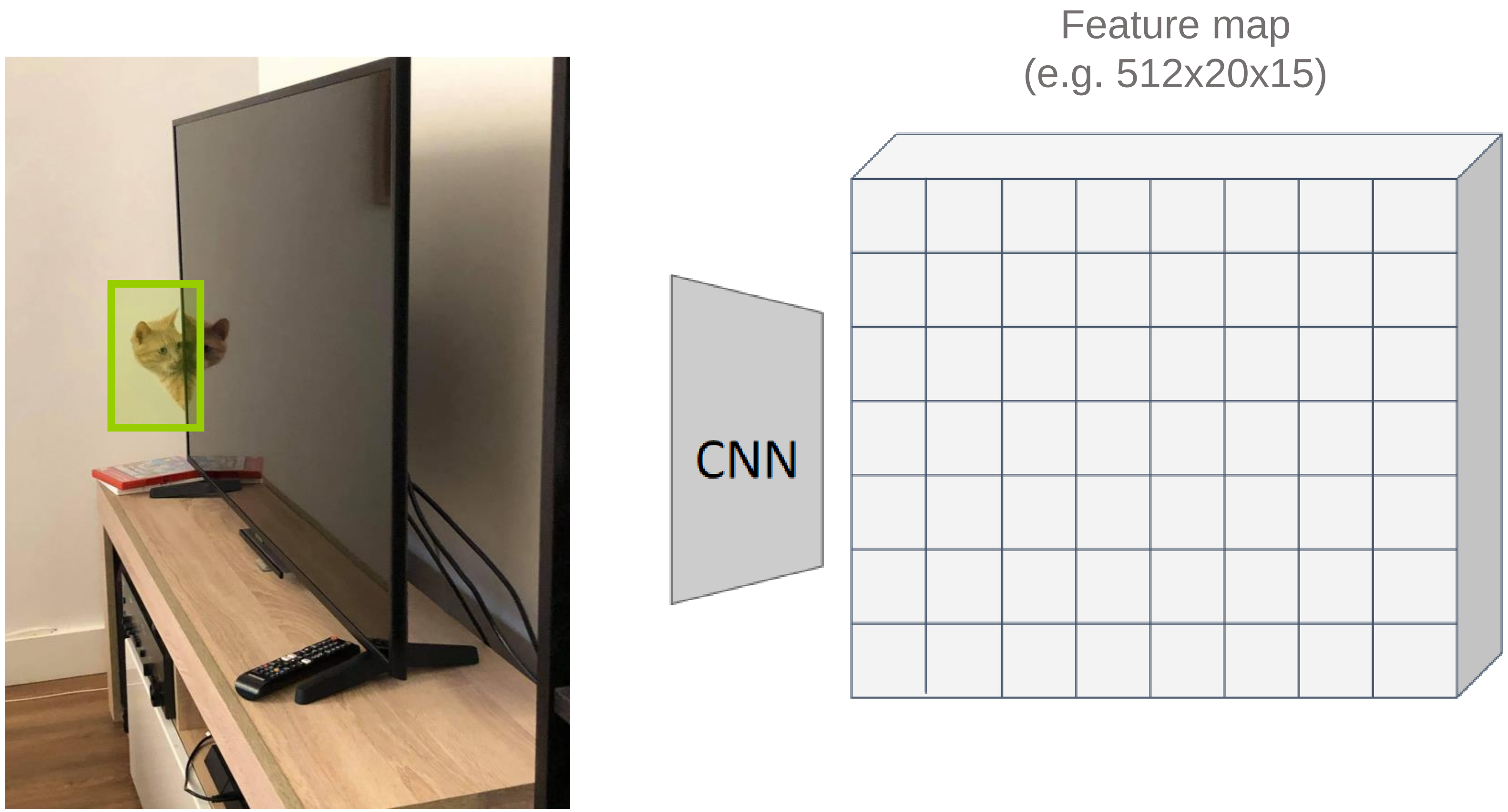
RoI pooling

Input image (e.g. 3x640x480)



Adapted from http://cs231n.stanford.edu/slides/2022/lecture_9_jiajun.pdf

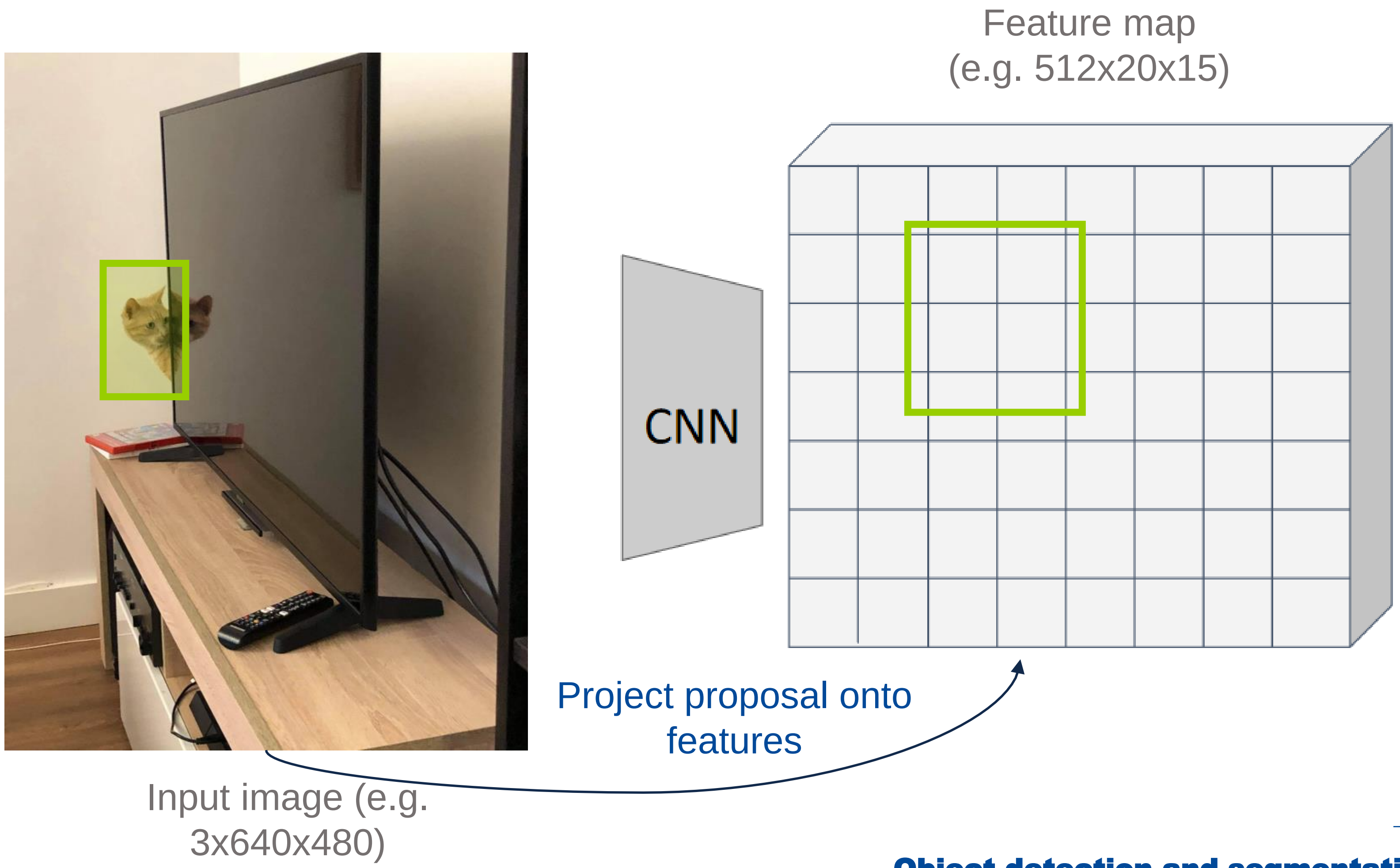
RoI pooling



Input image (e.g. 3x640x480)

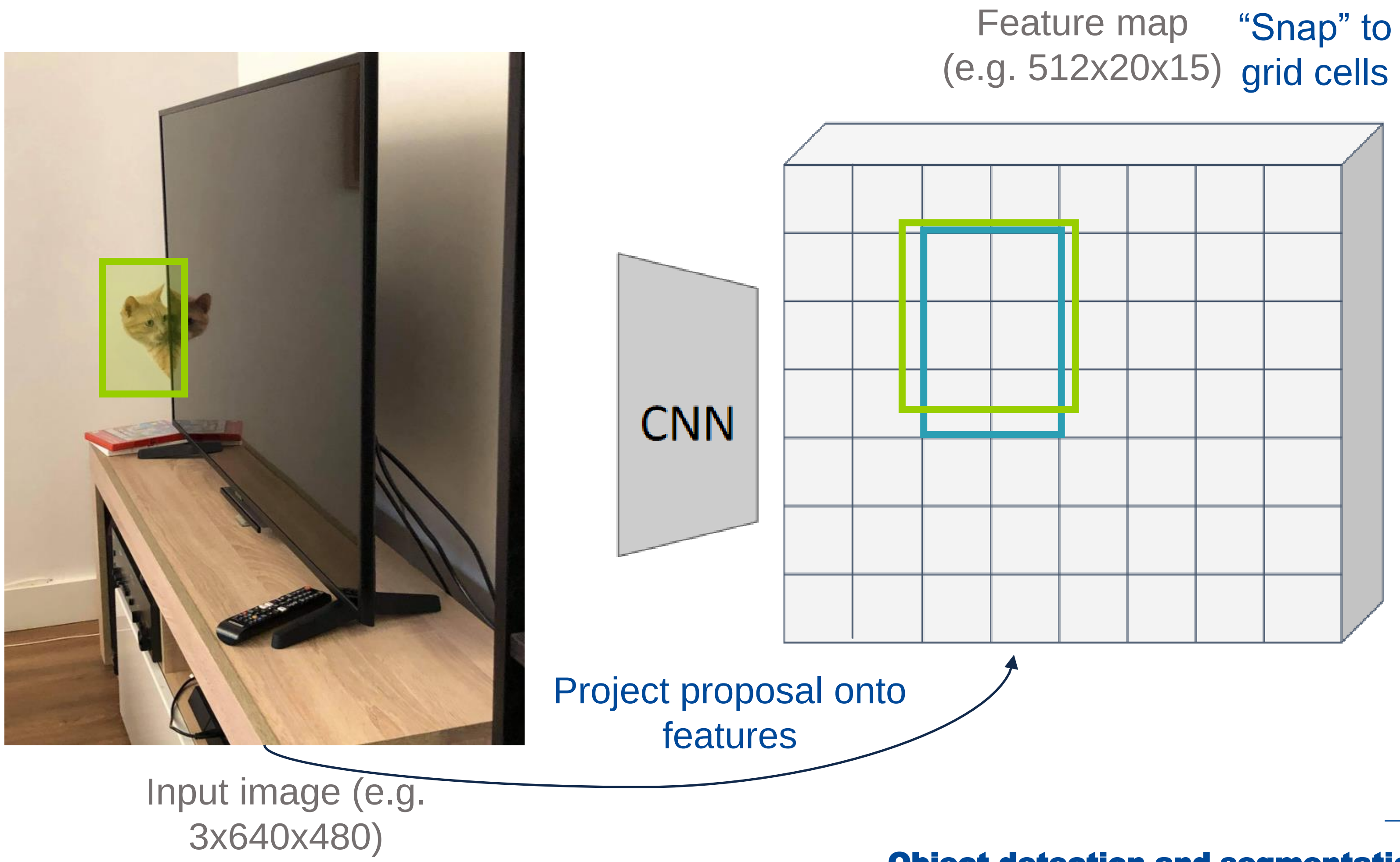
Adapted from http://cs231n.stanford.edu/slides/2022/lecture_9_jiajun.pdf

RoI pooling



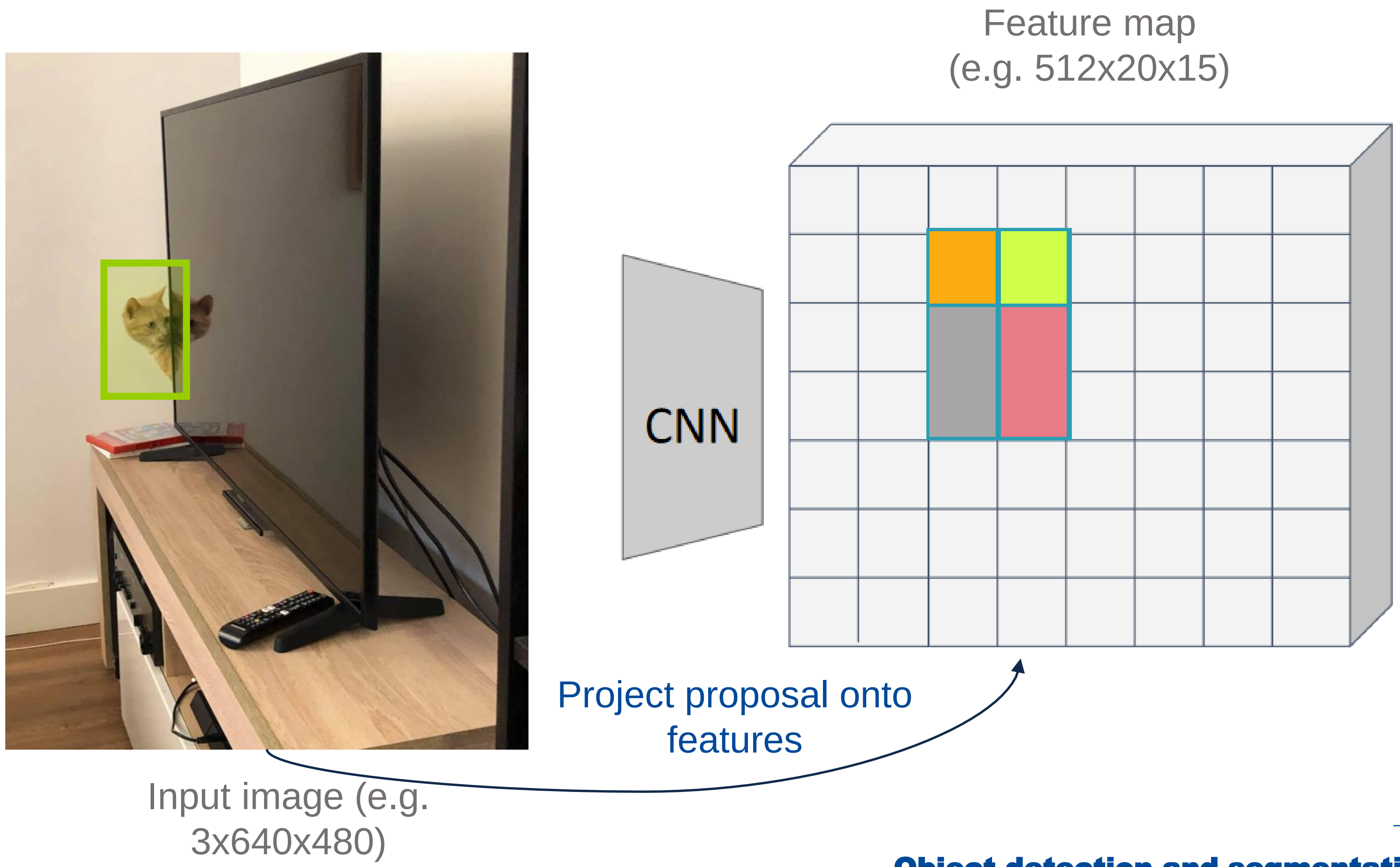
Adapted from http://cs231n.stanford.edu/slides/2022/lecture_9_jiajun.pdf

RoI pooling



Adapted from http://cs231n.stanford.edu/slides/2022/lecture_9_jiajun.pdf

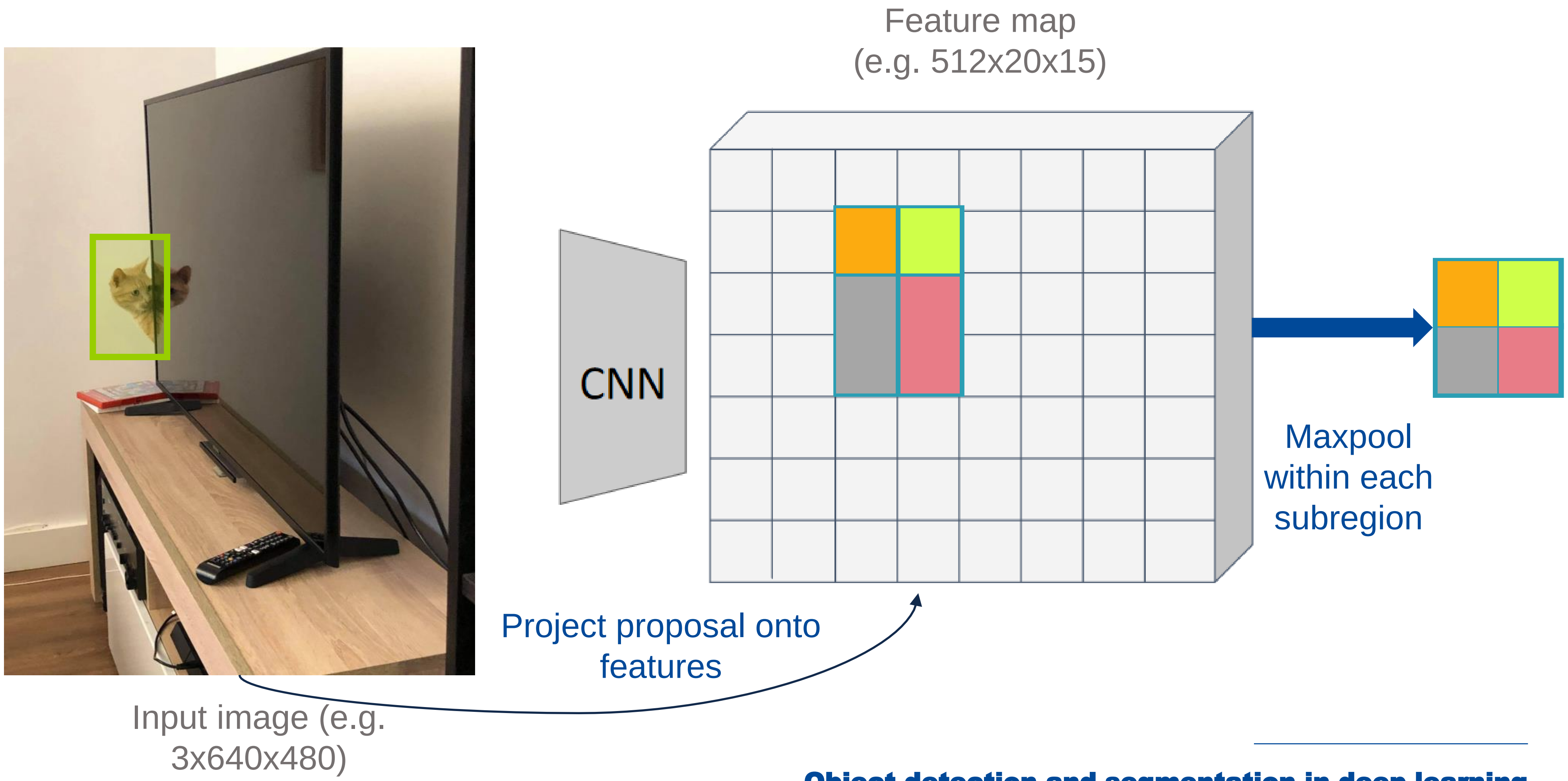
RoI pooling



Divide into 2x2 grid

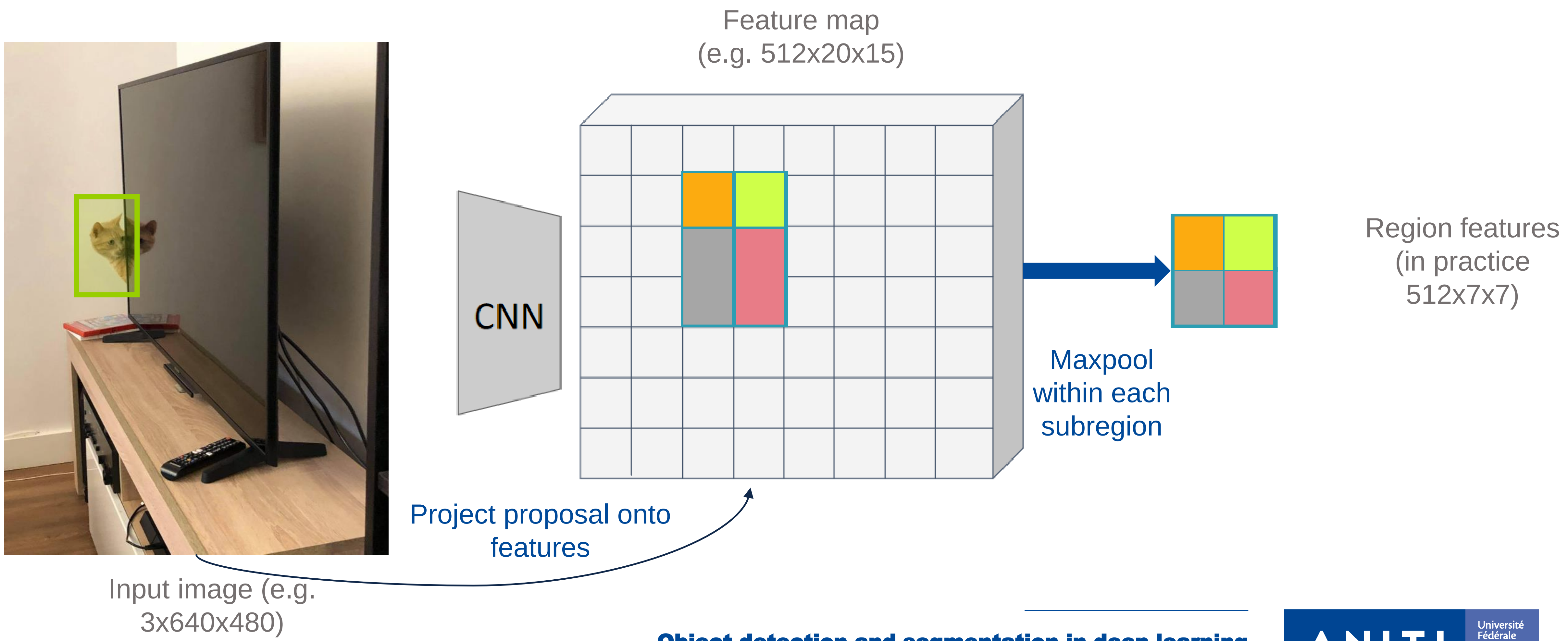
Adapted from http://cs231n.stanford.edu/slides/2022/lecture_9_jiajun.pdf

RoI pooling



Adapted from http://cs231n.stanford.edu/slides/2022/lecture_9_jiajun.pdf

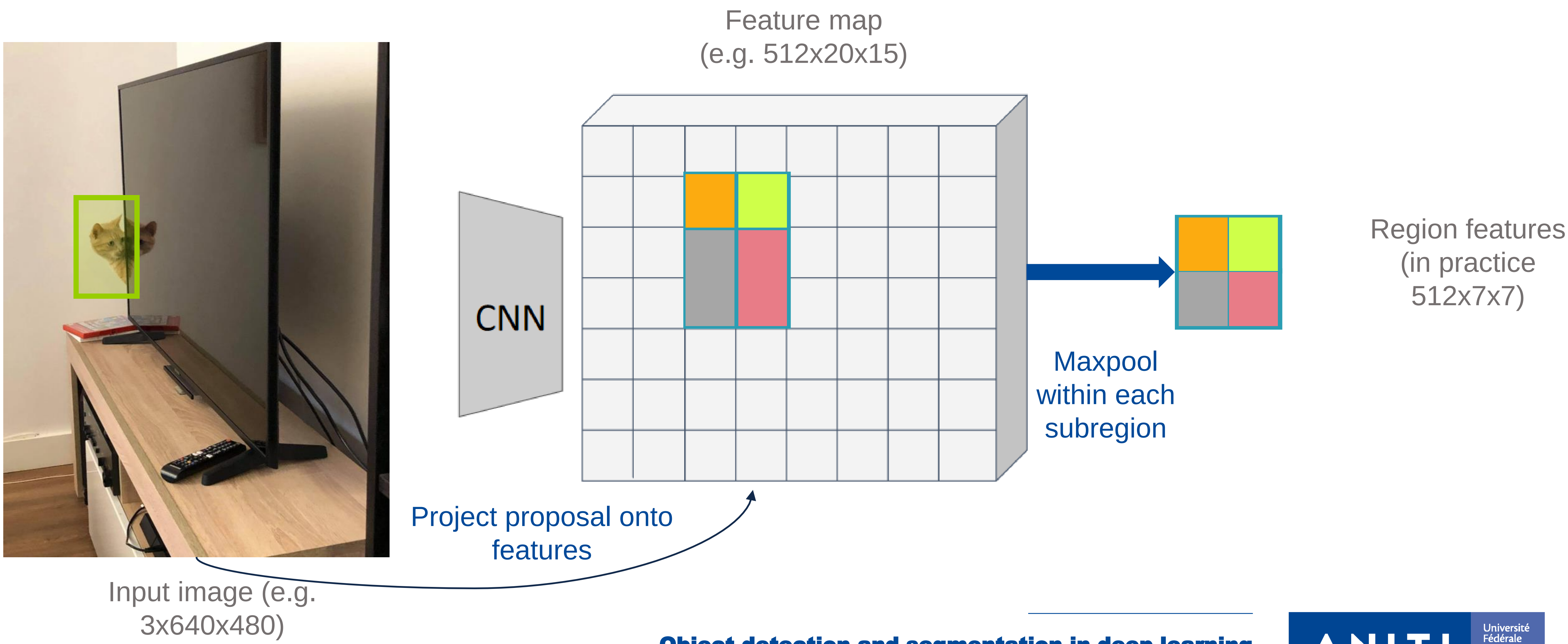
RoI pooling



Adapted from http://cs231n.stanford.edu/slides/2022/lecture_9_jiajun.pdf

Problem: region features might be slightly misaligned → RoI align

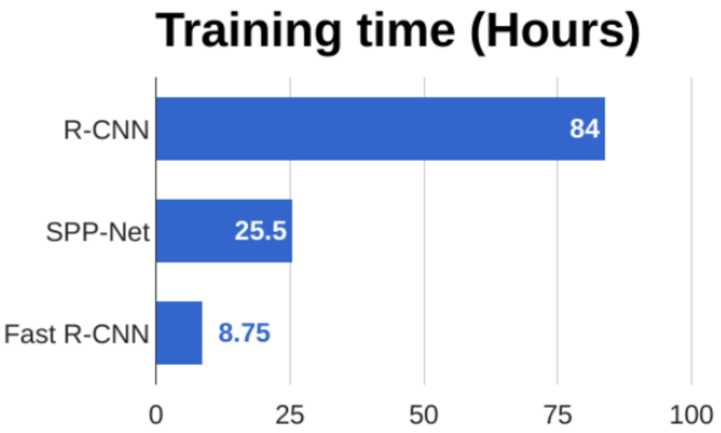
RoI pooling

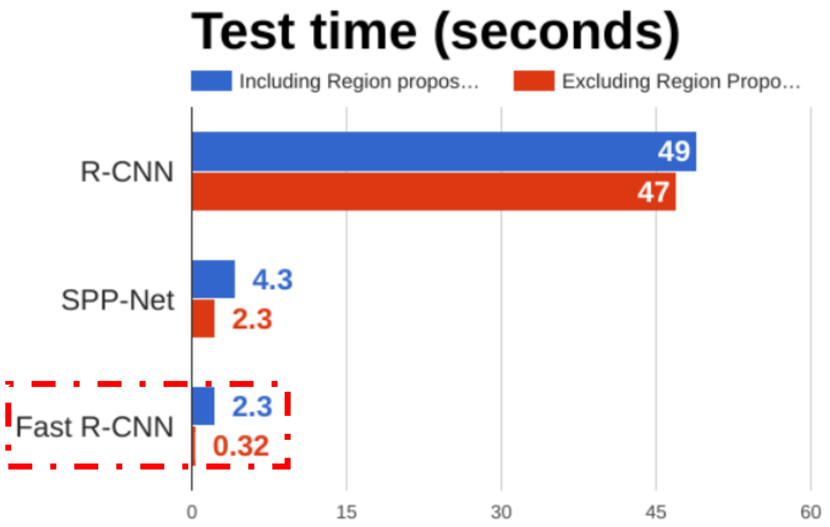


Adapted from http://cs231n.stanford.edu/slides/2022/lecture_9_jiajun.pdf

Object detection: Fast R-CNN vs. “Slow” R-CNN

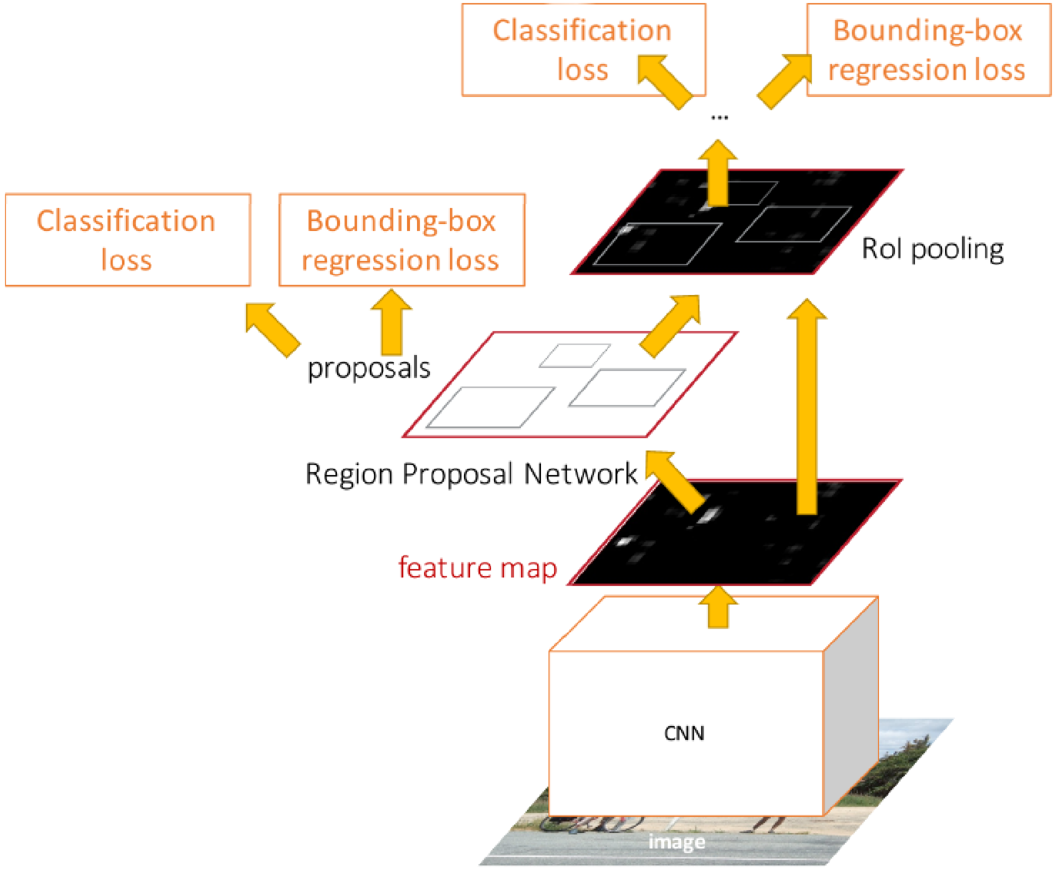
Runtime dominated by region proposals





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

Object detection: learn to *propose* regions, Faster R-CNN (Ren et al., 2015)



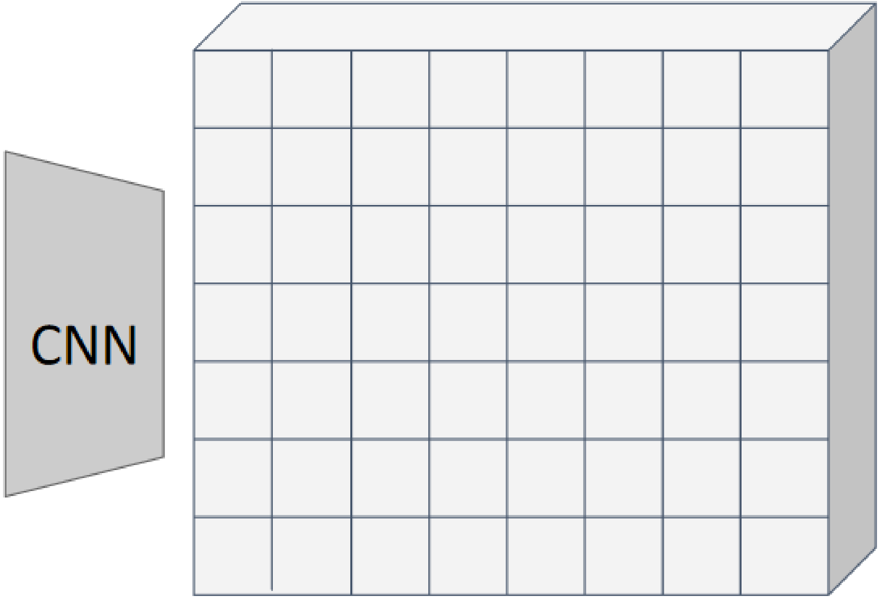
- Idea: insert a **Region Proposal Network (RPN)** to predict proposals from features

- Otherwise same as Fast R-CNN: crop features for each proposal, classify each one

Object detection: Region Proposal Network (RPN)

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

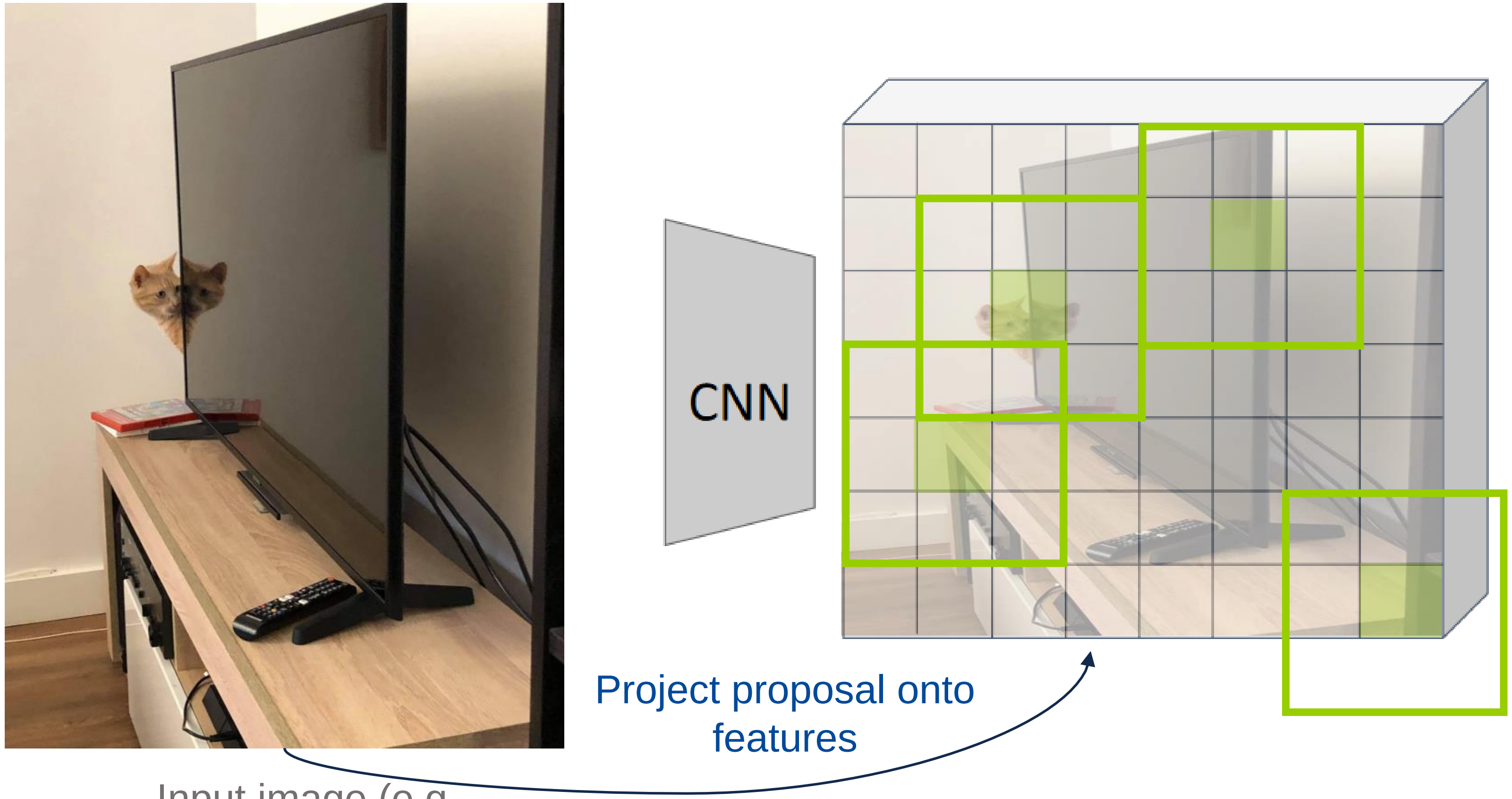
Input image (e.g. 3x640x480)



Adapted from http://cs231n.stanford.edu/slides/2022/lecture_9_jiajun.pdf

Object detection: Region Proposal Network (RPN)

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

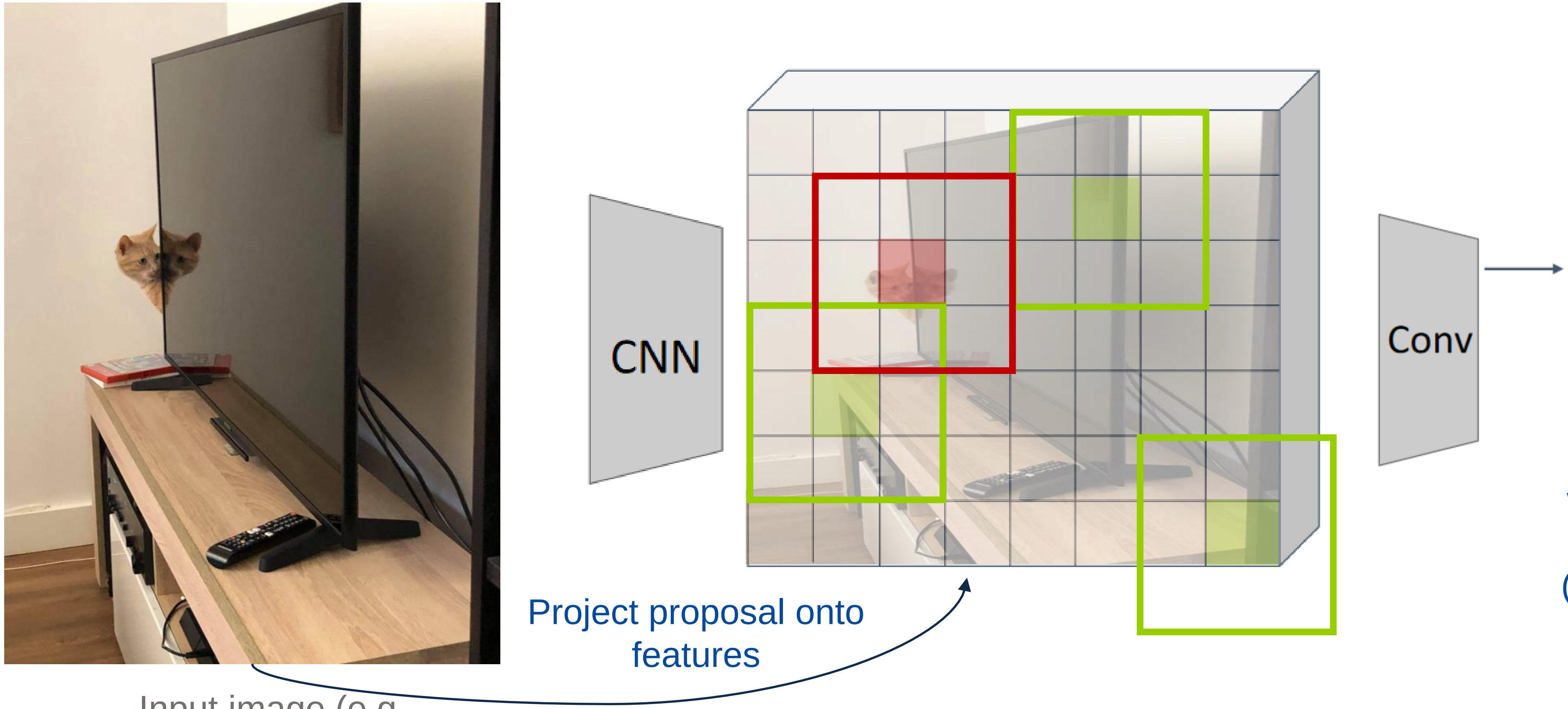


Input image (e.g. 3x640x480)

Adapted from http://cs231n.stanford.edu/slides/2022/lecture_9_jiajun.pdf

Object detection: Region Proposal Network (RPN)

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



Anchor is an object?

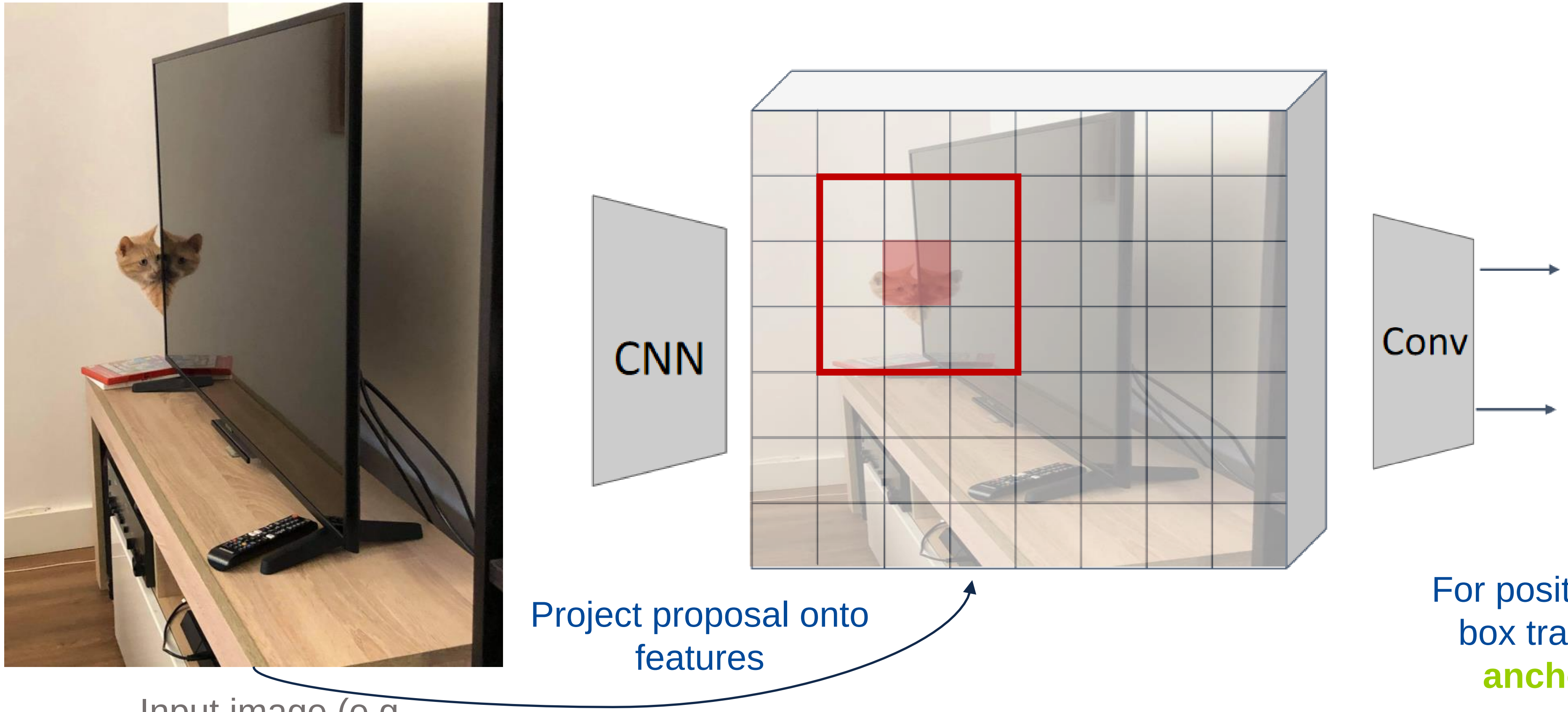
At each point, predict whether the corresponding anchor contains an object (per-cell logistic regression, predict scores with conv layer)

Input image (e.g. 3x640x480)

Adapted from http://cs231n.stanford.edu/slides/2022/lecture_9_jiajun.pdf

Object detection: Region Proposal Network (RPN)

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



Input image (e.g. 3x640x480)

Anchor is an object?

1x20x15

Box transforms

4x20x15

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

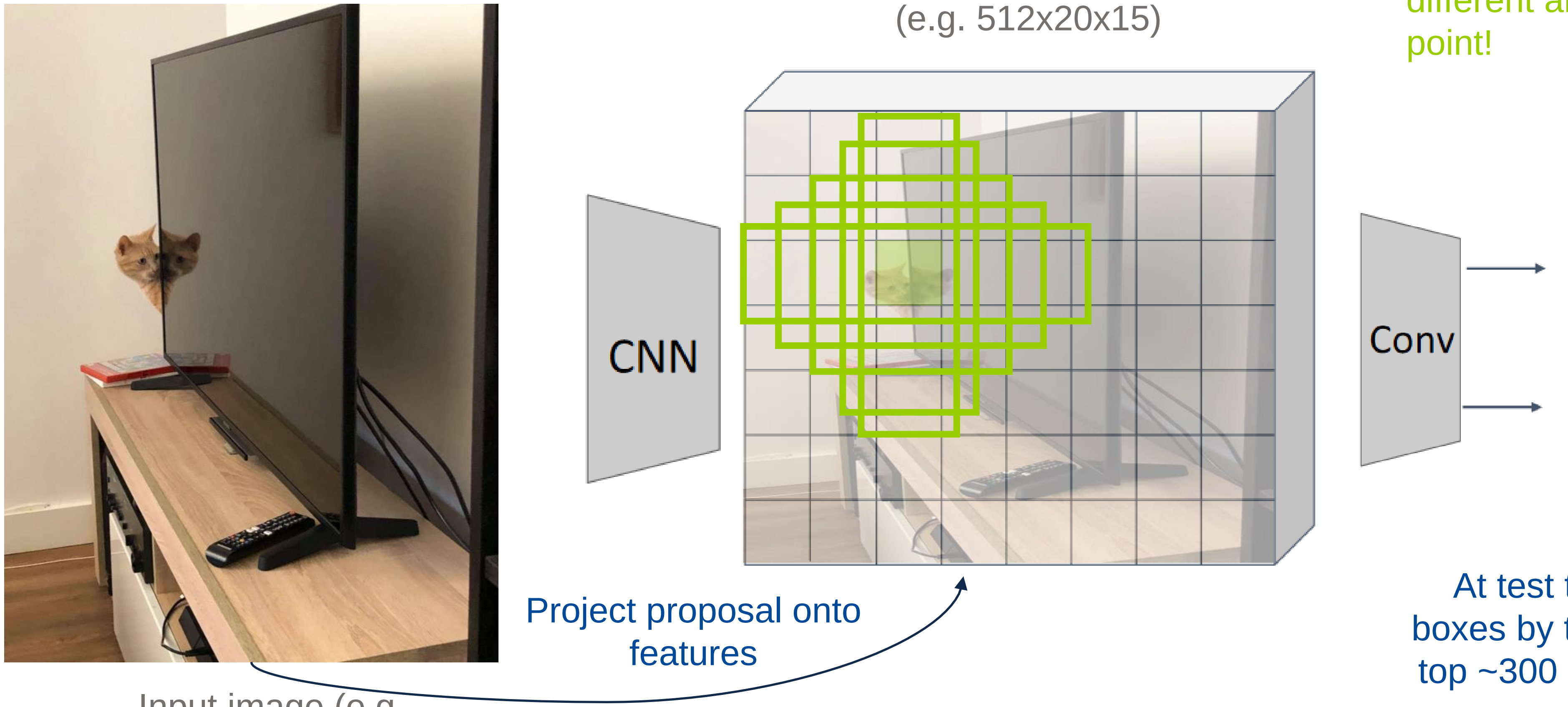
Adapted from http://cs231n.stanford.edu/slides/2022/lecture_9_jiajun.pdf

Object detection: Region Proposal Network (RPN)

Problem: Anchor boxes may have the wrong size/shape

Solution: In practice, we use K different anchor boxes at each point!

Feature map
(e.g. 512x20x15)



Project proposal onto features

Input image (e.g. 3x640x480)

Anchor is an object?

Kx20x15

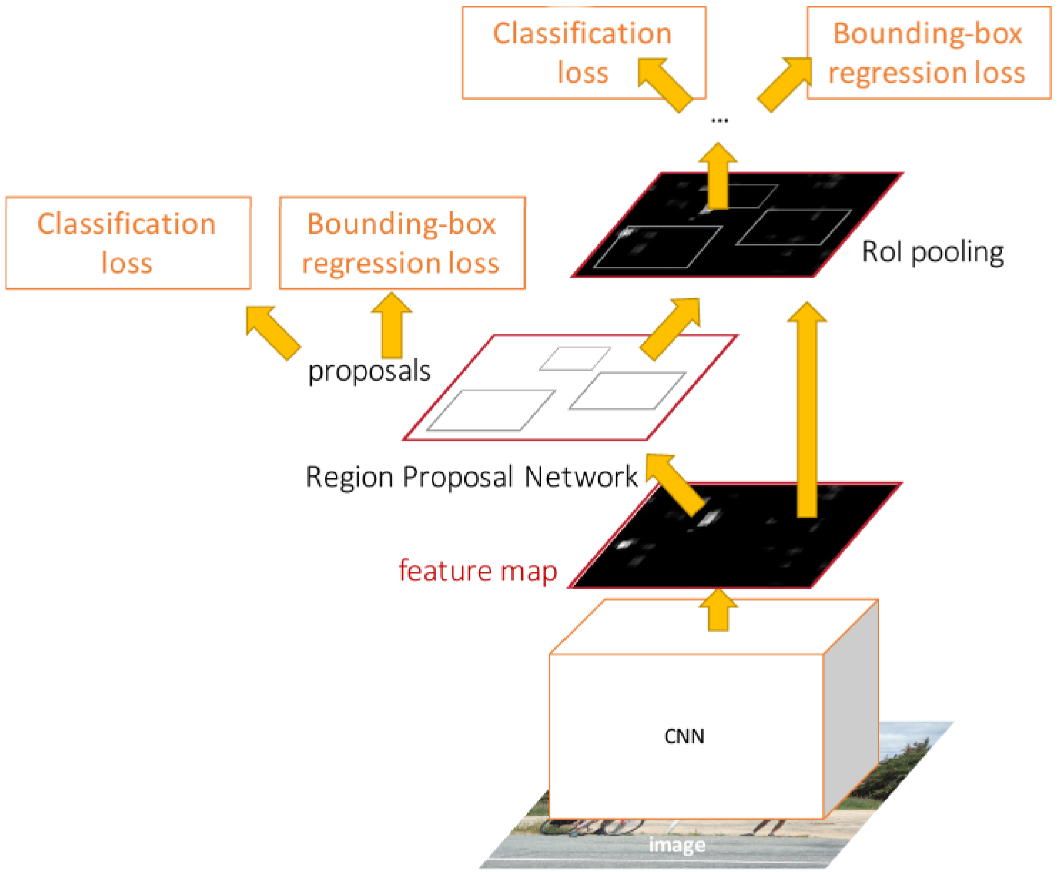
Box transforms

4**K**x20x15

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

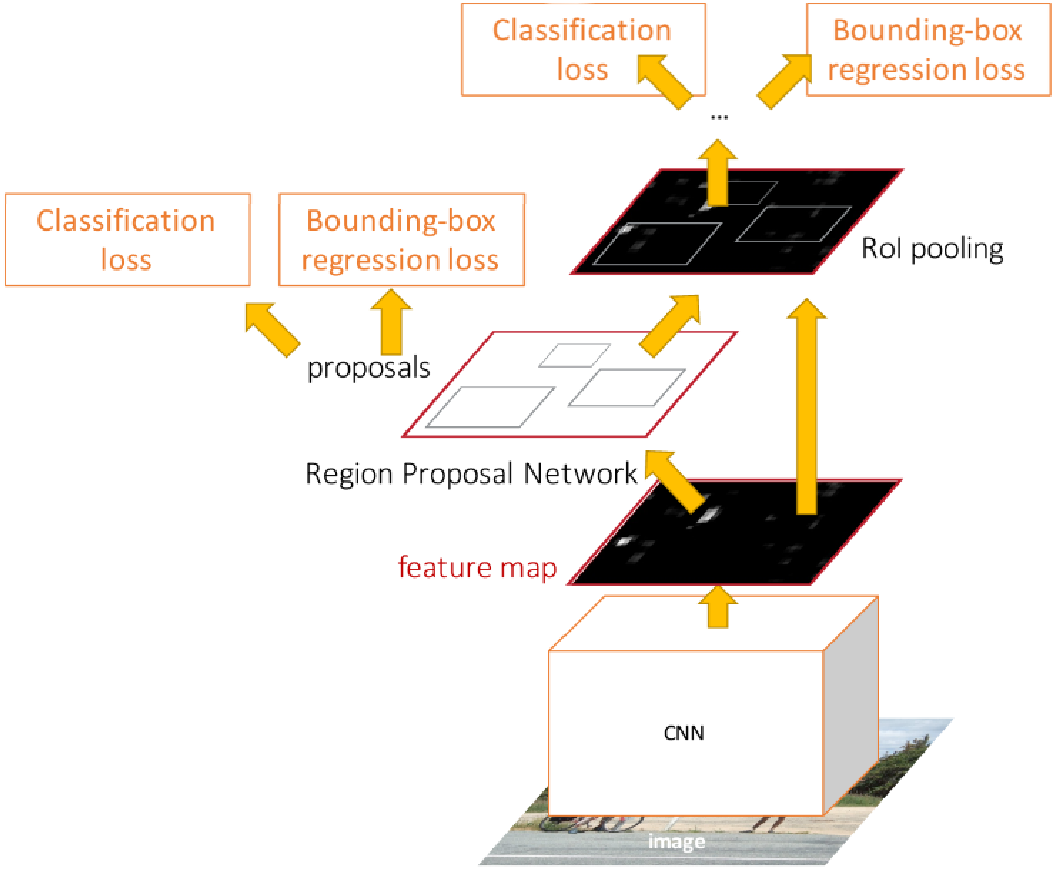
Adapted from http://cs231n.stanford.edu/slides/2022/lecture_9_jiajun.pdf

Object detection: learn to *propose* regions, Faster R-CNN (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

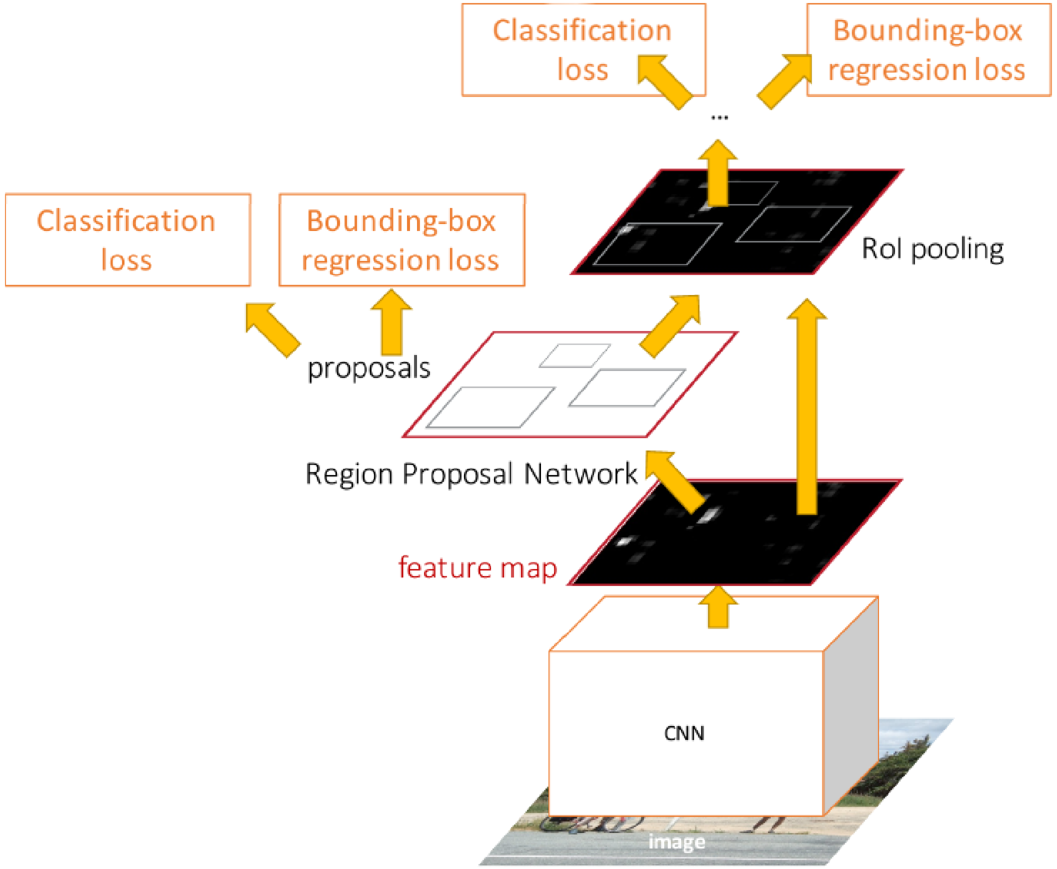
Object detection: learn to *propose* regions, Faster R-CNN (Ren et al., 2015)



- 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

Test time speed: 0.2 seconds vs 2.3 seconds (Fast R-CNN) vs 49 seconds (R-CNN)!

Object detection: learn to *propose* regions, Faster R-CNN (Ren et al., 2015)



- Faster R-CNN is a two-stage object detector:
  1. Extract features using a backbone network and propose regions (mostly background)
  2. For each region: crop features, predict object class and bounding box offset
- Single-stage object detectors: YOLO, SSD, RetinaNet

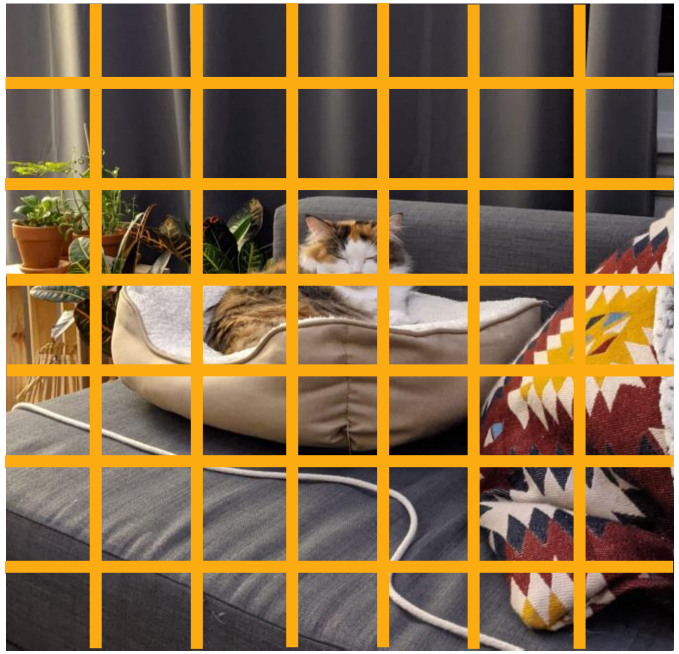
Object detection: single-stage object detectors (YOLO, SSD, RetinaNet)

- 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 box

Object detection: YOLO (Redmond et al., 2015)

Object detection: YOLO (Redmond et al., 2015)

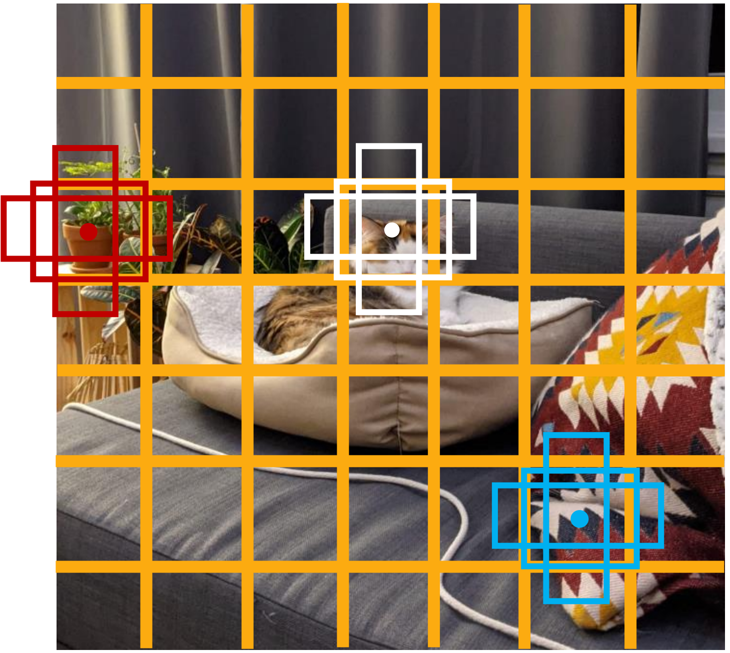
1. Divide the image into cells with an SxS grid



S=7

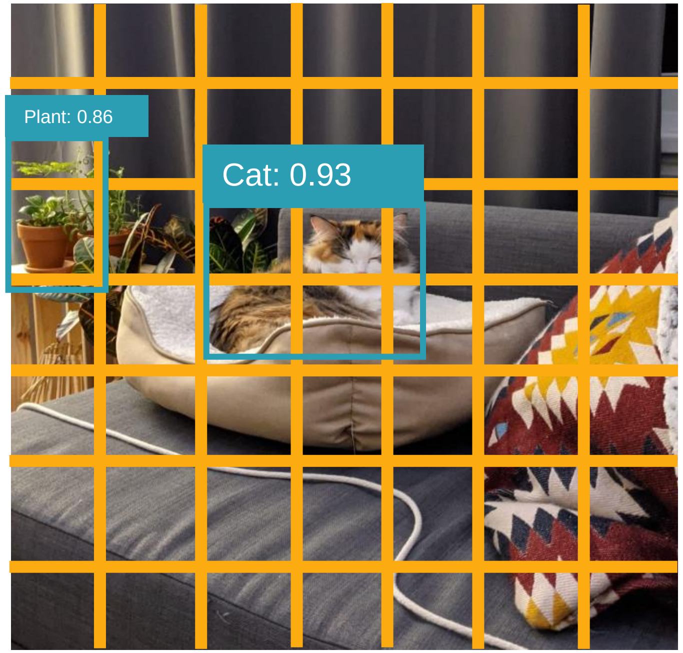
Object detection: YOLO (Redmond et al., 2015)

2. Each cell predicts B bounding boxes



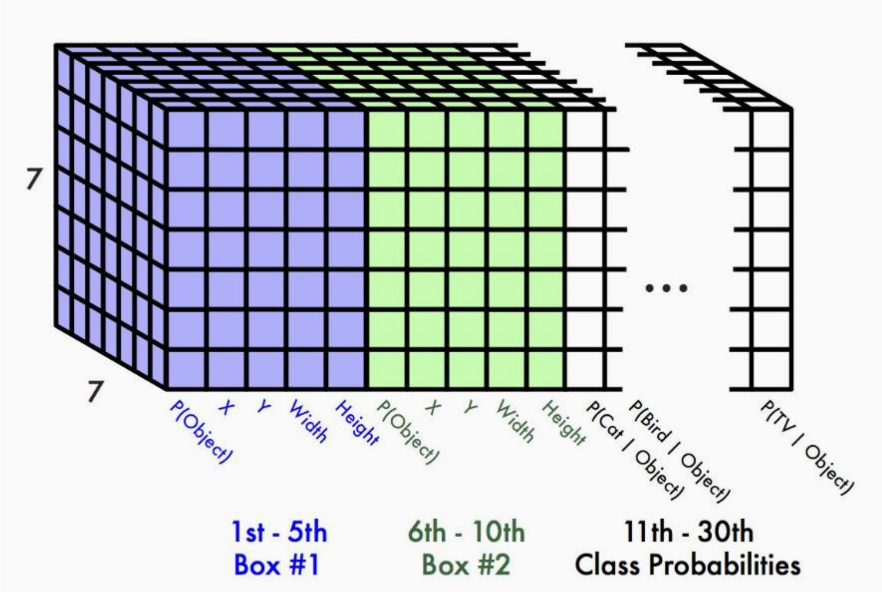
Object detection: YOLO (Redmond et al., 2015)

3. Return bounding boxes above confidence threshold

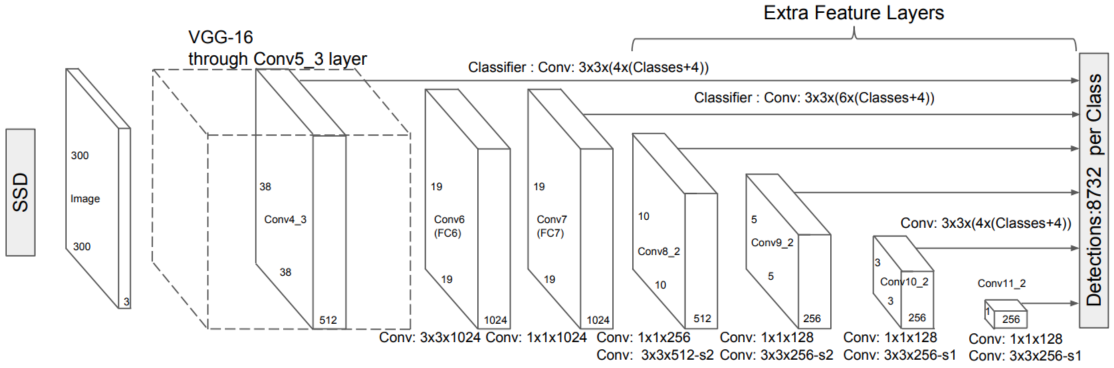


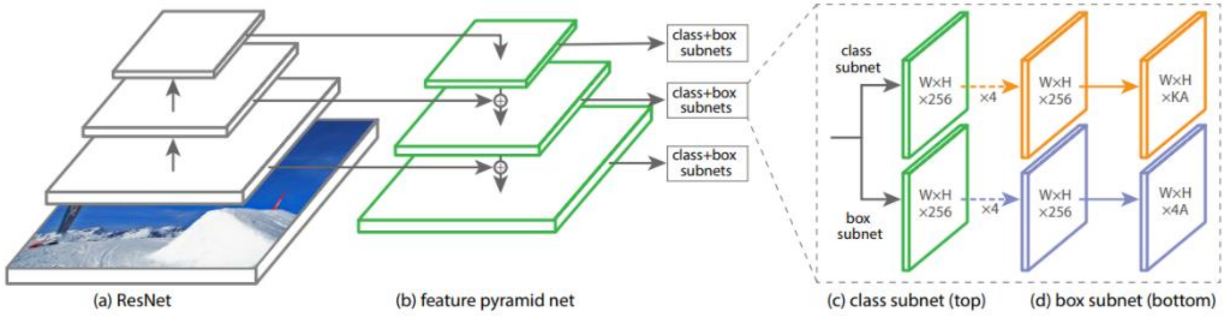
All other bounding boxes have a confidence probability less than the threshold (e.g 0.9) so they are suppressed.

- Each cell predicts:
  - For each anchor box:
    - 4 (box offset) coordinates (dx, dy, dh, dw)
    - 1 confidence value
  - Class probabilities (80 for COCO dataset, 20 for PASCAL-VOC)
- Output: 7x7x(5*B + C)
- Similar to RPN!



Object detection: single stage object detectors





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
Lin, T. Y., Goyal, P., Girshick, R., He, K., & Dollár, P. (2017). Focal loss for dense object detection. In *Proceedings of the IEEE international conference on computer vision* (pp. 2980-2988)..

Object detection (and segmentation): but...

Object detection (and segmentation): but...

Object detection/segmentation is a wide field of research, and many architectures exist.

Object detection (and segmentation): but…

Object detection/segmentation is a wide field of research, and many architectures exist.

- Alternatives to anchor-based methods exist:
  - CenterNet (Duan et al., 2019)
  - FCOS (Tian et al., 2019)
  - R-FCN (Dai et al., 2016)
  - DETR (object detection with transformers) (Carion et al., 2020)

Object detection (and segmentation): but...

Object detection/segmentation is a wide field of research, and many architectures exist.

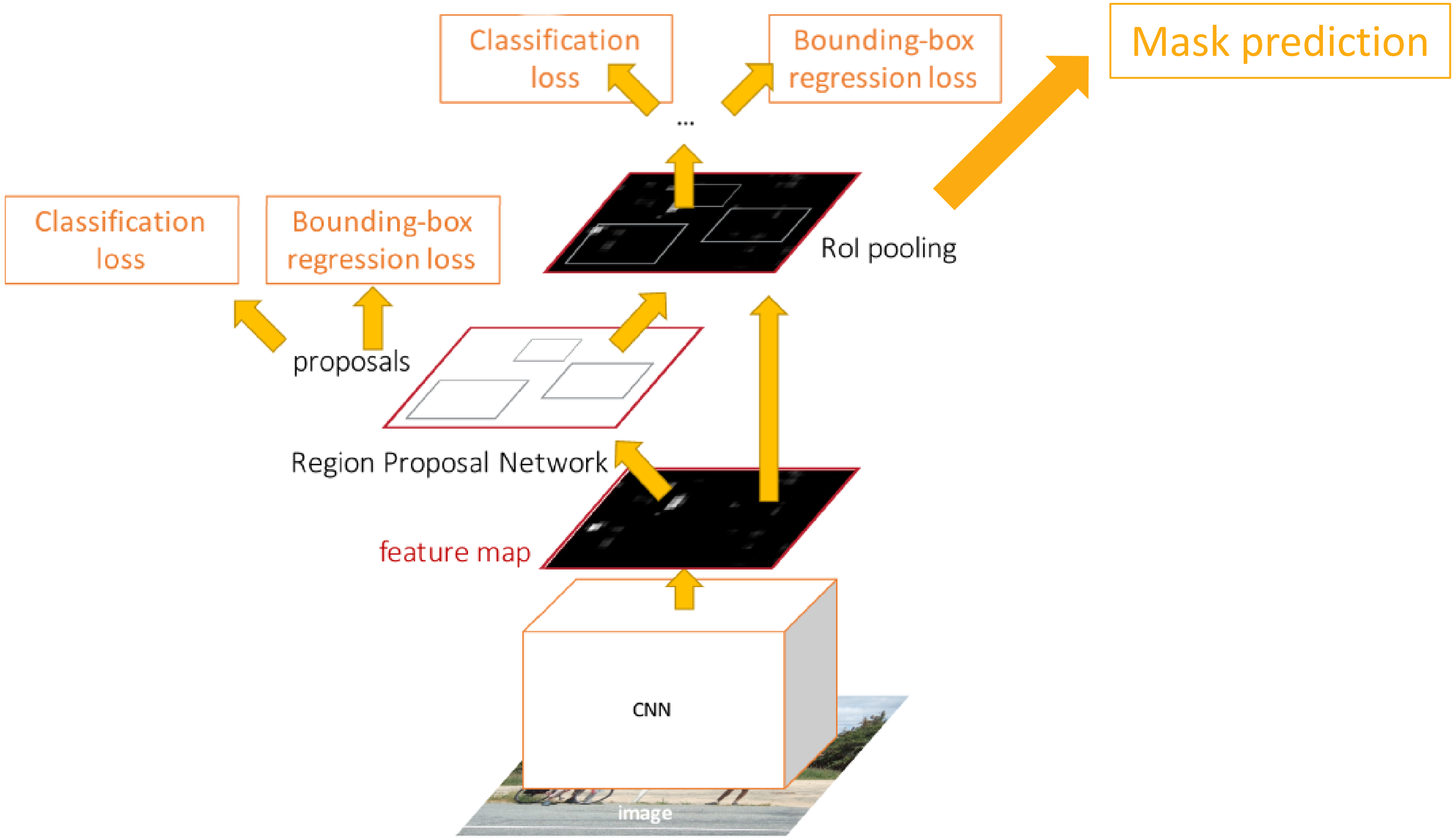
- Alternatives to anchor-based methods exist:
  - CenterNet (Duan et al., 2019)
  - FCOS (Tian et al., 2019)
  - R-FCN (Dai et al., 2016)
  - DETR (object detection with transformers) (Carion et al., 2020)
- Lot of variables:
  - Backbone network: VGG, ResNet, InceptionV2/V3, MobileNet, EfficientNet
  - Architecture style: two-stage, single-stage, hybrid…

Object detection (and segmentation): but…

Object detection/segmentation is a wide field of research, and many architectures exist.

- Alternatives to anchor-based methods exist:
  - CenterNet (Duan et al., 2019)
  - FCOS (Tian et al., 2019)
  - R-FCN (Dai et al., 2016)
  - DETR (object detection with transformers) (Carion et al., 2020)
- Lot of variables:
  - Backbone network: VGG, ResNet, InceptionV2/V3, MobileNet, EfficientNet
  - Architecture style: two-stage, single-stage, hybrid…
- Takeways:
  - Two-stage detectors are slower but more accurate
  - Single-stage detectors are faster but not as accurate
  - Bigger / Deeper backbones work better

Instance segmentation: Mask R-CNN

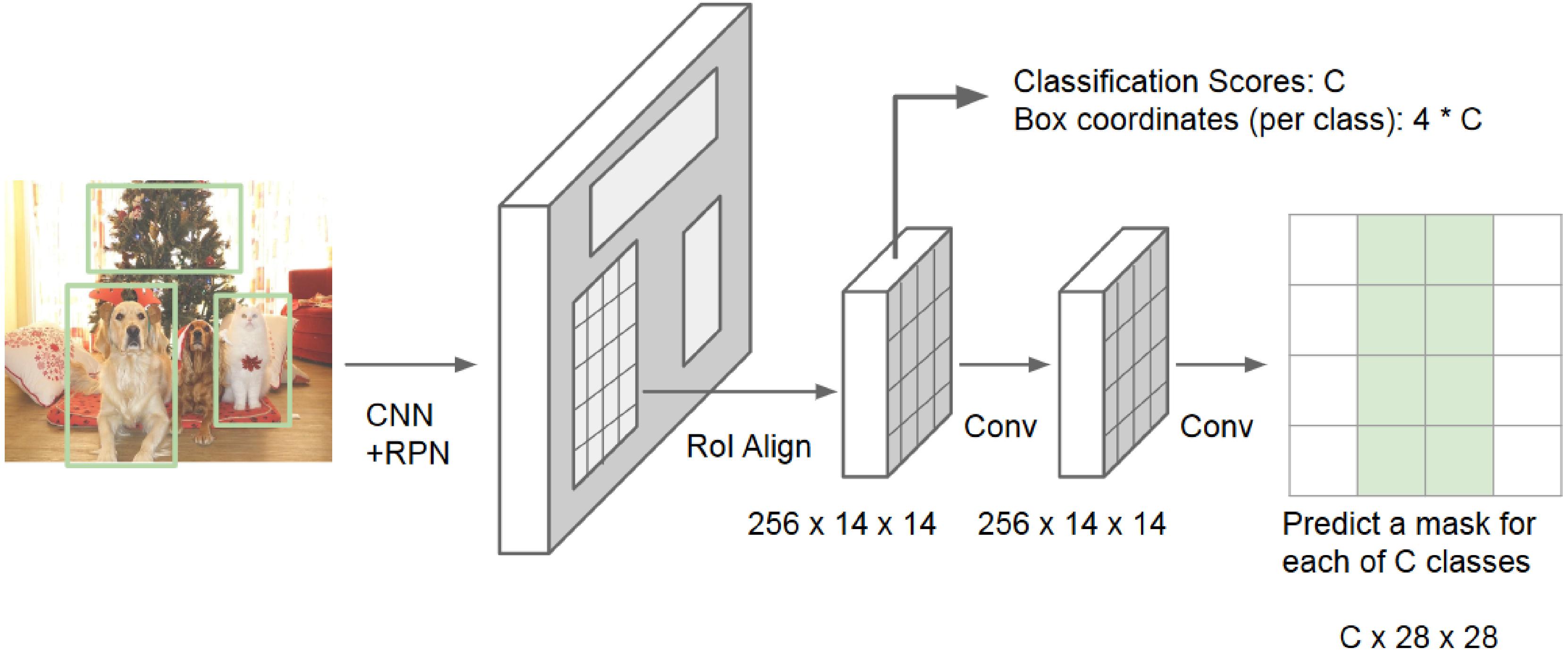


Add a small mask network that operates on each RoI and predicts 28x28 binary mask

From http://cs231n.stanford.edu/slides/2022/lecture_9_jiajun.pdf

Instance segmentation: Mask R-CNN

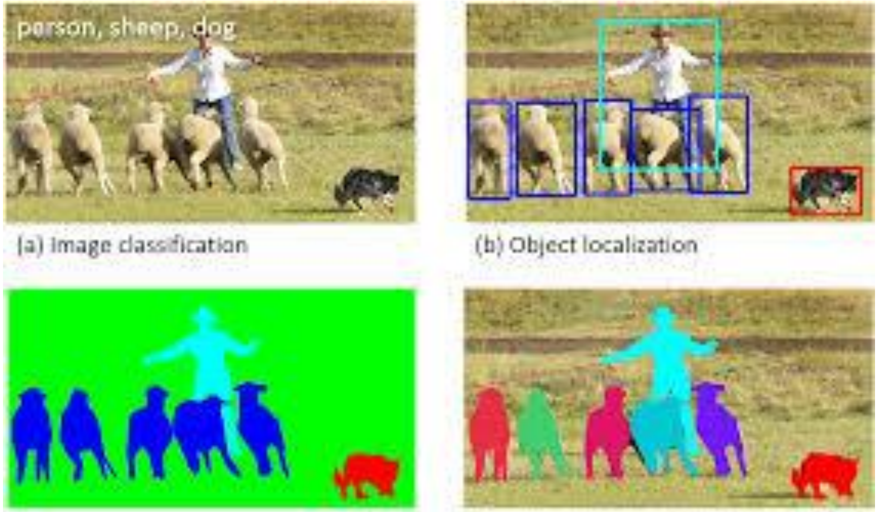
Add a small mask network that operates on each RoI and predicts 28x28 binary mask



From http://cs231n.stanford.edu/slides/2022/lecture_9_jiajun.pdf

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, semantic and instance segmentation, tracking…



(a) Image classification (b) Object localization (c) Semantic segmentation (d) This work

Object detection and segmentation: frameworks

Many implementations of the aforementioned model are available on GitHub.

TensorFlow Object Detection API: https://github.com/tensorflow/models/tree/master/research/object_detection

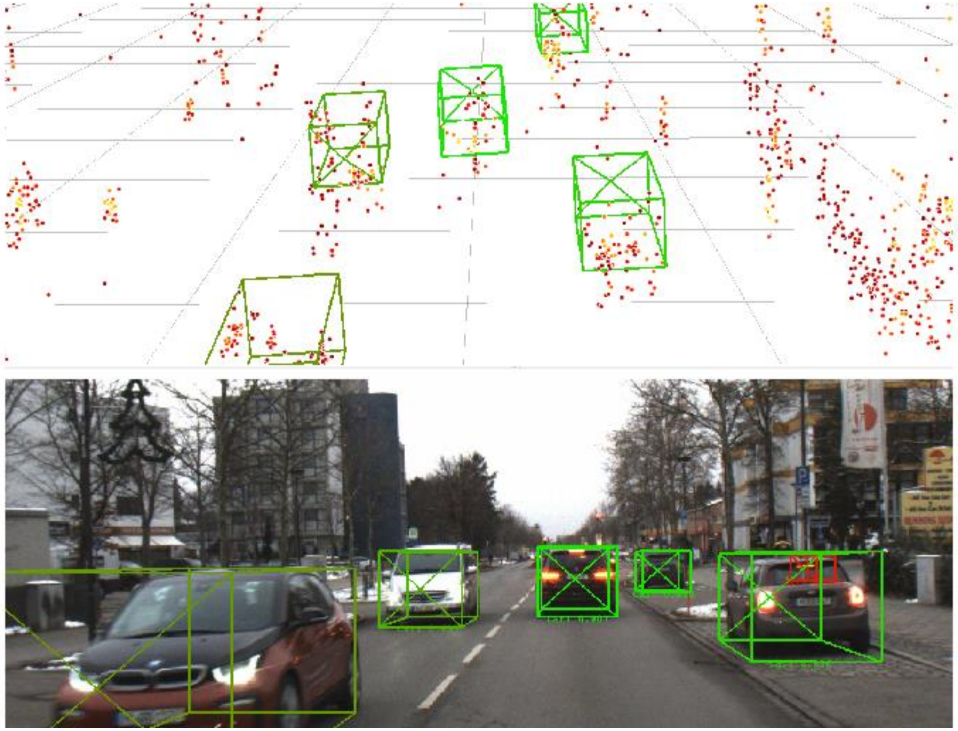
Detectron2 (PyTorch): https://github.com/facebookresearch/detectron2

Torchvision (PyTorch): https://pytorch.org/vision/stable/index.html

Use pre-trained model to finetune on your own dataset!

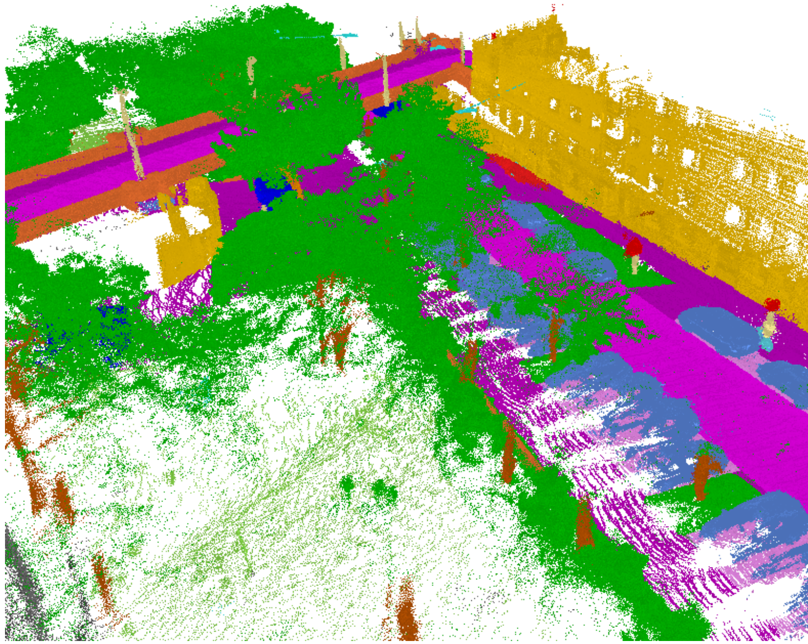
Other object detection/segmentation tasks

3D object detection



Key point object detection

3D semantic segmentation



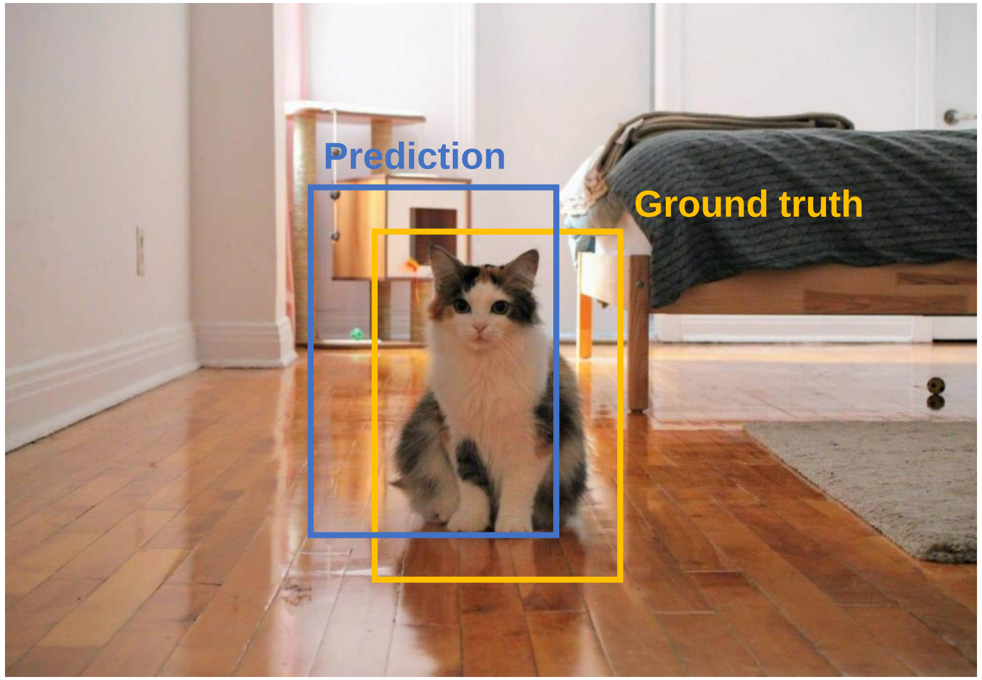
Appendix

ANITI

Université Fédérale Toulouse Midi-Pyrénées

Object detection (or segmentation): comparing boxes (or masks)

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

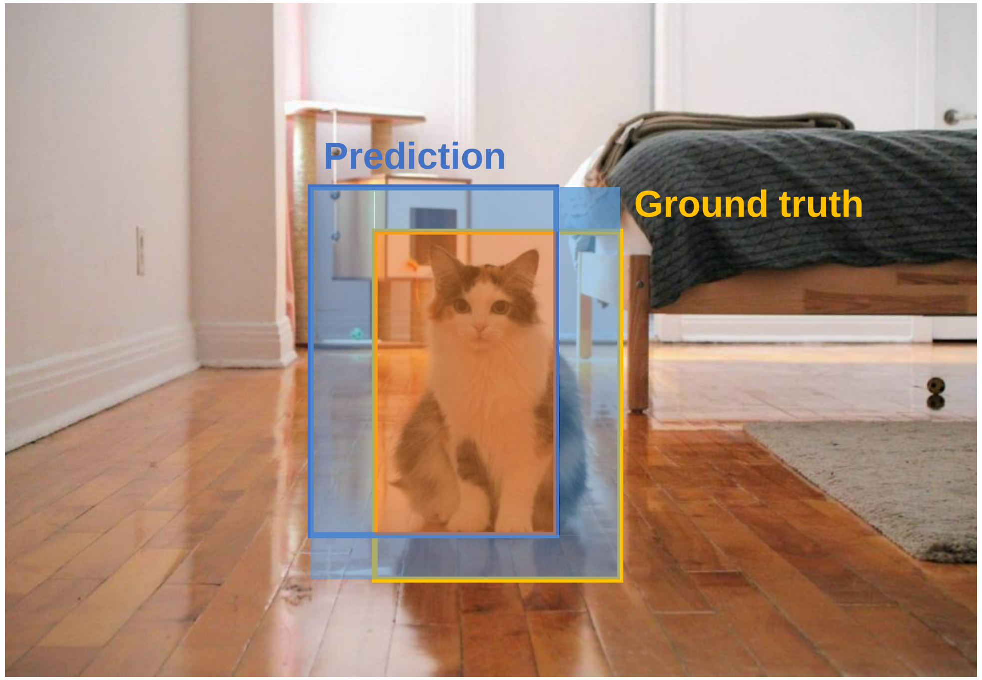


From: https://web.eecs.umich.edu/~justincj/slides/eecs498/498_FA2019_lecture15.pdf

Object detection (or segmentation): comparing boxes (or masks)

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

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



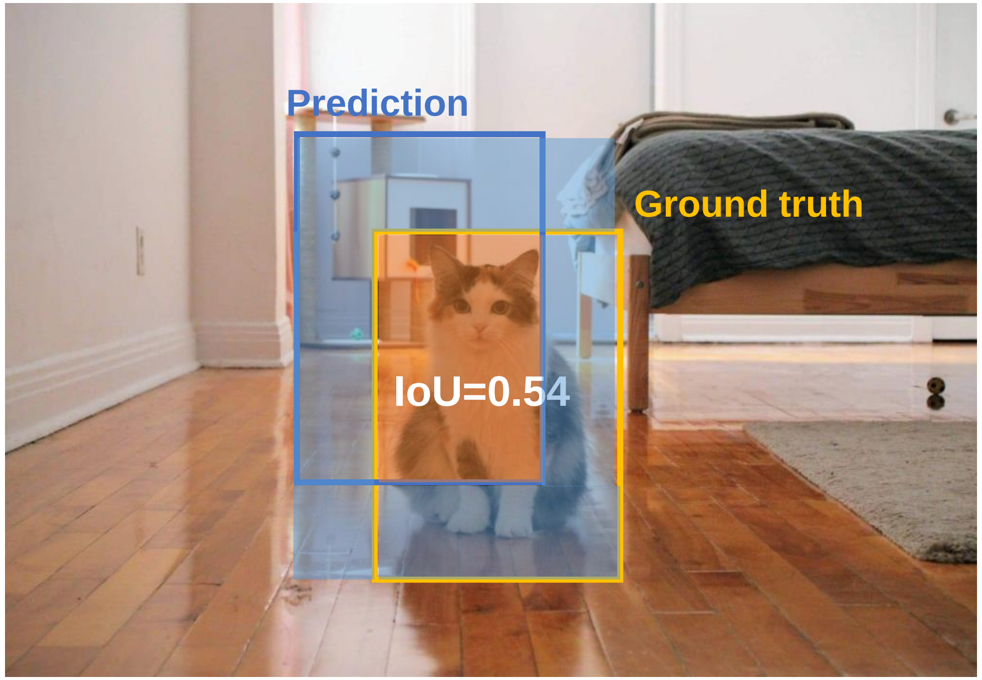
From: https://web.eecs.umich.edu/~justincj/slides/eecs498/498_FA2019_lecture15.pdf

Object detection (or segmentation): comparing boxes (or masks)

- 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”



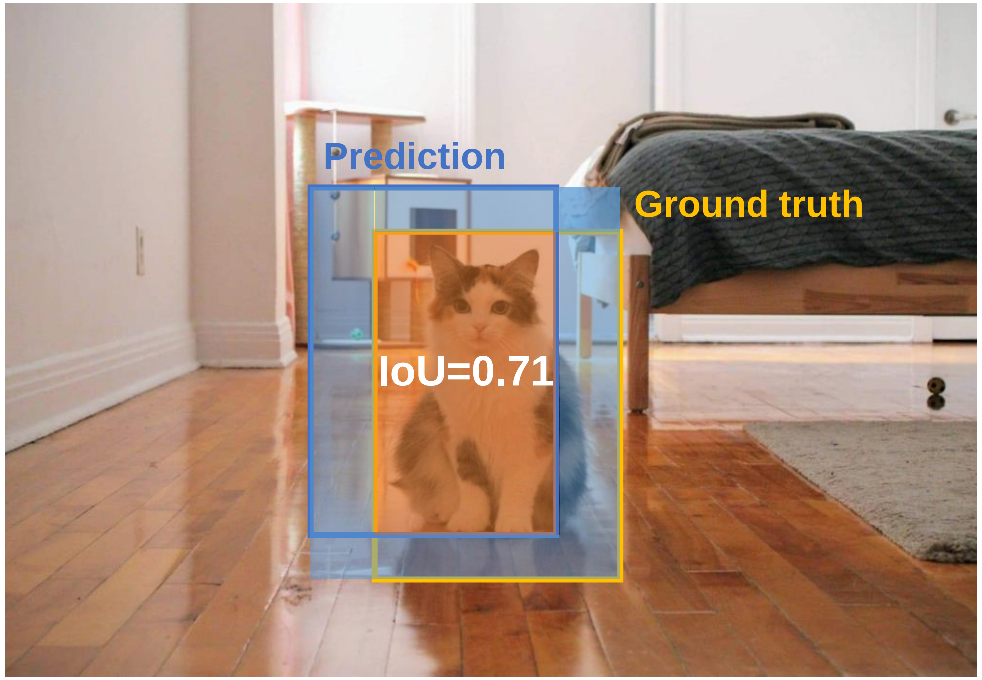
From: https://web.eecs.umich.edu/~justincj/slides/eecs498/498_FA2019_lecture15.pdf

Object detection (or segmentation): comparing boxes (or masks)

- 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"

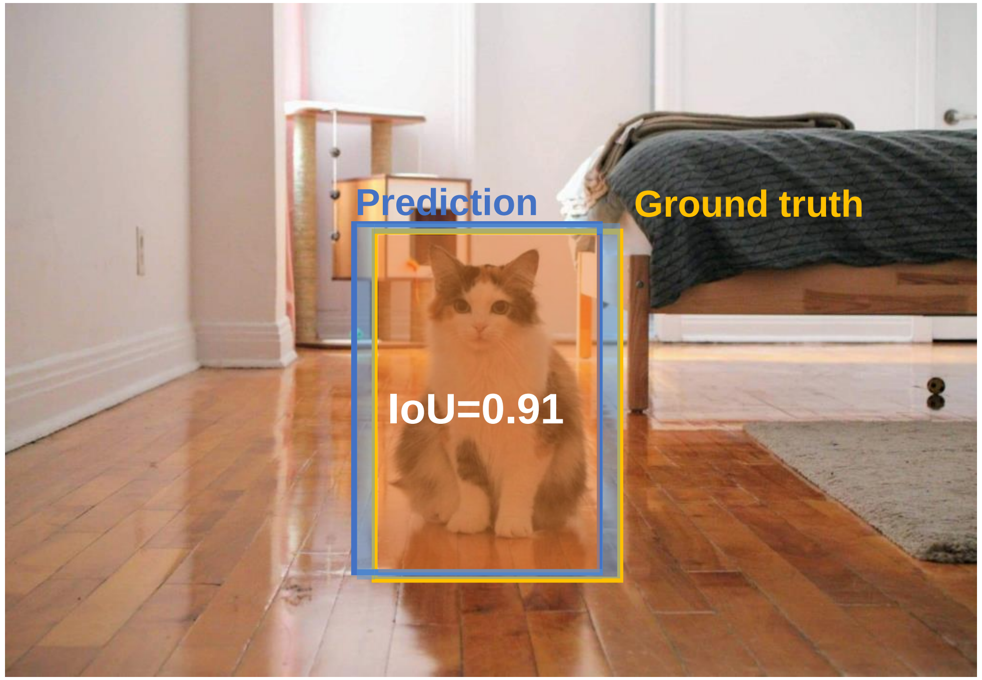


From: https://web.eecs.umich.edu/~justincj/slides/eecs498/498_FA2019_lecture15.pdf

Object detection (or segmentation): comparing boxes (or masks)

- 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”

- For segmentation masks this is done pixel-wise

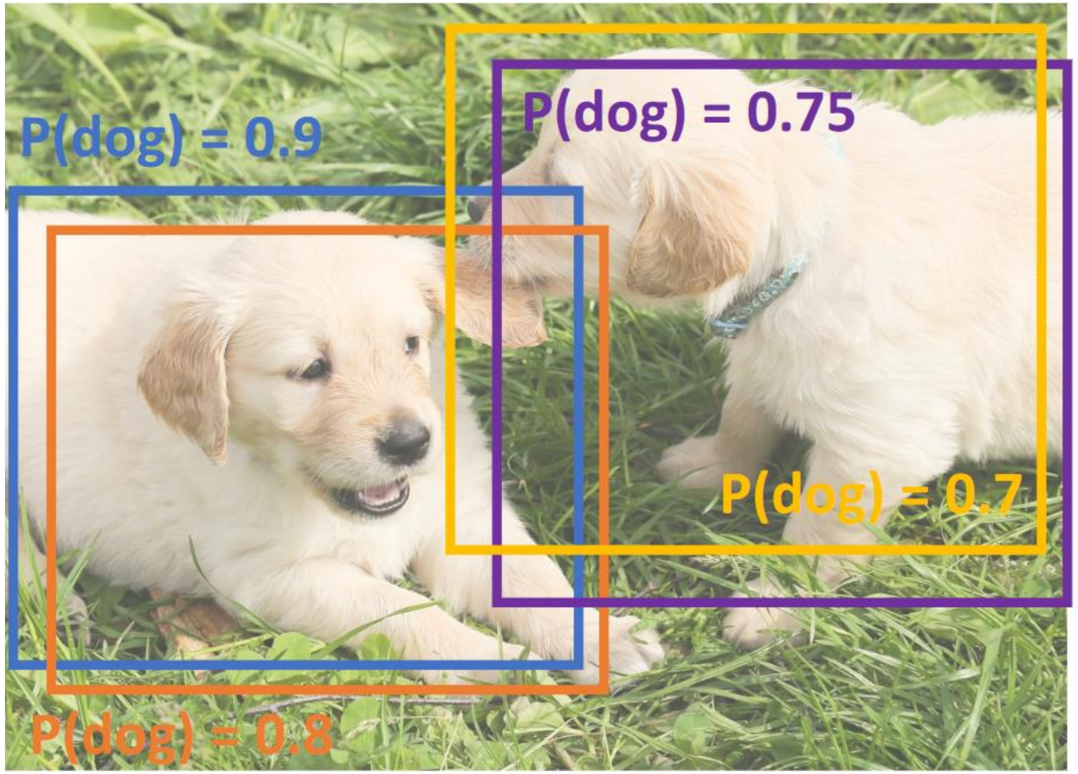
From: https://web.eecs.umich.edu/~justincj/slides/eecs498/498_FA2019_lecture15.pdf

Object detection: 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

Object detection: deal with overlapping boxes

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

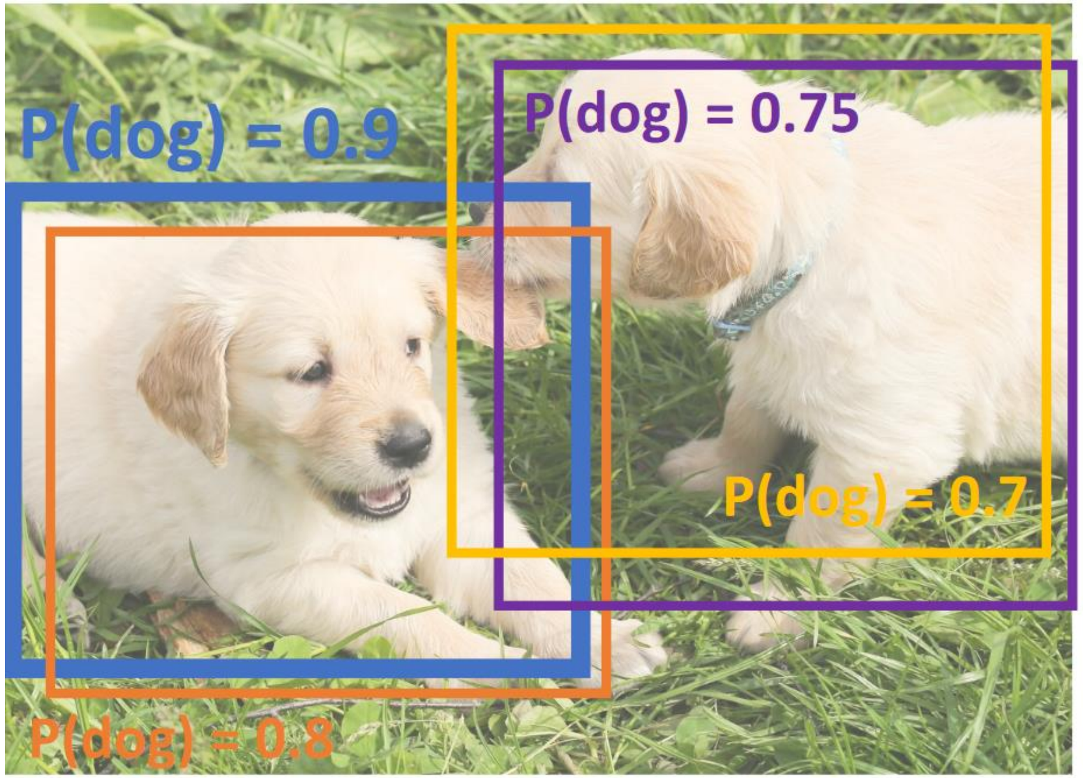
- **Solution**: post-process raw detections using **Non-Max Suppression (NMS)**

- <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

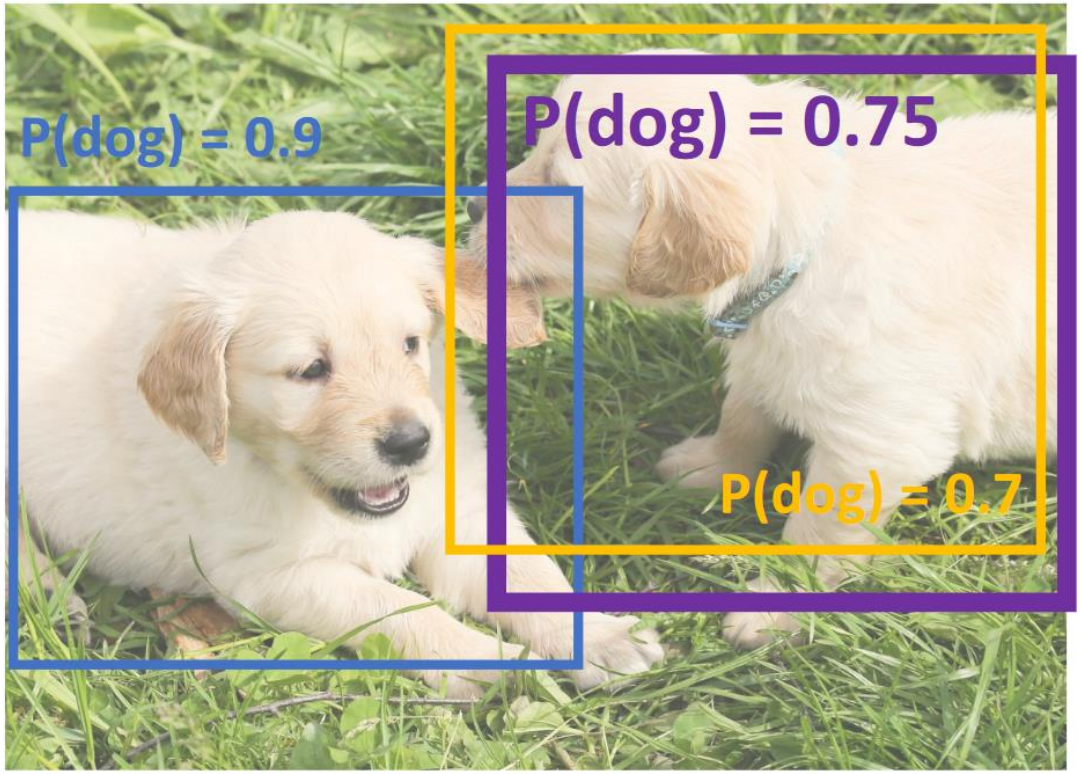
Object detection: deal with overlapping boxes

- **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(■ , ■) = 0.74



Puppy image is CC0 Public Domain

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

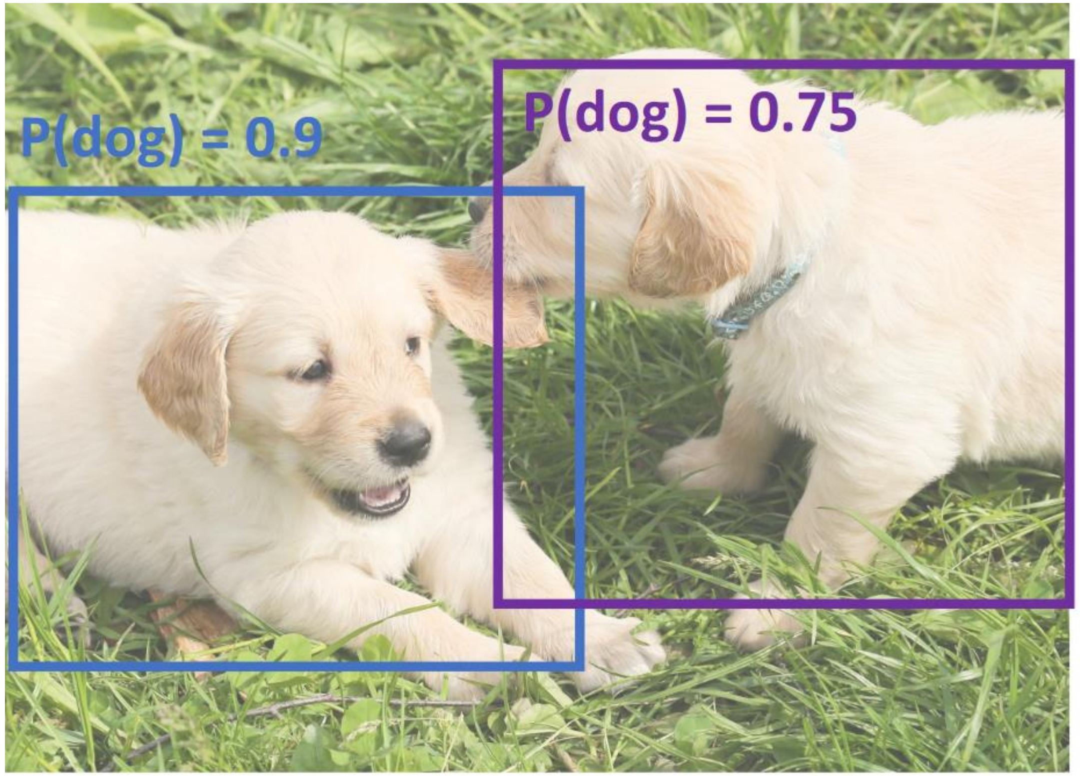
Object detection: deal with overlapping boxes

- **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

- **Problem**: NMS may eliminate "good" boxes when objects are highly overlapping…

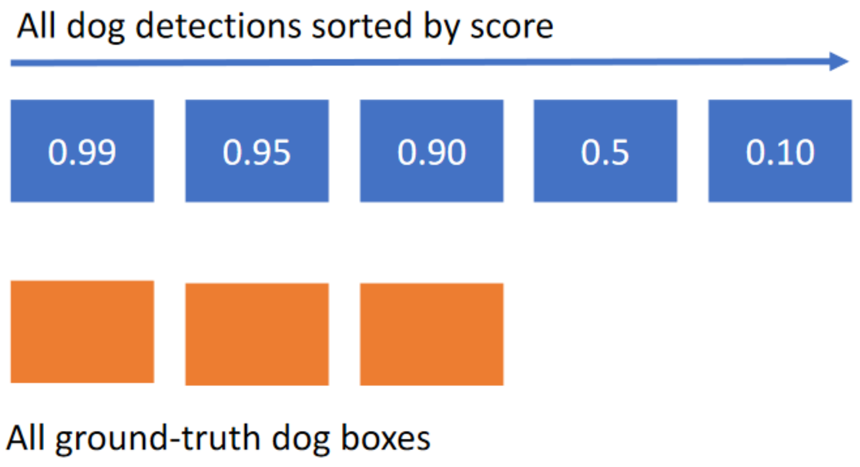


Puppy image is CC0 Public Domain

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

Evaluating object detectors – Mean Average Precision (mAP)

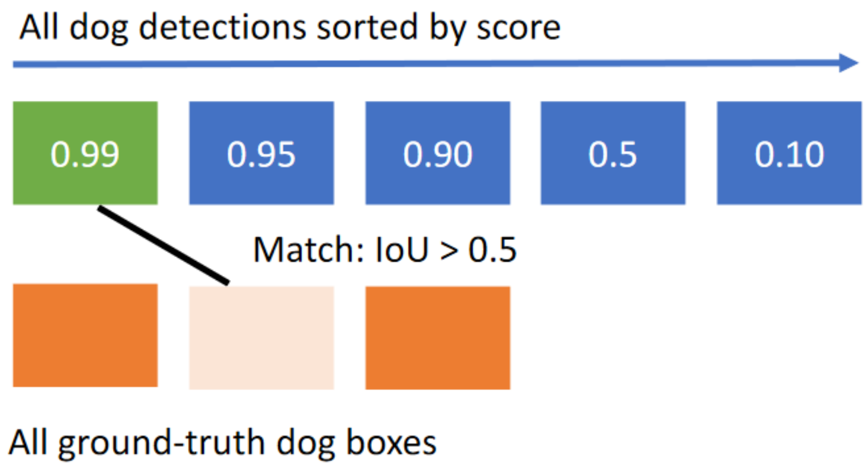
1. Run object detector on all test images (with NMS)
2. For each category, compute Average Precision (AP) = Area under Precision vs Recall Curve
    1. For each detection (highest score to lowest score)



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

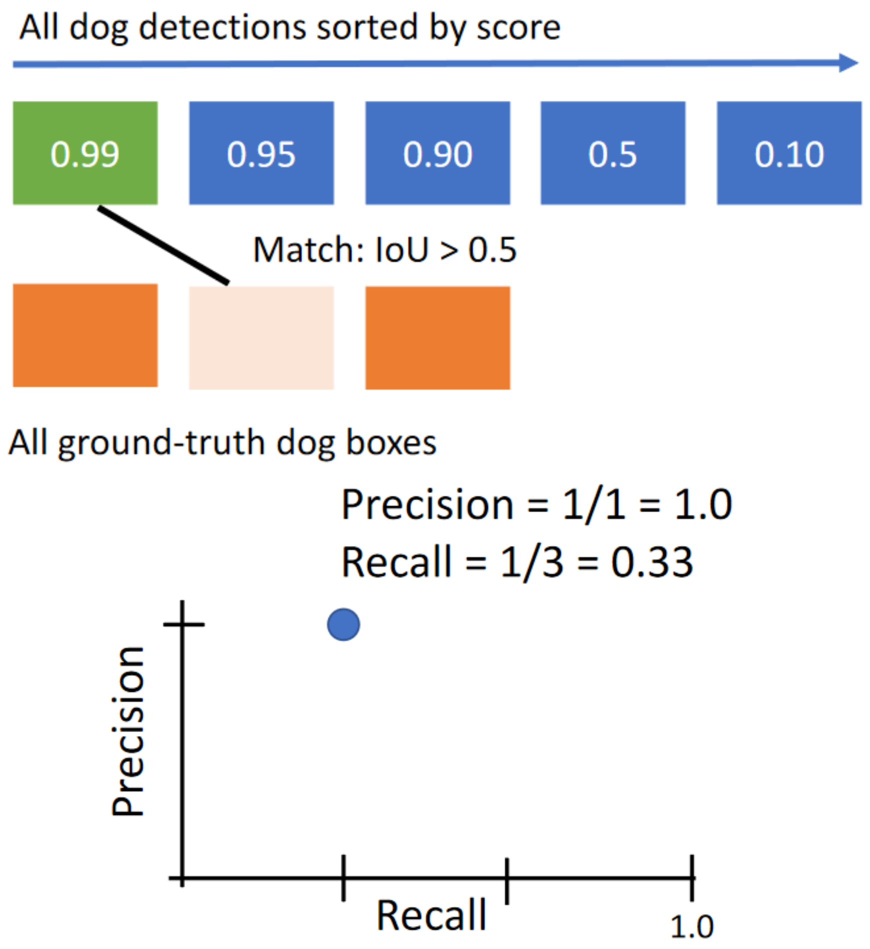


Object detection and segmentation in deep learning
08/02/2023

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

Evaluating object detectors – Mean Average Precision (mAP)

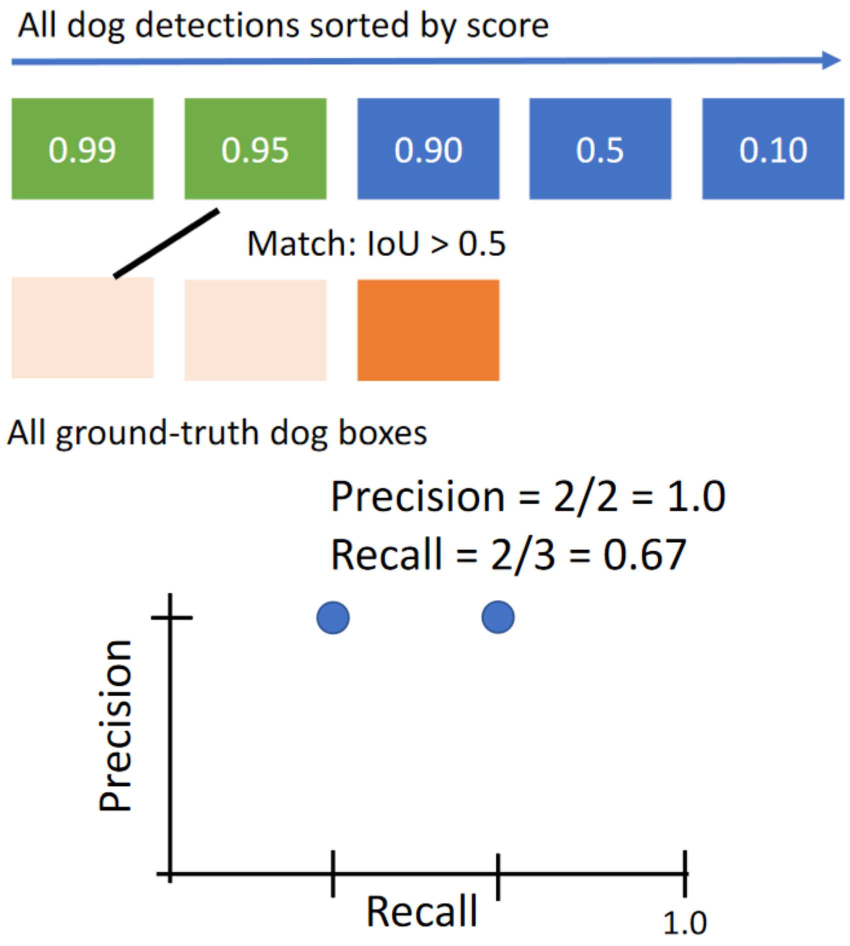
1. Run object detector on all test images (with NMS)
2. For each category, compute Average Precision (AP) = Area under Precision vs Recall Curve
    1. For each detection (highest score to lowest score)
        1. If it matches some GT box with IoU > 0.5, mark it as positive and eliminate the GT
   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

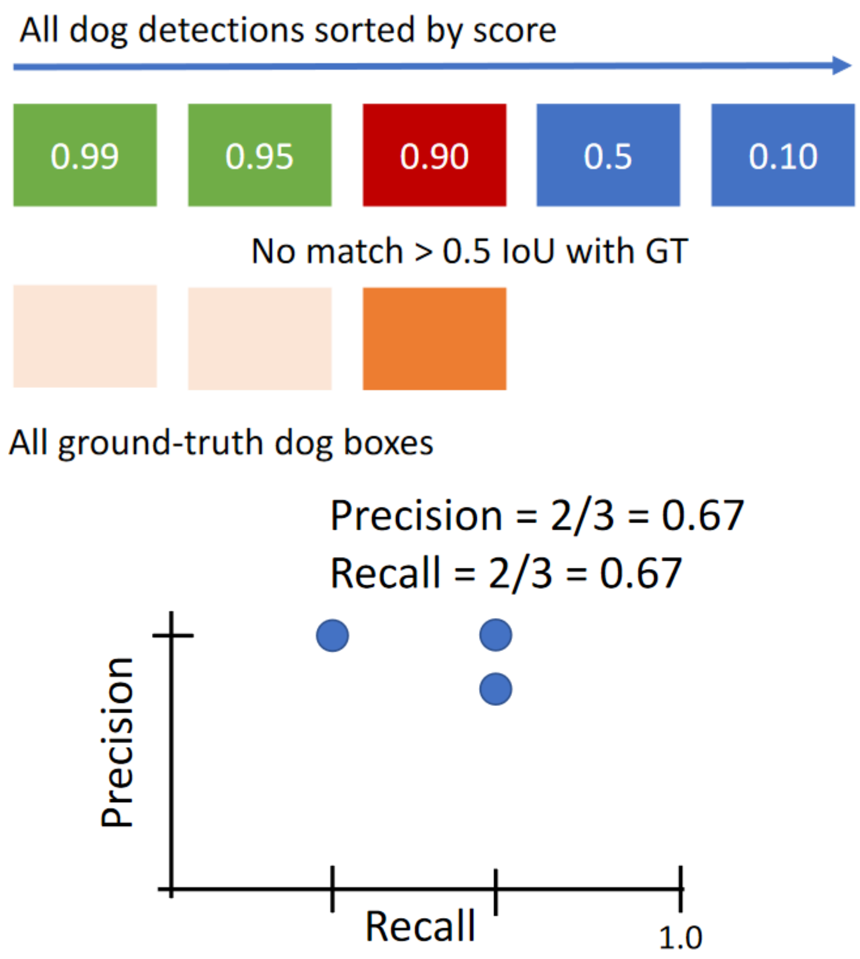
1. Run object detector on all test images (with NMS)
2. For each category, compute Average Precision (AP) = Area under Precision vs Recall Curve
    1. For each detection (highest score to lowest score)
        1. If it matches some GT box with IoU > 0.5, mark it as positive and eliminate the GT
   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

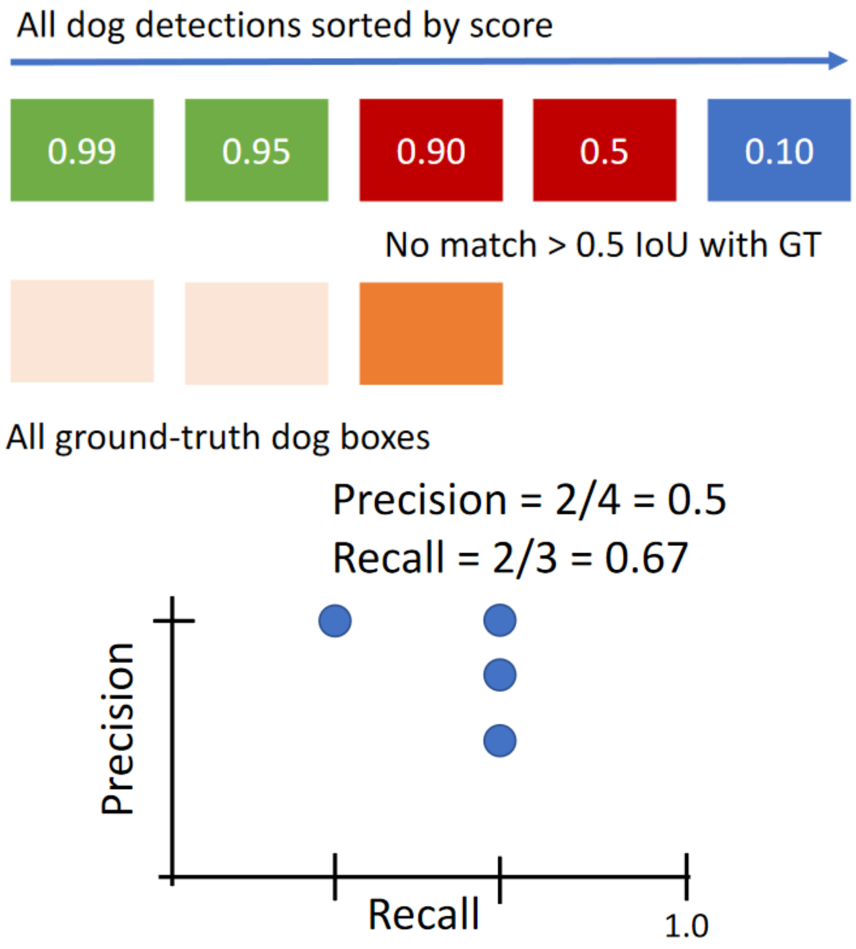
1. Run object detector on all test images (with NMS)
2. For each category, compute Average Precision (AP) = Area under Precision vs Recall Curve
   1. For each detection (highest score to lowest score)
      1. If it matches some GT box with IoU > 0.5, mark it as positive and eliminate the GT
      2.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

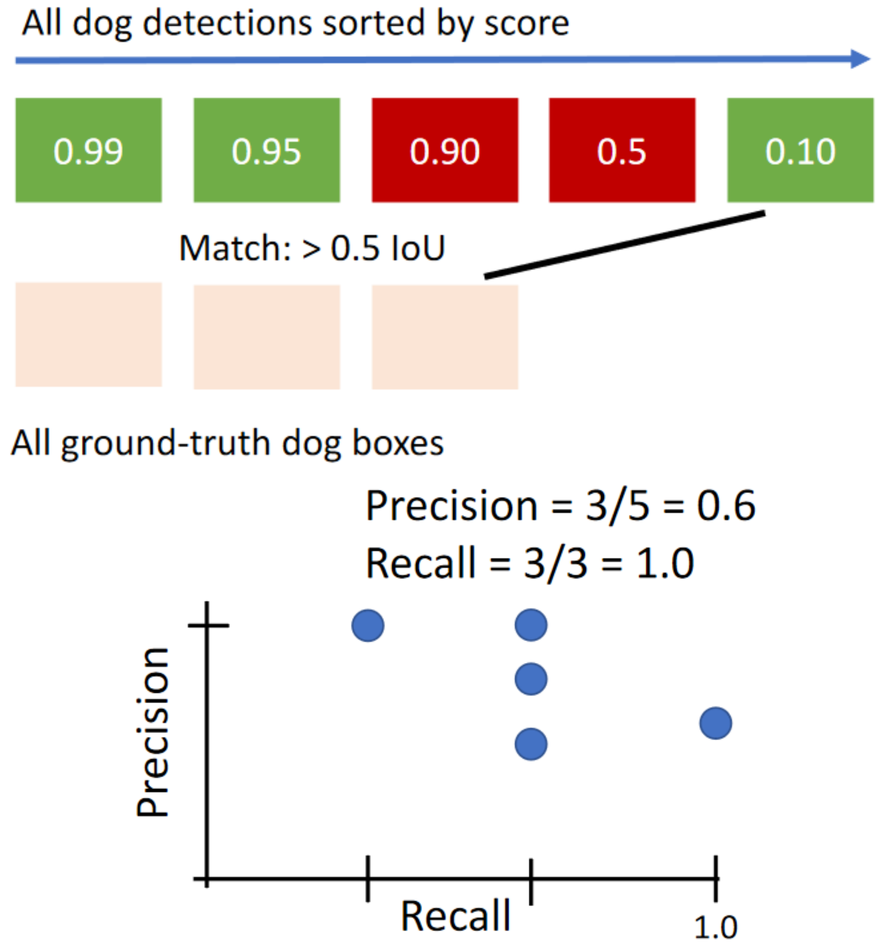
1. Run object detector on all test images (with NMS)
2. For each category, compute Average Precision (AP) = Area under Precision vs Recall Curve
    1. For each detection (highest score to lowest score)
        1. If it matches some GT box with IoU > 0.5, mark it as positive and eliminate the GT
   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