Topics:

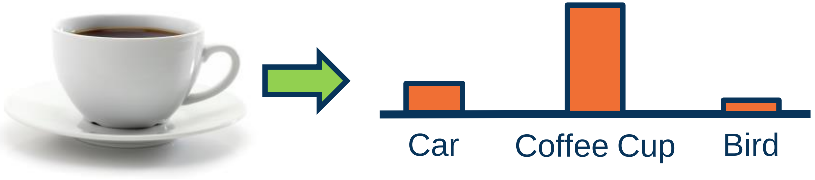
- Advanced Architectures: Segmentation and Detection

CS 4644-DL / 7643-A
ZSOLT KIRA

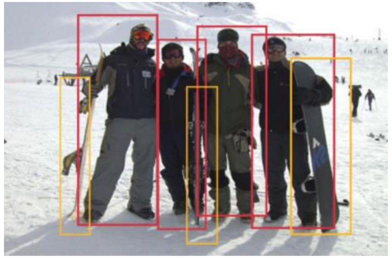
- **Assignment 3**
  - Due **March 8th 11:59pm EST**

- **Projects**
  - Project check-in due **March 14th**

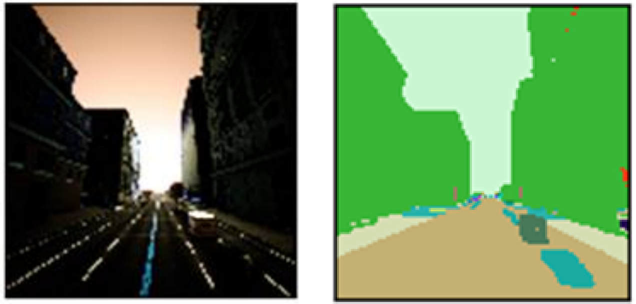
- Meta office hours Friday 3pm ET on attention models



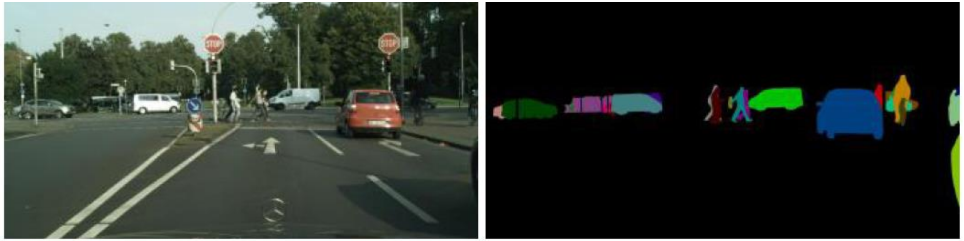
Classification
(Class distribution per image)



Object Detection
(List of bounding boxes with class distribution per box)



Semantic Segmentation
(Class distribution per pixel)

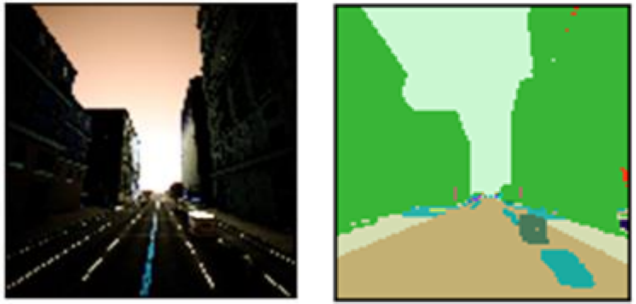


Instance Segmentation
(Class distribution per pixel with unique ID)

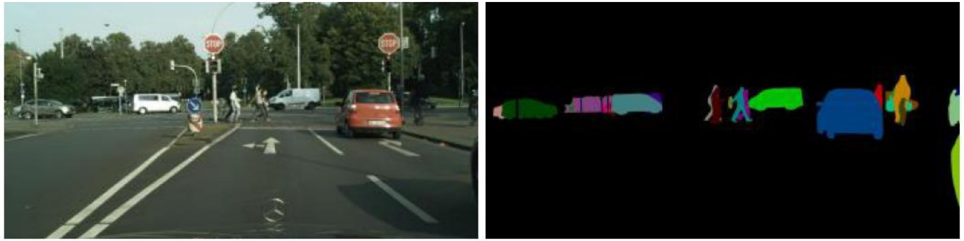
Computer Vision Tasks

Given an image, output another image

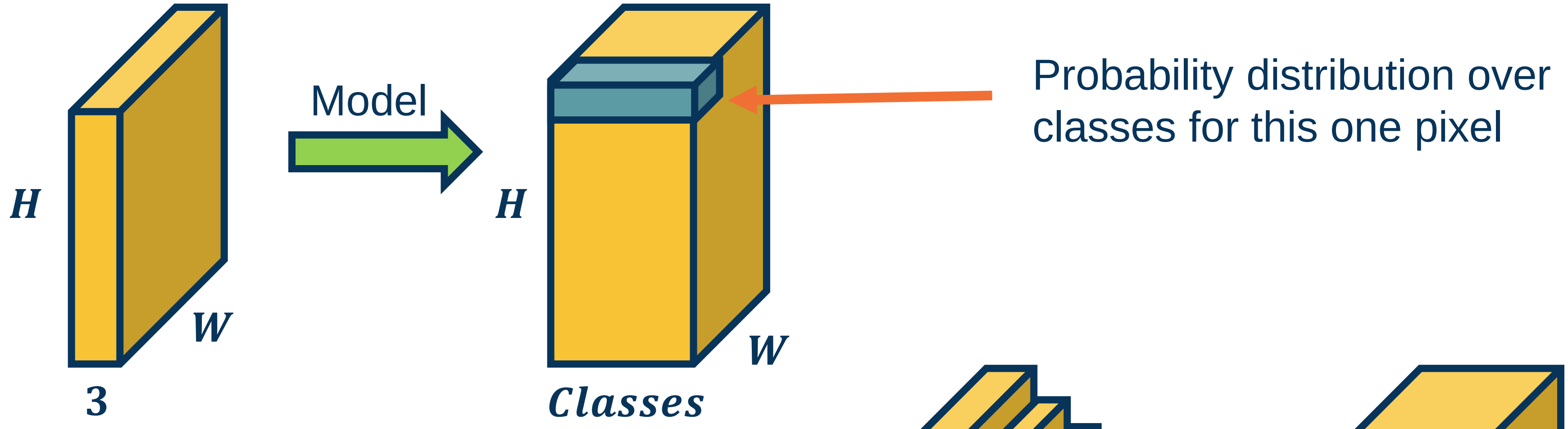
- Each output contains class distribution per pixel
- More generally an image-to-image problem

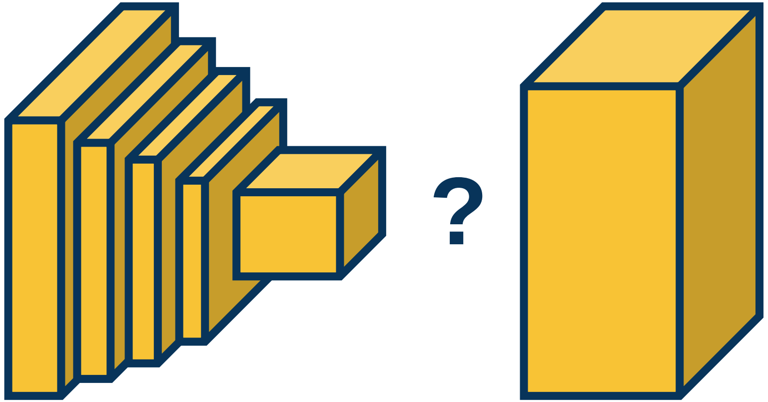


Semantic Segmentation
(Class distribution per pixel)

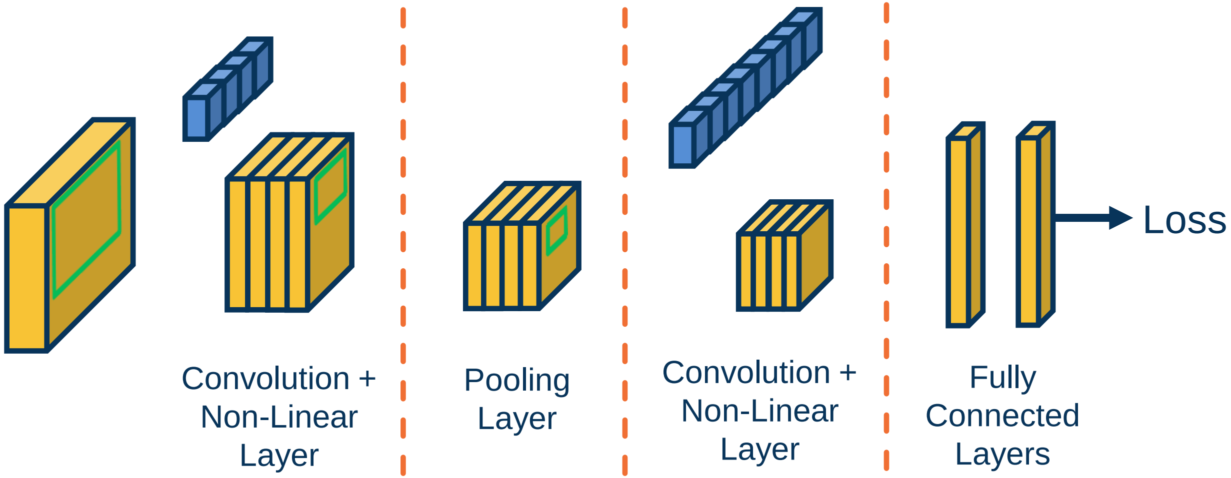


Instance Segmentation
(Class distribution per pixel with unique ID)





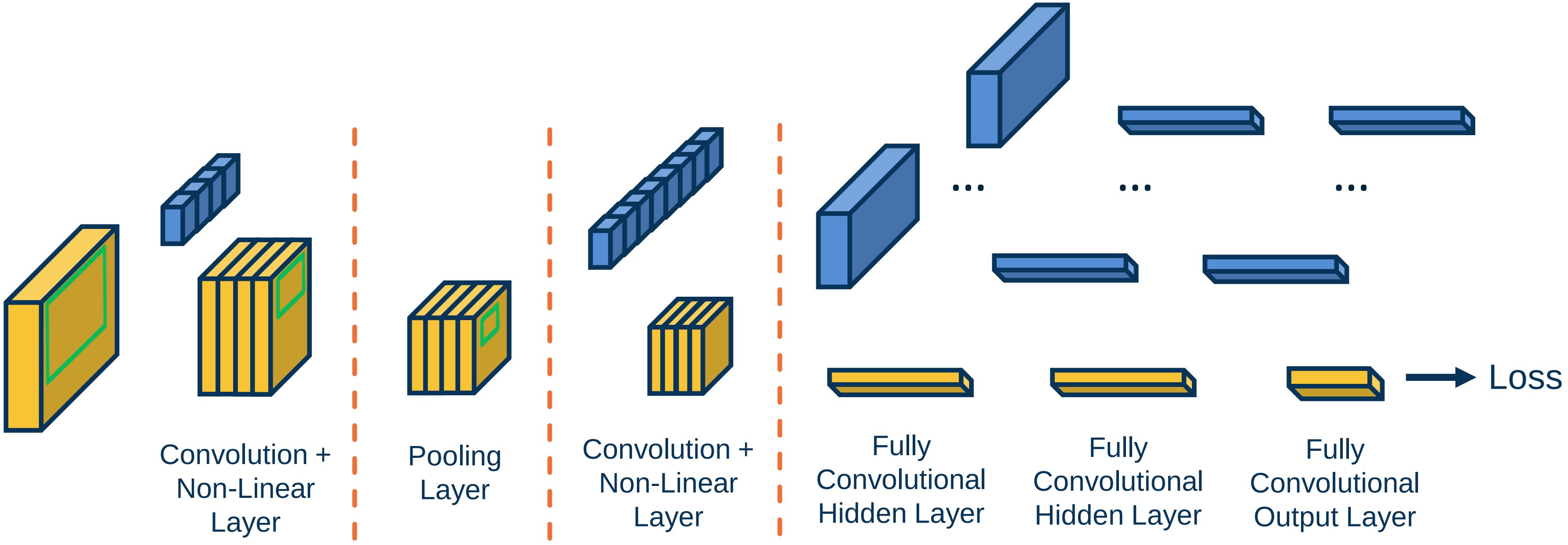
Input & Output



Fully connected layers no longer explicitly retain spatial information (though the network can still learn to do so)

Idea: Convert fully connected layer to convolution!

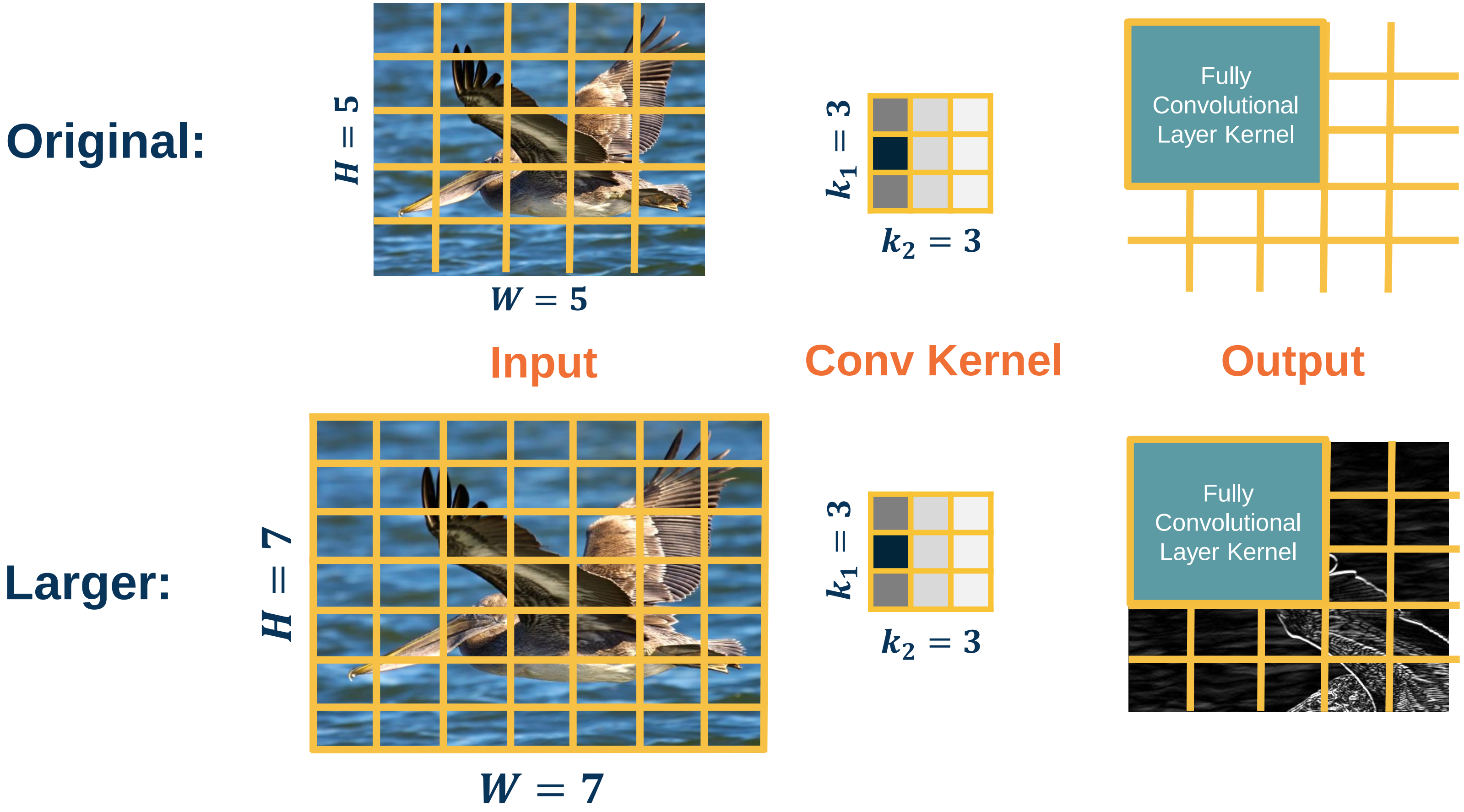
Idea 1: Fully-Convolutional Network



Each kernel has the size of entire input! (output is 1 scalar)

- This is equivalent to Wx+b!
- We have one kernel per output node

Converting FC Layers to Conv Layers



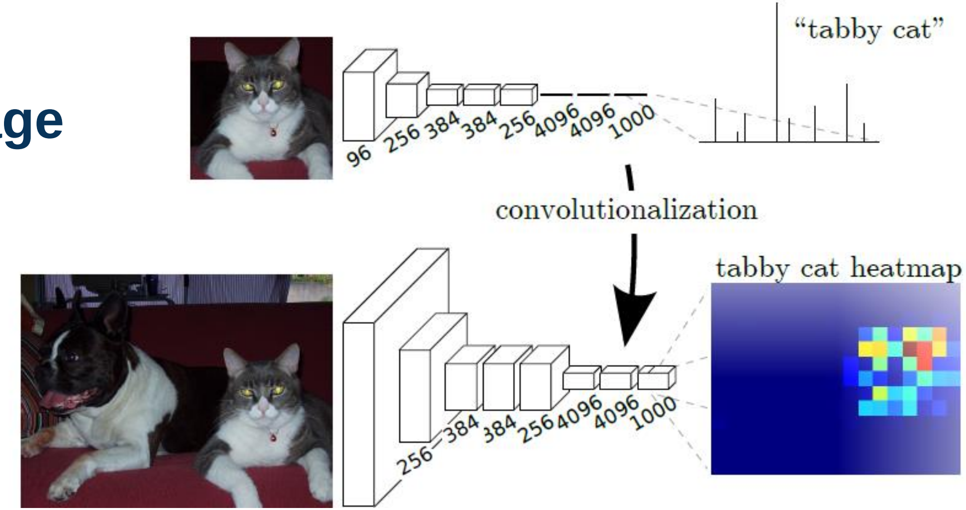
Same Kernel, Larger Input

Why does this matter?

- We can stride the “fully connected” classifier across larger inputs!
- Convolutions work on arbitrary input sizes (because of striding)

Original sized image

Larger Image

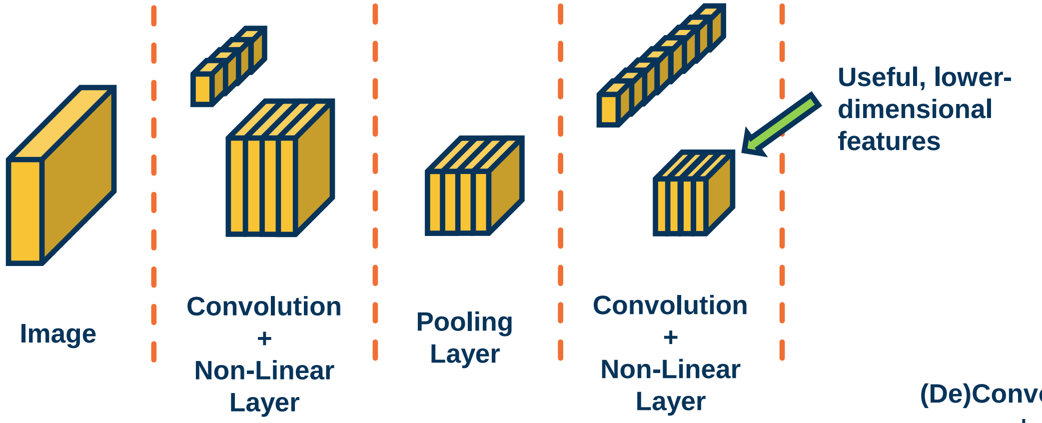


Larger Output Size!

Larger Output Maps

Long, et al., “Fully Convolutional Networks for Semantic Segmentation”, 2015

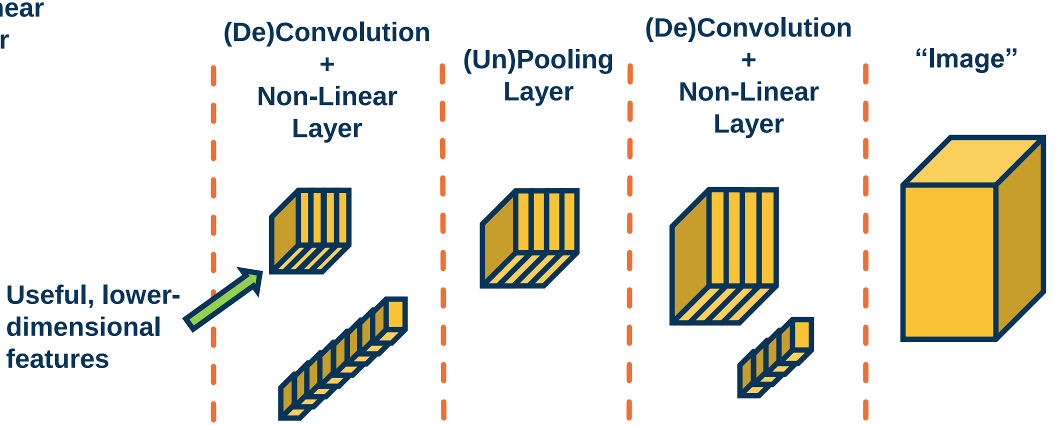
Convolutional Neural Network (CNN)



We can develop learnable or non-learnable upsampling layers!

Encoder

Decoder



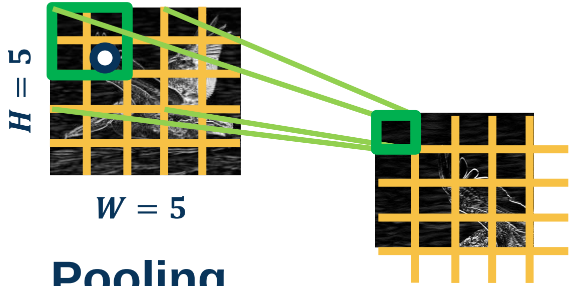
Idea 2: “De”Convolution and UnPooling

Georgia Tech

Example : Max pooling

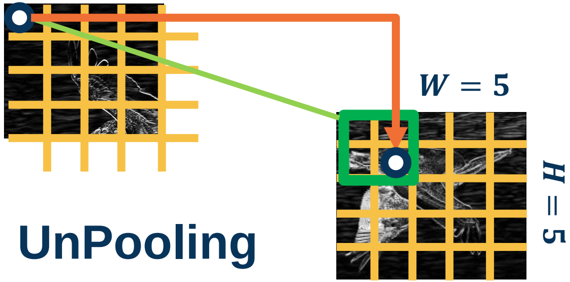
- Stride window across image but perform per-patch **max operation**

$$X(0{:}1, 0{:}1) = \begin{bmatrix} 100 & 150 \\ 100 & 200 \end{bmatrix} \Rightarrow \max(0{:}1,0{:}1) = 200$$



Pooling

Copy value to position chosen as max in encoder, fill reset of this window with zeros



UnPooling

Idea: Remember max elements in encoder! Copy value from equivalent position, rest are zeros

$$X = \begin{bmatrix} 120 & 150 & 120 \\ 100 & 50 & 110 \\ 25 & 25 & 10 \end{bmatrix} \xrightarrow[\text{2x2 max pool}]{} Y = \begin{bmatrix} 150 & 150 \\ 100 & 110 \end{bmatrix}$$

Encoder

Decoder

2x2 max unpool

$$X = \begin{bmatrix} 300 & 450 \\ 100 & 250 \end{bmatrix} \xrightarrow{} Y = \begin{bmatrix} 0 & 300 & - \\ 0 & 0 & - \\ - & - & - \end{bmatrix}$$

$$\mathbf{X}_{\mathbf{enc}} = \begin{bmatrix} 120 & 150 & 120 \\ 100 & 50 & 110 \\ 25 & 25 & 10 \end{bmatrix} \xrightarrow{\text{2x2 max pool}} \mathbf{Y}_{\mathbf{enc}} = \begin{bmatrix} 150 & 150 \\ 100 & 110 \end{bmatrix}$$

Contributions from multiple windows are summed

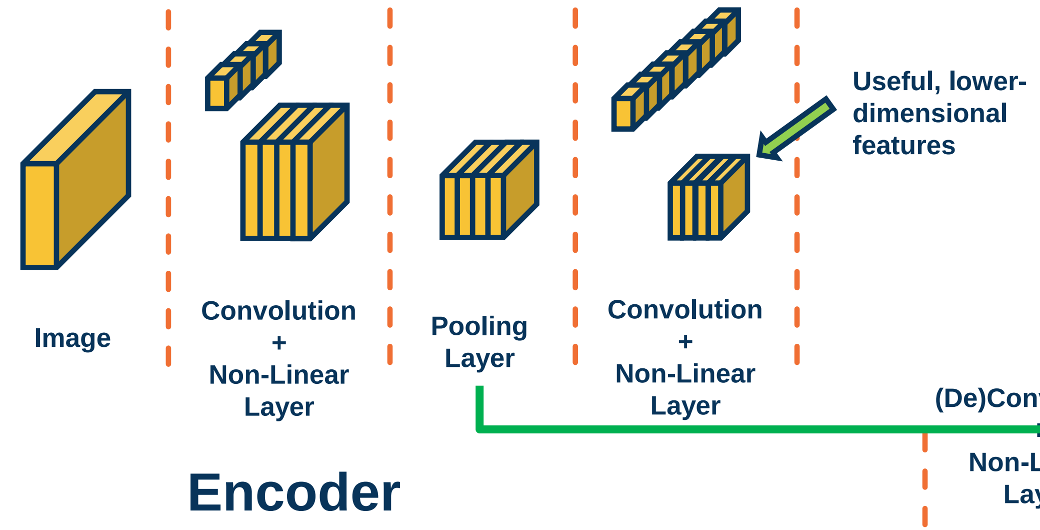
Encoder

Decoder

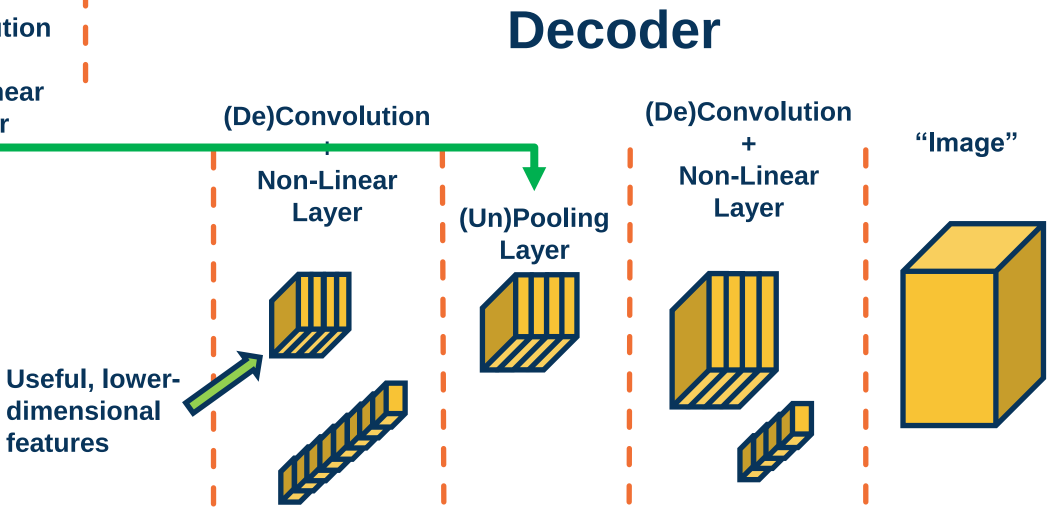
$$\mathbf{X}_{\mathbf{dec}} = \begin{bmatrix} 300 & 450 \\ 100 & 250 \end{bmatrix} \xrightarrow{\text{2x2 max unpool}} \mathbf{Y}_{\mathbf{dec}} = \begin{bmatrix} 0 & 300 + 450 & 0 \\ 100 & 0 & 250 \\ 0 & 0 & 0 \end{bmatrix}$$

Max Unpooling Example

Convolutional Neural Network (CNN)



We pull max indices from corresponding layers (requires symmetry in encoder/decoder)

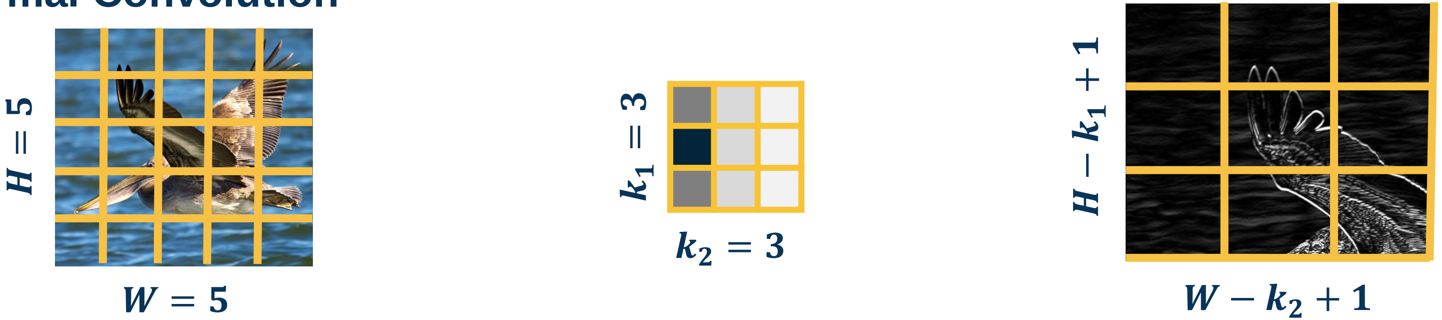


Symmetry in Encoder/Decoder

Georgia Tech

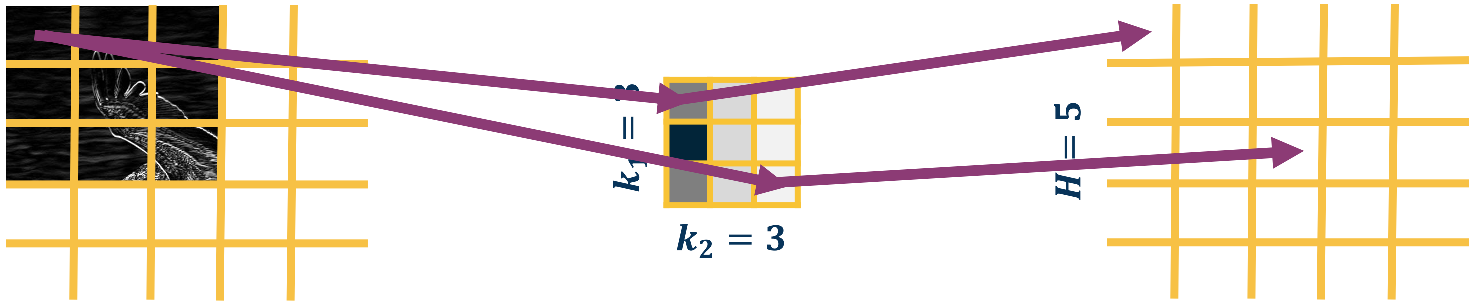
How can we *upsample* using convolutions and learnable kernel?

Normal Convolution



Transposed Convolution (also known as “deconvolution”, fractionally strided conv)

Idea: Take each input pixel, multiply by learnable kernel, “stamp” it on output



“De”Convolution (Transposed Convolution)

$$X = \begin{bmatrix} 120 & 150 & 120 \\ 100 & 50 & 110 \\ 25 & 25 & 10 \end{bmatrix} \qquad K = \begin{bmatrix} 1 & -1 \\ 2 & -2 \end{bmatrix}$$

Contributions from multiple windows are summed

$$\begin{bmatrix} 120 & -120 & 0 & 0 \\ 240 & -240 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

Incorporate
X(0,0)

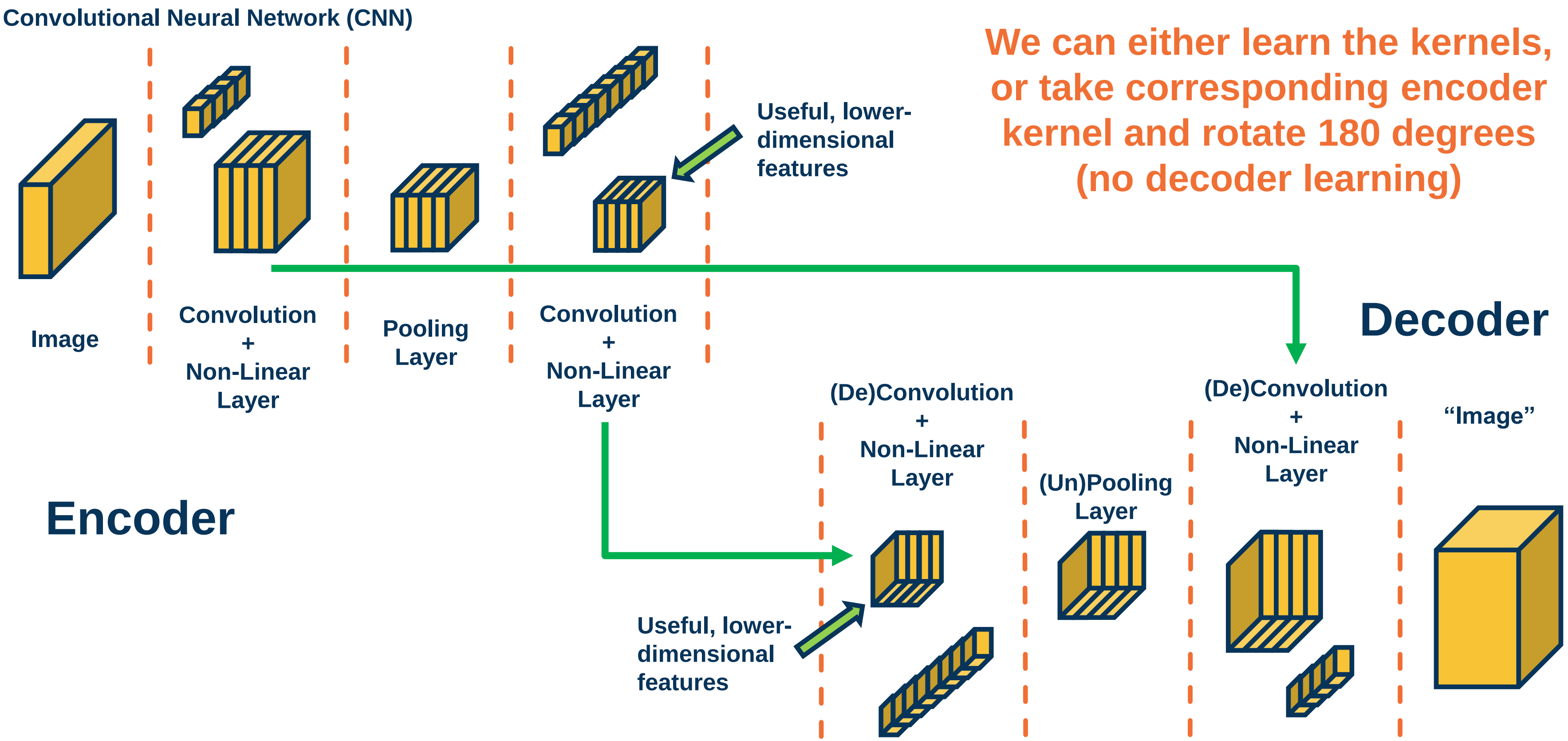
$$\begin{bmatrix} 120 & -120+150 & -150 & 0 \\ 240 & -240+300 & -300 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

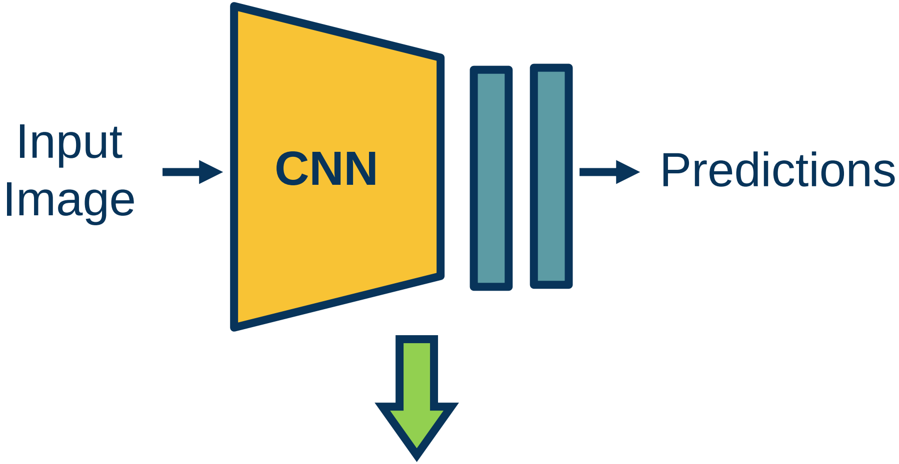
Incorporate
X(1,0)

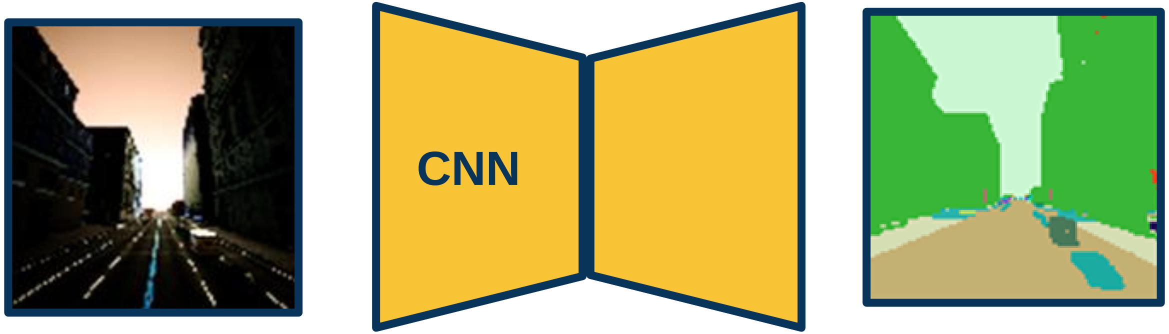
Transposed Convolution Example

Symmetry in Encoder/Decoder

We can either learn the kernels, or take corresponding encoder kernel and rotate 180 degrees (no decoder learning)



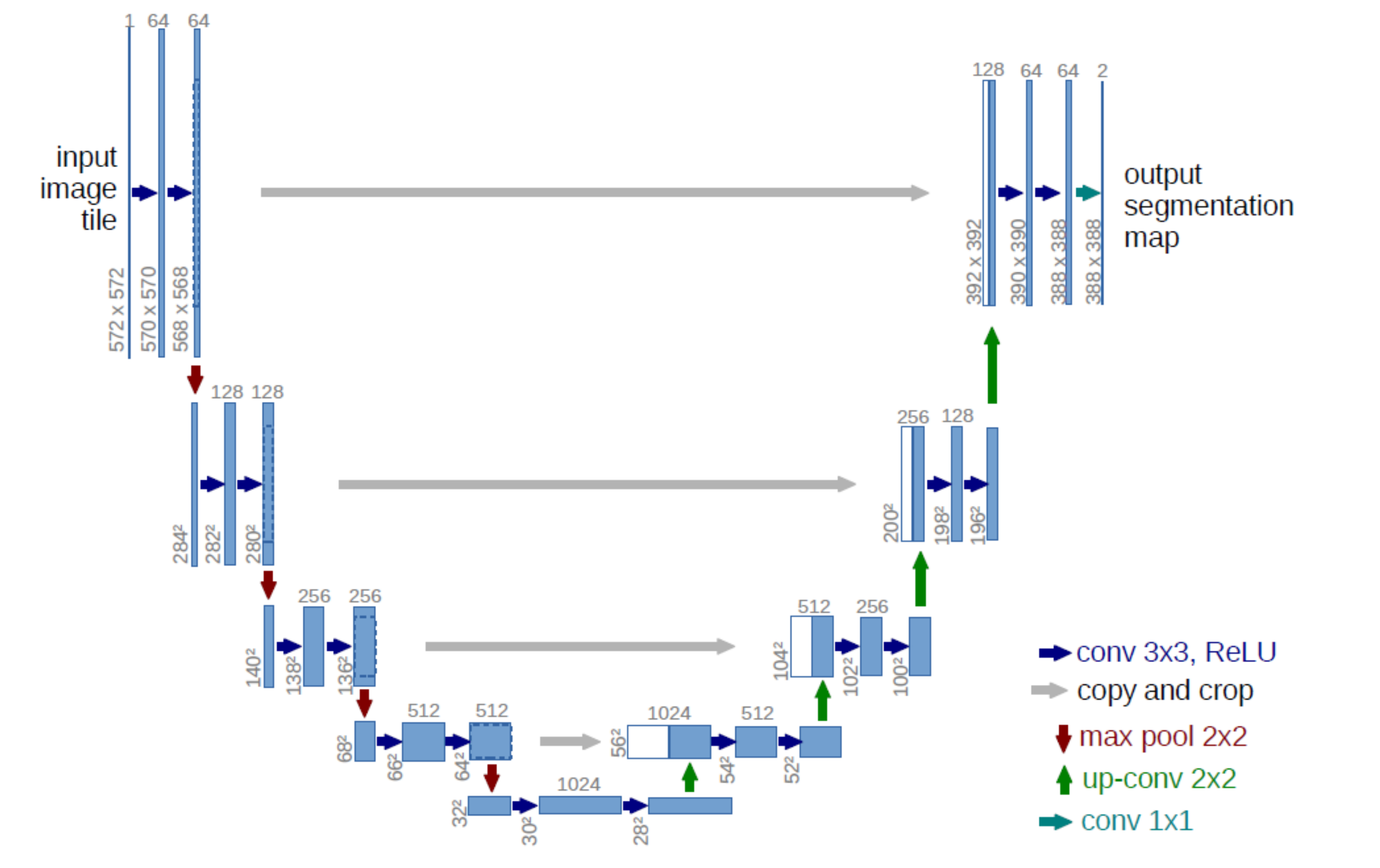




We can start with a pre-trained trunk/backbone (e.g. network pretrained on ImageNet)!

Transfer Learning

U-Net



You can have skip connections to bypass bottleneck!

Ronneberger, et al., "U-Net: Convolutional Networks for Biomedical Image Segmentation", 2015

Summary

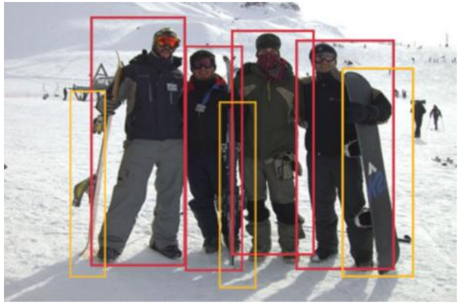
- Various ways to get **image-like outputs,** for example to predict segmentations of input images
- Fully convolutional layers essentially apply the striding idea to the output classifiers, supporting arbitrary input sizes
  - (without output size depending on what the input size is)
- We can have various upsampling layers that actually increase the size
- Encoder/decoder architectures are popular ways to leverage these to perform general image-to-image tasks

Single-Stage Object Detection

Given an image, output a list of bounding boxes with probability distribution over classes per box

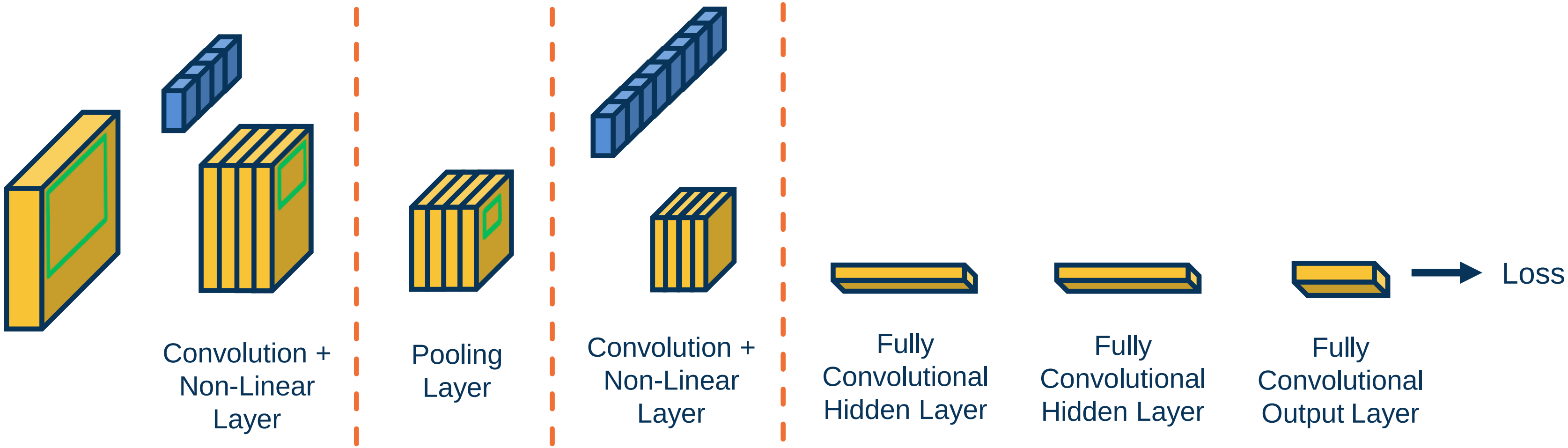
Problems:

- Variable number of boxes!
- Need to determine candidate regions (position and scale) first



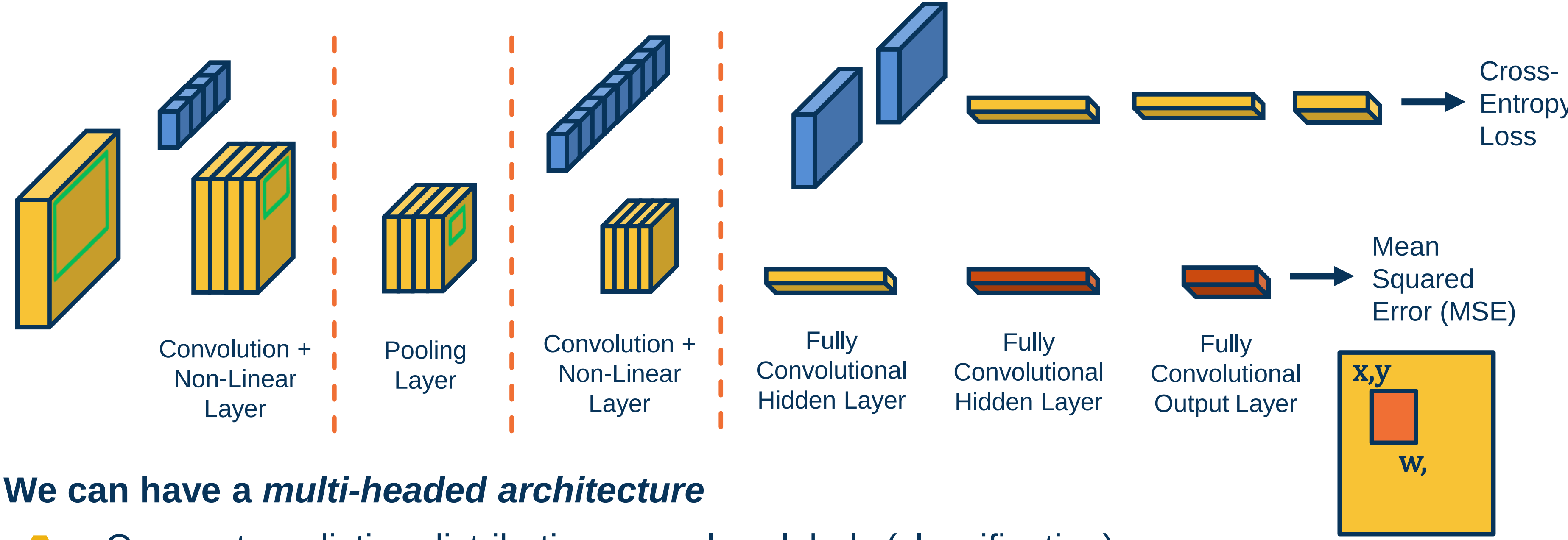
Object Detection

(List of bounding boxes with class distribution per box)



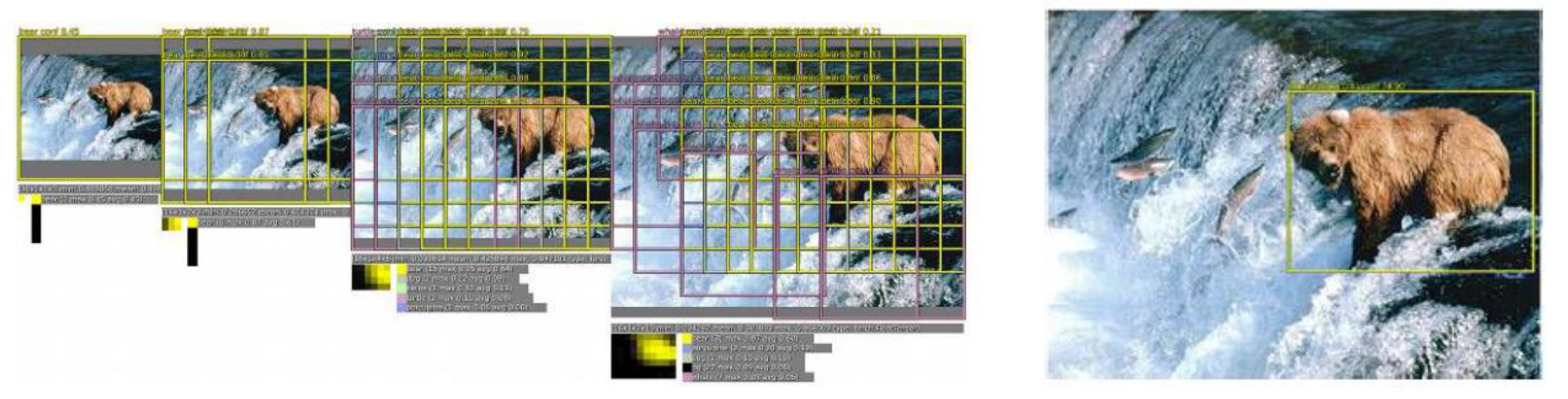
We can use the same idea of fully-convolutional networks

- Use ImageNet pre-trained model as backbone (e.g. taking in 224x224 image)
- Feed in larger image and get classifications for different windows in image



We can have a *multi-headed architecture*

- One part predicting distribution over class labels (classification)
- One part predicting a bounding box for each image region (regression)
  - Refinement to fit the object better (outputs 4 numbers)
- Both heads ***share features***! Jointly optimized (summing gradients)



Can also do this at multiple scales to result in a large number of detections

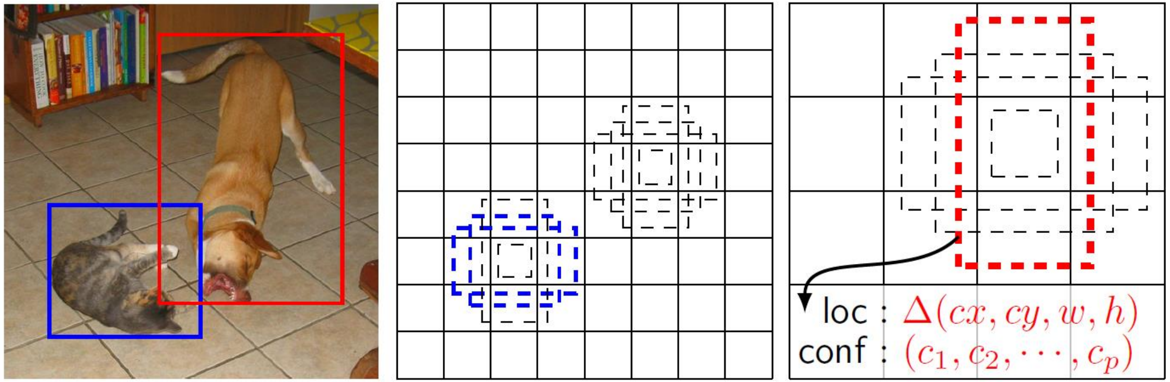
- Various tricks used to increase the resolution (decrease subsampling ratio)
- Redundant boxes are combined through **Non-Maximal Suppression (NMS)**

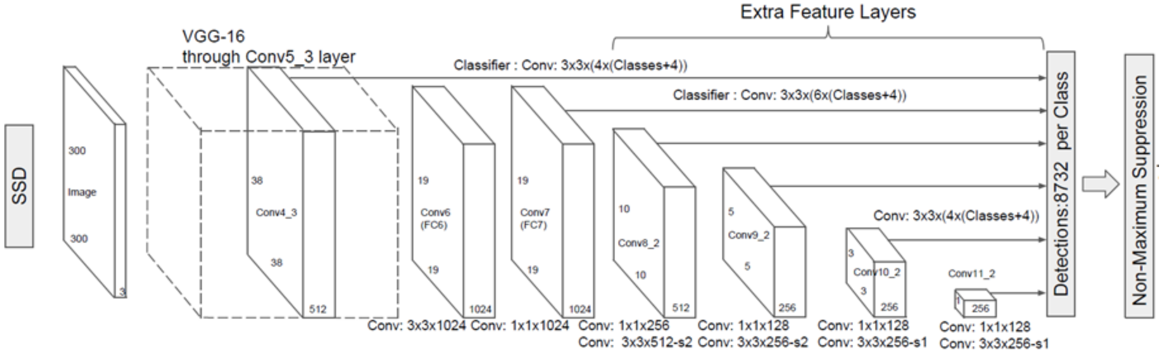
Sermanet, et al., “OverFeat: Integrated Recognition, Localization and Detection using Convolutional Networks”, 2013

Object Detection Tasks

Single-shot detectors use an idea of **grids** as anchors, with different scales and aspect ratios around them

- Various tricks used to increase the resolution (decrease subsampling ratio)

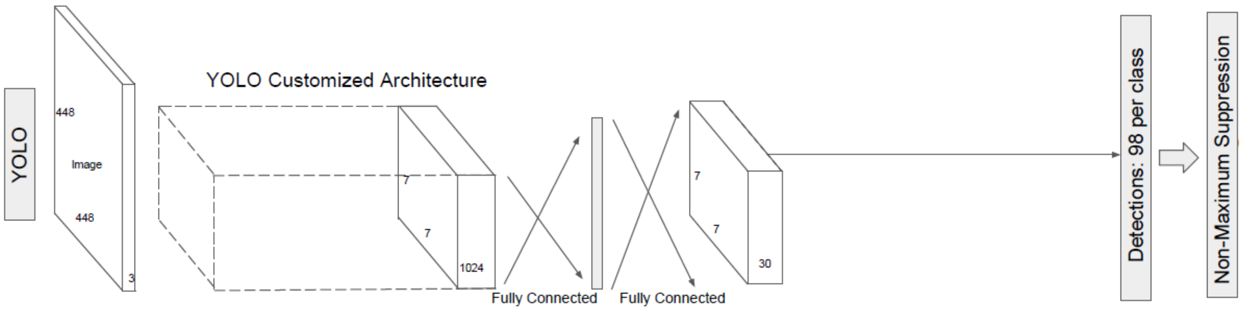


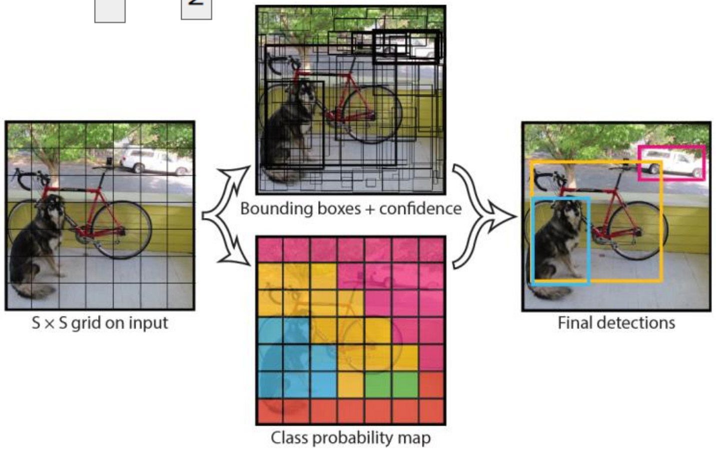


Liu, et al., "SSD: Single Shot MultiBox Detector", 2015

Single-Shot Detector (SSD)

Similar network architecture but single-scale (and hence faster for same size)



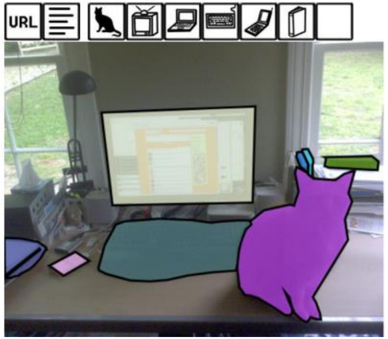


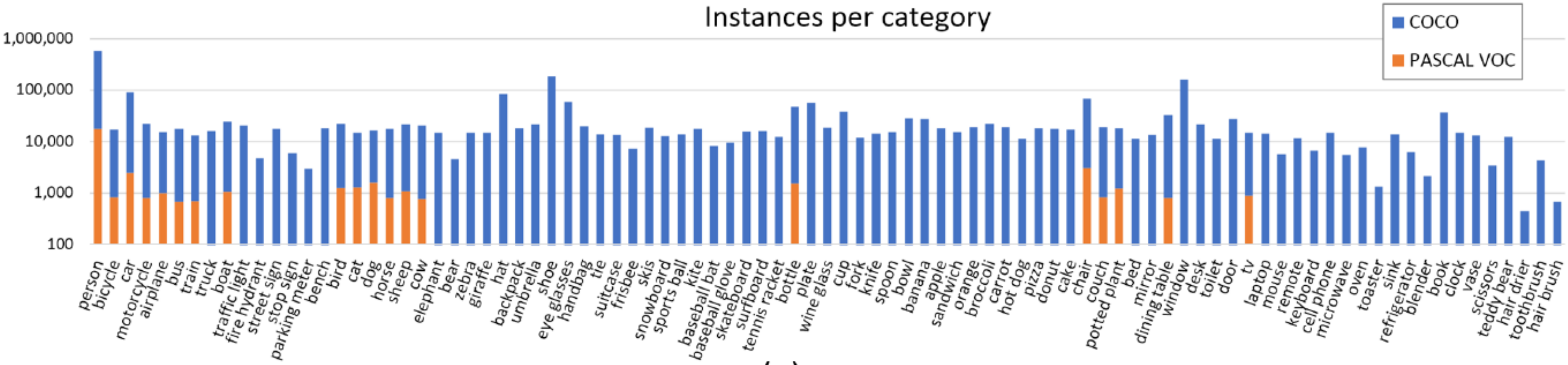
Redmon, et al., "You Only Look Once:Unified, Real-Time Object Detection", 2016

What is COCO?

COCO is a large-scale object detection, segmentation, and captioning dataset. COCO has several features:

- Object segmentation
- Recognition in context
- Superpixel stuff segmentation
- 330K images (>200K labeled)
- 1.5 million object instances
- 80 object categories





(a)

Categories per image

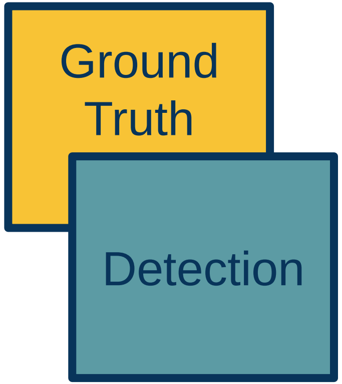
Instances per image

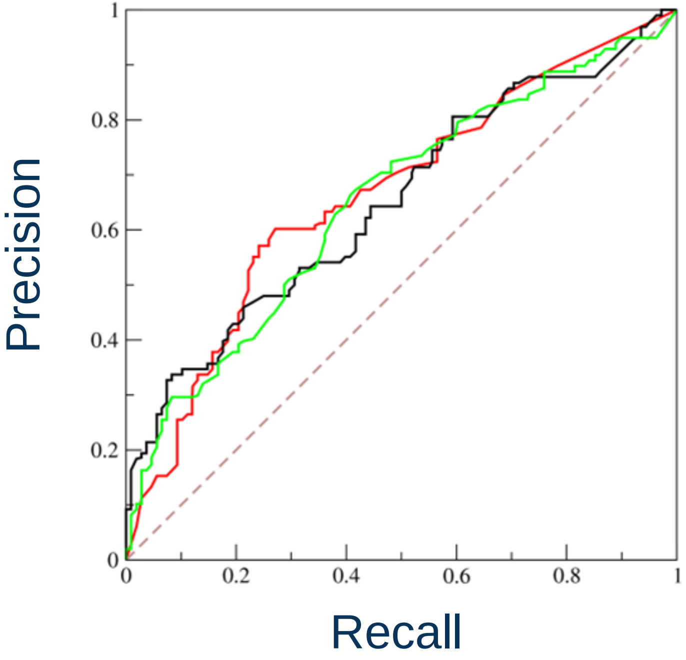
Lin, et al., "Microsoft COCO: Common Objects in Context", 2015. https://cocodataset.org/#explore

Datasets

Evaluation – Mean Average Precision (mAP)

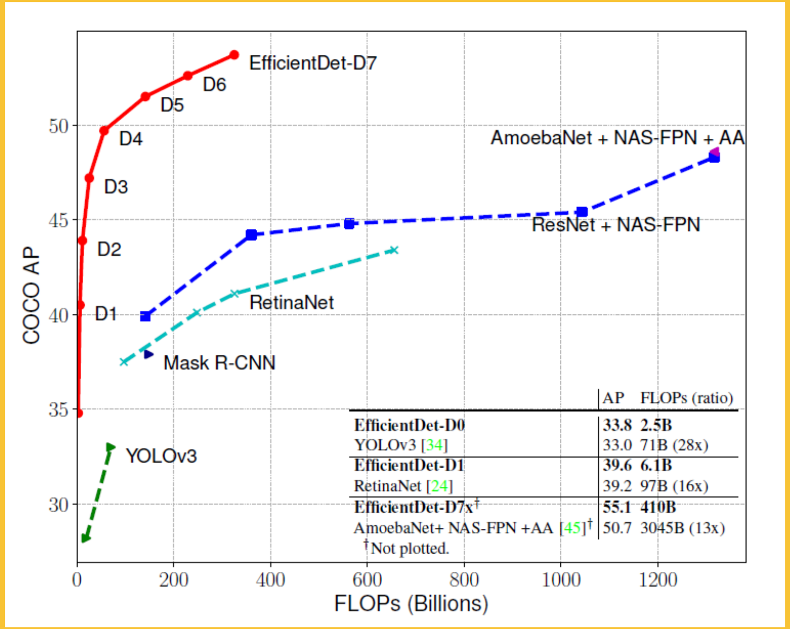
1. For each bounding box, calculate intersection over union (IoU)
2. Keep only those with IoU > threshold (e.g. 0.5)
3. Calculate precision/recall curve across classification probability threshold
4. Calculate **average precision (AP)** over recall of [0, 0.1, 0.2, …, 1.0]
5. Average over all categories to get mean Average Precision (mAP)



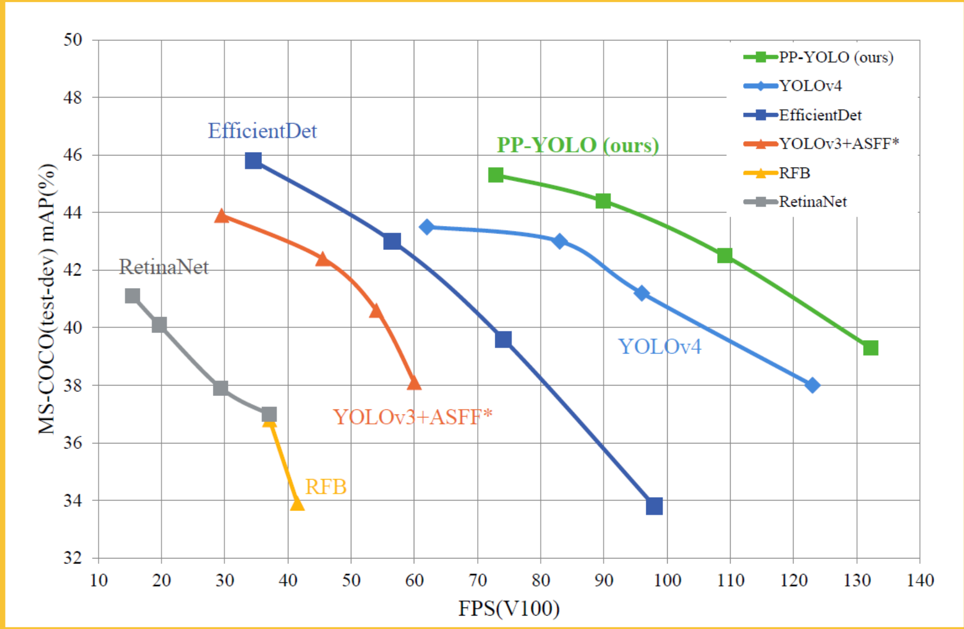


$$\boldsymbol{AP = \frac{1}{11} \sum_{i \in [0, 0.1, \ldots 1.0]} AP_i}$$

Results



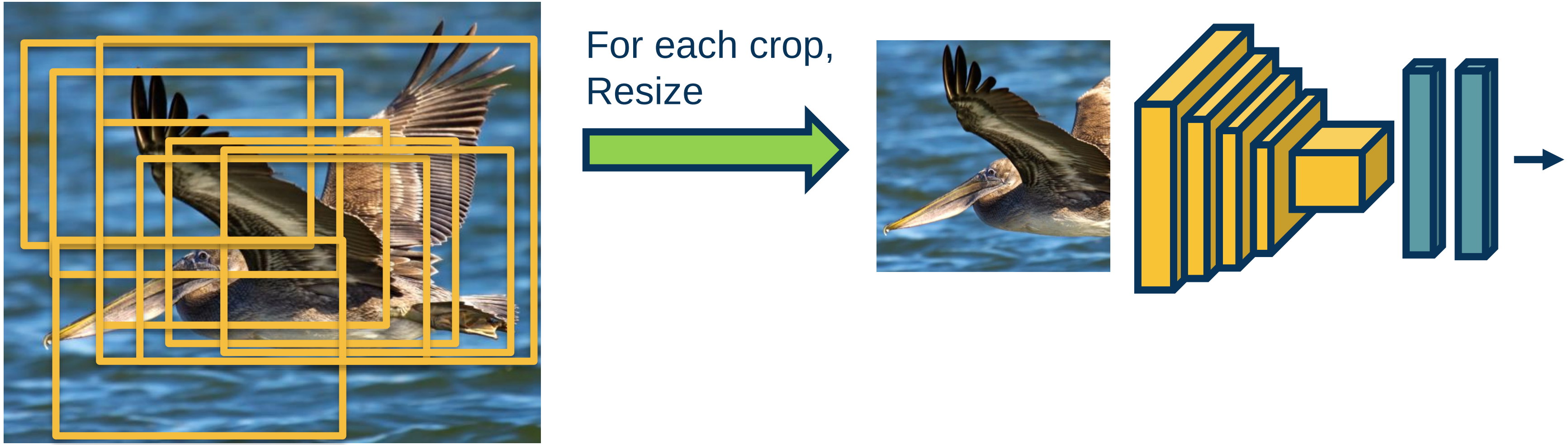
EfficientDet



PP-YOLO

Tan, et al., "EfficientDet: Scalable and Efficient Object Detection", 2020
Long et al., "PP-YOLO: An Effective and Efficient Implementation of Object Detector", 2020

Two-Stage Object Detectors



Instead of making dense predictions across an image, we can decompose the problem:

- Find regions of interest (ROIs) with object-like things
- Classifier those regions (and refine their bounding boxes)

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

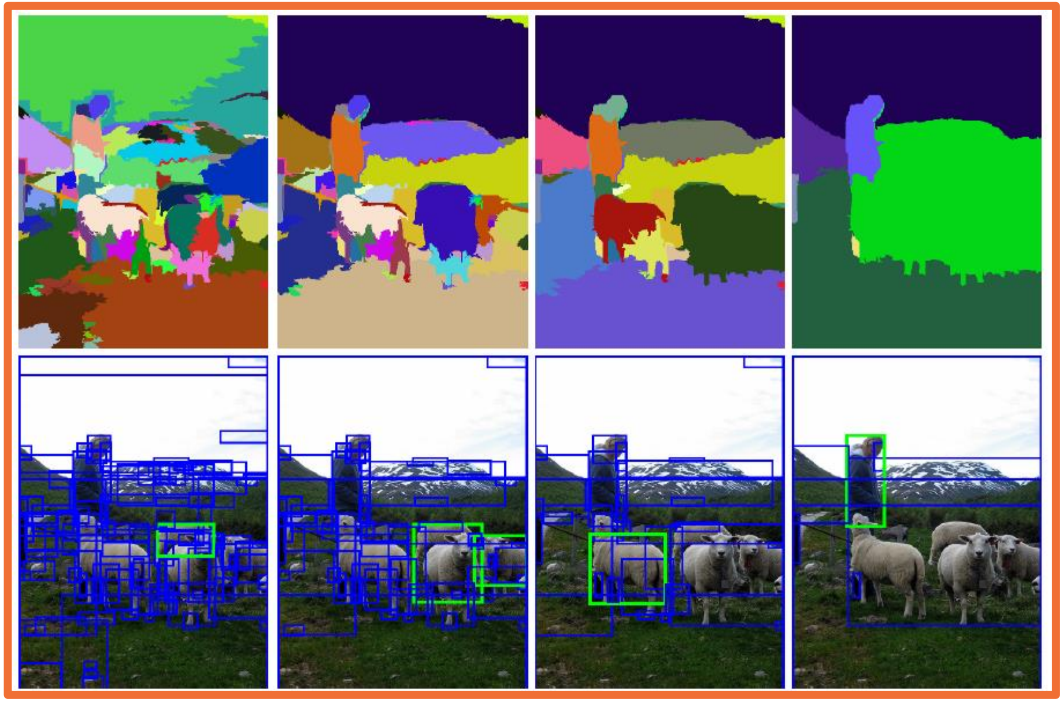
R-CNN

We can use **unsupervised (non-learned!) algorithms** for finding candidates

Downsides:

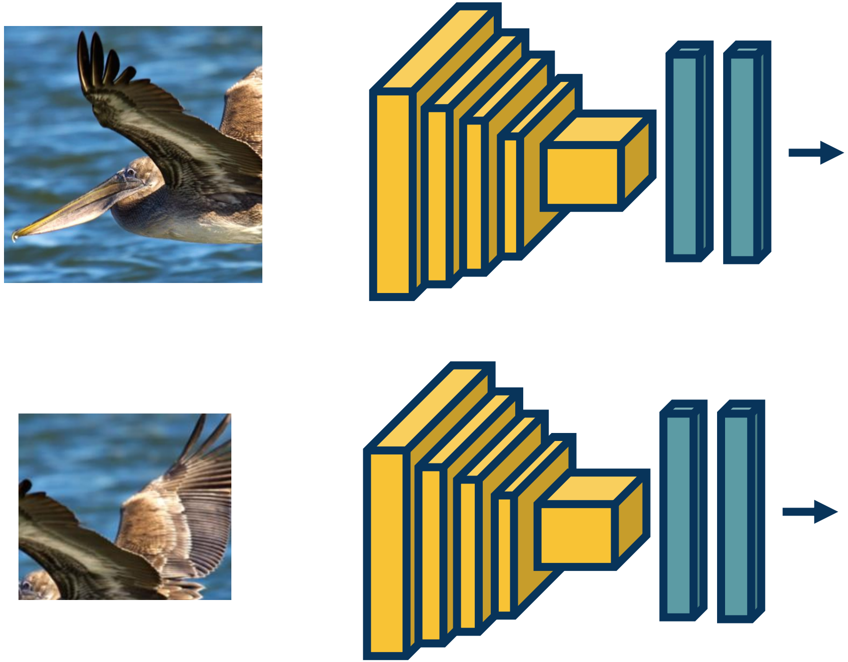
- Takes 1+ second per image
- Return thousands of (mostly background) boxes

Resize each candidate to full input size and classify



Uijlings, et al., "Selective Search for Object Recognition", 2012

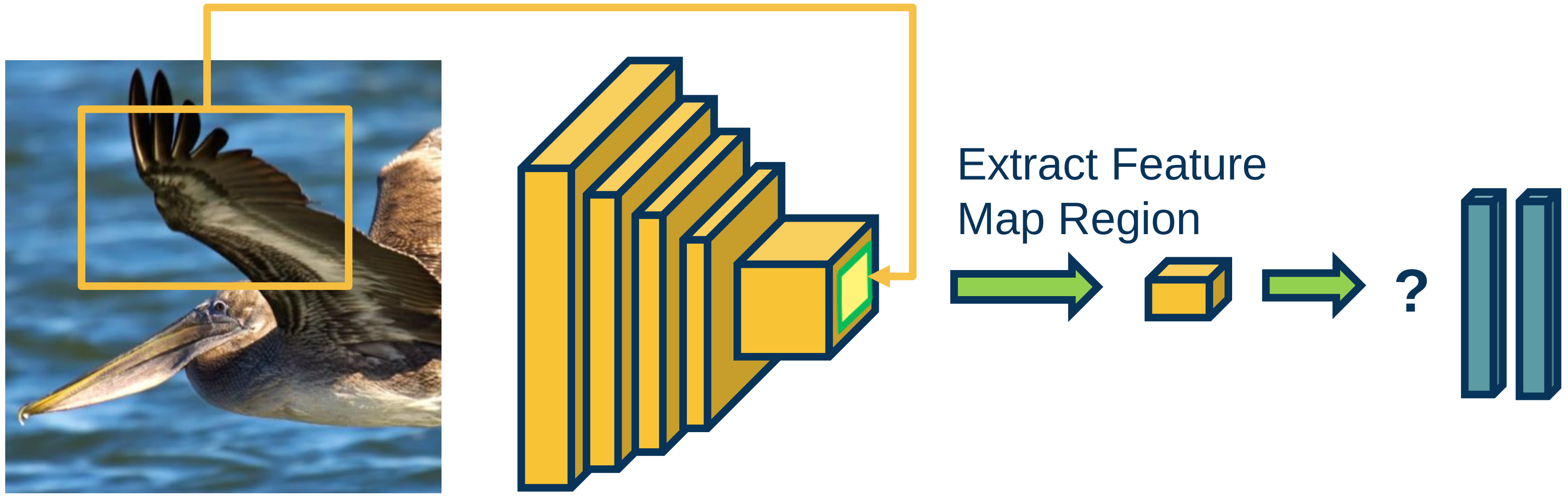
What is the problem with this?



Computation for convolutions re-done for each image patch, even if overlapping!

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

Map each ROI in image to corresponding region in feature maps

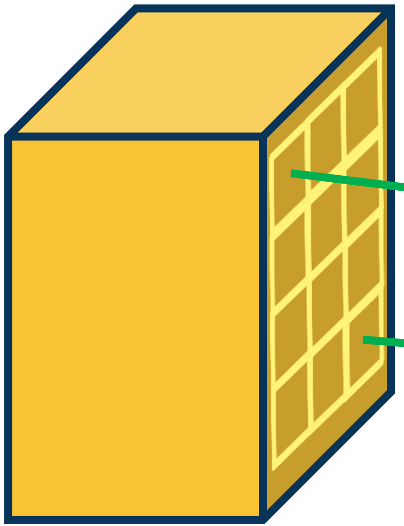


Idea: ***Reuse*** computation by finding regions in ***feature maps***

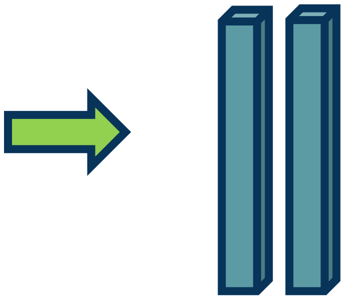
- Feature extraction only done once per image now!
- Problem: Variable input size to FC layers (different feature map sizes)

Girshick, "Fast R-CNN", 2015

For each grid element, max pool however many values there are to one scalar



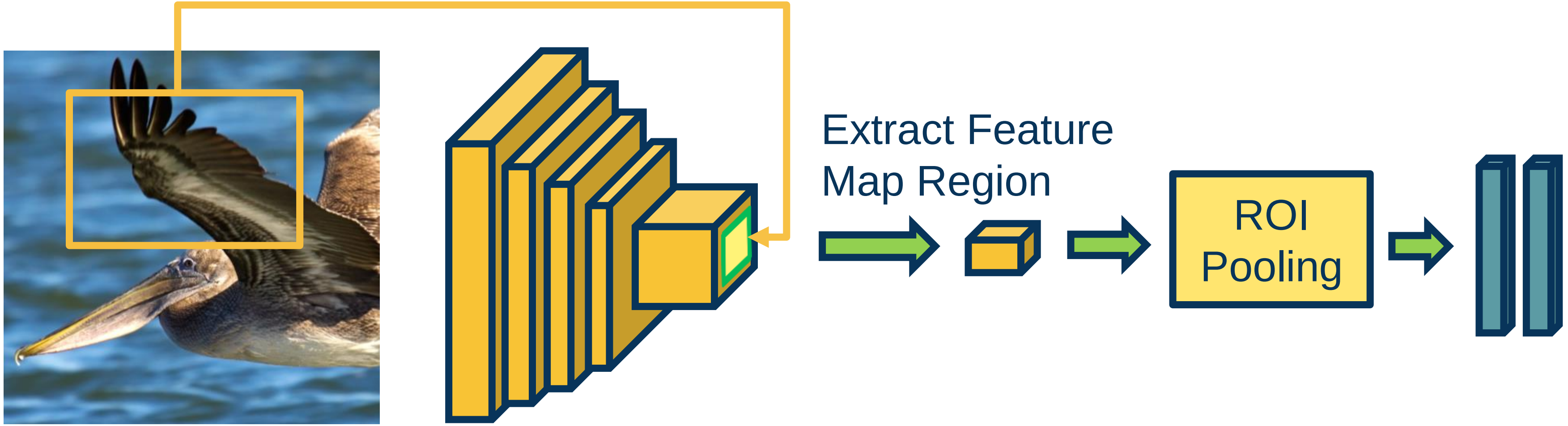
$$\begin{bmatrix} 120 & 150 & 120 \\ 100 & 50 & 110 \\ 25 & 25 & 10 \\ 65 & 75 & 10 \end{bmatrix}$$



Given an arbitrarily-sized feature map, we can use **pooling** across a grid (ROI Pooling Layer) to convert to fixed-sized representation

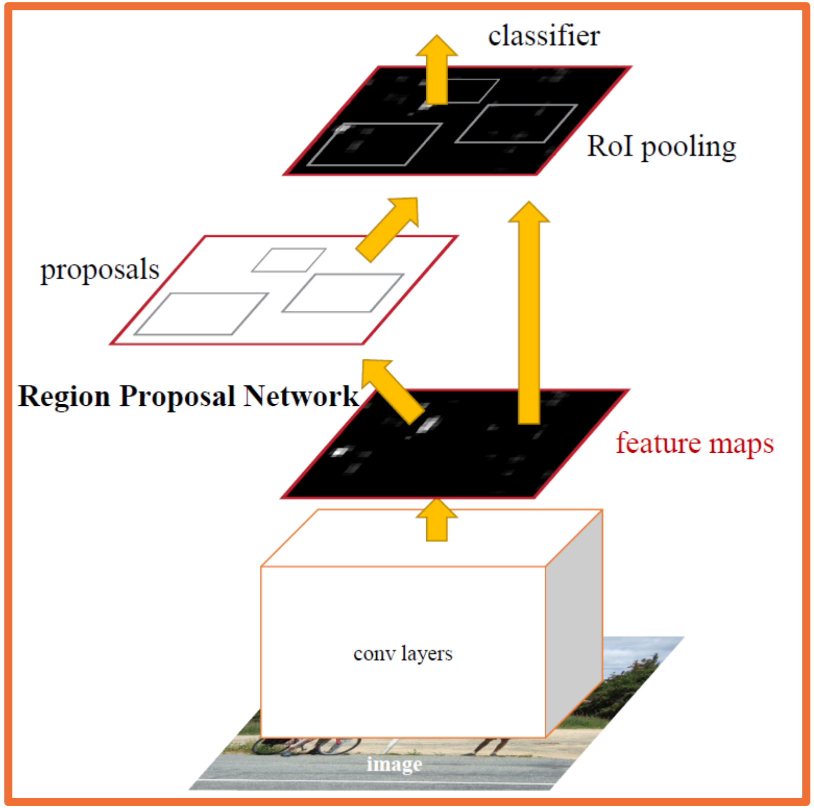
ROI Pooling

Map each ROI in image to corresponding are in feature maps



We can now train this model **end-to-end** (i.e. backpropagate through entire model including ROI Pooling)!

- **Idea:** Why not have the neural network *also* generate the proposals?
  - Region Proposal Network (RPN) uses same features!
- Outputs ***objectness score*** and bounding box
- Top k selected for classification
- Note some parts (gradient w.r.t. bounding box coordinates) not differentiable so some complexity in implementation

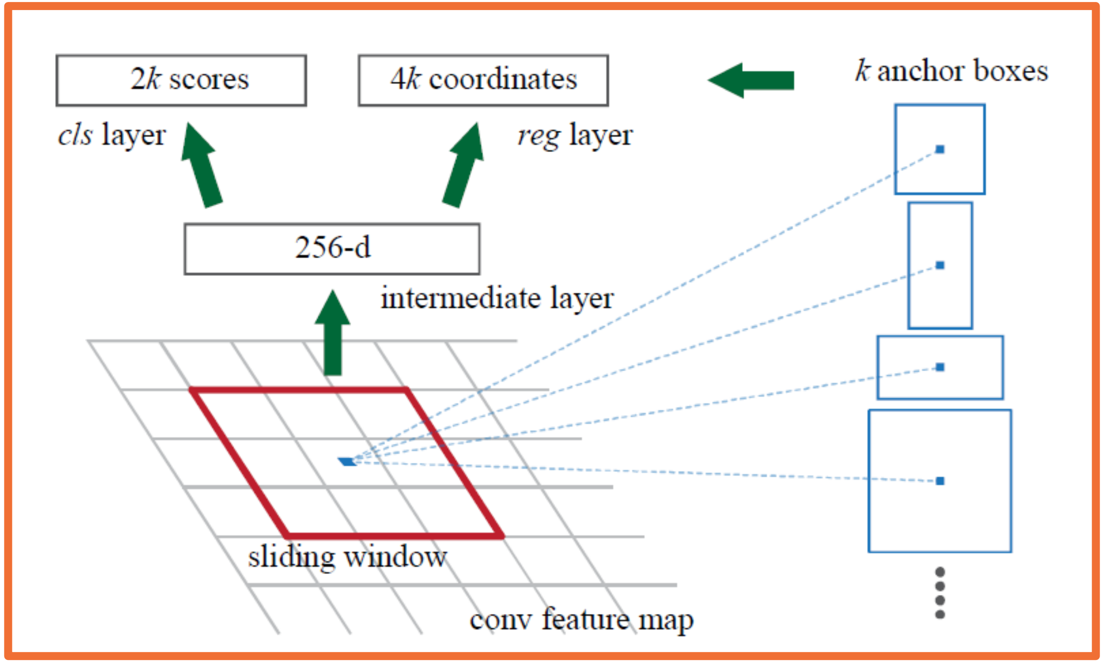


Ren, et al., "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks", 2016

Faster R-CNN

RPN also uses notion of **anchors in a grid**

Boxes of various sizes and scales classified with objectness score and refined bounding boxes refined

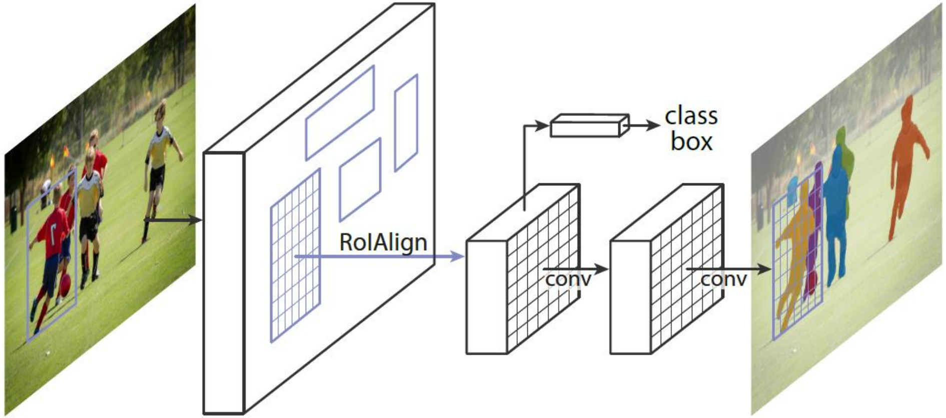


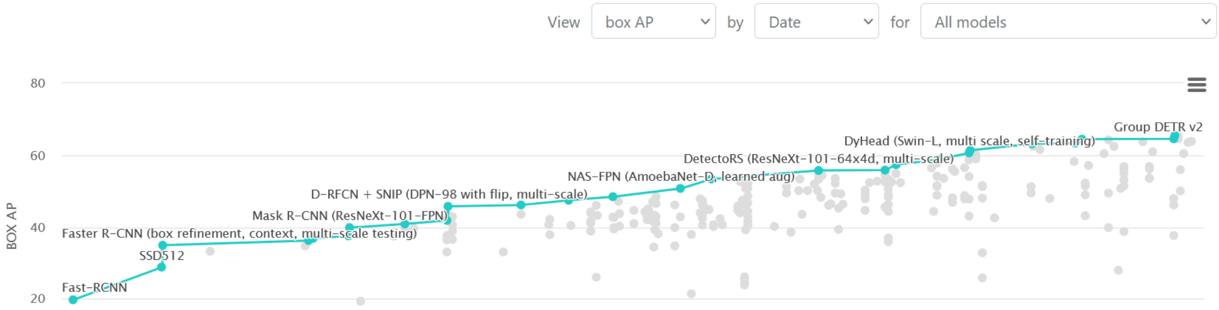
Ren, et al., "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks", 2016

Many new advancements have been made

For example, combining detection and segmentation

- Extract foreground (object) mask per bounding box





He, et al., "Mask R-CNN", 2018

https://paperswithcode.com/sota/object-detection-on-coco

Mask R-CNN

Summary

- A range of problems characterized by **density and type of output**
- **Semantic/instance segmentation:** Dense, spatial output
  - Leverage encoder/decoder architectures
- **Object detection:** Variable-length list of objects
  - Two-stage versus one-stage architectures
  - (Not covered): Anchor-based versus anchor-free methods

DETR

DETR
End-to-End Object Detection with Transformers

End-to-End Object Detection with Transformers

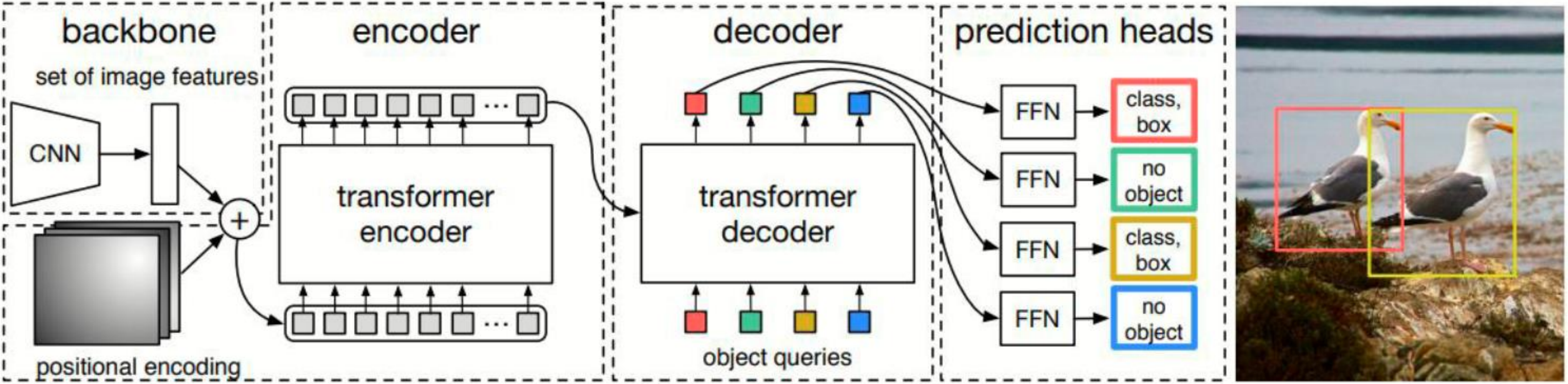
arXiv:2005.12872v3 [cs.CV] 28 May 2020

Nicolas Carion*, Francisco Massa*, Gabriel Synnaeve, Nicolas Usunier,
Alexander Kirillov, and Sergey Zagoruyko

Facebook AI

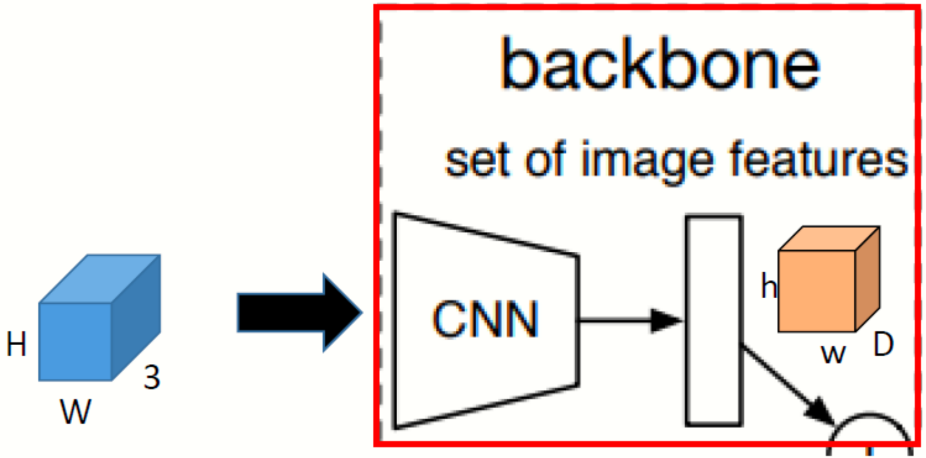
Abstract. We present a new method that views object detection as a direct set prediction problem. Our approach streamlines the detection pipeline, effectively removing the need for many hand-designed components like a non-maximum suppression procedure or anchor generation that explicitly encode our prior knowledge about the task. The main ingredients of the new framework, called DEtection TRansformer or DETR, are a set-based global loss that forces unique predictions via bipartite matching, and a transformer encoder-decoder architecture. Given a fixed small set of learned object queries, DETR reasons about the relations of the objects and the global image context to directly output the final set of predictions in parallel. The new model is conceptually simple and does not require a specialized library, unlike many other modern detectors. DETR demonstrates accuracy and run-time performance on par with the well-established and highly-optimized Faster R-CNN baseline on the challenging COCO object detection dataset. Moreover, DETR can be easily generalized to produce panoptic segmentation in a unified manner. We show that it significantly outperforms competitive baselines. Training code and pretrained models are available at https://github.com/facebookresearch/detr.

DEtector TRansformer - DETR overview



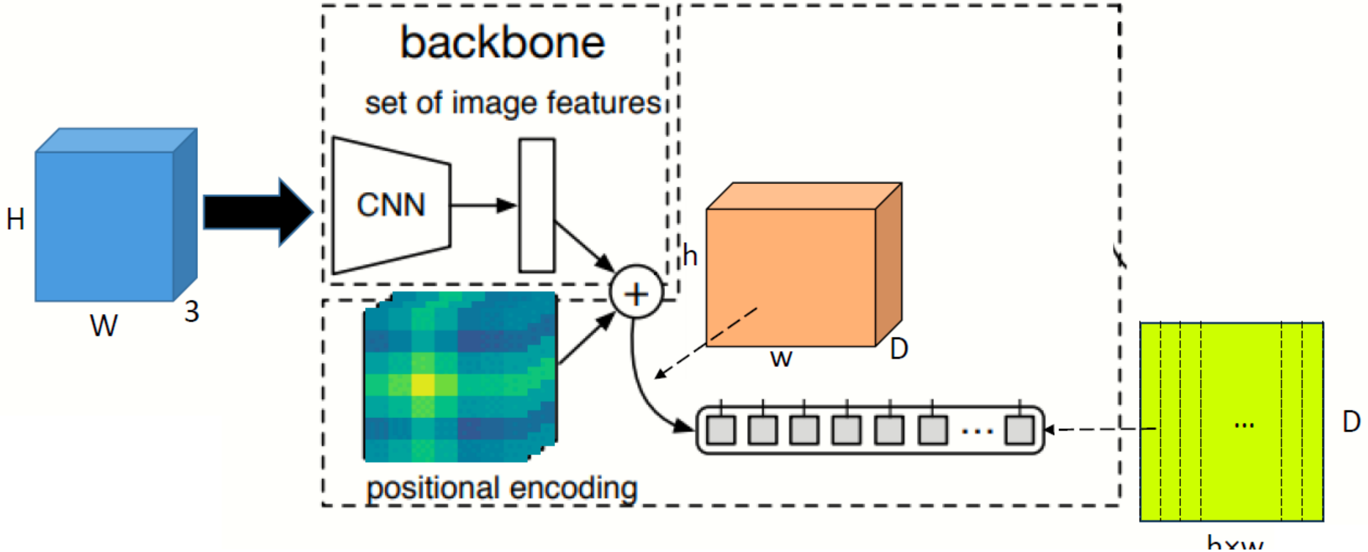
DEtector TRansformer - DETR backbone

A conventional CNN backbone to learn a 2D representation of an input image.



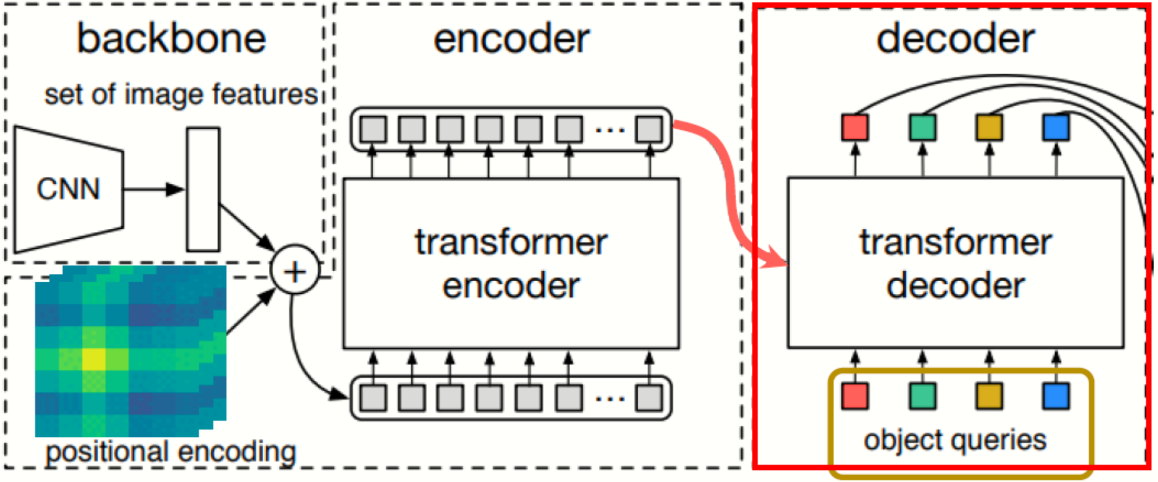
DEtector TRansformer - DETR
transformer encoder

DETR supplements the features with a **positional encoding** and **flattens** them before passing them into a **transformer encoder**.



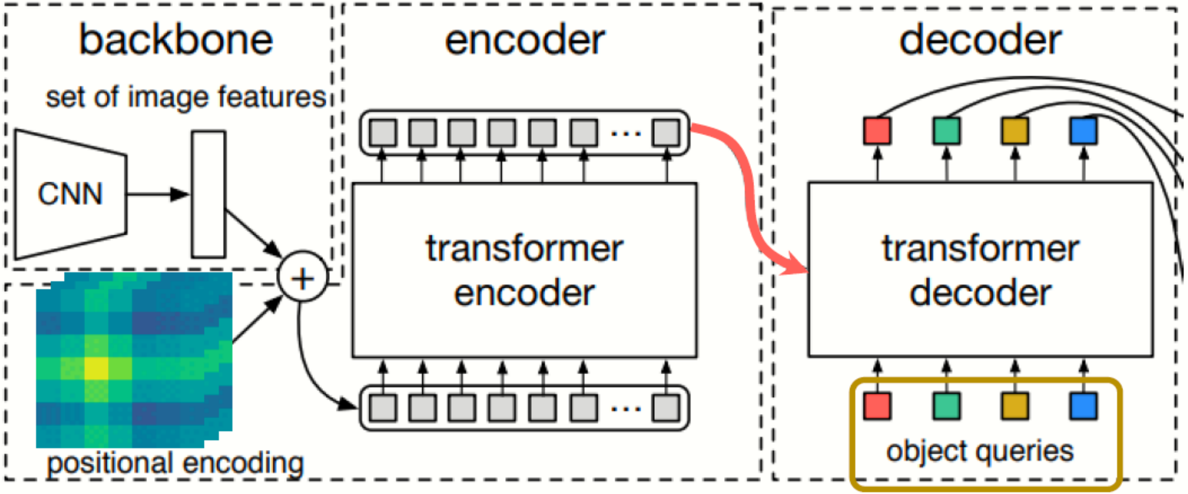
DEtector TRansformer - DETR
transformer decoder

The transformer decoder takes as input a small number of **learned positional** and content embeddings (object queries) and additionally attends to the encoder output.



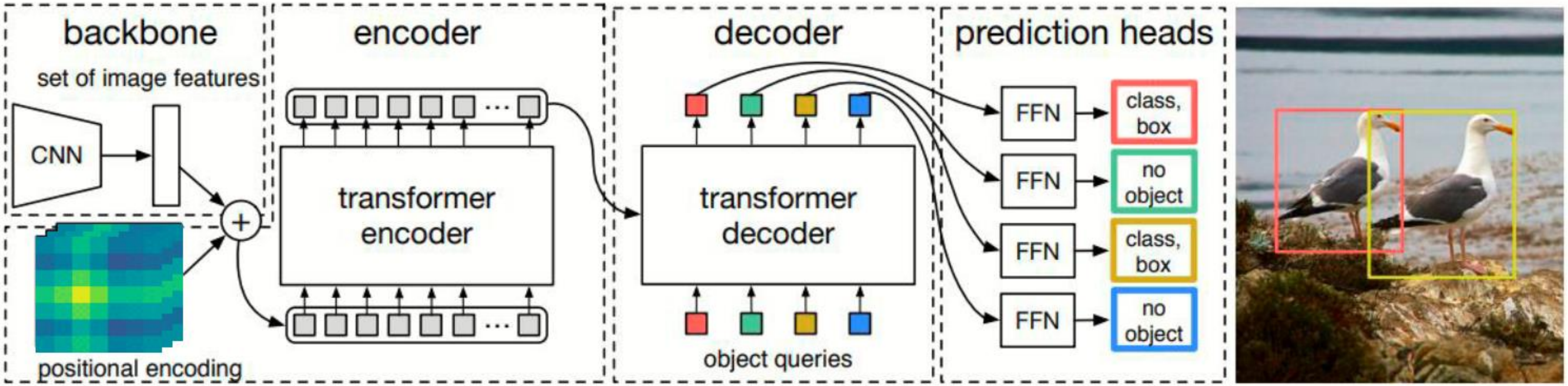
DEtector TRansformer - DETR
Object queries

- are randomly initialized embeddings,
- refined through the course of training, and
- then fixed for evaluation.



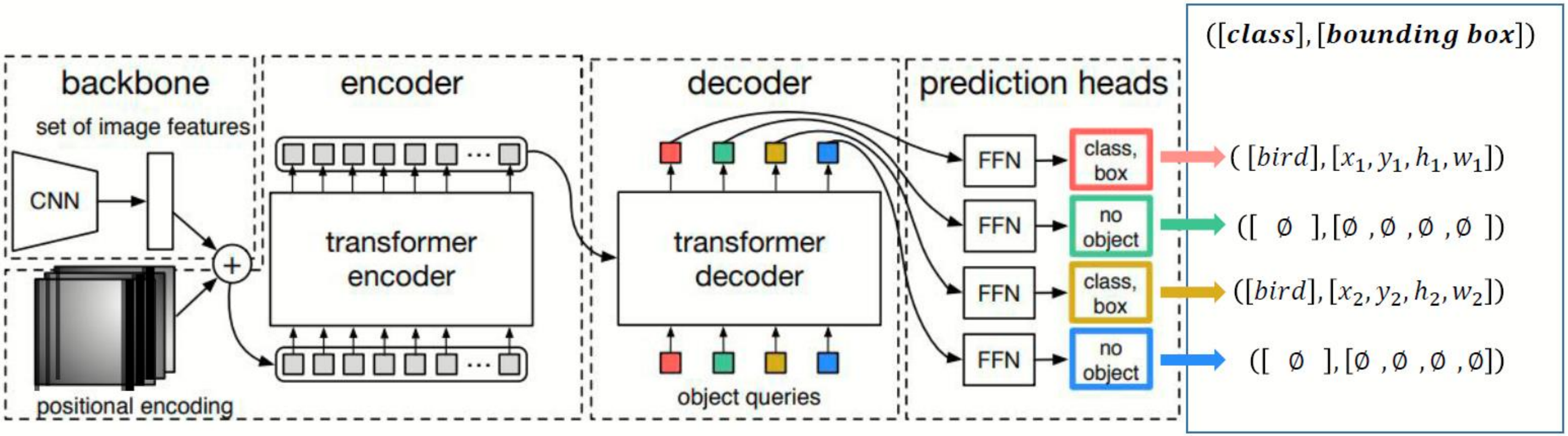
DEtector TRansformer - DETR prediction heads

Each embedding at the decoder output feeds a shared feed-forward network (FFN) that predicts either a detection (class and bounding box) or a "no object" class.



DEtector TRansformer - DETR prediction heads

Each output embedding of the decoder to a shared feed forward network (FFN) that predicts either a detection (class and bounding box) or a “no object" class.



Matching bounding boxes during training with the Reference

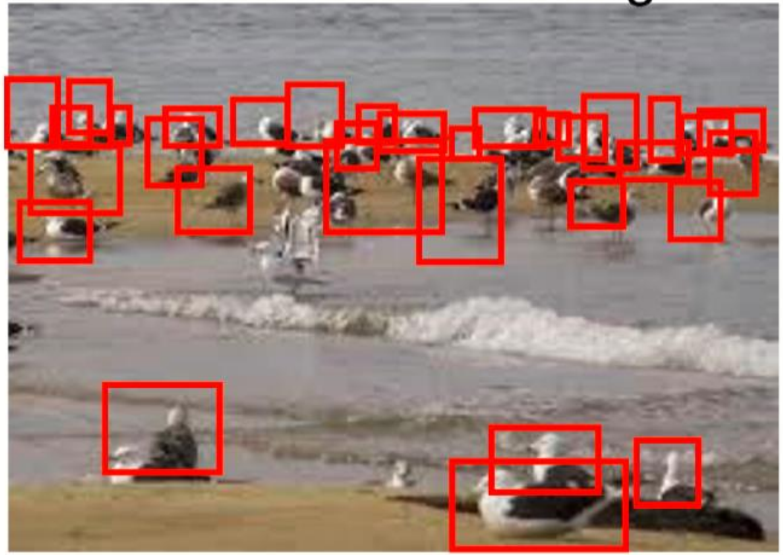
What is the target for each bounding box during training?

DETR while training

Reference – Ground Truth

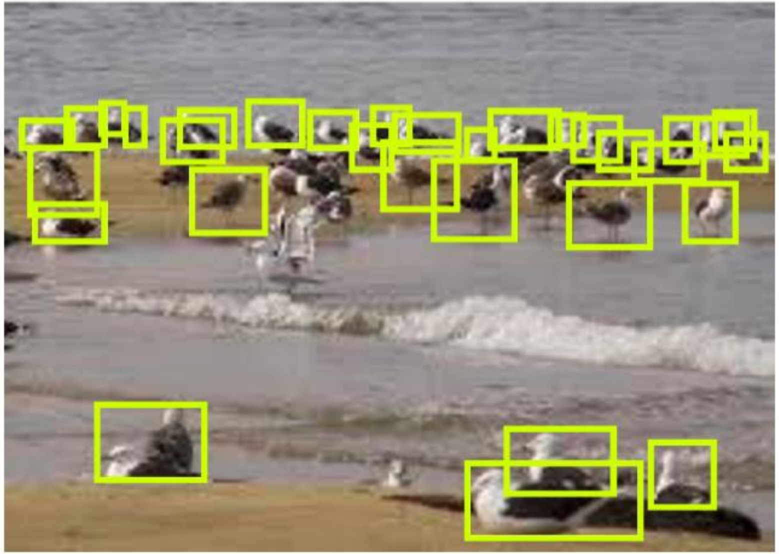
The complexity of matching grows with *N*!

DETR while training



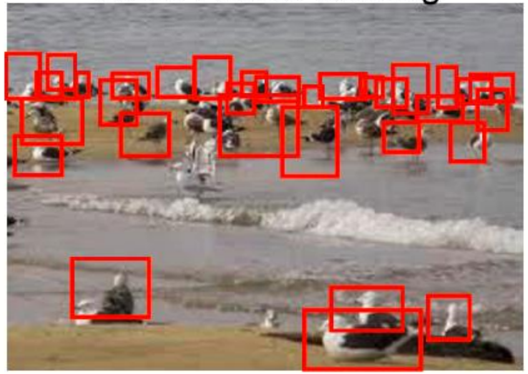
N!

Reference – Ground Truth

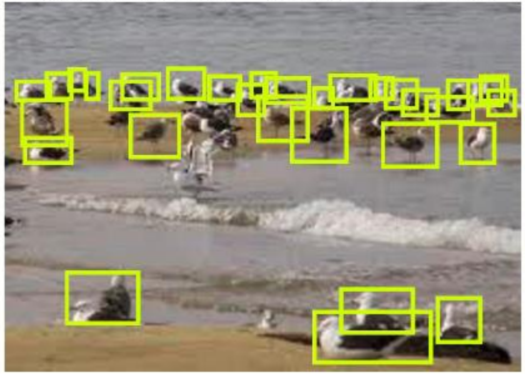


The Hungarian algorithm used for bipartite matching

DETR while training

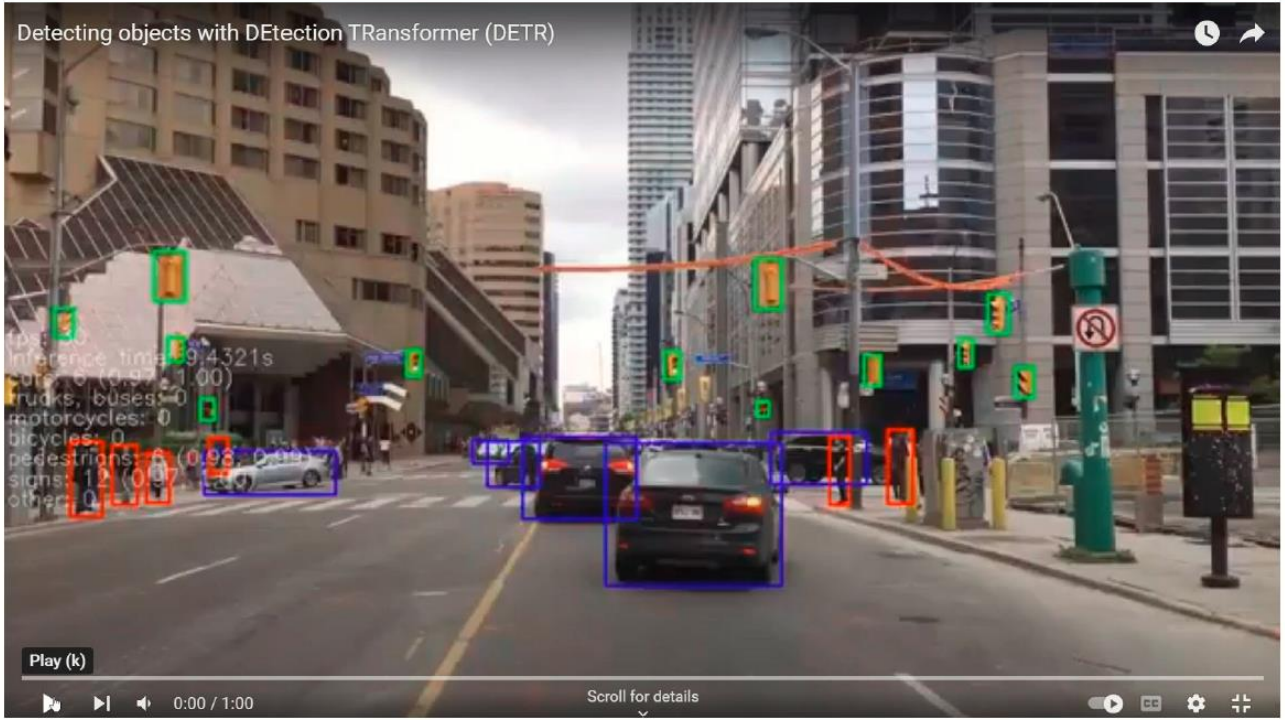


Reference – Ground Truth



The **Hungarian algorithm**[1] computes the optimal assignment efficiently. It considers both the class prediction and the similarity of predicted and ground truth boxes.

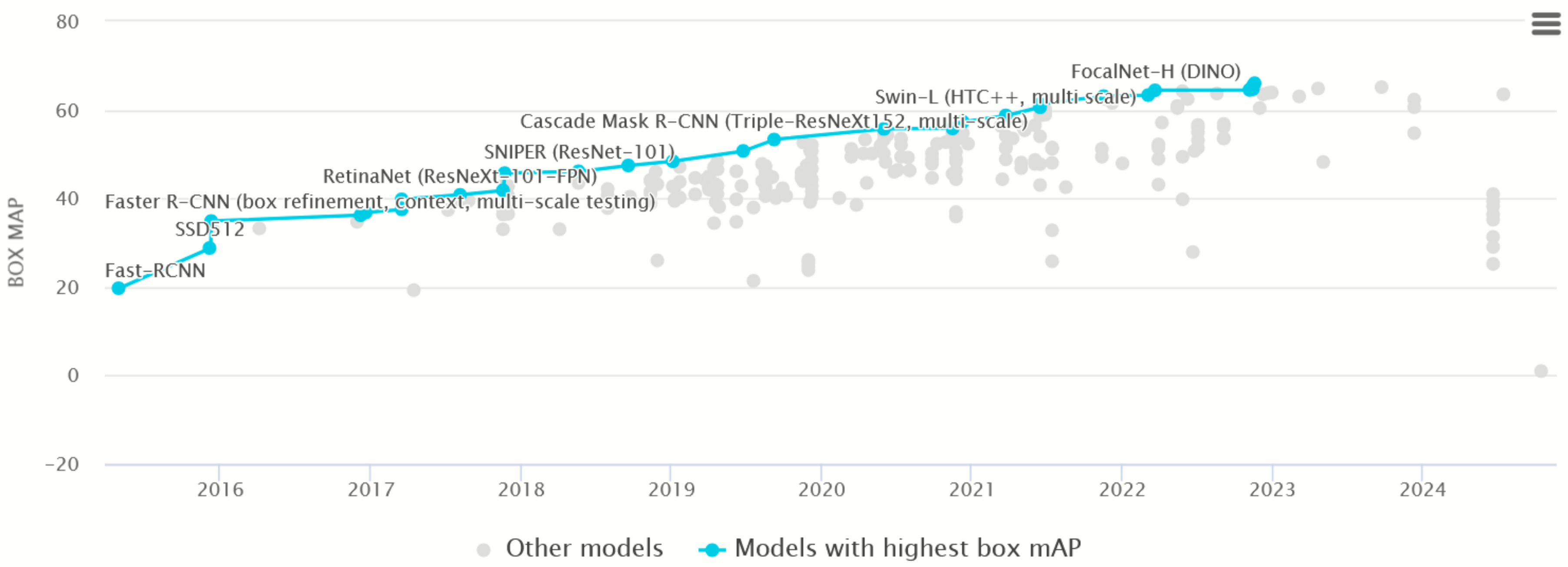
DETR Demo


Detecting objects with DEtection TRansformer (DETR)
Play (k)
0:00 / 1:00
Scroll for details

https://paperswithcode.com/sota/object-detection-on-coco

Leaderboard   Dataset

View box mAP by Date for All models



Georgia Tech

Leaderboard Dataset

View mask AP by Date for All models