
HE VD
IG

Intelligence Artificielle pour les systèmes autonomes (IAA)

Modular pipeline: Object detection

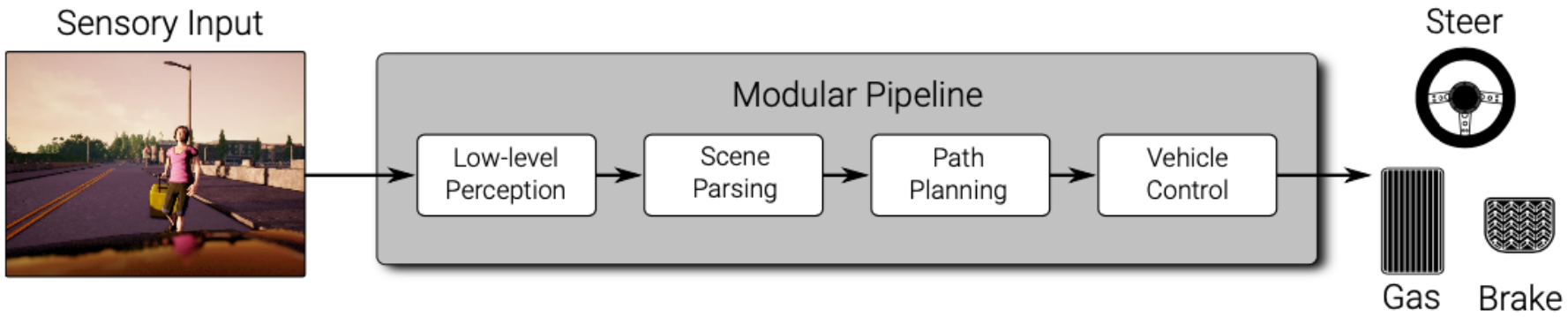
Prof. Yann Thoma - Prof. Marina Zapater

Février 2024

Basé sur le cours du Prof. A. Geiger

Modular Pipeline

Reminder of main blocks



→ Vehicle control

→ Low-level perception : Odometry, SLAM and global localization

→ **Scene Parsing (Object detection and tracking)**

→ Path planning

Summary

Today's lesson

→ **Object detection basics**

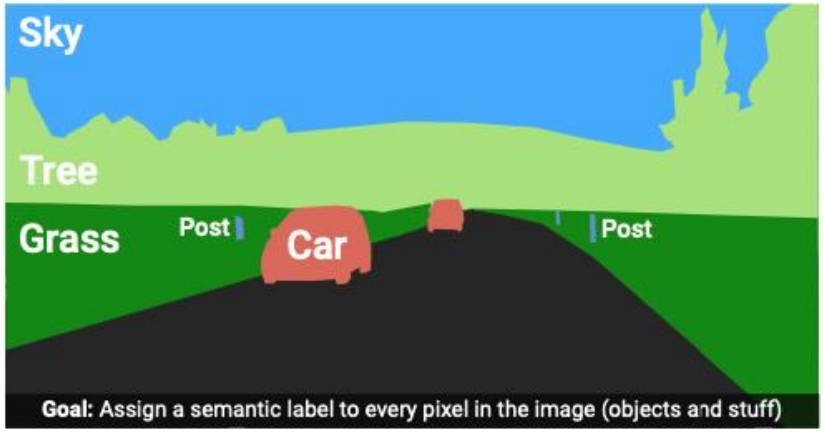
→ From classical to deep learning strategies

→ 3D Object detection

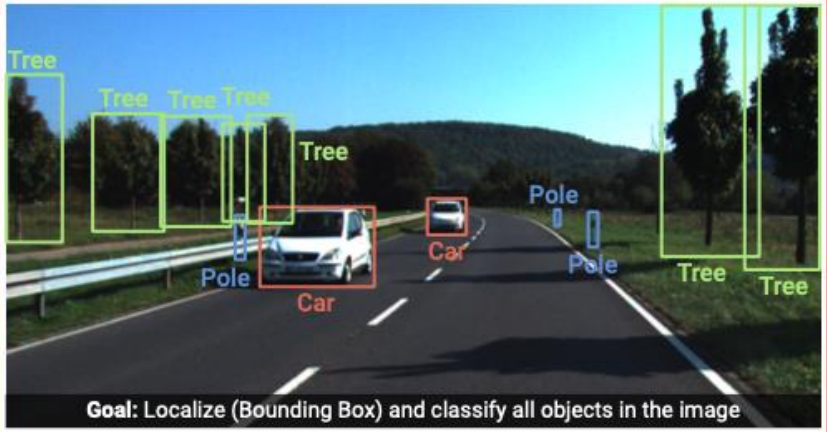
Object detection

Definition and differences wrt what we have seen so far

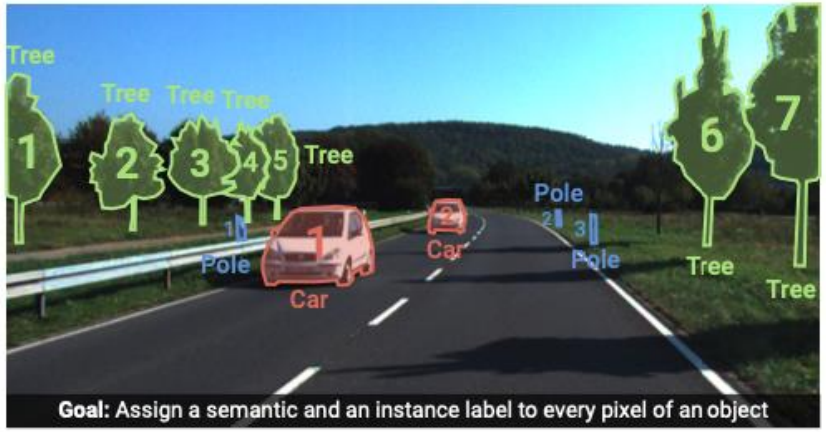
Image Classification



Semantic Segmentation



Object Detection



Instance Segmentation

An interesting TED talk

By Chris Urmson

https://www.youtube.com/watch?v=tiwVMrTLUWg

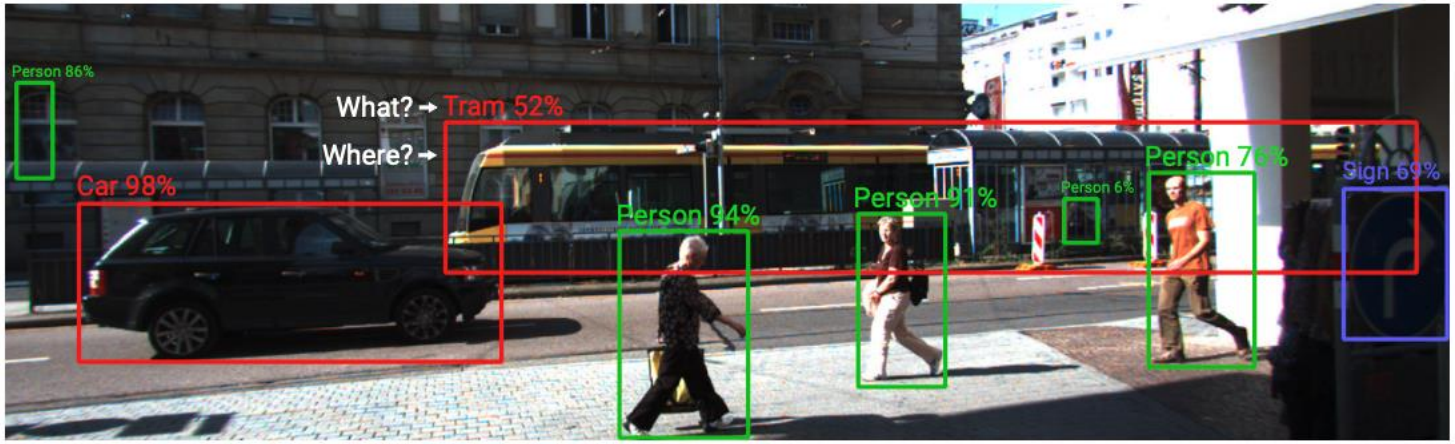
Problem settings

→ **Input**: RGB image

→ **Output**: set of 2D/3D bounding boxed with category label and confidence

→ **Issue**: there are infinite possible boxes and the number of objects is unknown

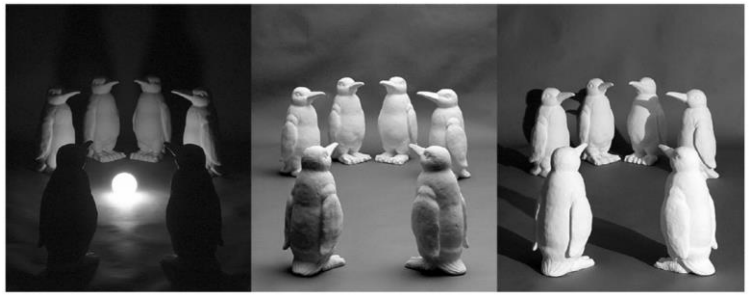
- And many challenges associated



Challenges

For object detection

Illumination changes



View-point variation

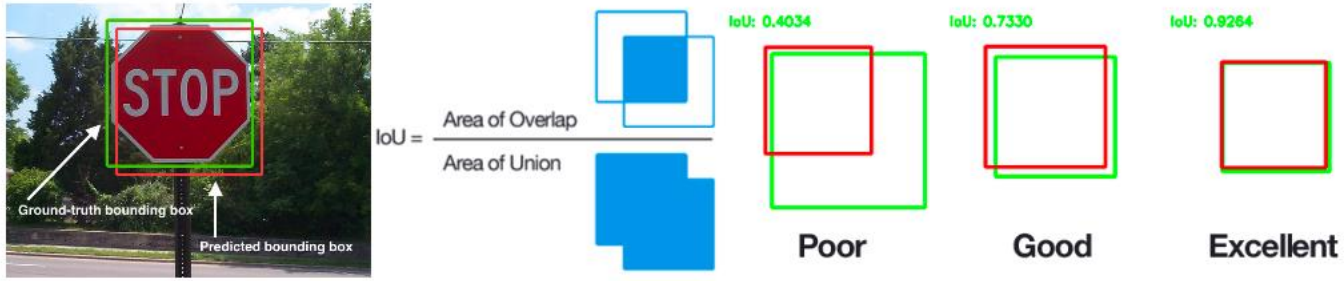
Intra-class variation

Deformation

Occlusion

How to measure performance?

Measuring bounding box alignment (wrt a ground truth)



→ Intersection-over-Union (IoU) → predicted box vs true box

→ Detection considered successful if IoU > 0.5

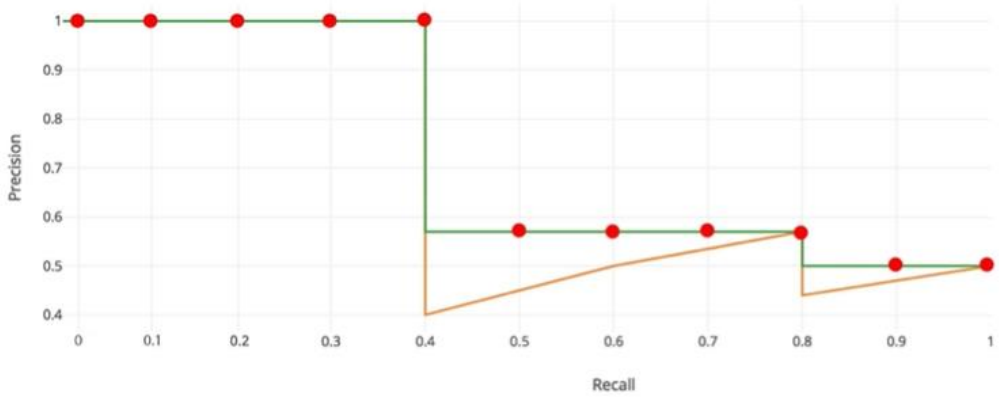
→ And if there are multiple objects?

- Number of detections depends on detector threshold and is detector specific

How to measure Detection Performance?

Average precision metric (used in **The Pascal Visual Object Classes (VOC) Challenge, 2010**)

1. Run detector with varying thresholds
2. Assign detections to closest objects
3. Count True Positives (TP),
   False Positives (FP) and False Negatives (FN)
4. Compute **Average Precision** (AP)



True Positives (TP): Number of objects correctly detected (IoU>0.5)

False Positives (FP): Wrong detections

False Negatives (FN): Number of true objects not detected (IoU<0.5)

$$\text{Precision } P = \frac{TP}{TP + FP}$$

$$\text{Recall } R = \frac{TP}{TP + FN}$$

$$\text{Avg. Prec. } AP = \frac{1}{11} \sum_{R \in \{0,\ldots,1\}} \max_{R' \geq R} P(R')$$

Summary

Today's lesson

→ Object detection basics

→ **From classical to deep learning strategies**

→ 3D Object detection

Object Detection Techniques

Some interesting approaches to the problem

Classical approaches (no deep learning yet)

→ Sliding window detection (2010)
- Classical approach, not necessarily using deep learning

→ Part-based models **(**2010)

Revolution of DL models (2015): Use of Convolutional Neural Networks (CNN)

→ Region-based CNNs (R-CNN, Fast R-CNN)

→ Feature pyramid network

→ Single-stage detector (another revolution!)
- **YOLO** : You Only Look Once

→ Now using transformers…

Sliding window object detection

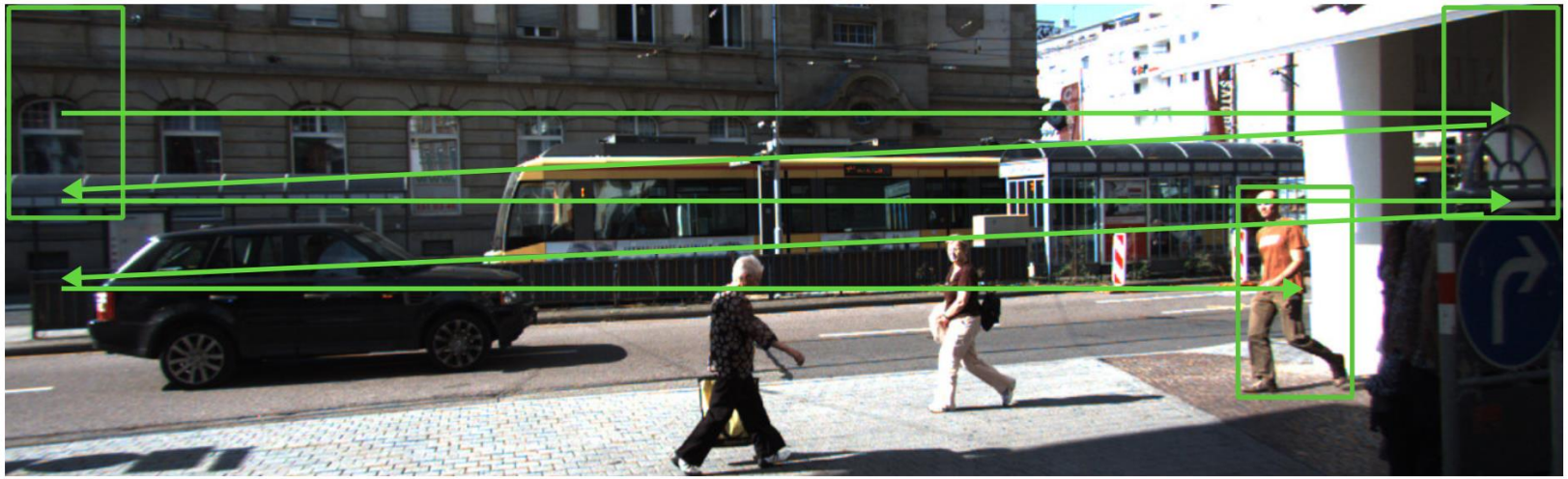
Main idea (dating from 2010)

→ Run sliding window of fixed size over the image; extract features for each window

→ Classify each crop (object vs. background) using SVM, random forest, boosting,

→ Search across aspect ratios/scales to recover objects of varying size/distance

→ Non-maxima suppression (=clustering) to retrain only 1 prediction per object



Part-based models

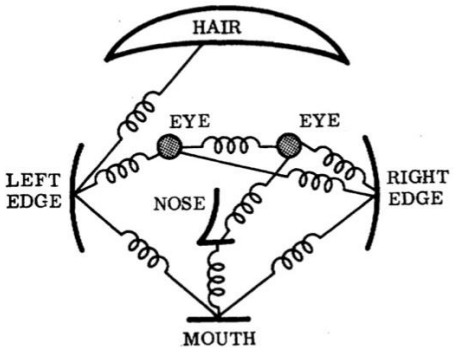
Idea: Model object based on its parts

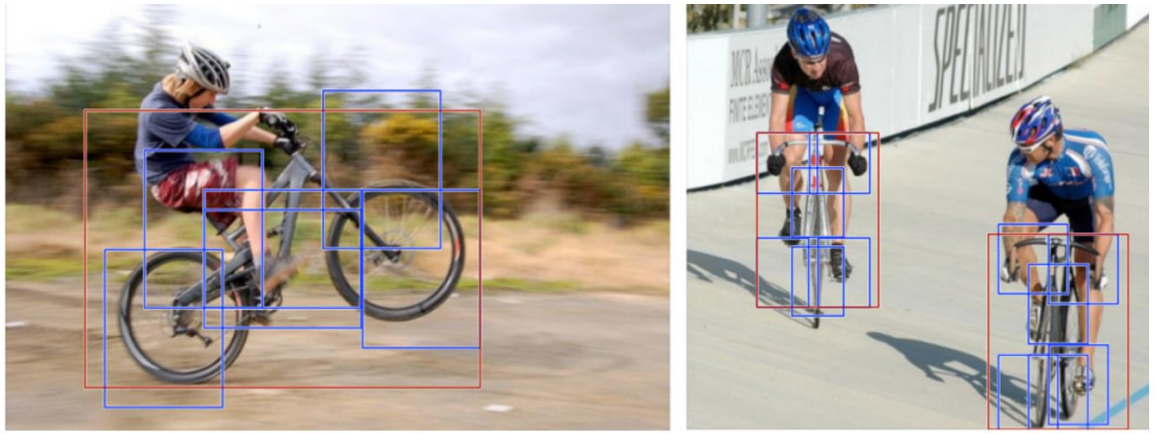
→ We will model the distribution of part configurations

→ Allows for more invariance (learns the parts of the objects, so it can work well even with image deformations)

→ Way more complex (therefore, slower inference)

→ Again, a technique of 2010





https://ieeexplore.ieee.org/document/5255236

Using CNNs

How can we detect objects using Neural Networks?

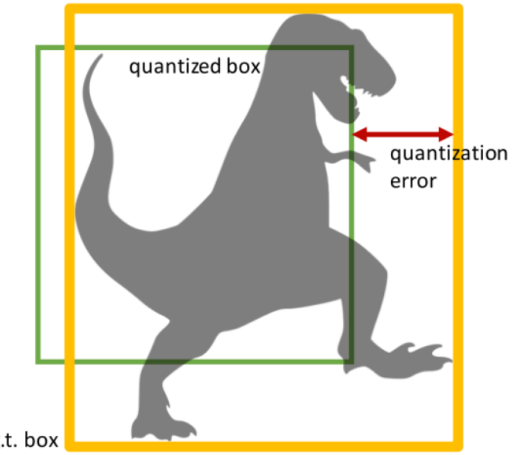
→ Idea: classify all boxes of an image using a classification network

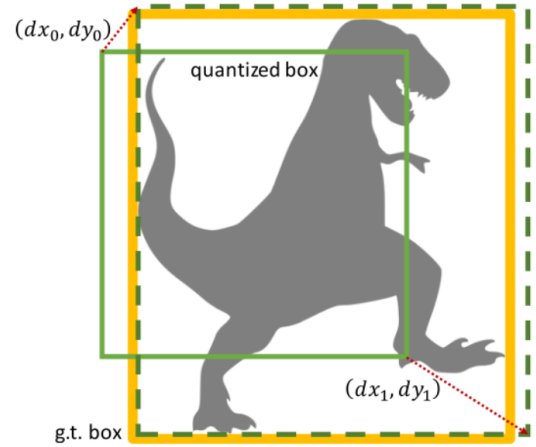
→ Problem: too many boxes to classify (even by space discretization)

→ Better idea: a 2-stage object detector

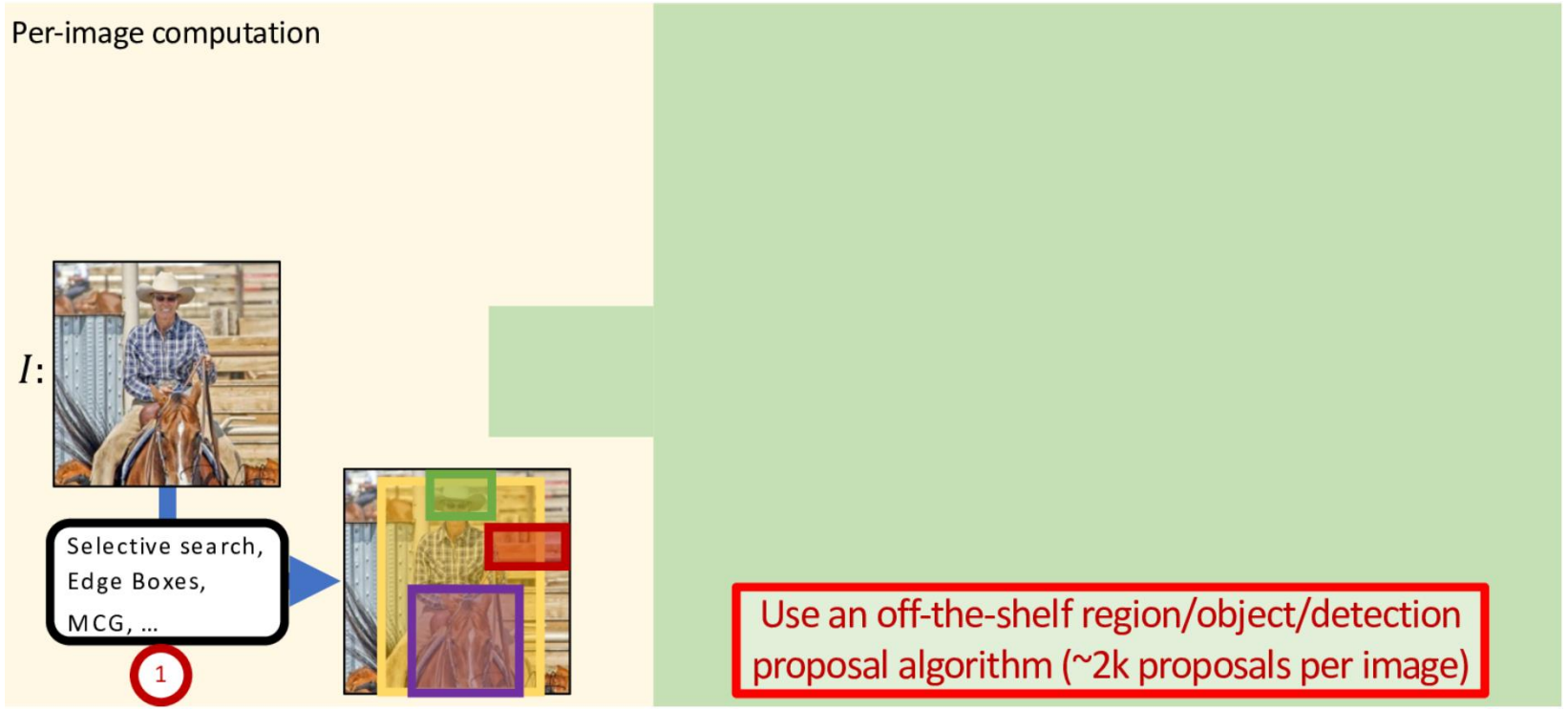
- **Quantize:** Detect candidate boxes that might contain an object
  - Approximation with a finite set of boxes
  - Leads to quantization error
- **Regress**: Classify and then refine the location of boxes using a CNN
  - Recover loss of localization accuracy by regressing the location offset
- **Clustering:** Remove redundant predictions using non-maximum supression

We split the original problem into two subproblems

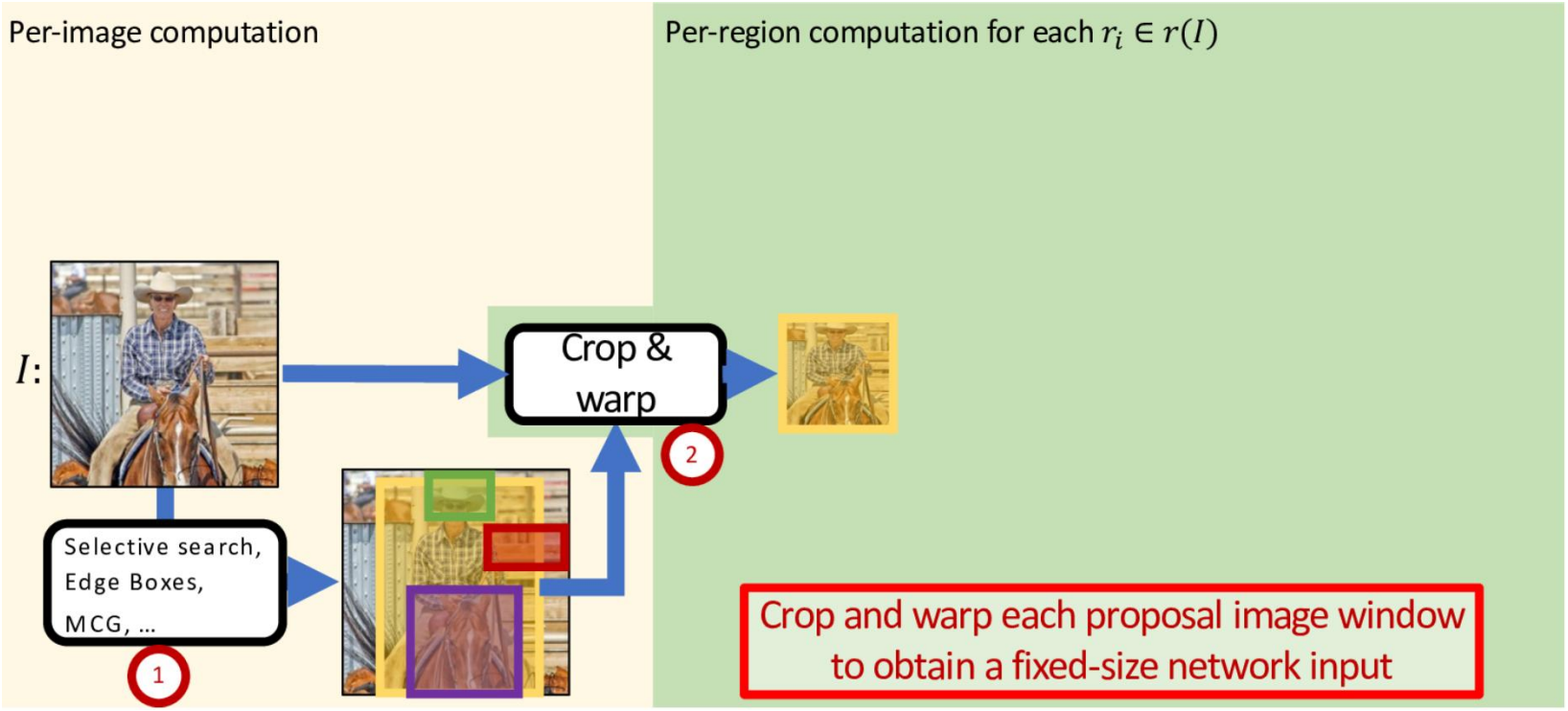




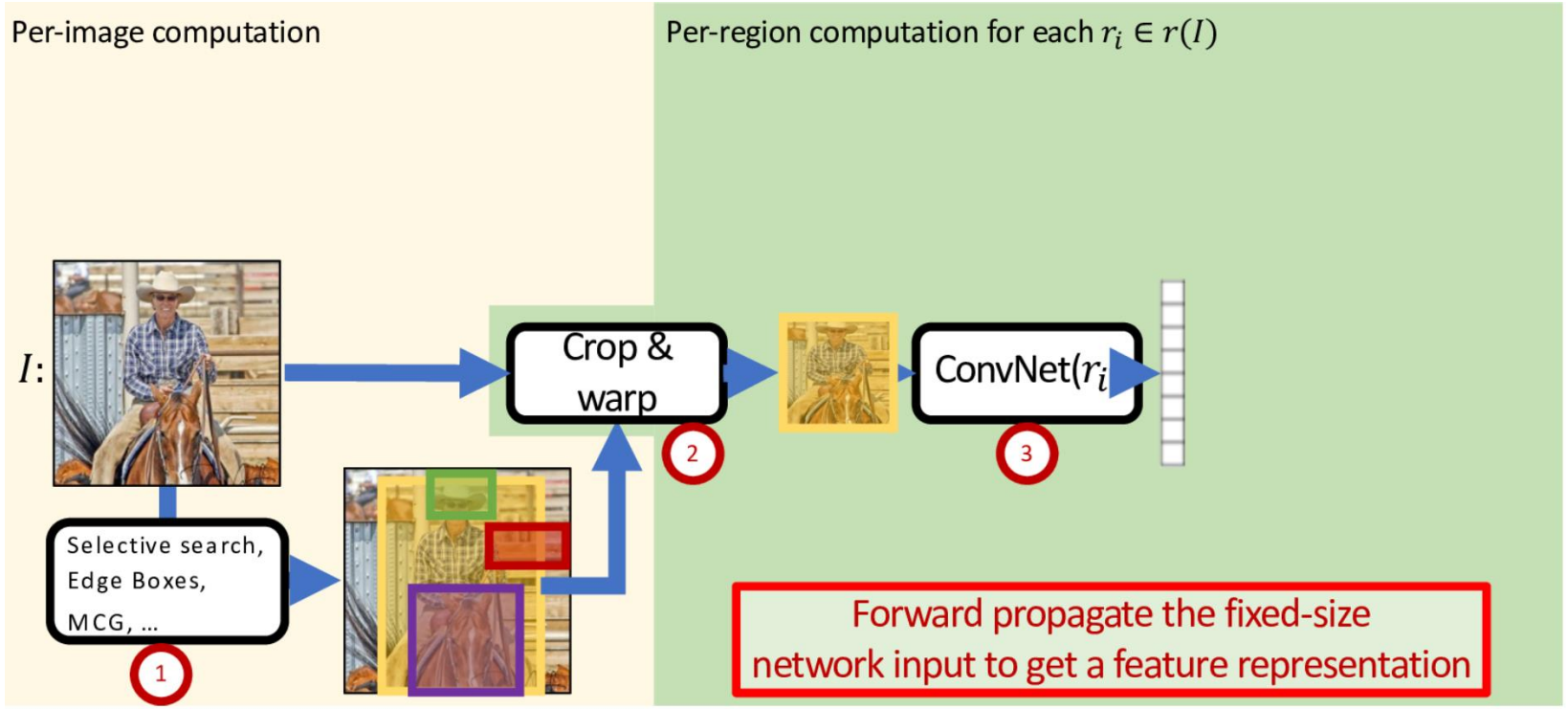
R-CNN: Region-based Convolutional Neural Network



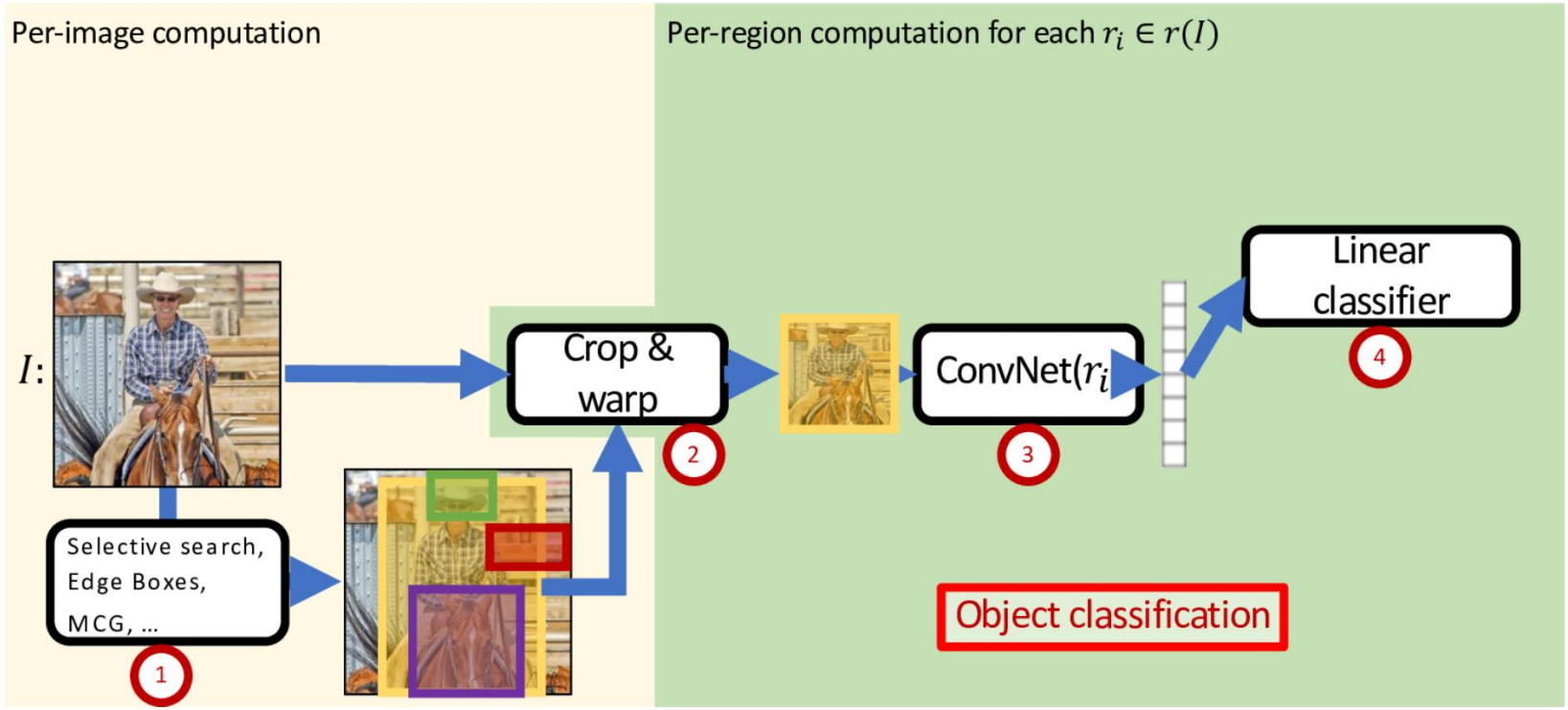
R-CNN: Region-based Convolutional Neural Network



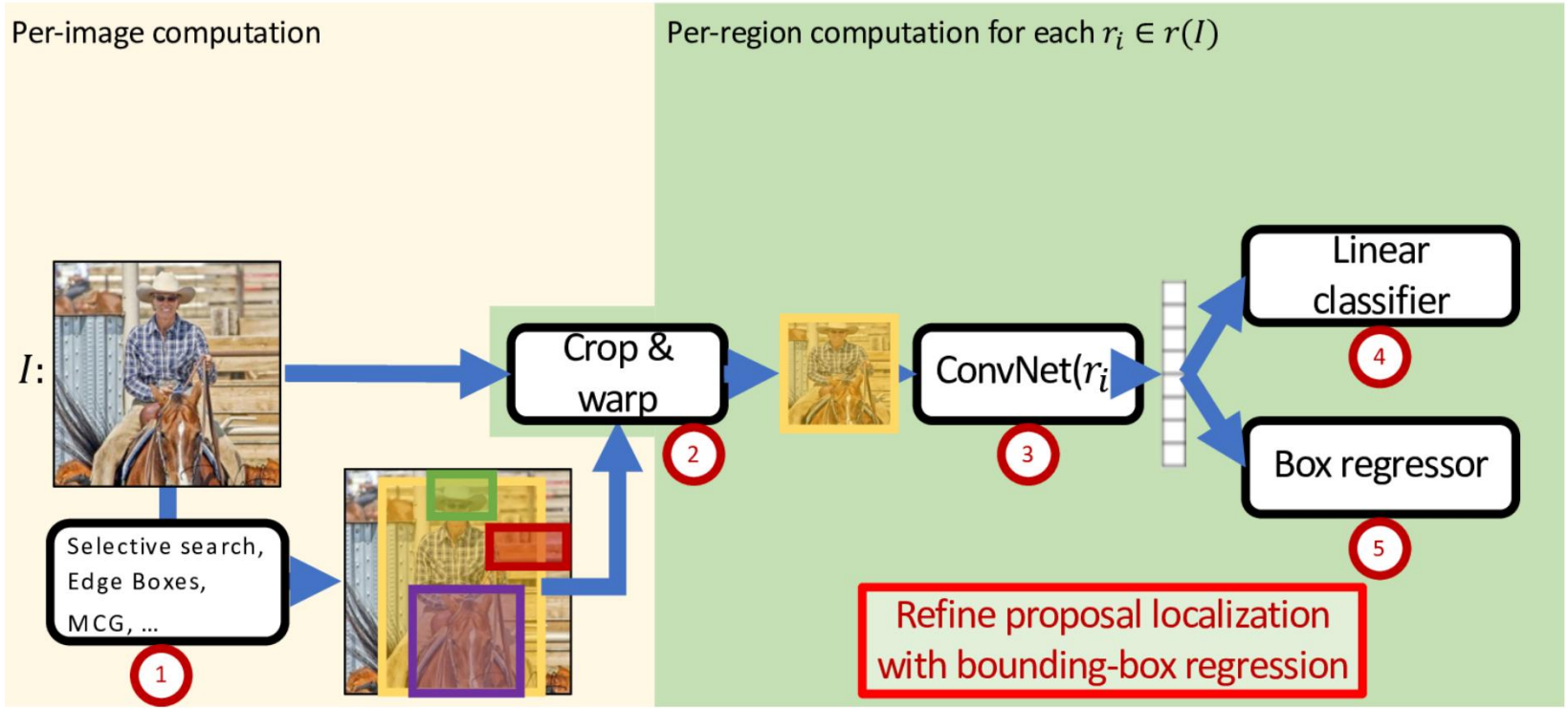
R-CNN: Region-based Convolutional Neural Network



R-CNN: Region-based Convolutional Neural Network



R-CNN: Region-based Convolutional Neural Network



Main issues with R-CNN

Heavy computation!

→ Heavy per-region computation (e.g., 2000 full network evaluations)

→ No computation/feature sharing

→ Slow region proposal methods adds to runtime

→ Generic region proposal techniques have limited recall

Solution?

→ Fast R-CNN

Fast R-CNN

Using a backbone architecture

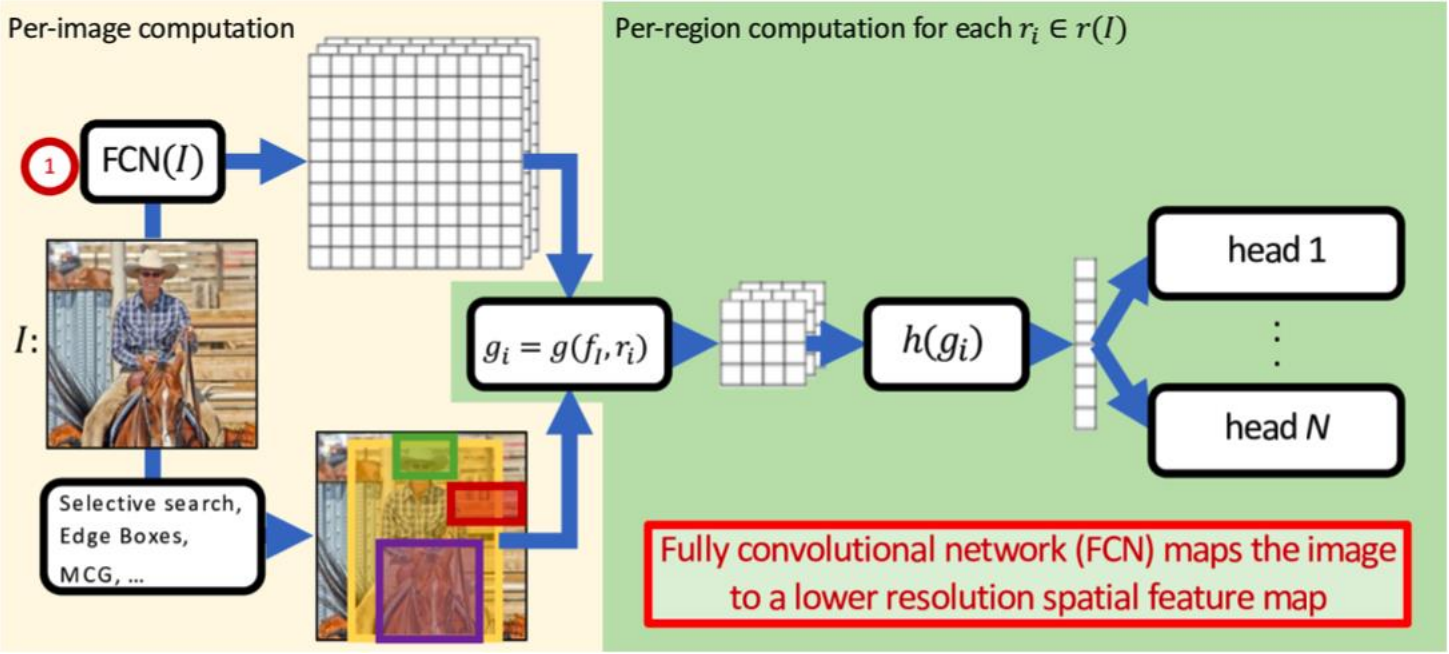
→ Use any standard CNN as “backbone architecture”
- AlexNet, VGG, ResNet, Inception… or others

→ Remove global pooling : output spatial dimensions proportional to input dimensions

→ A good network exploits the strongest recognition backbone

https://arxiv.org/abs/1504.08083

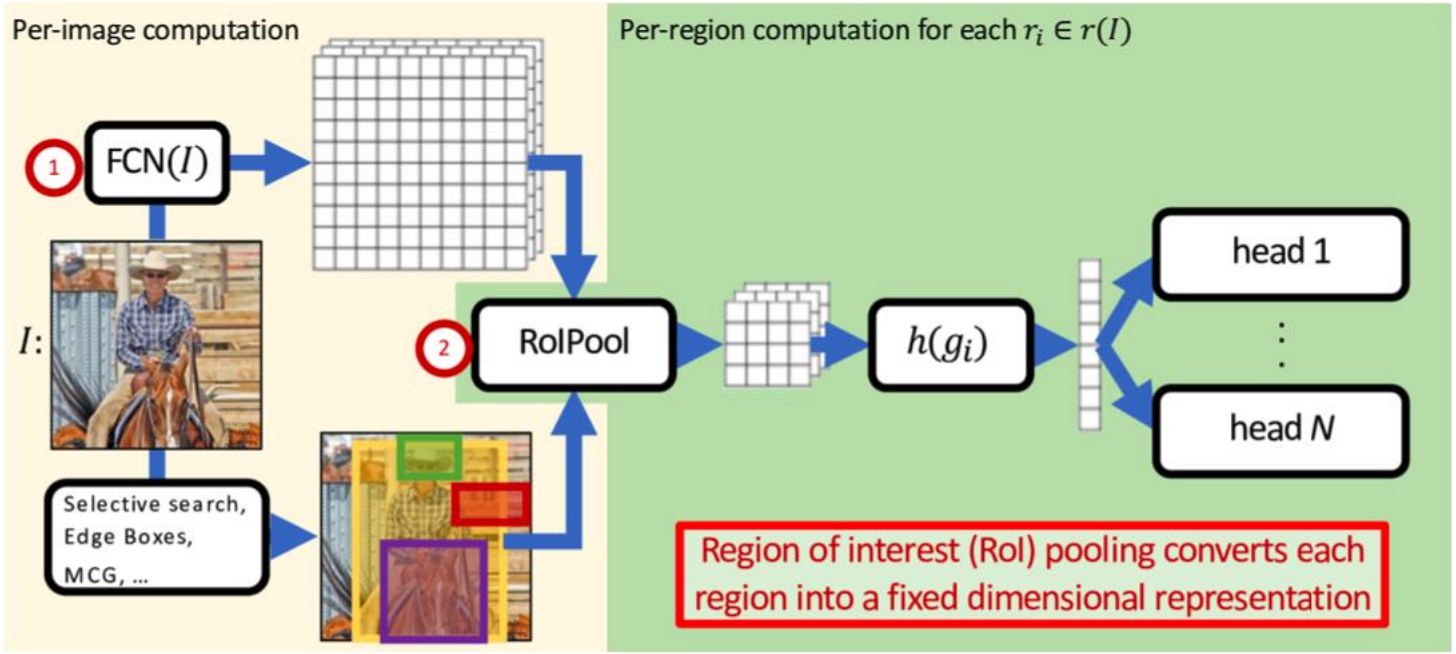
Fast R-CNN



→ Lighter weight per-region computation : end-to-end trainable version

https://arxiv.org/abs/1504.08083

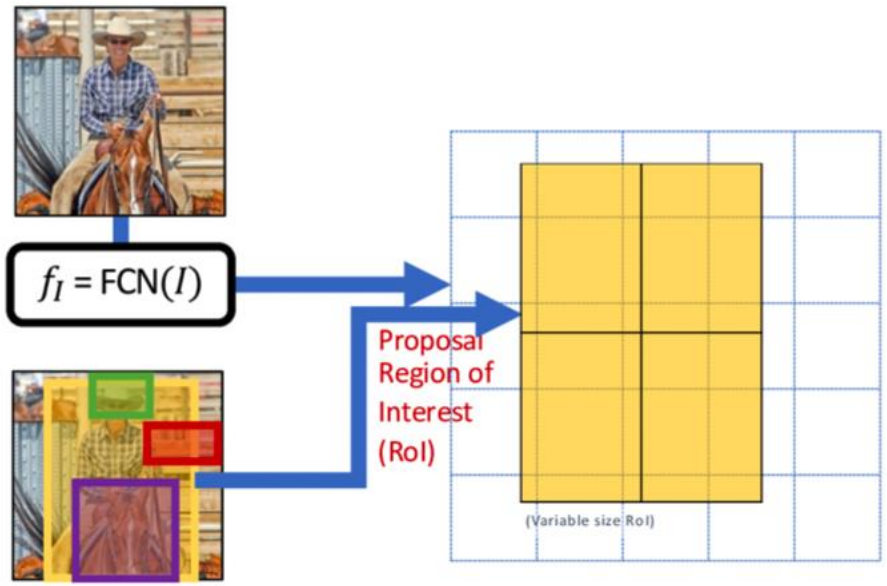
Fast R-CNN



https://arxiv.org/abs/1504.08083

Fast R-CNN

Region-of-Interest (RoI) Pooling on each Proposal

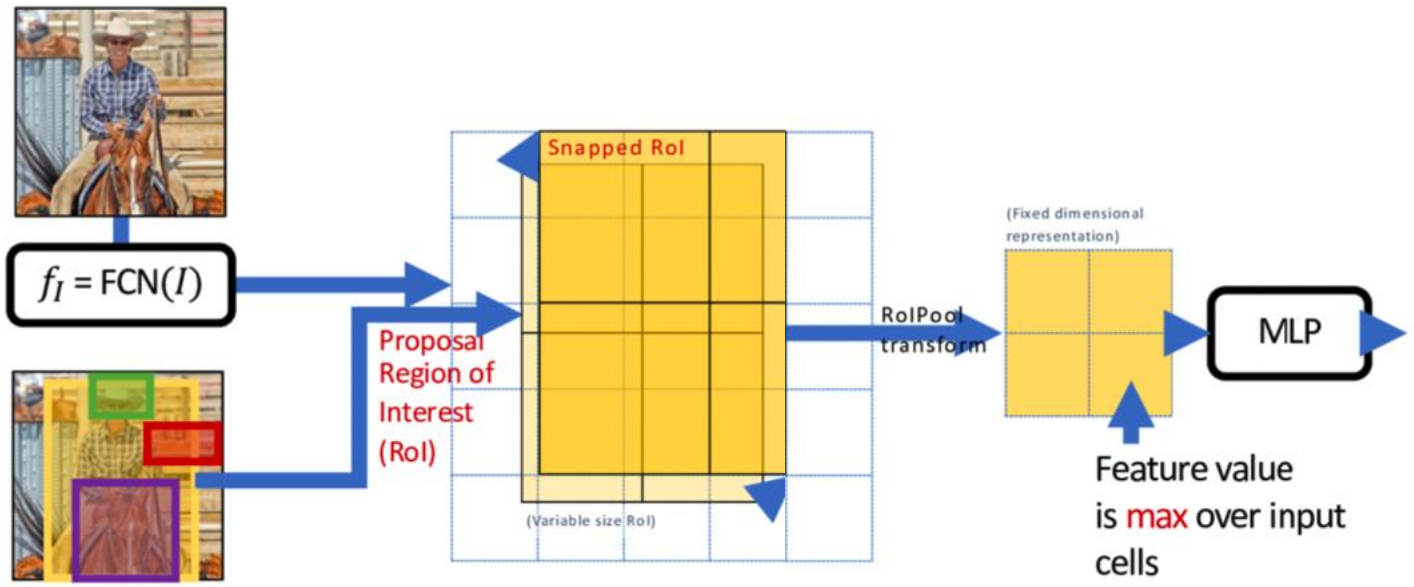


Key innovation in SPP-net
[He et al. 2014]

https://arxiv.org/abs/1504.08083

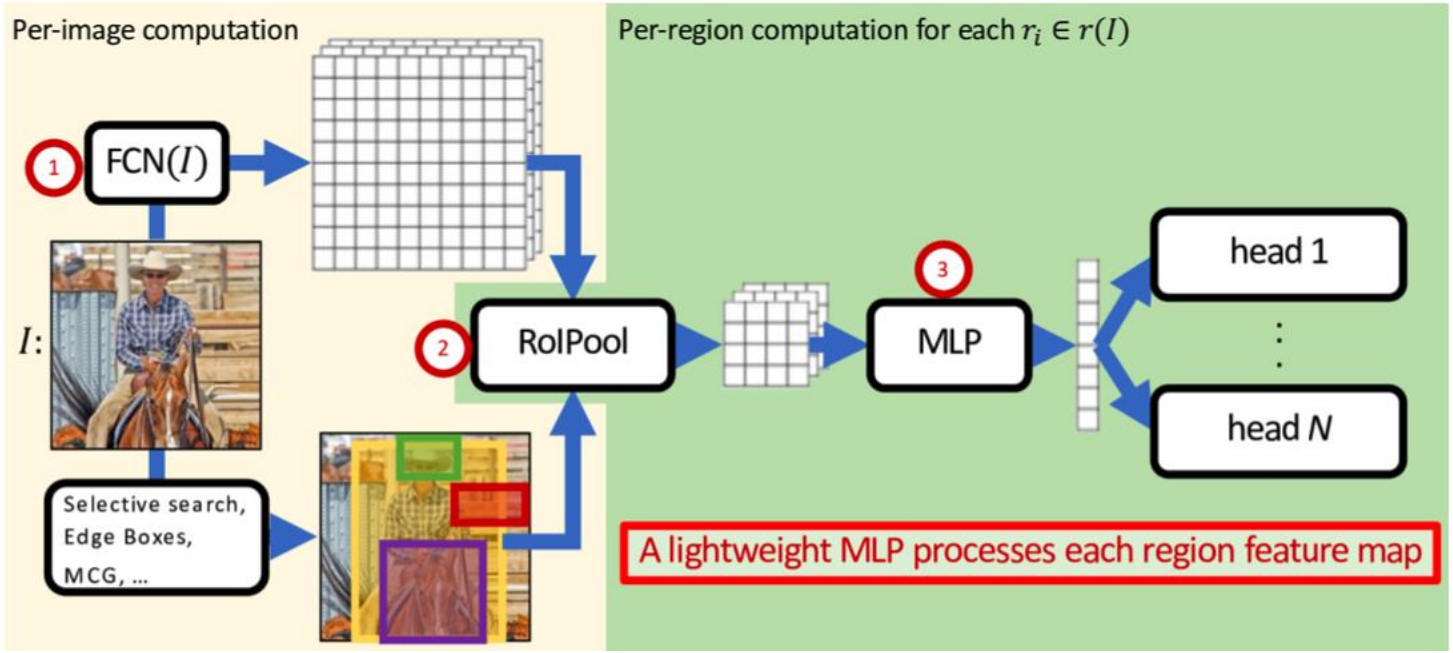
Fast R-CNN

Region-of-Interest (RoI) Pooling on each Proposal



https://arxiv.org/abs/1504.08083

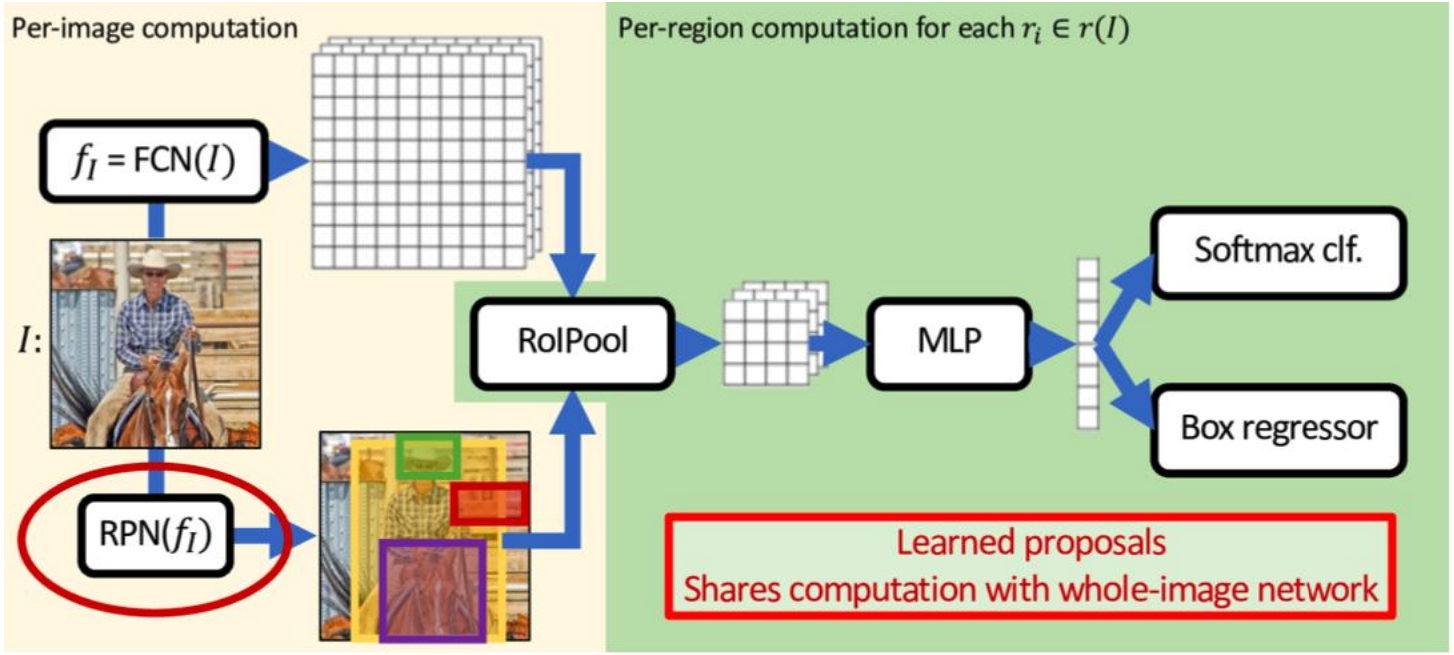
Fast R-CNN



https://arxiv.org/abs/1504.08083

Faster R-CNN

A faster implementation of Fast R-CNN



→ Comparison point (baseline) for single-stage detection

https://arxiv.org/abs/1506.01497

Single-stage detection

YOLO family models : You Only Look Once

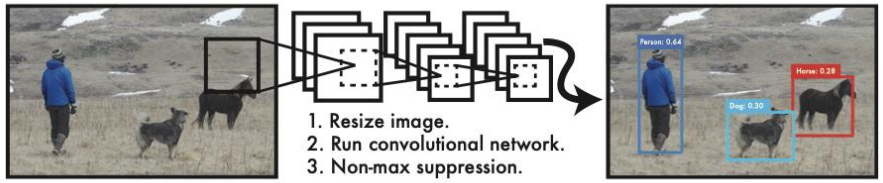


Figure 1: The YOLO Detection System. Processing images with YOLO is simple and straightforward. Our system (1) resizes the input image to 448 × 448, (2) runs a single convolutional network on the image, and (3) thresholds the resulting detections by the model's confidence.

→ A breakthrough in object detection:
just one forward pass of a CNN on the image

→ YOLO (v1) published in 2016

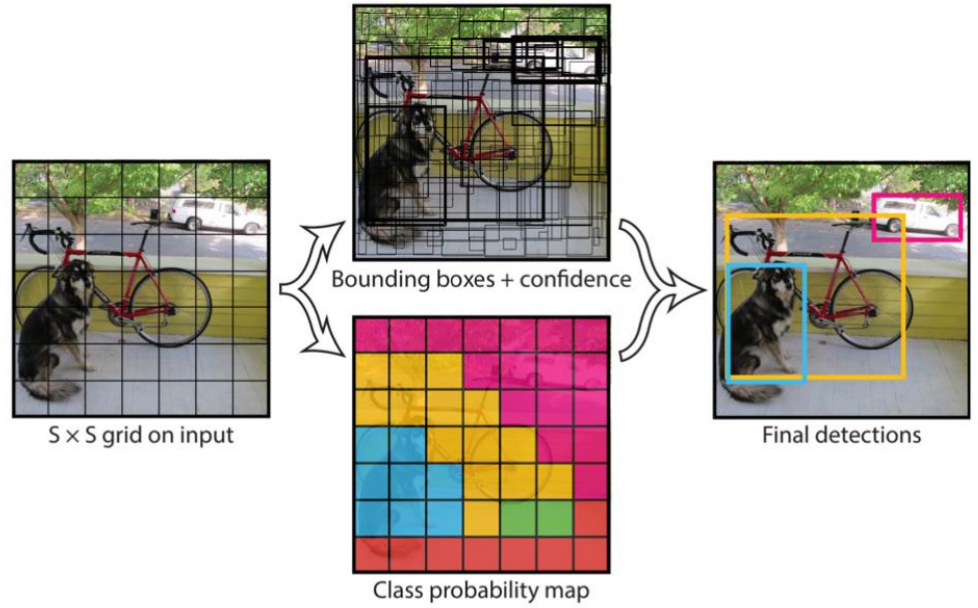
→ YOLO v3 very widely used (2018)

→ Now we find up to YOLO v8

How does it work?

→ Regress boxes and predict class probability in a single stage

→ Faster, but worse performance due to subsampling of output space



Yolo v1: https://arxiv.org/pdf/1506.02640.pdf
Yolo v3: https://arxiv.org/abs/1804.02767

A new paradigm shift?

Use of Transformers for Detection

→ DETR: DEtection TRansformer

→ End-to-end object detection using transformers

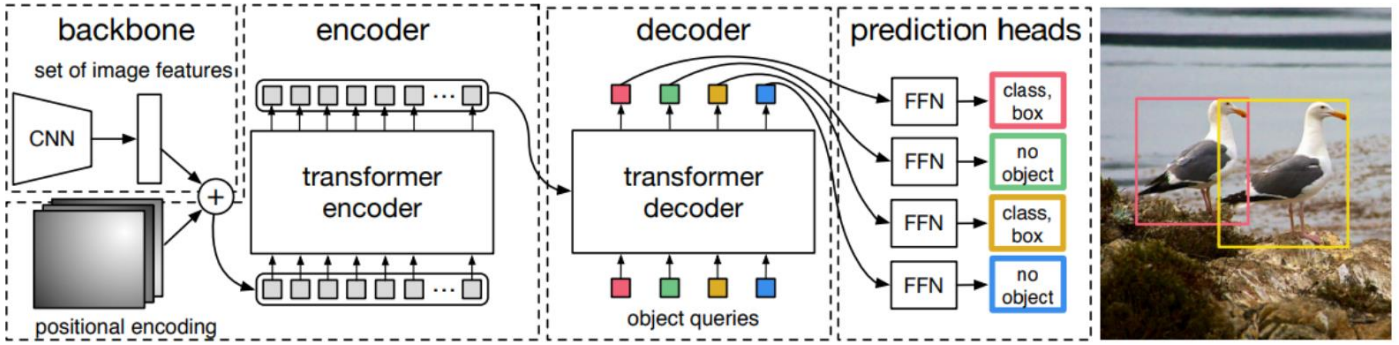


Fig. 2: DETR uses a conventional CNN backbone to learn a 2D representation of an input image. The model flattens it and supplements it with a positional encoding before passing it into a transformer encoder. A transformer decoder then takes as input a small fixed number of learned positional embeddings, which we call *object queries*, and additionally attends to the encoder output. We pass 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.

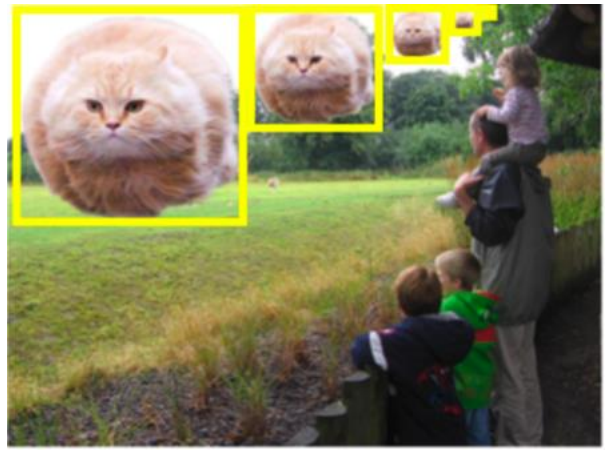
https://arxiv.org/abs/2005.12872

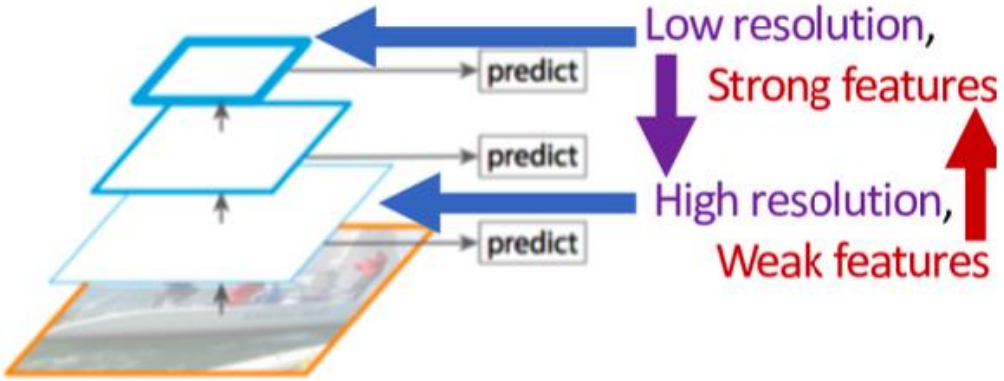
Feature Pyramid Network (FPN)

Improving scale equivariance

→ Detectors need to classify and localize over a wide range of scales

→ Building a pyramid of features





Summary

Today's lesson

→ Object detection basics

→ From classical to deep learning strategies

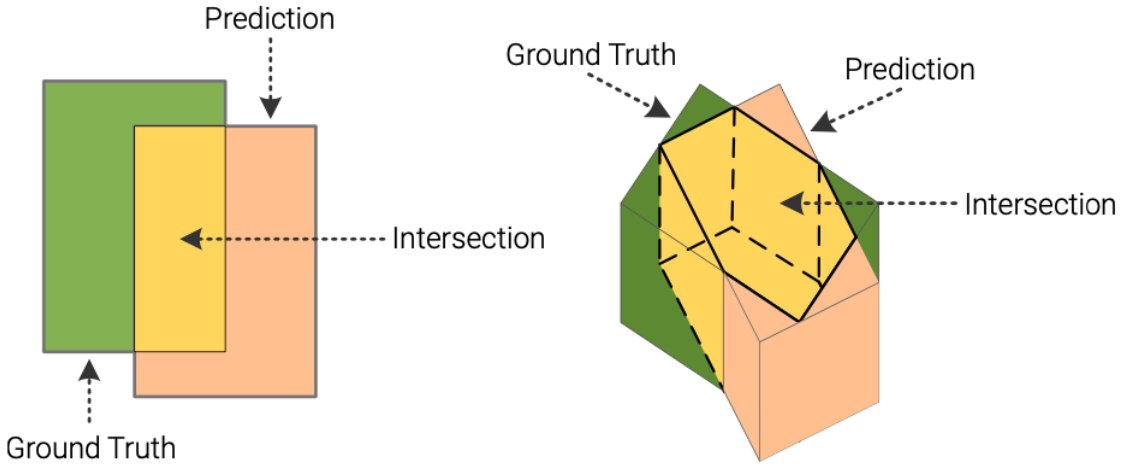
→ **3D Object detection**

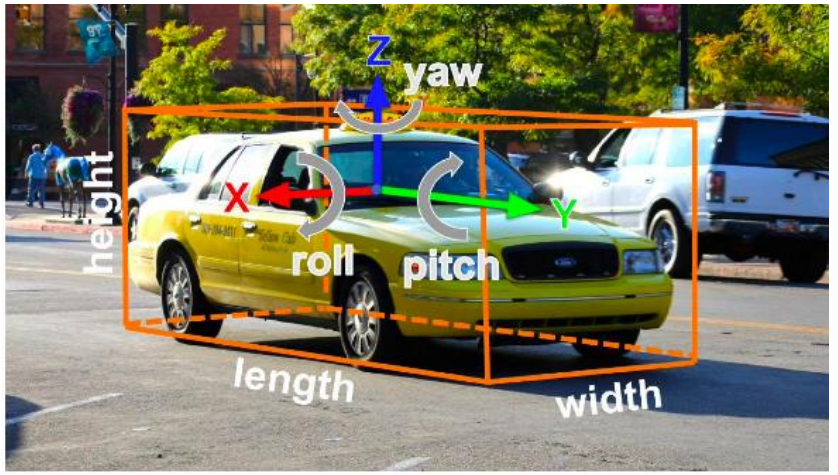
3D object detection

Comparison vs 2D object detection

→ In 3D detection we predict the bounding box for objects of interest

→ 3D IoU can be calculated in a similar way to 2D IoU





→ Much harder than 2D!!

→ Simplification: assume zero roll and pitch

Different possibilities for input modality

And difficulty will depend on input modality

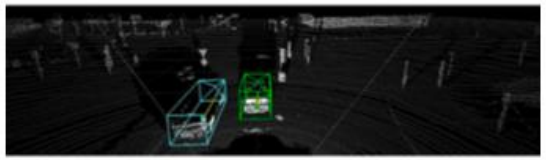
→ Regressing 3D boxes from monocular images requires good object size views

→ Stereo information helps localizing the box in 3D space

→ Lidars are sparse but provide good 3D accuracy

→ Lidar and 2D images are complementary and work well together.

Image



Lidar

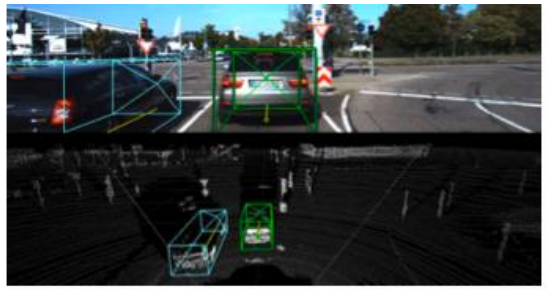


Image + Lidar

Image-based 3D object detection

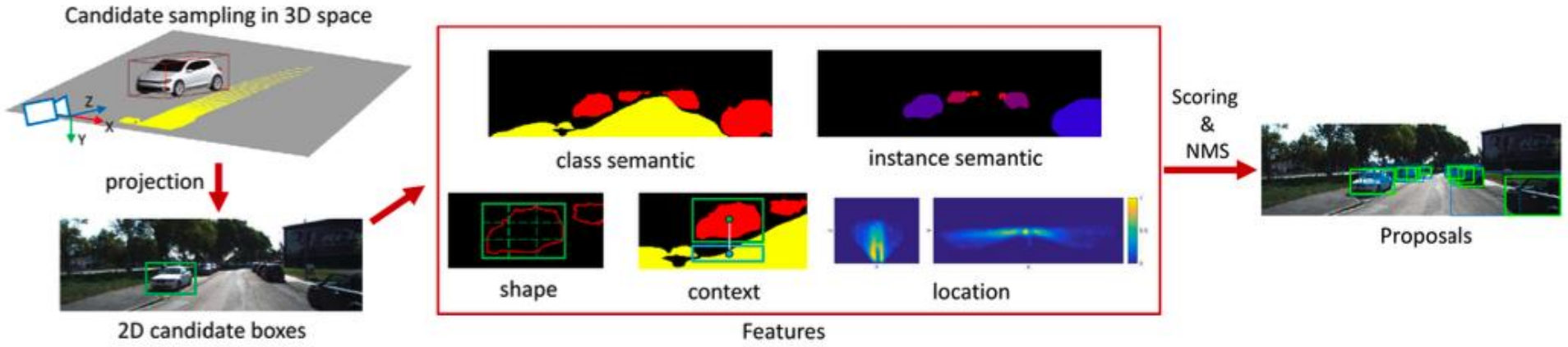
Monocular 3D object detection

→ Getting candidate 3D boxes on the ground plane in non-road areas

→ Project the boxes into image plane

→ Feature-based scoring

→ Final scoring and 2D refinement using Fast R-CNN.



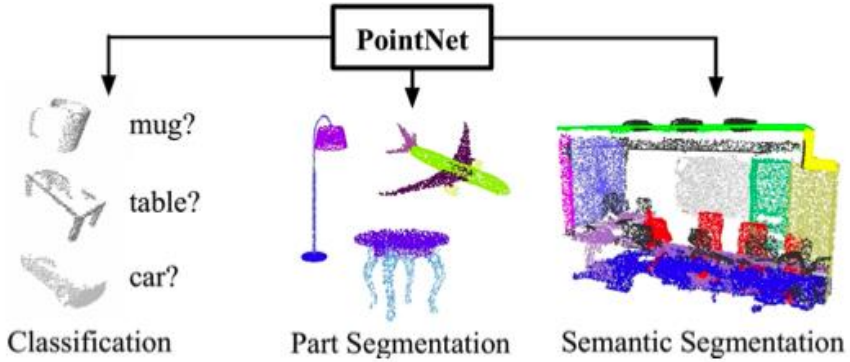
Lidar-based object detection

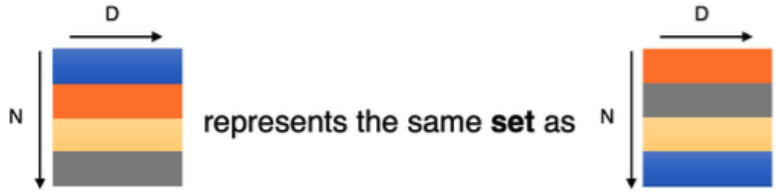
PointNet: Learning with Point Clouds

→ Challenges:

- Accomodating point clouds of **arbitrary size**
- Invariant to all (N!) **permutations**
- Invariant to **rotations** of the point cloud

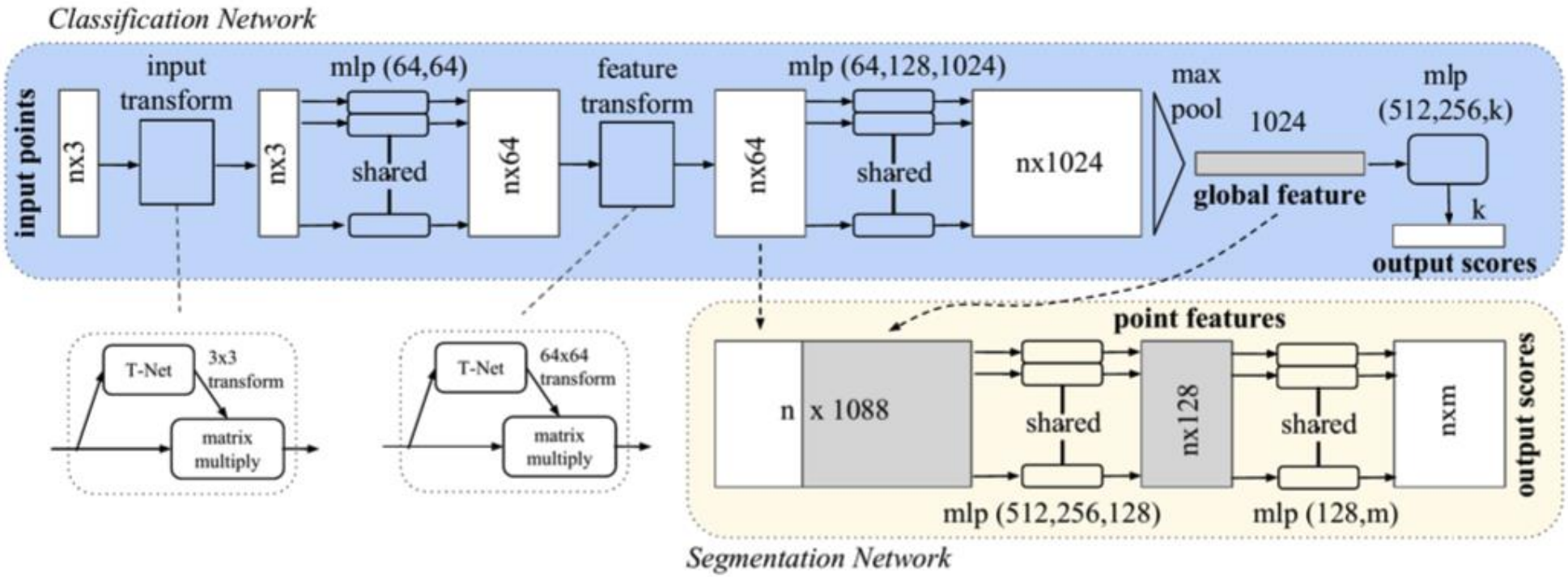
→ Way more complex NNs : using MLPs





PointNet

Applying MLP per point, followed by max-pooling and a final (global) MLP

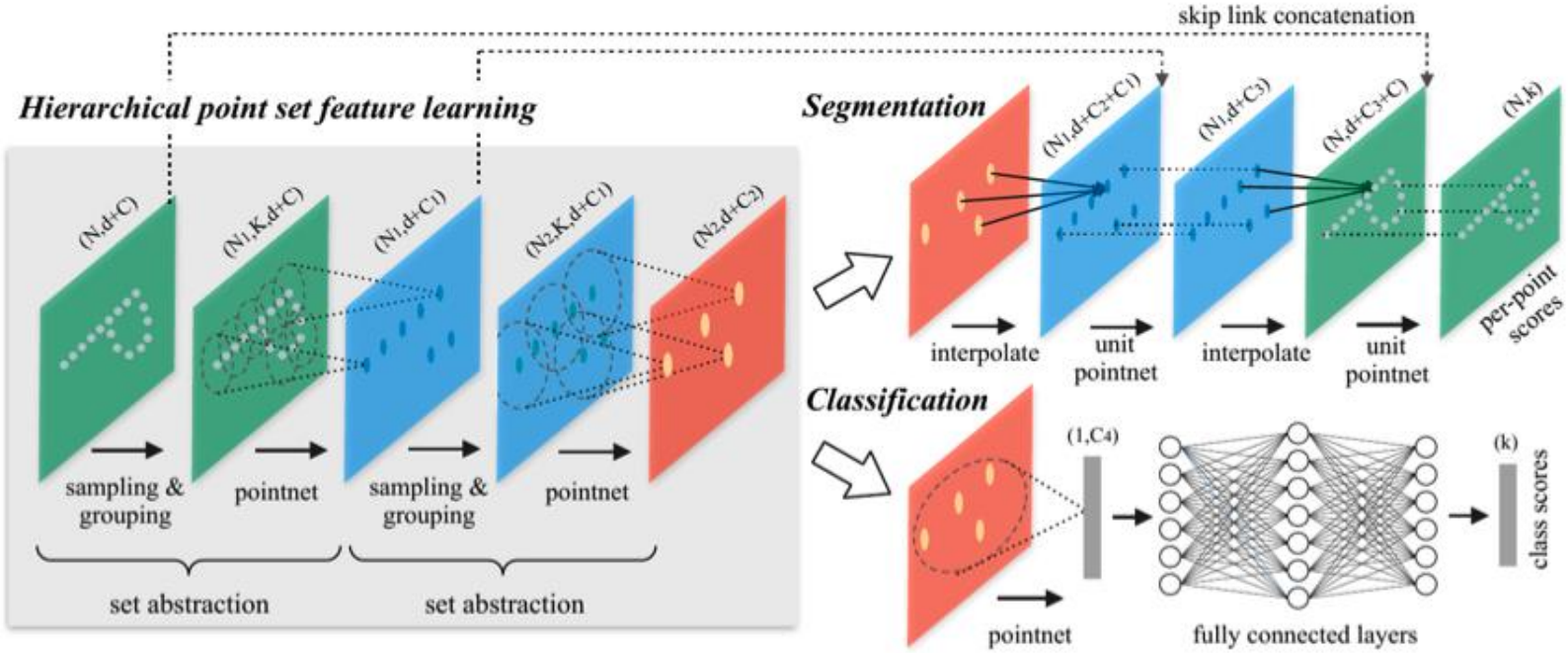


https://arxiv.org/abs/1612.00593

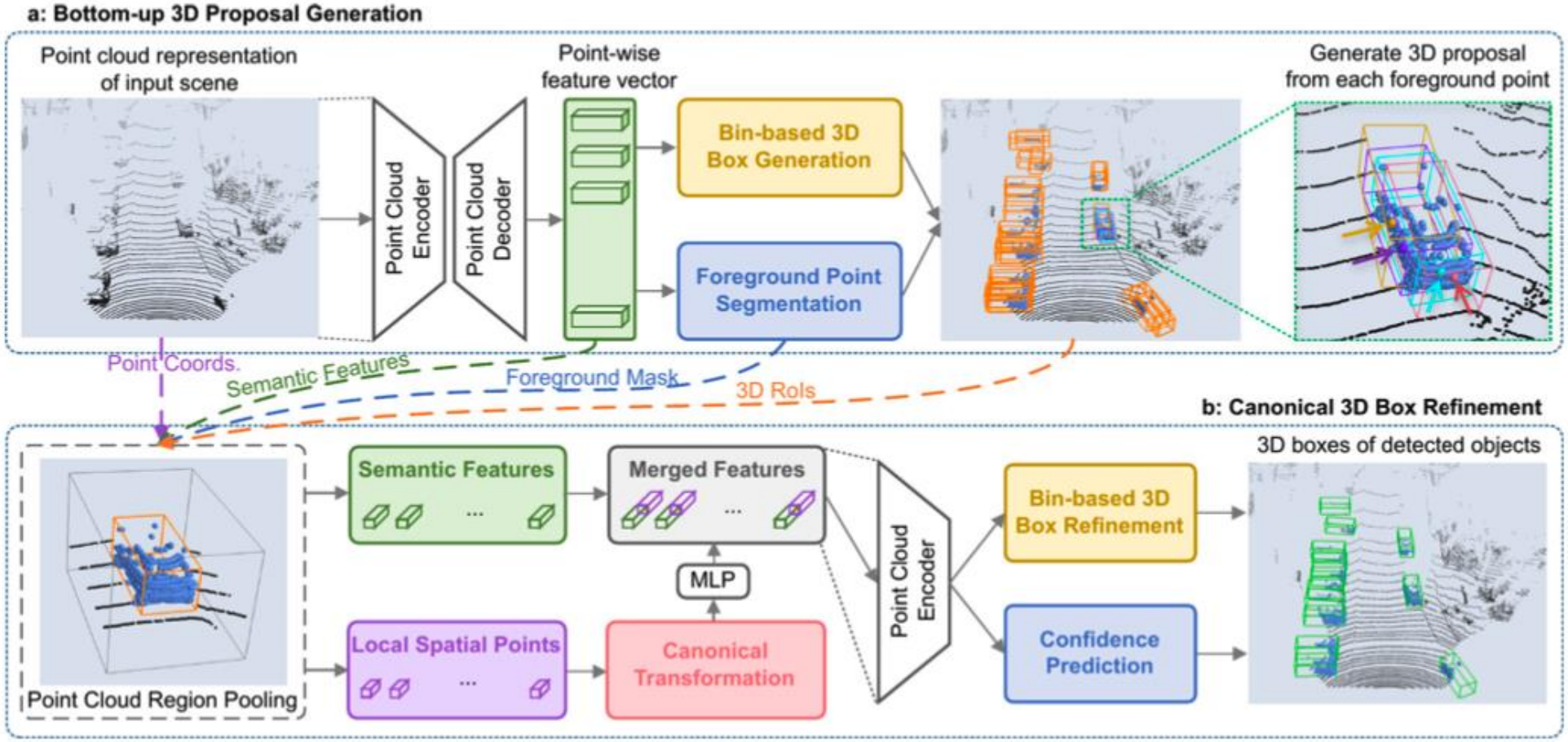
PointNet++

Solves the issue of PointNet not capturing local structures

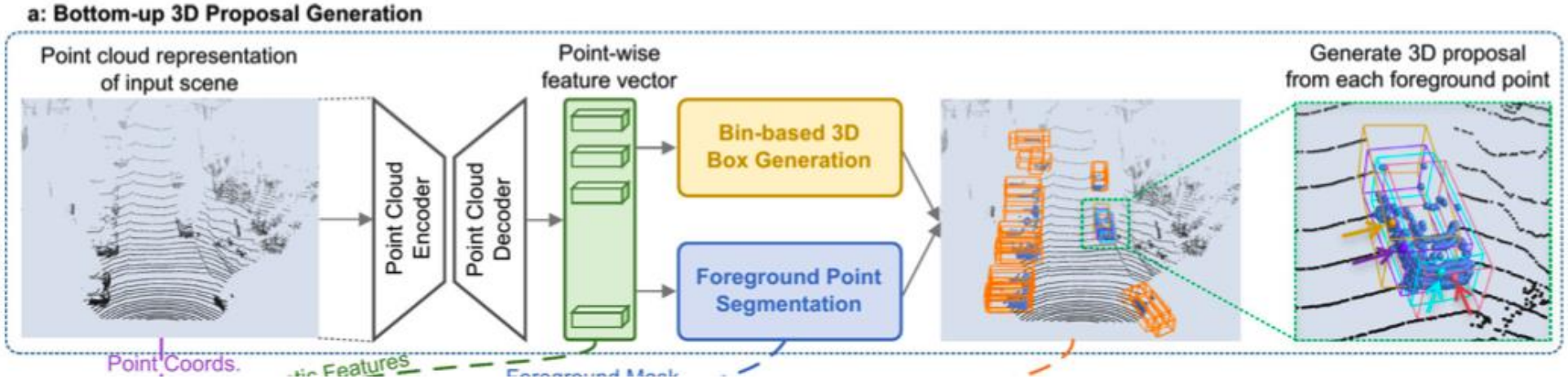
→ Applying PointNet **recursively** on **nested partitions** of the point set



PointRCNN: Faster R-CNN for Point Clouds



PointRCNN: Faster R-CNN for Point Clouds



→ Backbone to learn point-wise features with PointNet++

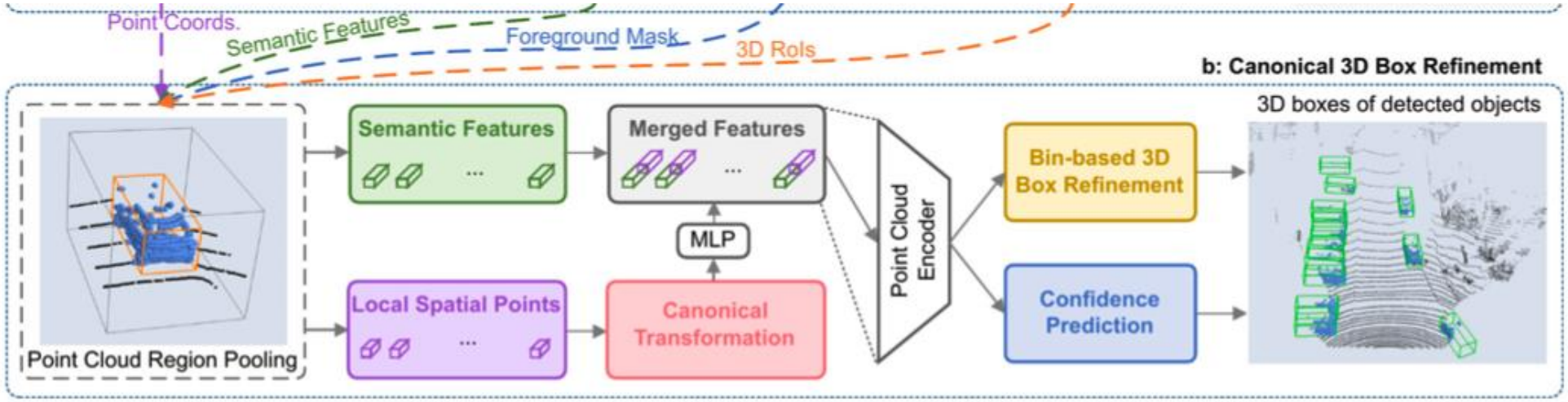
→ 3D box proposal regression (Faster R-CNN) and foreground segmentation

PointRCNN: Faster R-CNN for Point Clouds

→ Pooling of 3D point features based on 3D box proposals

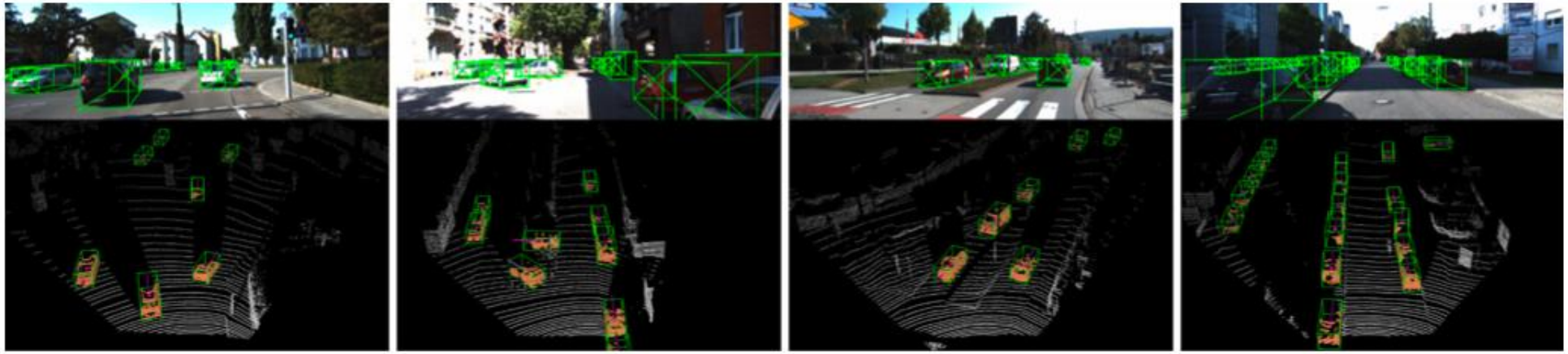
→ Transform pooled points to 3D box coordinates

→ 3D box refinement



PointRCNN: Faster R-CNN for Point Clouds

Outperforms multi-modal methods… except if pedestrians are small



Method	Modality	Car (IoU=0.7)			Pedestrian (IoU=0.5)			Cyclist (IoU=0.5)		
		Easy	Moderate	Hard	Easy	Moderate	Hard	Easy	Moderate	Hard
MV3D [4]	RGB + LiDAR	71.09	62.35	55.12	-	-	-	-	-	-
UberATG-ContFuse [17]	RGB + LiDAR	82.54	66.22	64.04	-	-	-	-	-	-
AVOD-FPN [14]	RGB + LiDAR	81.94	71.88	66.38	50.80	42.81	**40.88**	64.00	52.18	46.61
F-PointNet [25]	RGB + LiDAR	81.20	70.39	62.19	**51.21**	**44.89**	40.23	71.96	56.77	50.39
VoxelNet [43]	LiDAR	77.47	65.11	57.73	39.48	33.69	31.51	61.22	48.36	44.37
SECOND [40]	LiDAR	83.13	73.66	66.20	51.07	42.56	37.29	70.51	53.85	46.90
Ours	LiDAR	**85.94**	**75.76**	**68.32**	49.43	41.78	38.63	**73.93**	**59.60**	**53.59**

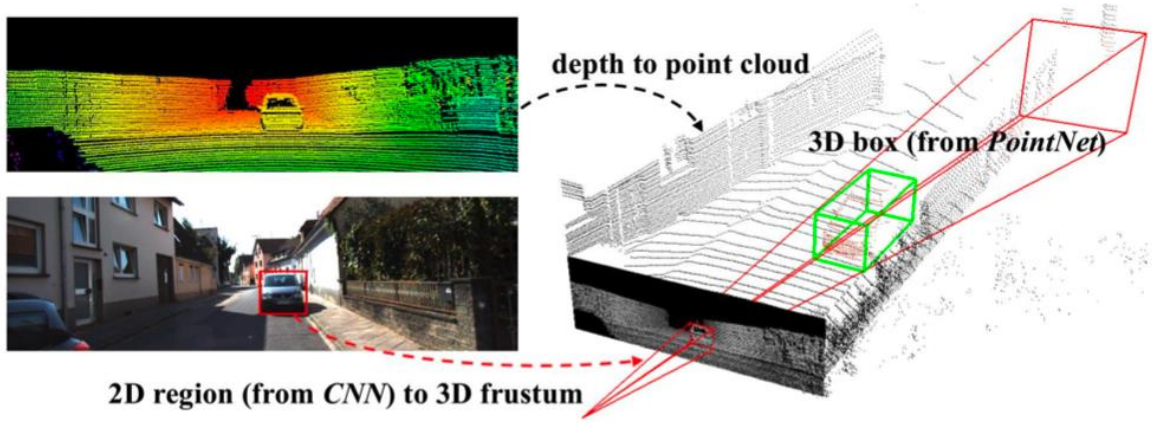
Multi-modal methods

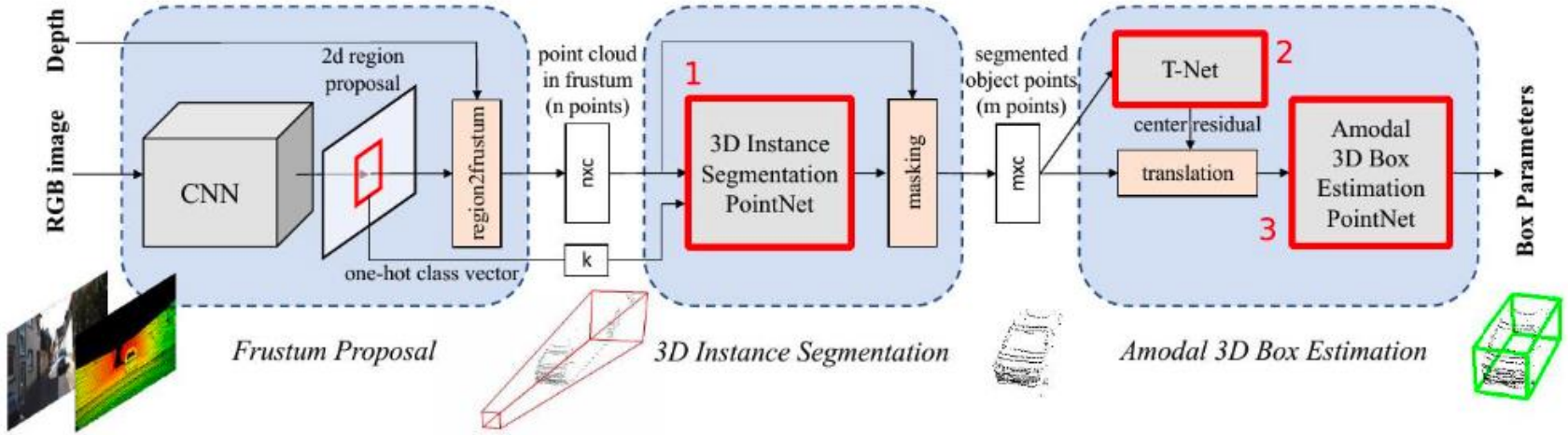
Frustrum PointNets: Using 2D cameras plus Lidar

→ Generate 2D object proposals in the RGB image

→ Extract 3D point cloud from Lidar

→ Predict 3D bounding box via PointNet





Hes·so
Haute Ecole Spécialisée
de Suisse occidentale