Object detection

Machine Learning 2023/2024

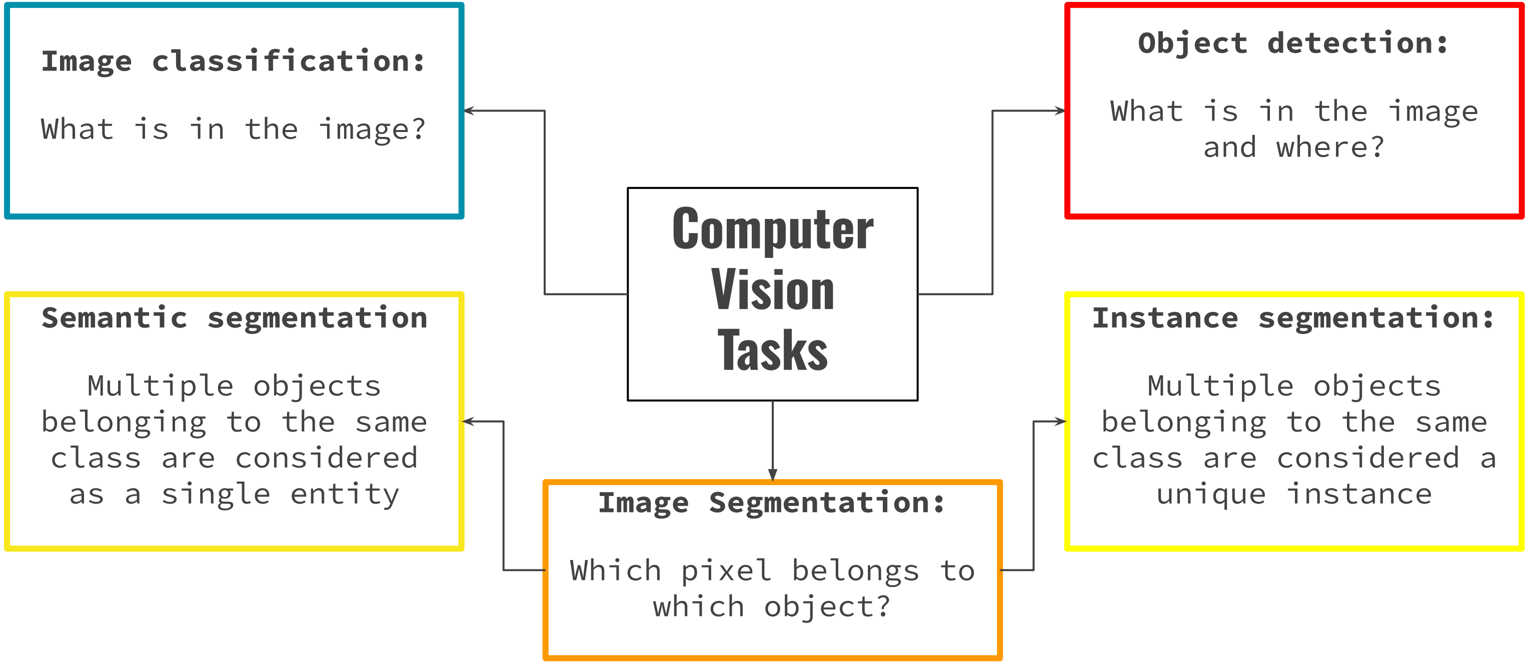
Damiano Brunori, Michela Proietti

SAPIENZA
UNIVERSITÀ DI ROMA

Outline

- Problem’s overview
- Examples of architectures and applications
  - YOLO
  - Faster R-CNN
  - Mask R-CNN
  - Detection Transformers
- Open challenges

Computer Vision Tasks (1)



Computer Vision Tasks

Image classification:

What is in the image?

Object detection:

What is in the image and where?

Image Segmentation:

Which pixel belongs to which object?

Semantic segmentation

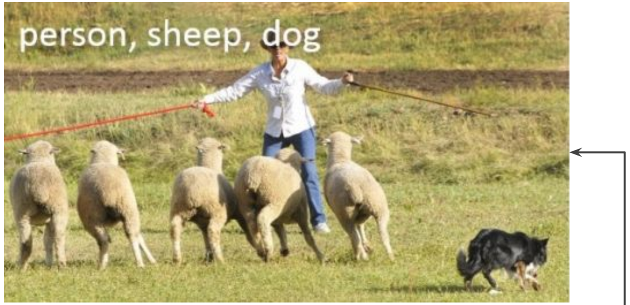
Multiple objects belonging to the same class are considered as a single entity

Instance segmentation:

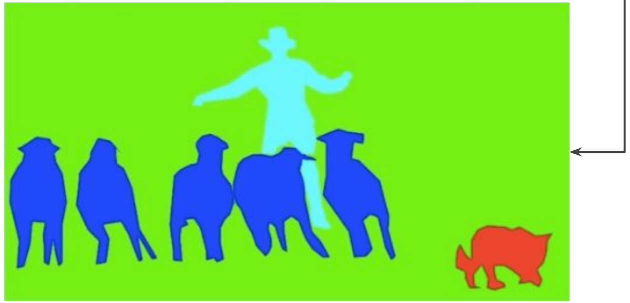
Multiple objects belonging to the same class are considered a unique instance

Computer Vision Tasks (2)

Image classification

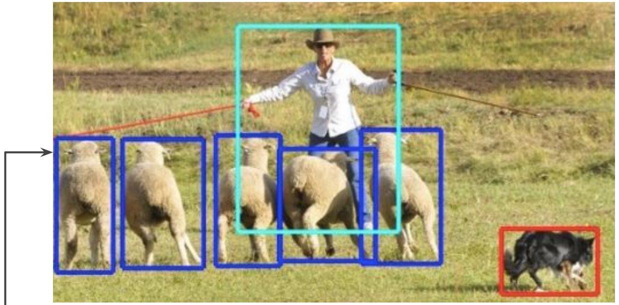


Semantic segmentation

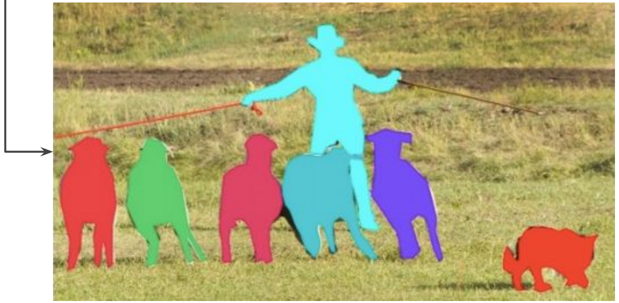


Computer Vision Tasks

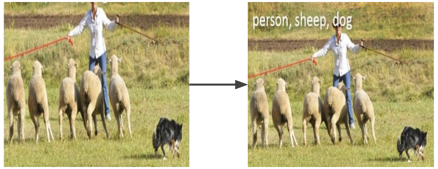
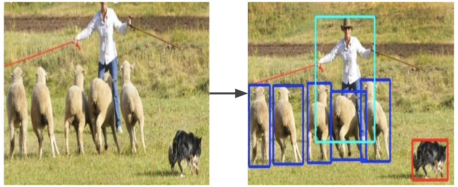
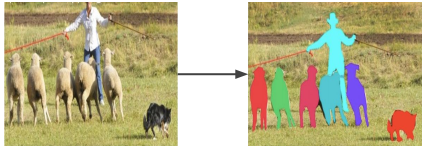
Object detection



Instance segmentation



Computer Vision Tasks (3)

Task	Output	Example
Classification	Probability of the class C given the image I → $P(C\|I)$	
Object Detection	Localization of bounding boxes $b_i = \{x,y,w,h\}$ AND probability of class C given the bounding box b_i and the image I → $P(C\|b_i, I)$	
Image Segmentation	Probability of class $C_{x,y}$ for pixel (x,y) of the image I → $P(C_{x,y}\|I)$	

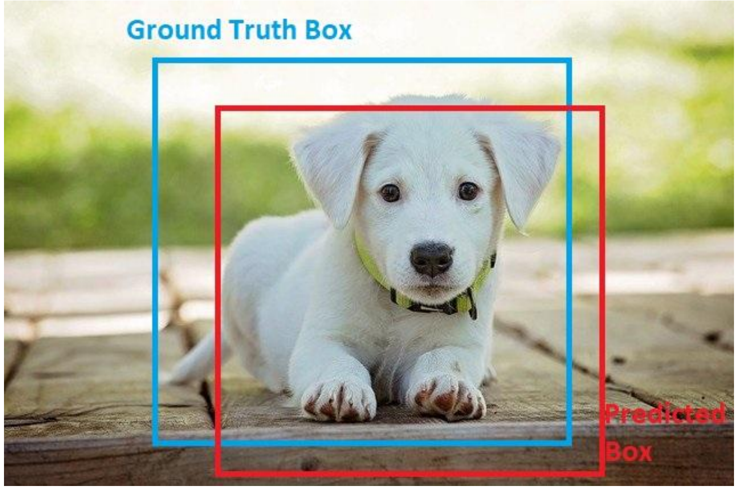
Object Detection

- Multiple classes in the same image
- Multiple instances of the same class in the same image
- Localization of the bounding boxes
- Prediction of confidence scores, classes and bounding boxes
- Determine a true or false positive for each detection
- Evaluation metric based on Intersection Over Union (IOU) for bounding boxes:

  $$\text{Area}(GT \cap D) \,/\, \text{Area}(GT \cup D) > 0.5$$

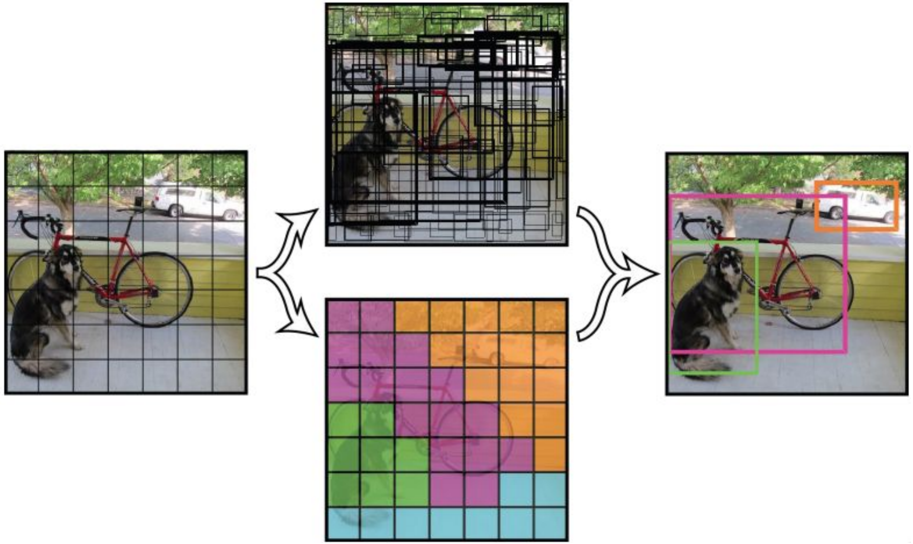
- Consider only one true positive for multiple detections of the same GT

GT → Ground Truth; D → Detected



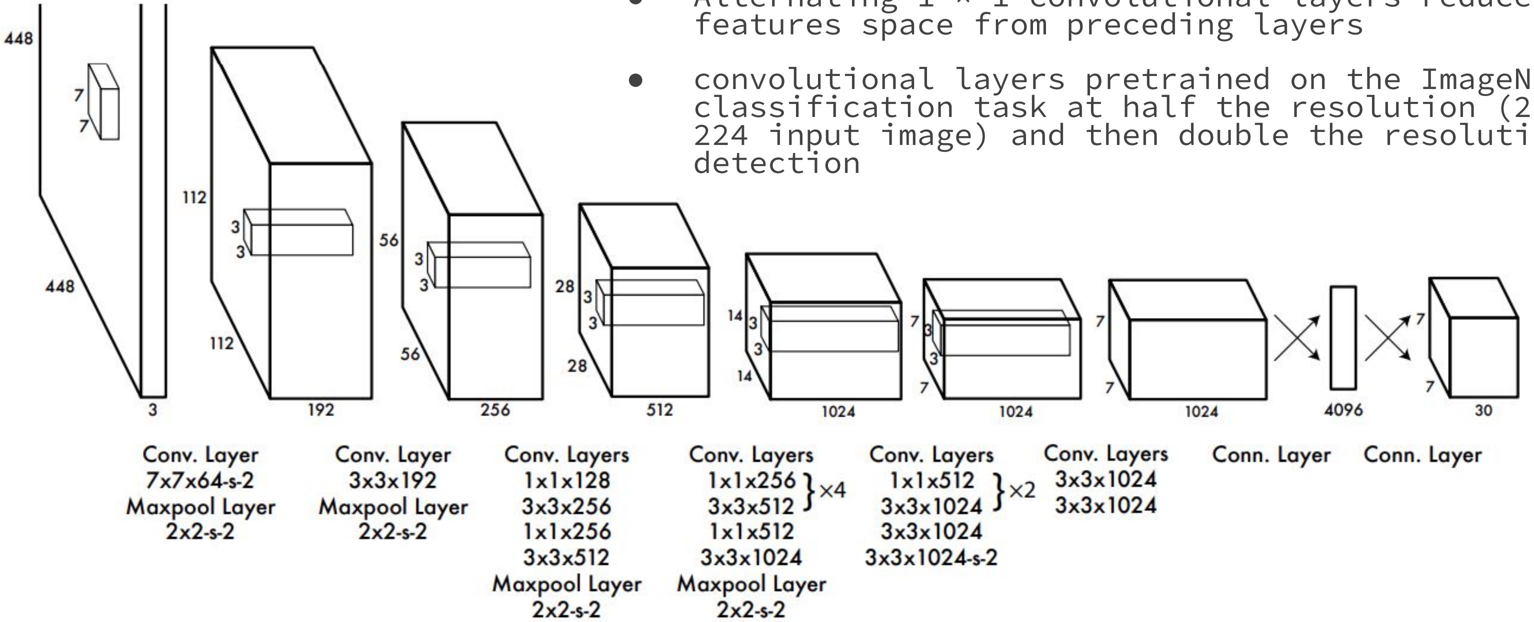
YOLO (You Only Look Once)

- Predicts all objects in a single forward pass (indeed, it looks once)
- Divide the image into a SxS coarse grid and directly predict class label and a few candidate boxes for each grid cell
- A grid cell will detect the object whose center falls in the considered grid cell
- Each grid cell predicts:
    - bounding boxes
    - confidence scores as P(Obj)*IOU
    - Conditional class probability as $P(C_i|Obj)$



YOLO Architecture

- 24 convolutional layers followed by 2 fully connected layers
- Alternating 1 × 1 convolutional layers reduce the features space from preceding layers
- convolutional layers pretrained on the ImageNet classification task at half the resolution (224 × 224 input image) and then double the resolution for detection



YOLO Loss (1)

Regression loss

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

Confidence loss

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

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

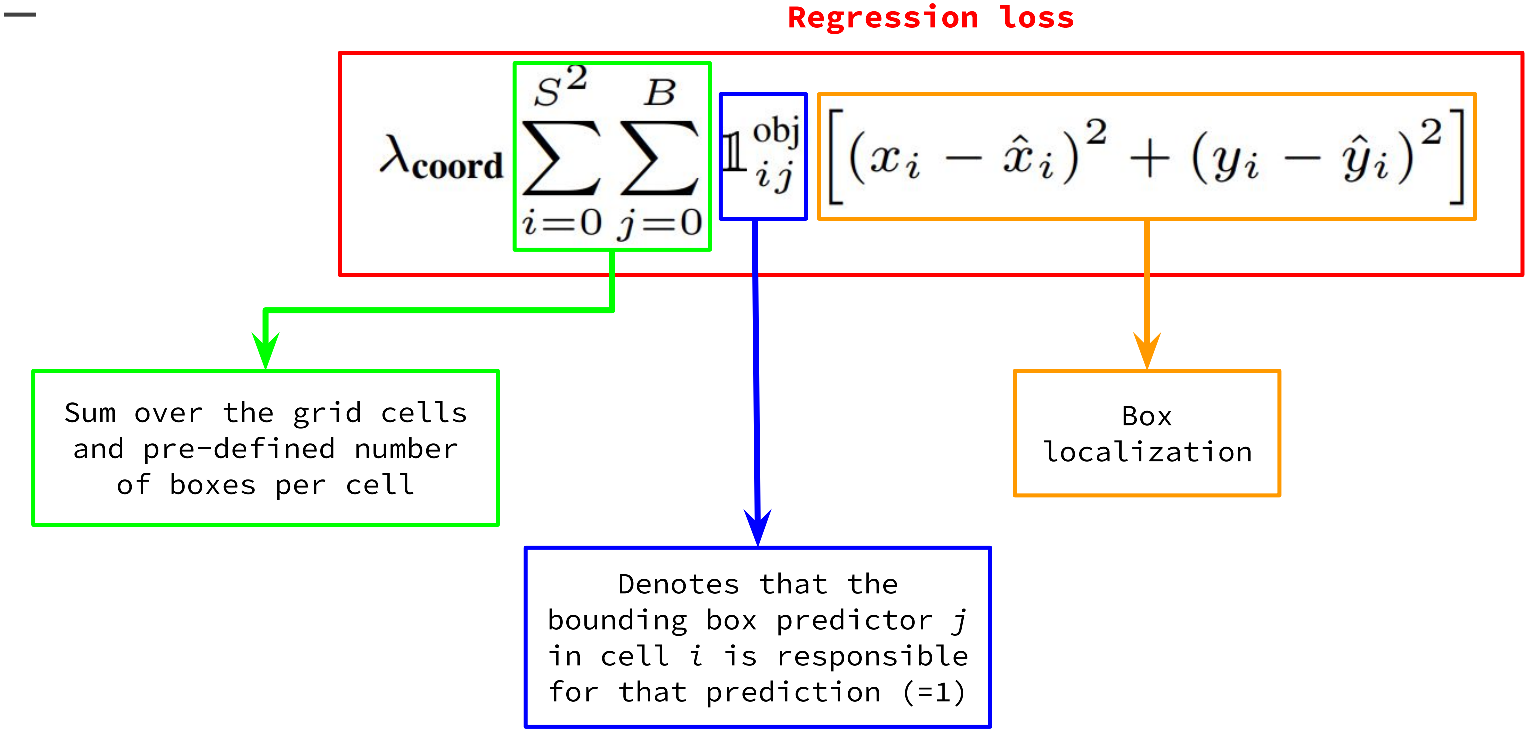
Classification loss

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

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

λ_{coord}, λ_{noobj} are coefficients to regulate each loss component

YOLO Loss (2)



Regression loss

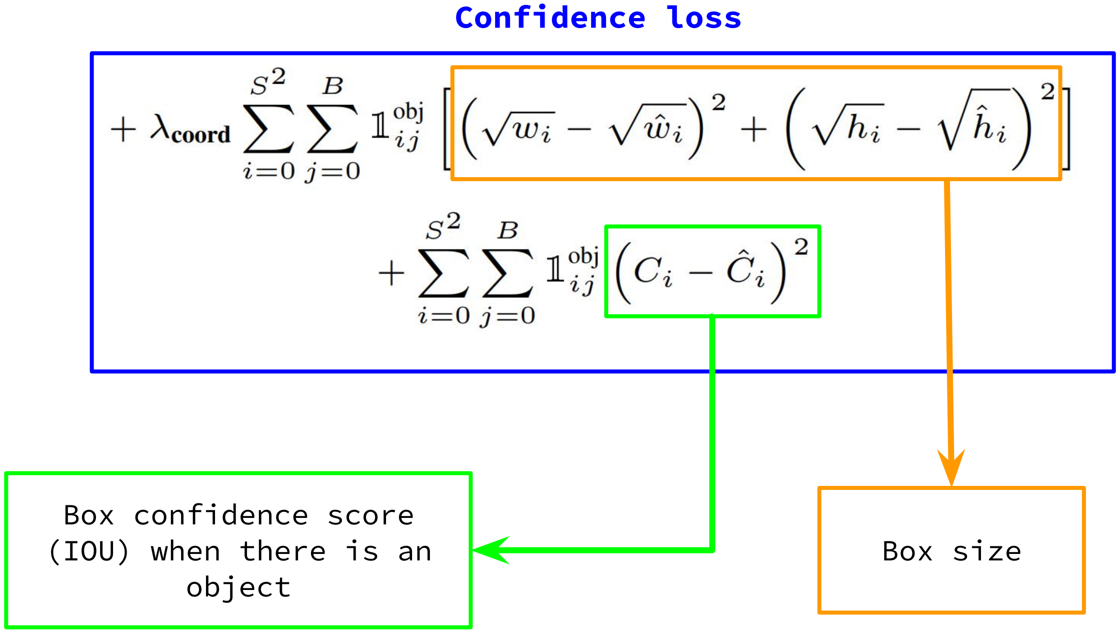
$$\lambda_{\textbf{coord}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{obj}} \left[(x_i - \hat{x}_i)^2 + (y_i - \hat{y}_i)^2 \right]$$

Sum over the grid cells and pre-defined number of boxes per cell

Denotes that the bounding box predictor *j* in cell *i* is responsible for that prediction (=1)

Box localization

YOLO Loss (3)



Confidence loss

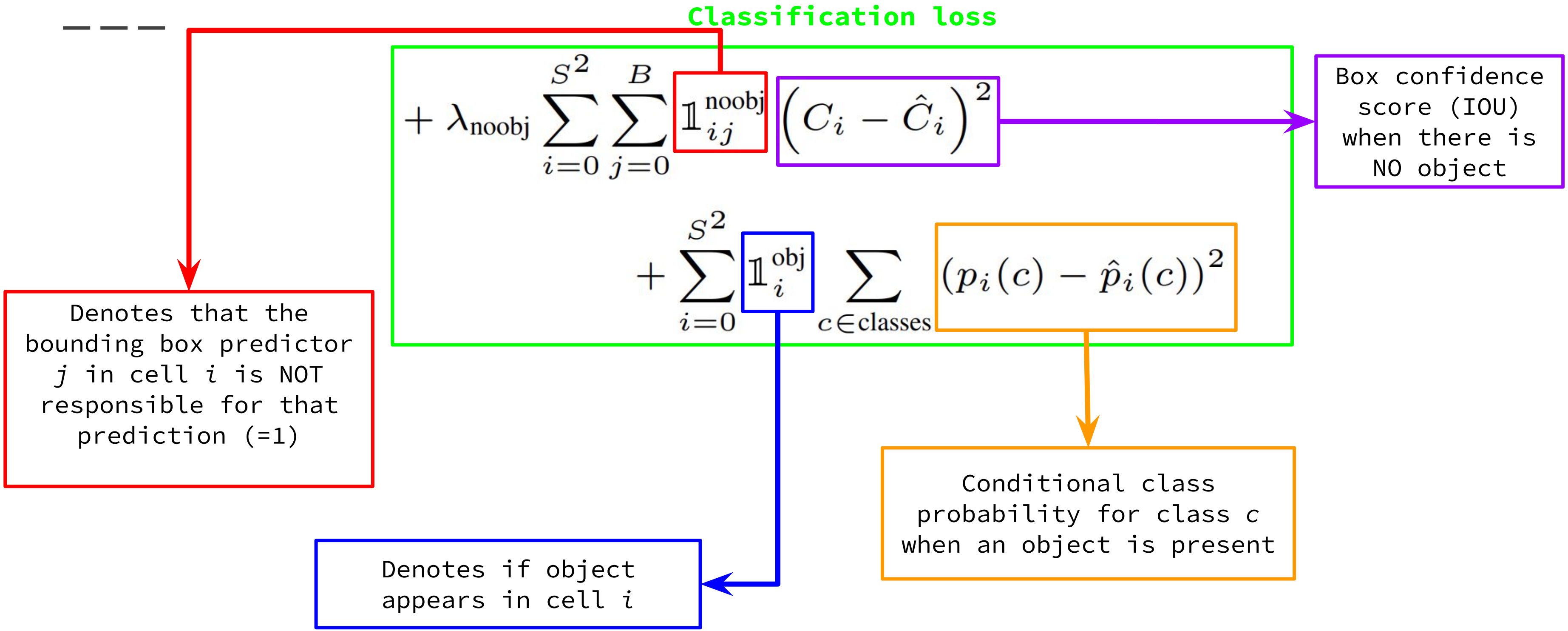
$$+ \lambda_{\textbf{coord}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{obj}} \left[\left(\sqrt{w_i} - \sqrt{\hat{w}_i} \right)^2 + \left(\sqrt{h_i} - \sqrt{\hat{h}_i} \right)^2 \right]$$

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

Box confidence score (IOU) when there is an object

Box size

YOLO Loss (4)



Classification loss

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

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

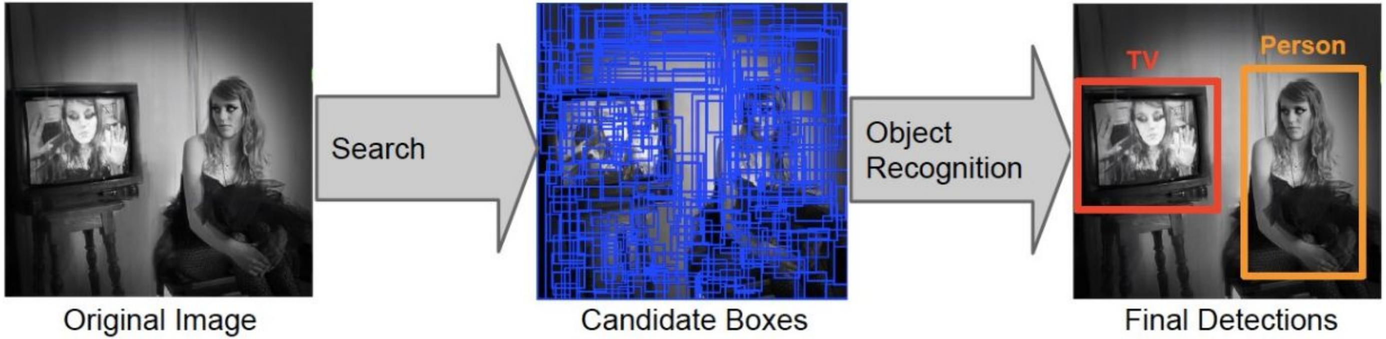
Denotes that the bounding box predictor *j* in cell *i* is NOT responsible for that prediction (=1)

Box confidence score (IOU) when there is NO object

Denotes if object appears in cell *i*

Conditional class probability for class c when an object is present

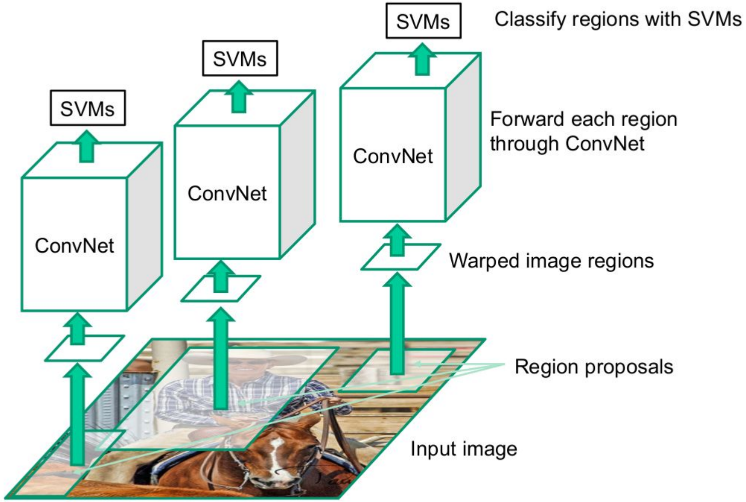
Faster RCNN (Background)

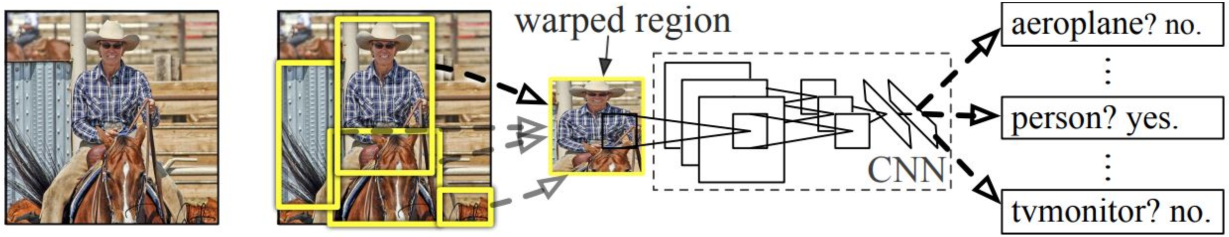


- Generate and evaluate hundreds of region
- The proposal detection can be category-specific or category-independent, hand-crafted or trained
- Classifier can be slower but more powerful

Faster RCNN (RCNN Background)

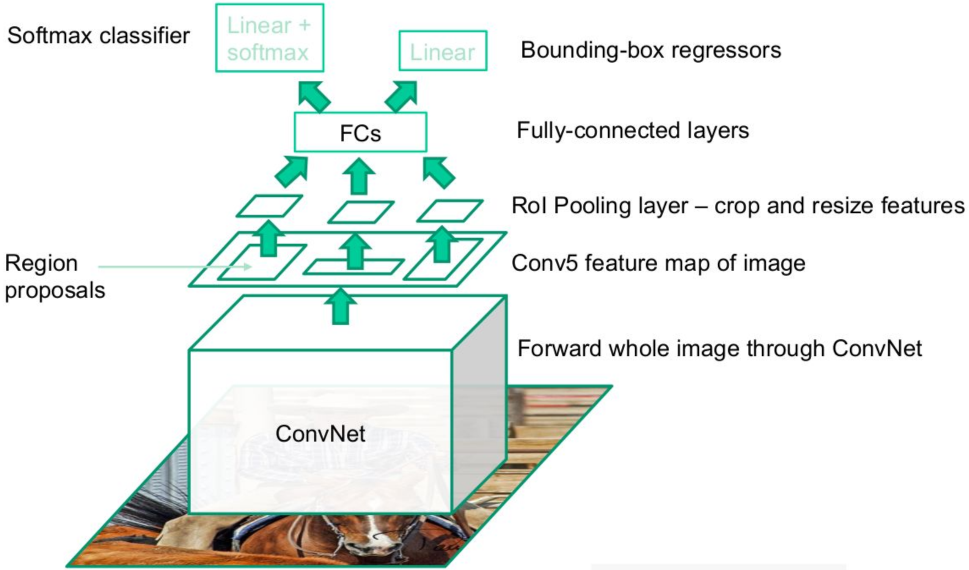
- Accurate
- Any architecture (e.g. AlexNet) can be *plugged in*
- Training and detection are slow (2000 region proposals per image)





Faster RCNN (Fast RCNN Background 1)

- Region of Interest (RoI) pooling:
    - Crop and resample a fixed-size feature that represents a RoI out of the outputs of the last conv. layer
    - is a type of max pooling where the pool size depends on the input size
- ROI pooling uses a single feature map for all the regions: this warps ROIs into one single layer
- ROI pooling layer uses max pooling to convert the features



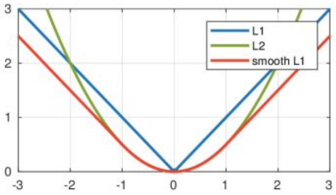
Faster RCNN (Fast RCNN Background 2)

- Multi-task loss for ground truth class *y*, the predicted class probabilities *P(y)*, the ground truth box *b*, and the predicted box $\hat{b}$:

$$L(y,P(y),b,\hat{b}) = -\log P(y) + \lambda[y\geq 1]L_{reg}(b,\hat{b})$$

- $-\log P(y)$ is the softmax loss; L_{reg} is the regression loss; λ is a hyperparameter that determines the relative weight of the regression loss VS the classification loss to the overall loss; *[y≥1]* evaluates to 1 when y≥1 and 0 otherwise.

- The regression loss is defined as the *smooth L1* loss on top of log space offsets relative to proposal:



$$L_{reg}(b,\hat{b}) = \Sigma_{i=\{x,y,,w,h\}} smooth_{L1}(b_i - \hat{b}_i$$

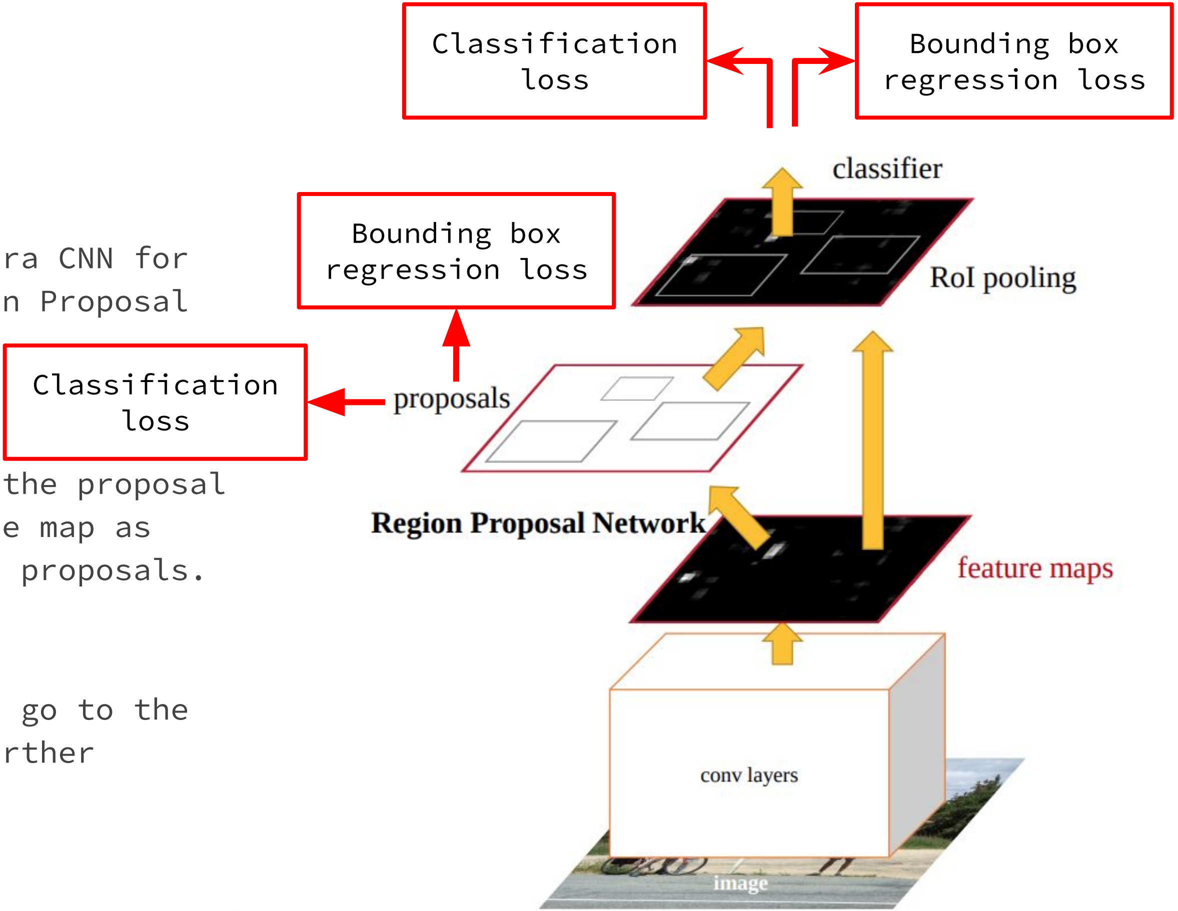
$$\text{smooth}_{L_1}(x) = \begin{cases} 0.5x^2 & \text{if } |x| < 1 \\ |x| - 0.5 & \text{otherwise} \end{cases}$$

Faster RCNN

- Faster R-CNN owns an extra CNN for regional proposal (Region Proposal Network)

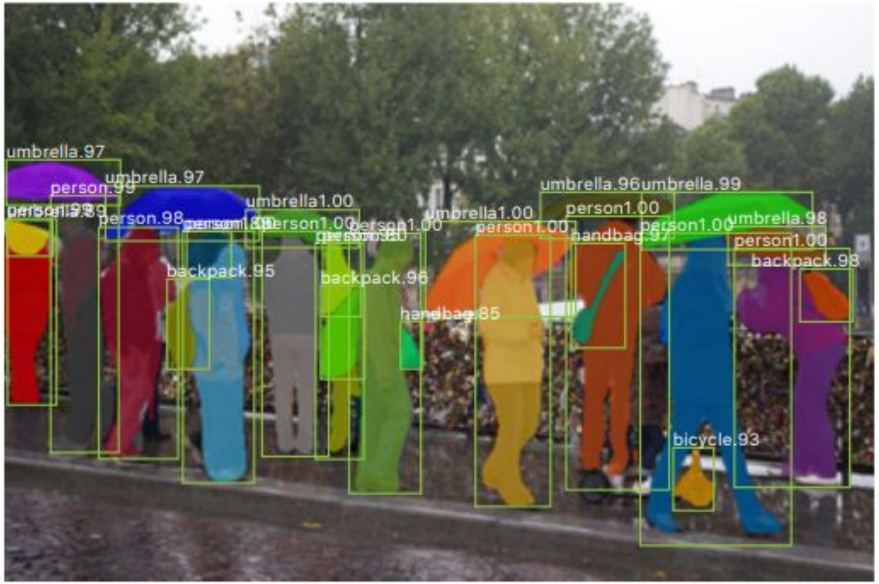
- In the training region, the proposal network takes the feature map as input and outputs region proposals.

- Finally, these proposals go to the ROI pooling layer for further procedure.



Mask RCNN

- Builds on top of Faster RCNN
- For each candidate object, it outputs:
  1. Discrete probability distribution per RoI
  2. Bounding-box regression offsets
  3. **Object masks**



1. $p = (p_0, \ldots, p_K)$
2. $t^k = \left(t^k_{\mathrm{x}}, t^k_{\mathrm{y}}, t^k_{\mathrm{w}}, t^k_{\mathrm{h}}\right)$
3. K mxm binary masks

Mask RCNN

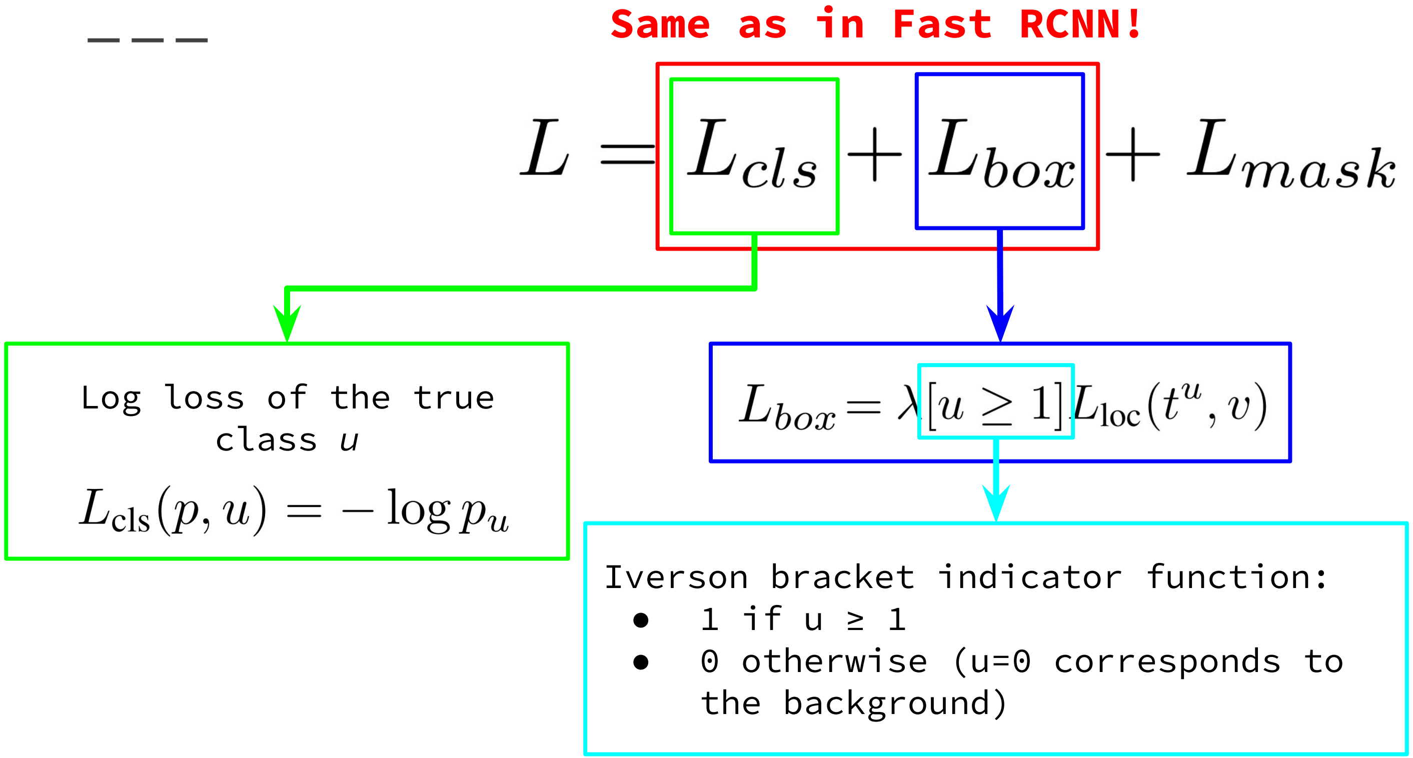
Same as in Fast RCNN!

$$L = \boxed{\boxed{L_{cls}} + L_{box}} + L_{mask}$$

Log loss of the true class u

$$L_{\text{cls}}(p, u) = -\log p_u$$

Mask RCNN



Same as in Fast RCNN!

$$L = \boxed{\boxed{L_{cls}} + \boxed{L_{box}}} + L_{mask}$$

Log loss of the true class *u*

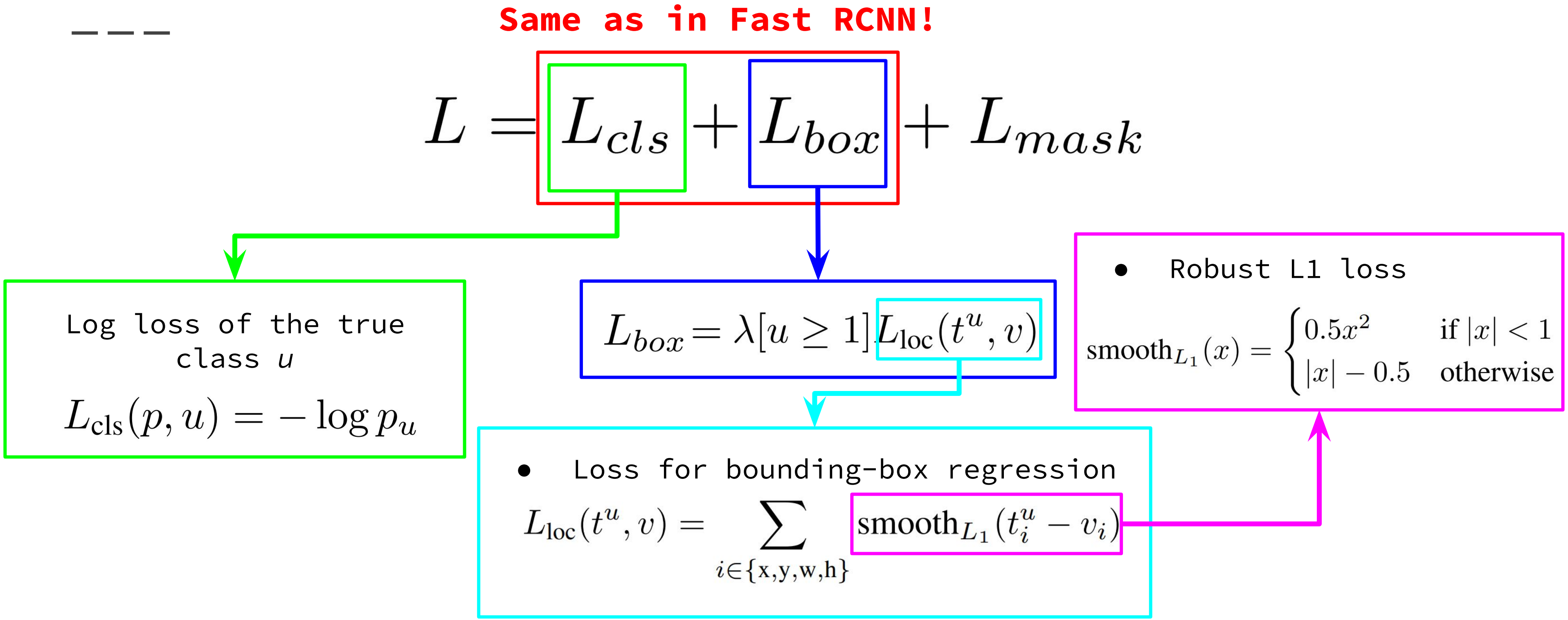
$$L_{\text{cls}}(p, u) = -\log p_u$$

$$L_{box} = \lambda\boxed{[u \geq 1]}L_{\text{loc}}(t^u, v)$$

Iverson bracket indicator function:

- 1 if u ≥ 1
- 0 otherwise (u=0 corresponds to the background)

Mask RCNN



Same as in Fast RCNN!

$$L = \boxed{\boxed{L_{cls}} + \boxed{L_{box}}} + L_{mask}$$

Log loss of the true class *u*

$$L_{\text{cls}}(p, u) = -\log p_u$$

$$L_{box} = \lambda[u \geq 1] \boxed{L_{\text{loc}}(t^u, v)}$$

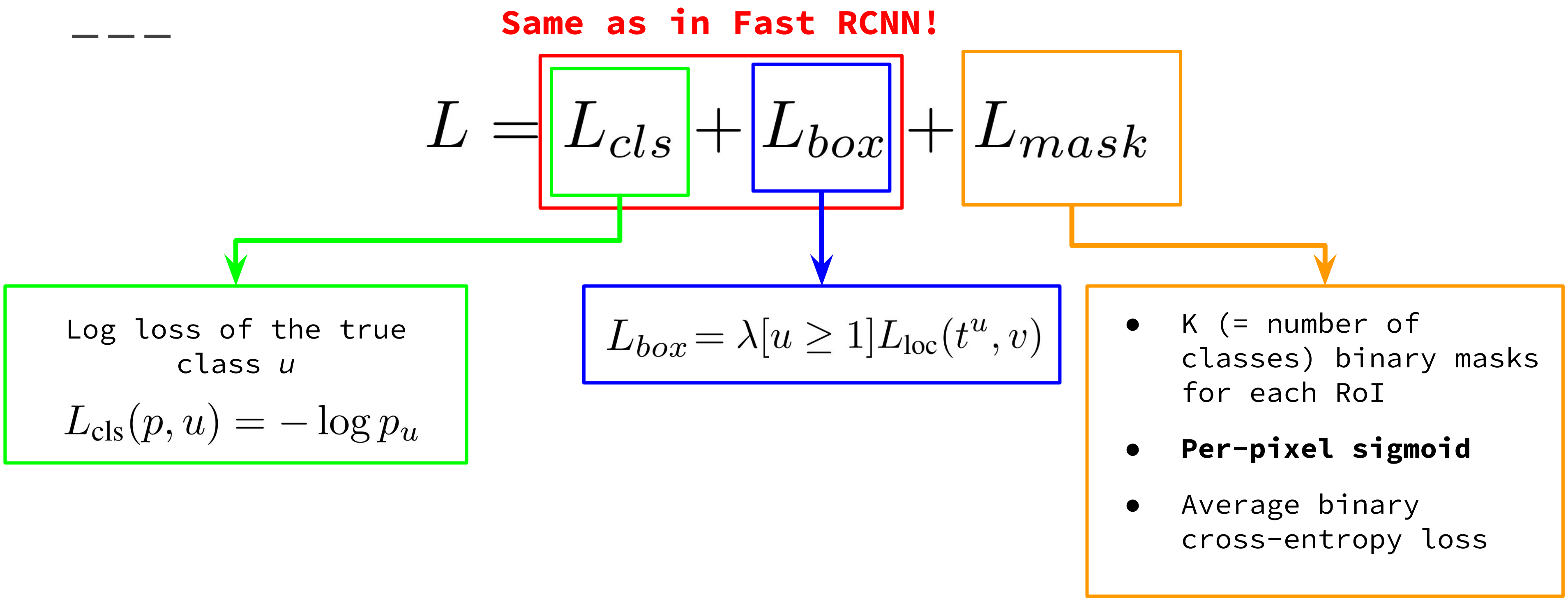
- Loss for bounding-box regression

$$L_{\text{loc}}(t^u, v) = \sum_{i \in \{\text{x,y,w,h}\}} \boxed{\text{smooth}_{L_1}(t_i^u - v_i)}$$

- Robust L1 loss

$$\text{smooth}_{L_1}(x) = \begin{cases} 0.5x^2 & \text{if } |x| < 1 \\ |x| - 0.5 & \text{otherwise} \end{cases}$$

Mask RCNN



Same as in Fast RCNN!

$$L = \boxed{\boxed{L_{cls}} + \boxed{L_{box}}} + \boxed{L_{mask}}$$

Log loss of the true class *u*

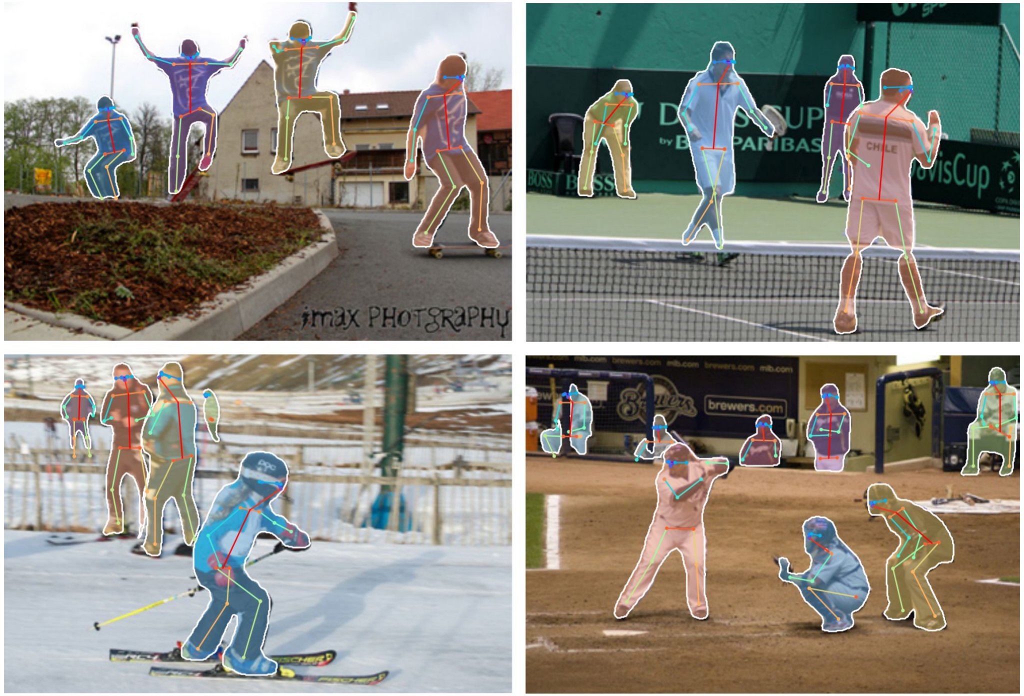
$$L_{\text{cls}}(p, u) = -\log p_u$$

$$L_{box} = \lambda[u \geq 1] L_{\text{loc}}(t^u, v)$$

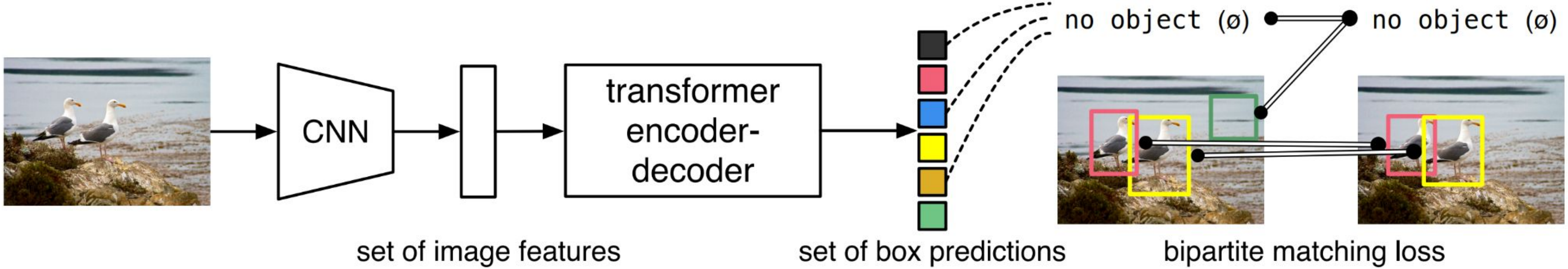
- K (= number of classes) binary masks for each RoI
- **Per-pixel sigmoid**
- Average binary cross-entropy loss

Mask RCNN for pose estimation

- Keypoint’s location modeled as a one-hot mask
- Mask R-CNN predicts K masks, one for each of K keypoint types (e.g., left shoulder, right elbow)



DEtection TRansformers (DETR)

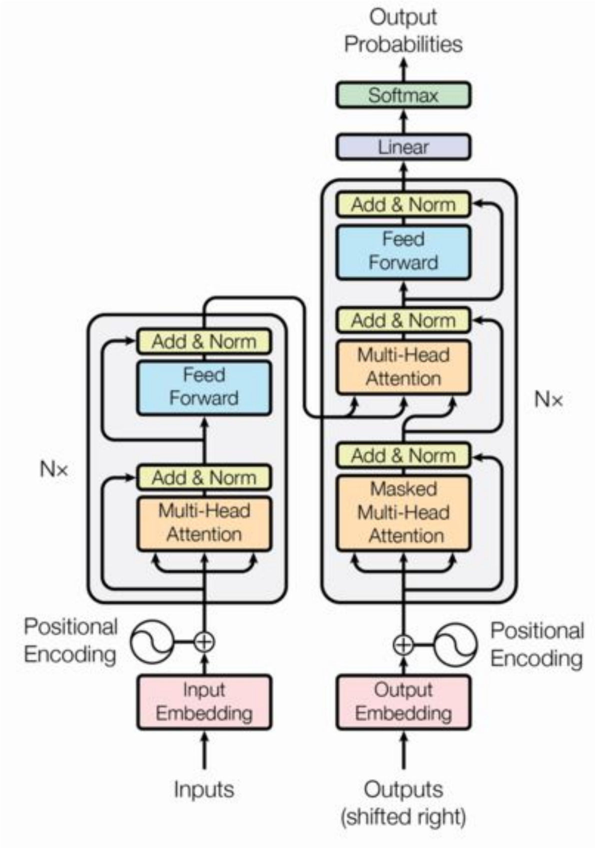


- Does not rely on region proposal networks
- Exploit transformer architecture

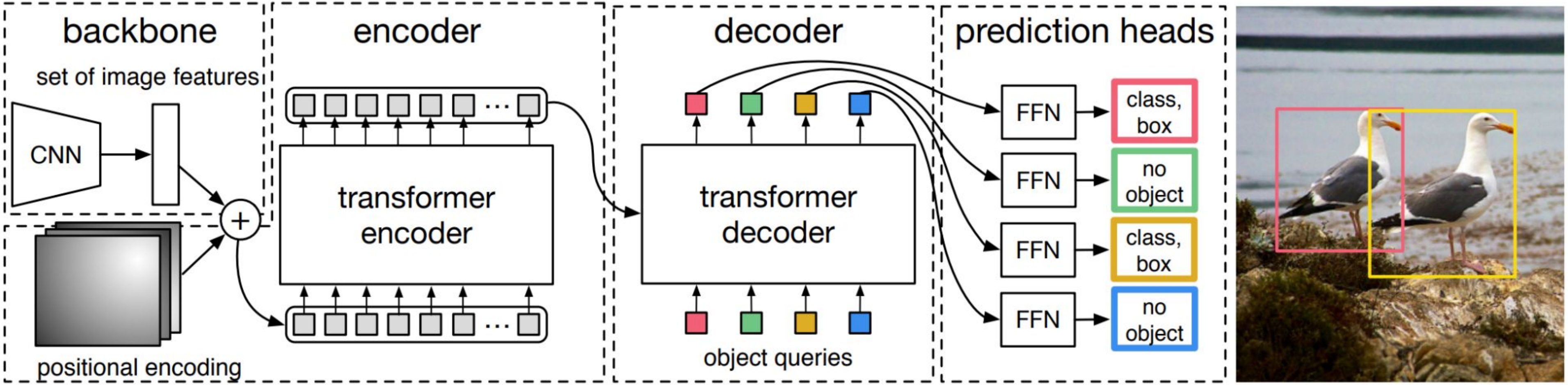
DEtection TRansformers (DETR)

Transformer architecture

- Encoder-decoder architecture
- Multi-head self-attention mechanism
- Positional encoding

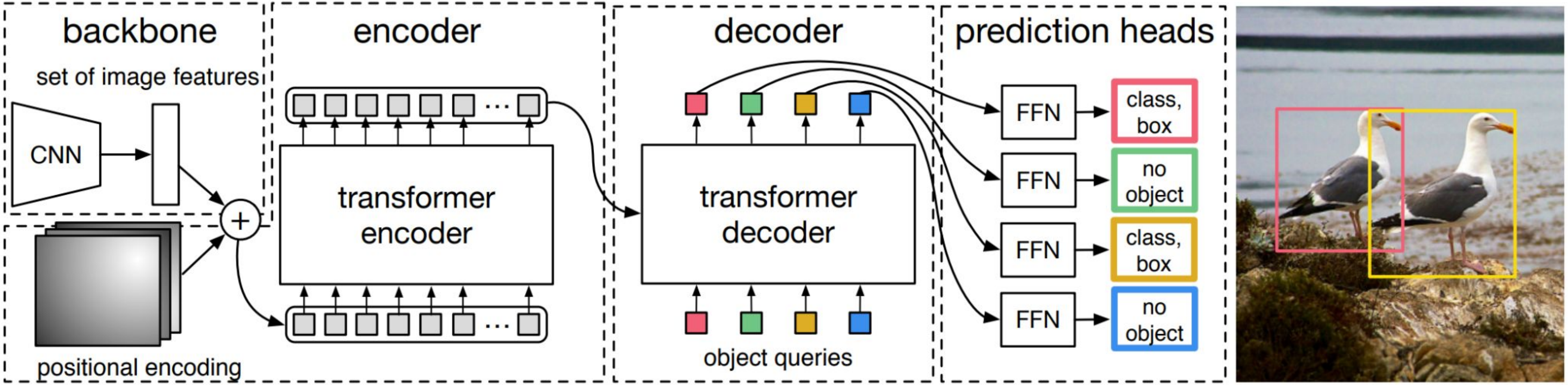


DEtection TRansformers (DETR)



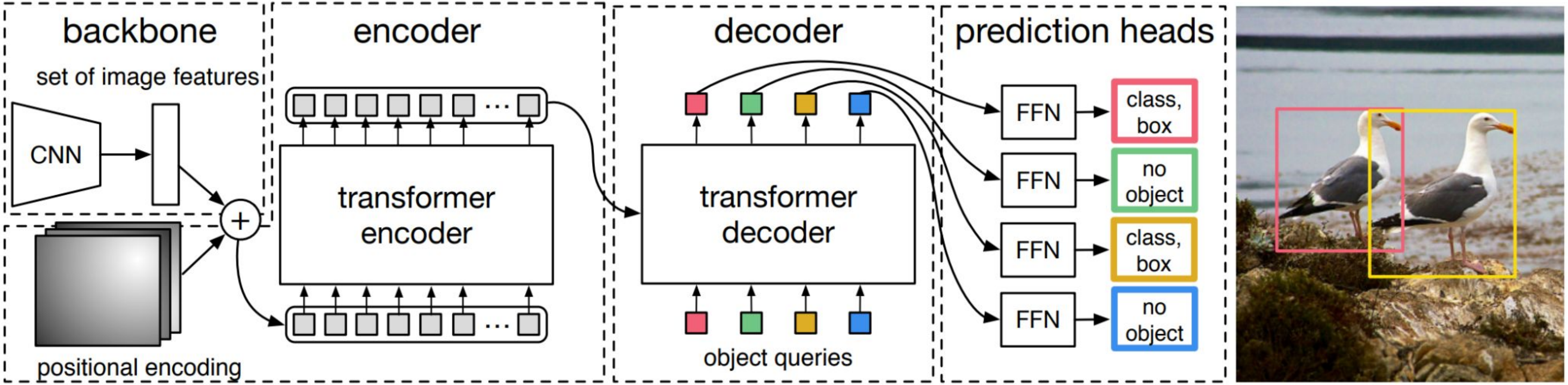
1. A **CNN** is used to extract a feature map that is flattened to a sequence of feature vectors.

DEtection TRansformers (DETR)



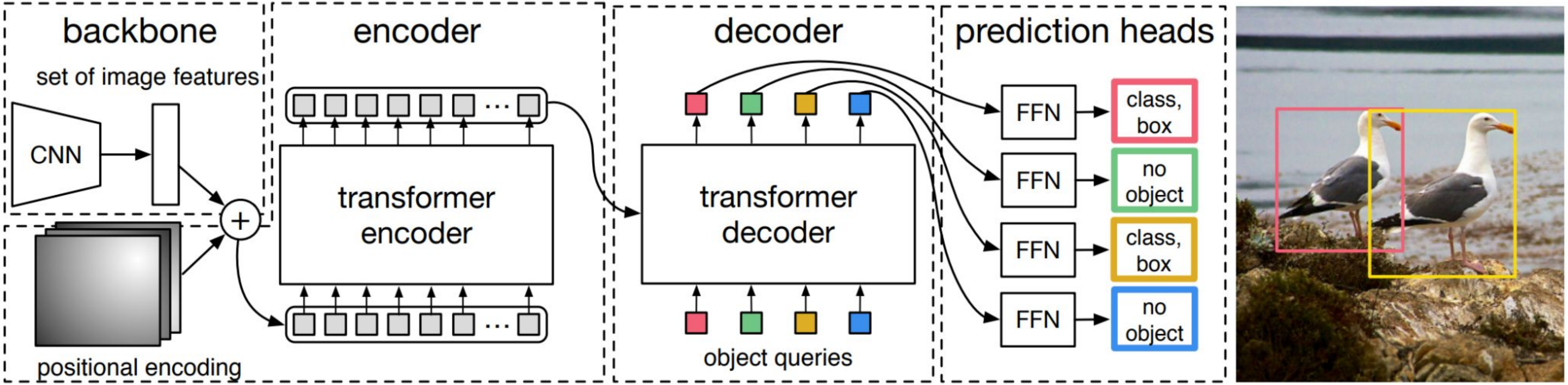
2. The feature vectors and positional encodings are passed to a **transformer encoder**.

DEtection TRansformers (DETR)



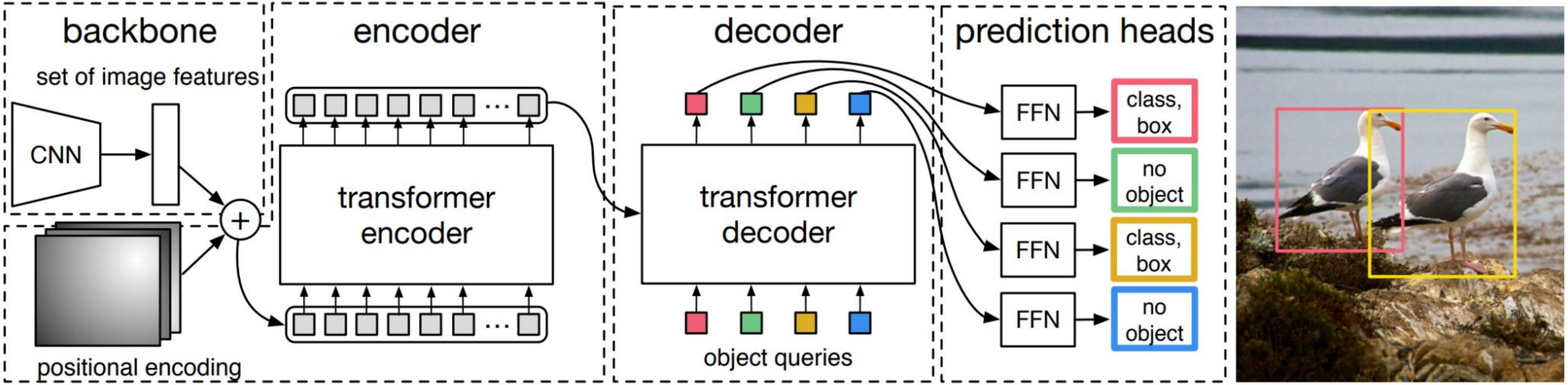
3. A set of **learned object queries**, that represent potential object instances are used to query the transformer encoder output. The number of object queries is fixed and determines the maximum number of objects DETR can predict per image.

DEtection TRansformers (DETR)



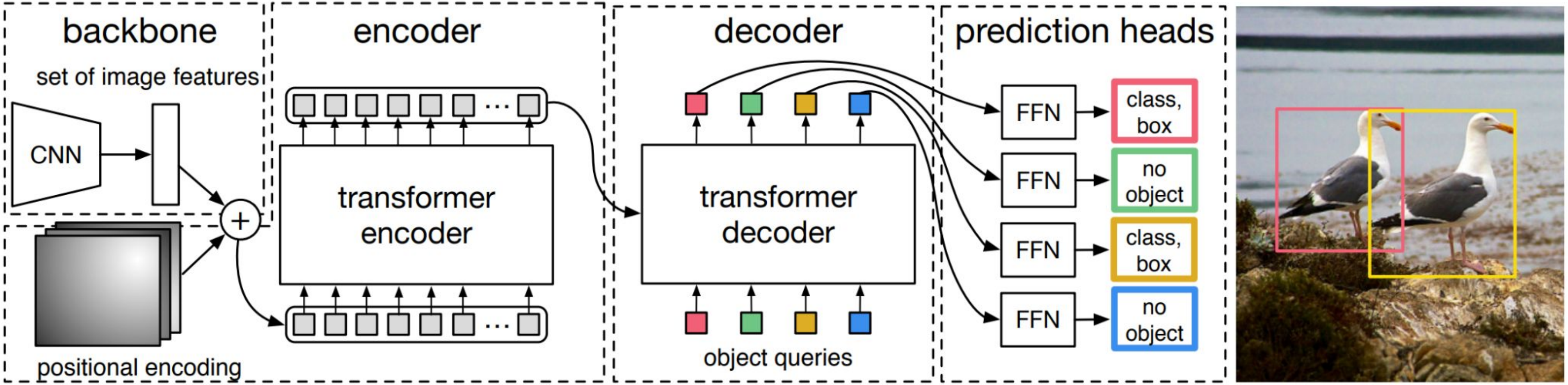
4. The **transformer decoder** takes the object queries and their corresponding transformer encoder outputs and refines the predictions

DEtection TRansformers (DETR)



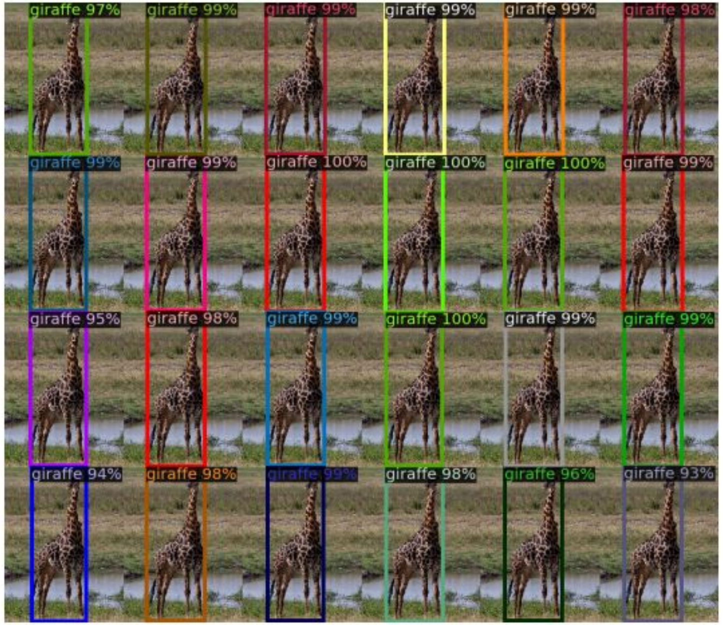
5. Two **object detection heads**:
    - Probability distribution over object classes (background included)
    - Bounding-boxes (center, width, heigh)

DEtection TRansformers (DETR)

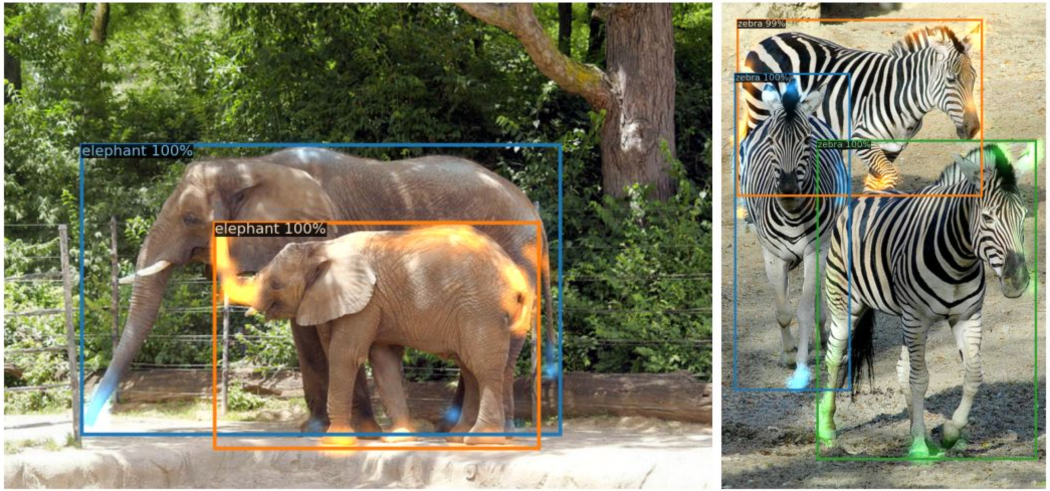


6. **Bipartite matching loss**
   - Match predicted object queries to ground truth objects based on a bipartite matching algorithm, minimizing the assignment cost.

DEtection TRansformers (DETR)

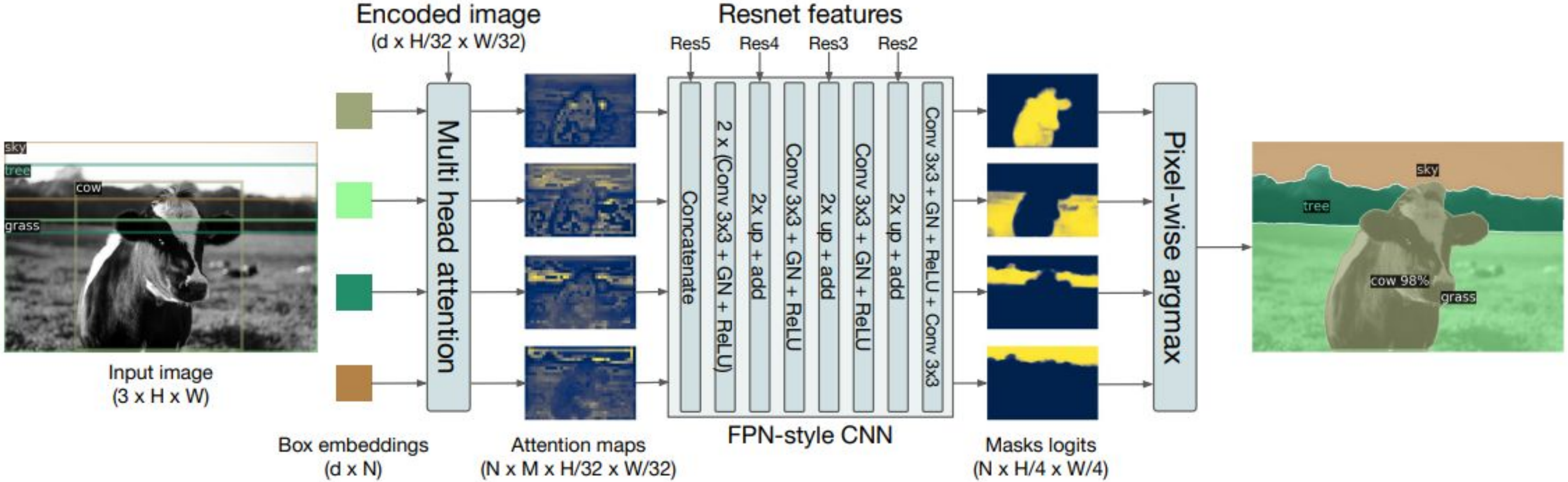


Out of distribution generalization



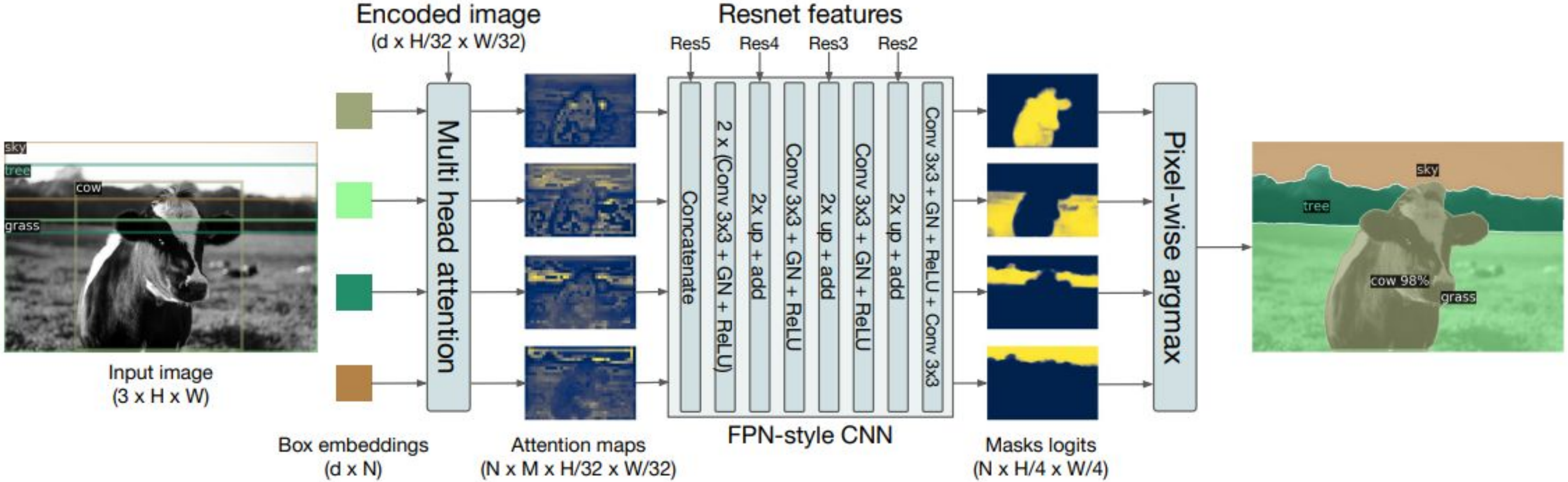
Decoder attention for the predicted objects

DEtection TRansformers (DETR) - Panoptic segmentation



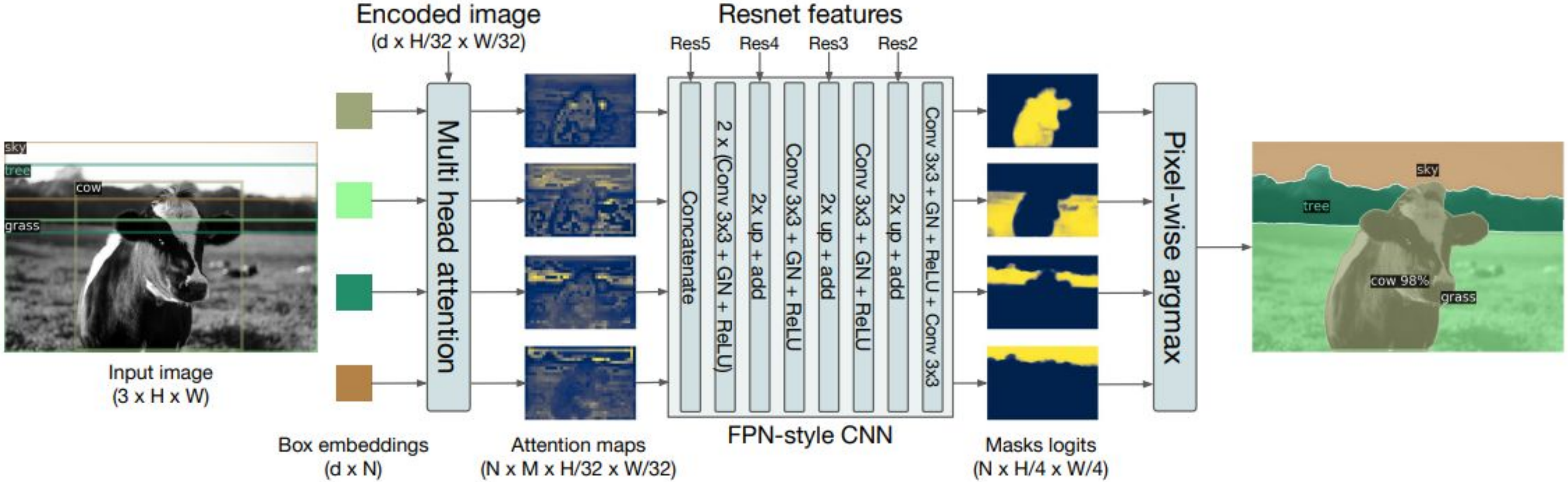
1. The input is the output of the transformer decoder for each object.

DEtection TRansformers (DETR) - Panoptic segmentation



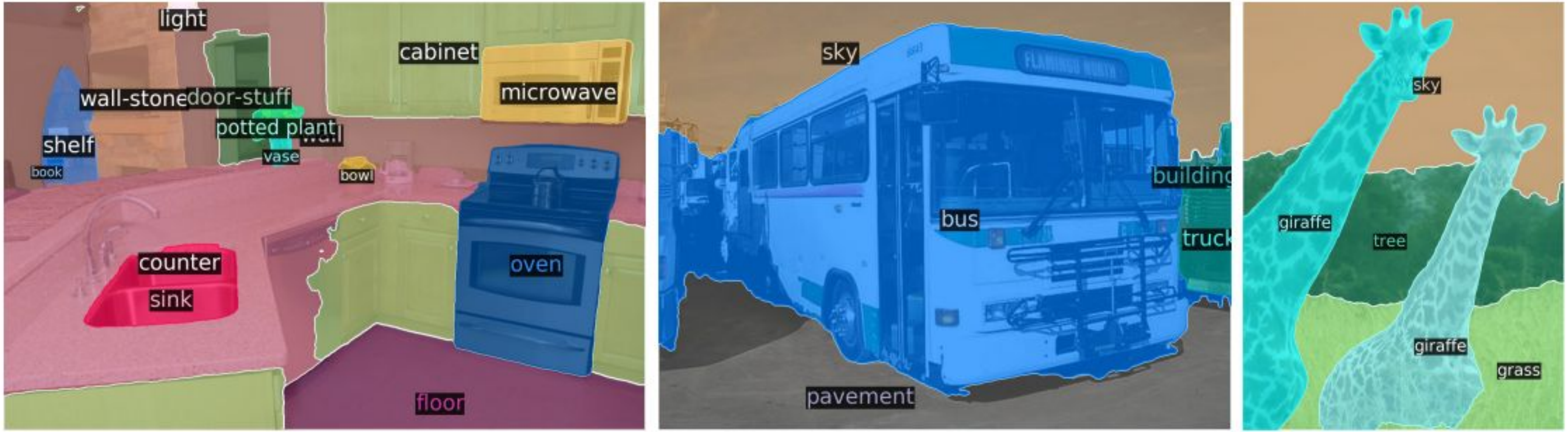
2. Compute multi-head (M) attention scores of the input over the output of the encoder, thus generating M heatmaps per object.

DEtection TRansformers (DETR) - Panoptic segmentation



3. Pass the heatmaps to a CNN to make the final prediction and increase the resolution.

DEtection TRansformer (DETR) - Panoptic segmentation



Open Challenges

- Handling small objects
- Scale variation
- Robustness in environmental changes
- Handling occlusions
- Reducing annotation efforts