Advanced Deep Learning

CSA401

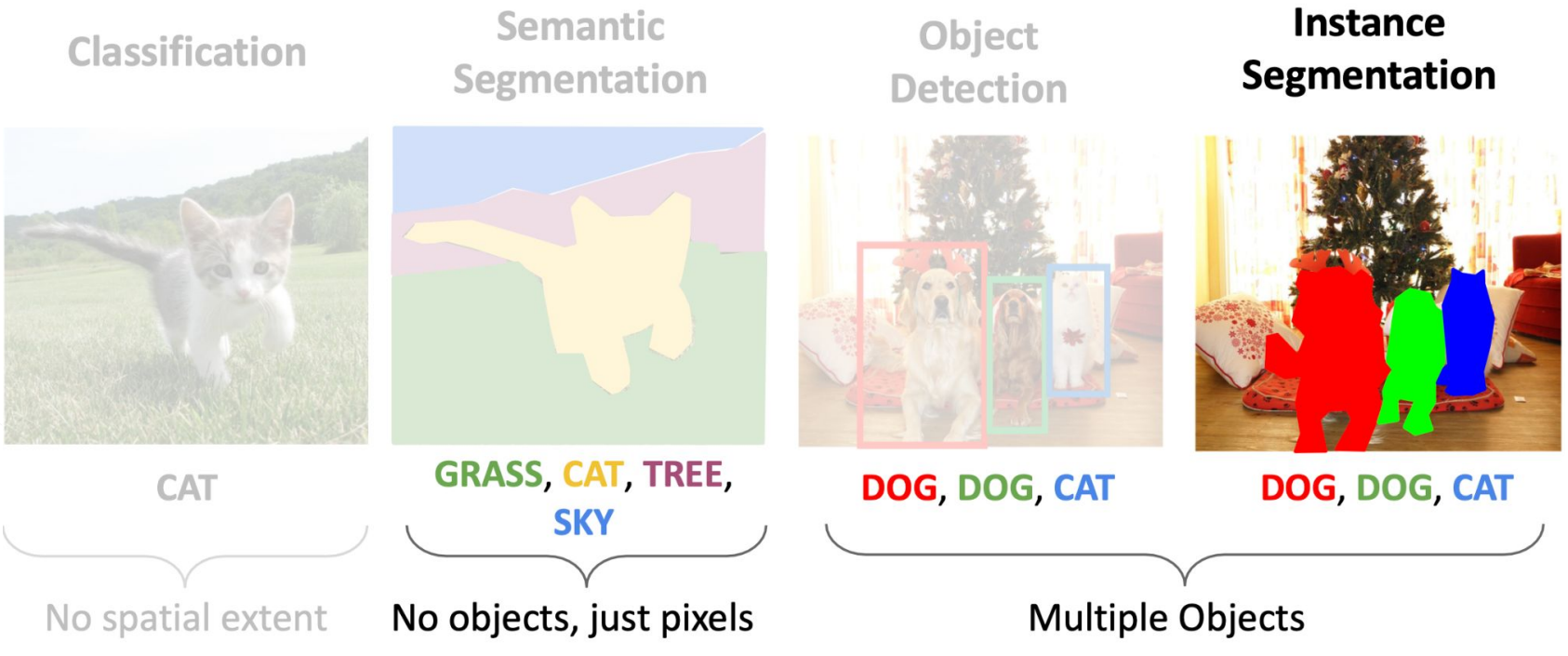
Mask R-CNN



Learning Outcomes

- Instance Segmentation
- Understand Mask R-CNN architecture
- Implement Mask R-CNN in PyTorch

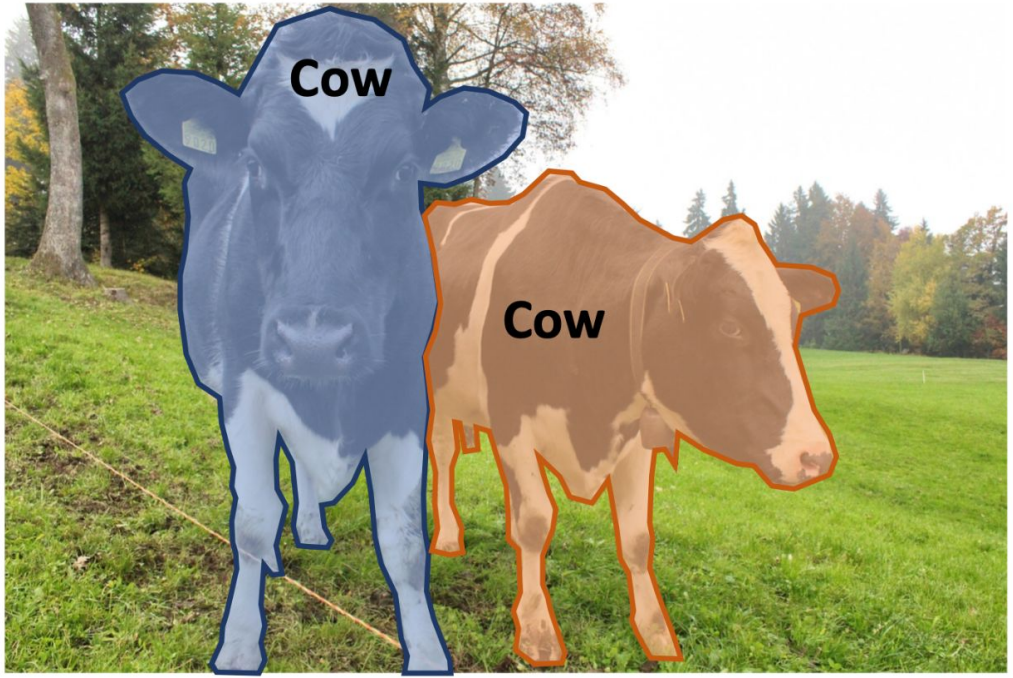
Computer Vision Tasks: Instance Segmentation



Computer Vision Tasks: Instance Segmentation

Instance Segmentation: Detect all objects in the image, and identify the pixels that belong to each object (Only things!)

Approach: Perform object detection, then predict a segmentation mask for each object!



Instance Segmentation

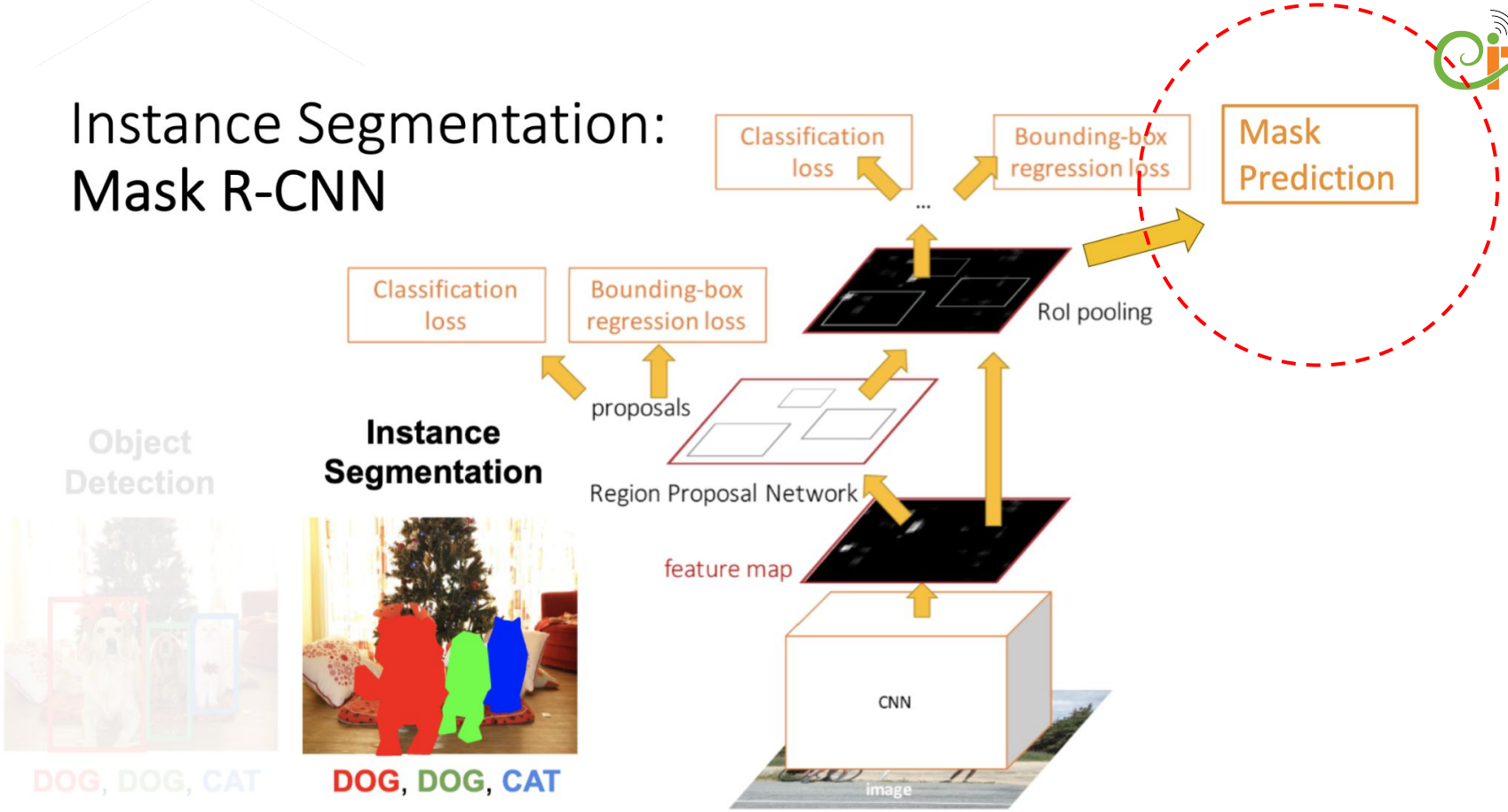
- Instance segmentation is a computer vision task that involves *identifying* and *delineating* individual objects within an image while assigning a unique *label* to each pixel.
- Techniques employed in instance segmentation:
  - Single-shot instance segmentation - **YOLACT** (You Only Look At Coefficients)
  - Transformer-based methods: capturing **long-range dependencies** in data - **DETR (DEtection TRansformer)**
  - Detection-based instance segmentation - **Mask R-CNN**

Object Detection: Faster R-CNN



Ren et al, "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks", NeurIPS 2015

Instance Segmentation: Mask R-CNN



He et al, "Mask R-CNN", ICCV 2017

https://arxiv.org/pdf/1703.06870.pdf

Mask R-CNN Architecture

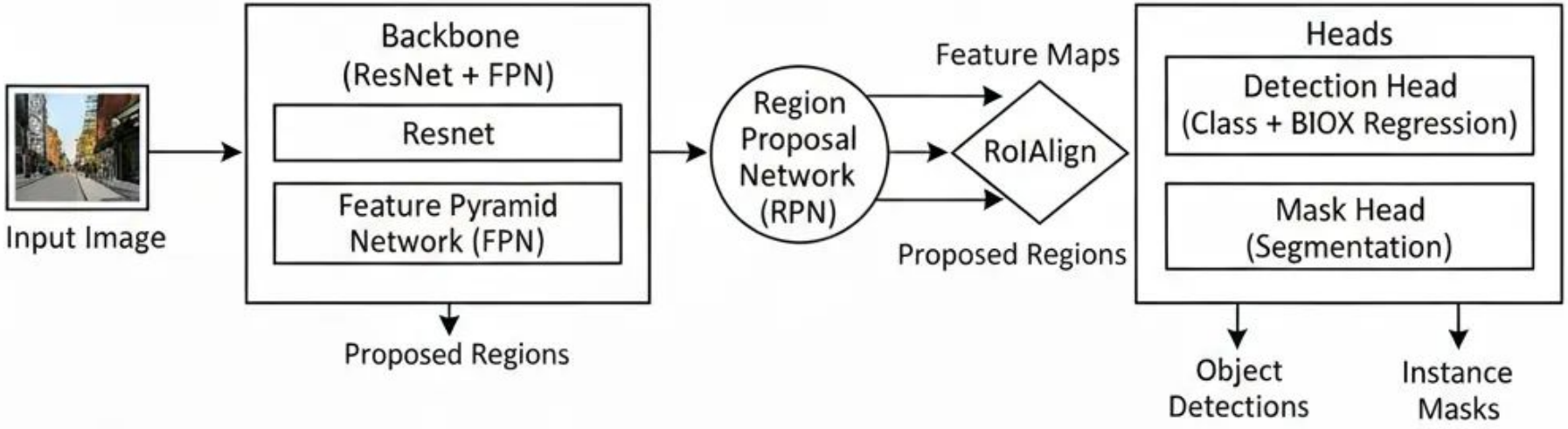
- Mask R-CNN is an extension of the Faster R-CNN.
- It builds on its predecessor by adding a branch for **predicting segmentation masks** on detected objects, operating in parallel with the existing branch for **bounding box recognition**.
- This dual functionality allows Mask R-CNN to detect objects and precisely **segregate** them within the image, making it invaluable for tasks requiring detailed object understanding.
- The Mask R-CNN framework has revolutionized the field of computer vision, offering improved accuracy and efficiency in tasks like instance segmentation.

Mask R-CNN Architecture

Mask R-CNN core component:

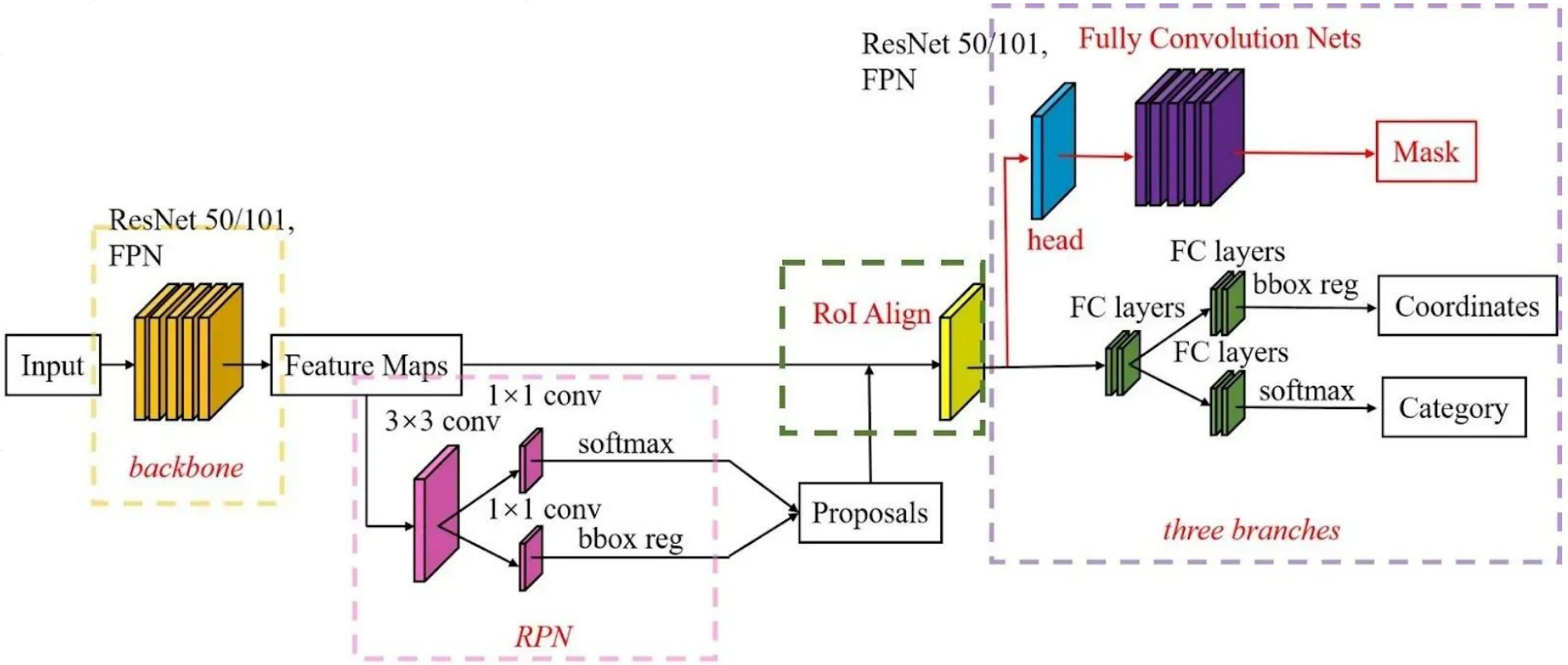
1. **Backbone (e.g., ResNet + FPN**): Extracts features from the input image.
2. **Region Proposal Network (RPN)**: Proposes object candidates (Regions of Interest).
3. **RoI Align**: Fixes the problem of quantization (RoI Pool used in Faster R-CNN).
4. **Three branches**:
    - Classification branch: What is the object?
    - Bounding box regression: Where is the object?
    - Mask branch: Predicts a binary mask for the object.

Mask R-CNN



source

Mask R-CNN


ResNet 50/101,
FPN
Fully Convolution Nets
Mask
head
FC layers
bbox reg
Coordinates
FC layers
FC layers
softmax
Category
RoI Align
ResNet 50/101,
FPN
Input
Feature Maps
1×1 conv
3×3 conv
softmax
1×1 conv
bbox reg
Proposals
backbone
RPN
three branches

Core components of Mask R-CNN

- **Backbone Network:** The backbone network in Mask R-CNN is typically a pre-trained convolutional neural network, such as **ResNet** or ResNeXt. This backbone processes the input image and extracts high-level features.
- **Feature Pyramid Network**: An FPN is then added on **top of this backbone** network to enhance the model's **ability to detect objects of different sizes** across the image.
- **Region Proposal Network (RPN)**: The RPN generates potential object regions (proposals) within the feature map.
- **RoI Align**: RoI Align layer which fixes the **misalignment issue** caused by the **RoI Pooling** process used in previous models. It does this by preserving the exact spatial locations of the features, leading to more accurate mask predictions.
- **Classification and Bounding Box Regression**: to classify the object and refine its bounding box. It uses a fully connected network to output a class label and bounding box coordinates.
- **Mask Prediction**: In addition to the classification and bounding box outputs, there's a parallel branch for mask prediction. This branch is a small *Fully Convolutional Network (FCN)* that outputs a **binary mask for each RoI**.

Backbone Network (Feature Extraction)

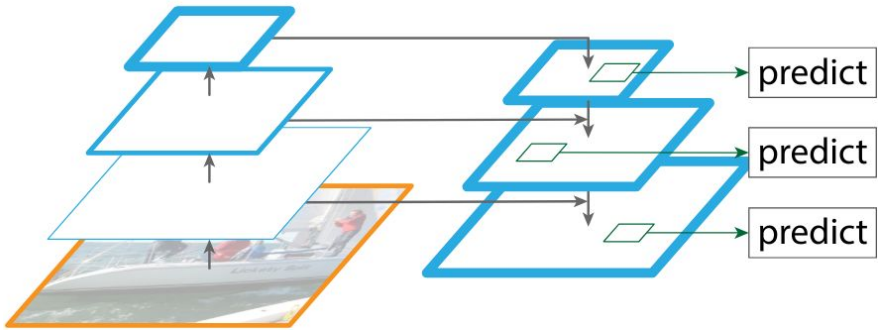
- Mask R-CNN uses a deep convolutional network such as ResNet-50 or ResNet-101 as its backbone.
- The backbone extracts feature maps, detailed numerical representations of the input image.
- To enhance these features across different scales, it usually combines the backbone with a Feature Pyramid Network (FPN).
- FPN merges low-level (fine) and high-level (semantic) features, helping the model detect both small and large objects.

  Input: Original image

  Output: Multi-scale feature maps

Feature Pyramid Network

- **FPN** is modular architectural extension that is added on top of a backbone model that reorganizes and enhances those features to make them **multi-scale** and more useful for detection/segmentation tasks.
- **Feature Fusion**: FPN creates connections between different levels of the backbone network to create a top-down pathway.
- This **top-down pathway** combines **high-level semantic information** with **lower-level feature maps**, allowing the model to **reuse** features at different scales.

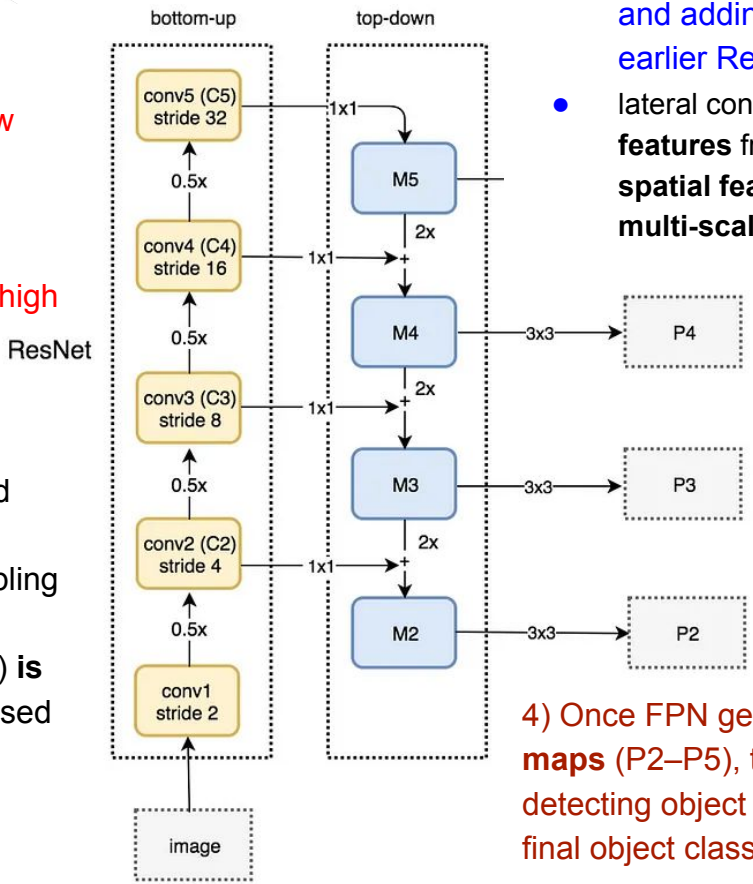


https://tinyurl.com/36yc7vu5

Feature Pyramid Network

1) ResNet processes the image and produces several layers of feature maps: **Bottom-Up** Pathway:

- C2: early features (high resolution, low semantics)
- C3: mid-level features
- C4: deeper features
- C5: deepest features (low resolution, high semantics)



ResNet

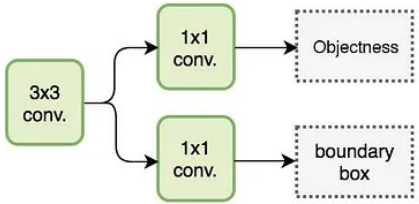
3) After merging, each feature map is passed through a **3×3** convolution to:

- Clean up artifacts from upsampling
- Further refine the features

- Each of these **feature maps** (P2,P3..) **is rich and semantically strong**, and used by the Region Proposal Network

2) **Top-Down** Pathway:

- Build a top-down pyramid by **upsampling higher-level features** (starting from C5) and adding them to lateral connections from earlier ResNet layers.
- lateral connections- merge **high-level semantic features** from deeper layers with **fine-grained spatial features** from shallower layers to create **multi-scale feature maps**.

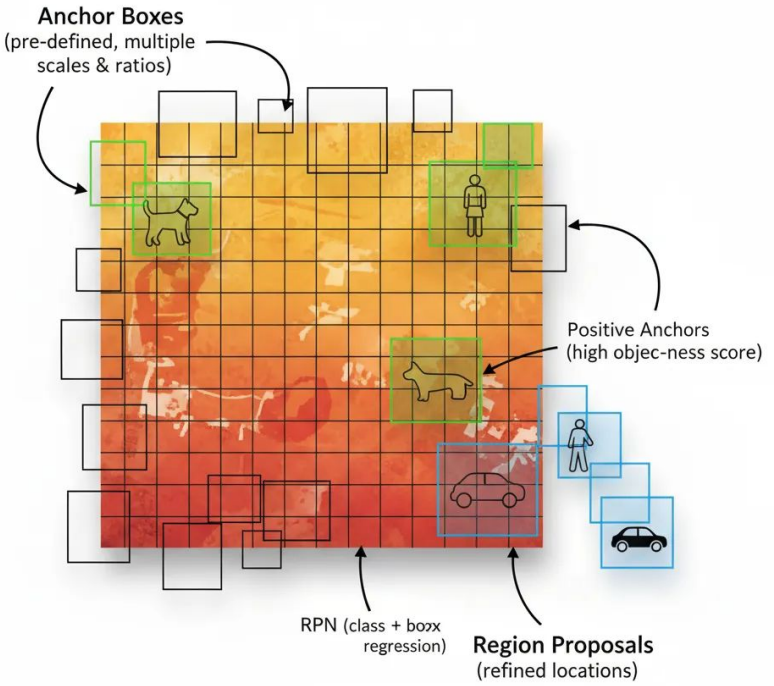


predictor head

4) Once FPN generates the **multi-scale feature maps** (P2–P5), these are fed into RPN for for detecting object proposals and the RoI heads for final object classification and segmentation.

Region Proposal Network (RPN)

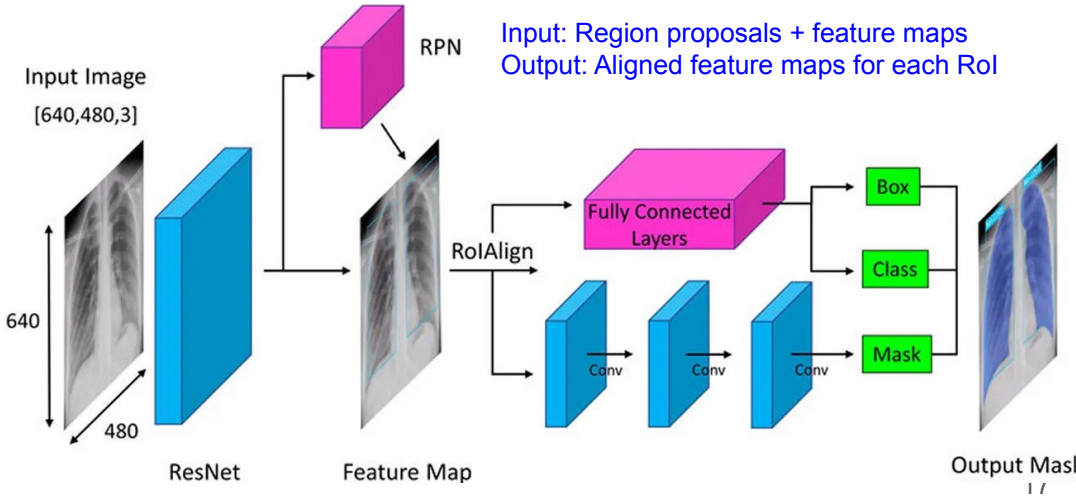
- The RPN scans the feature maps to propose candidate regions that may contain objects.
  - *Slides small filters over the feature map.*
  - *Generates multiple anchors (predefined boxes) at each position.*
  - *Predicts whether each anchor contains an object or background.*
  - *Refines the* coordinates of these boxes.
- Then, NMS removes overlapping or duplicate boxes, keeping only the best proposals (usually 200–300).
- *Input: Feature maps from backbone*
- *Output: Candidate object regions (RoIs)*

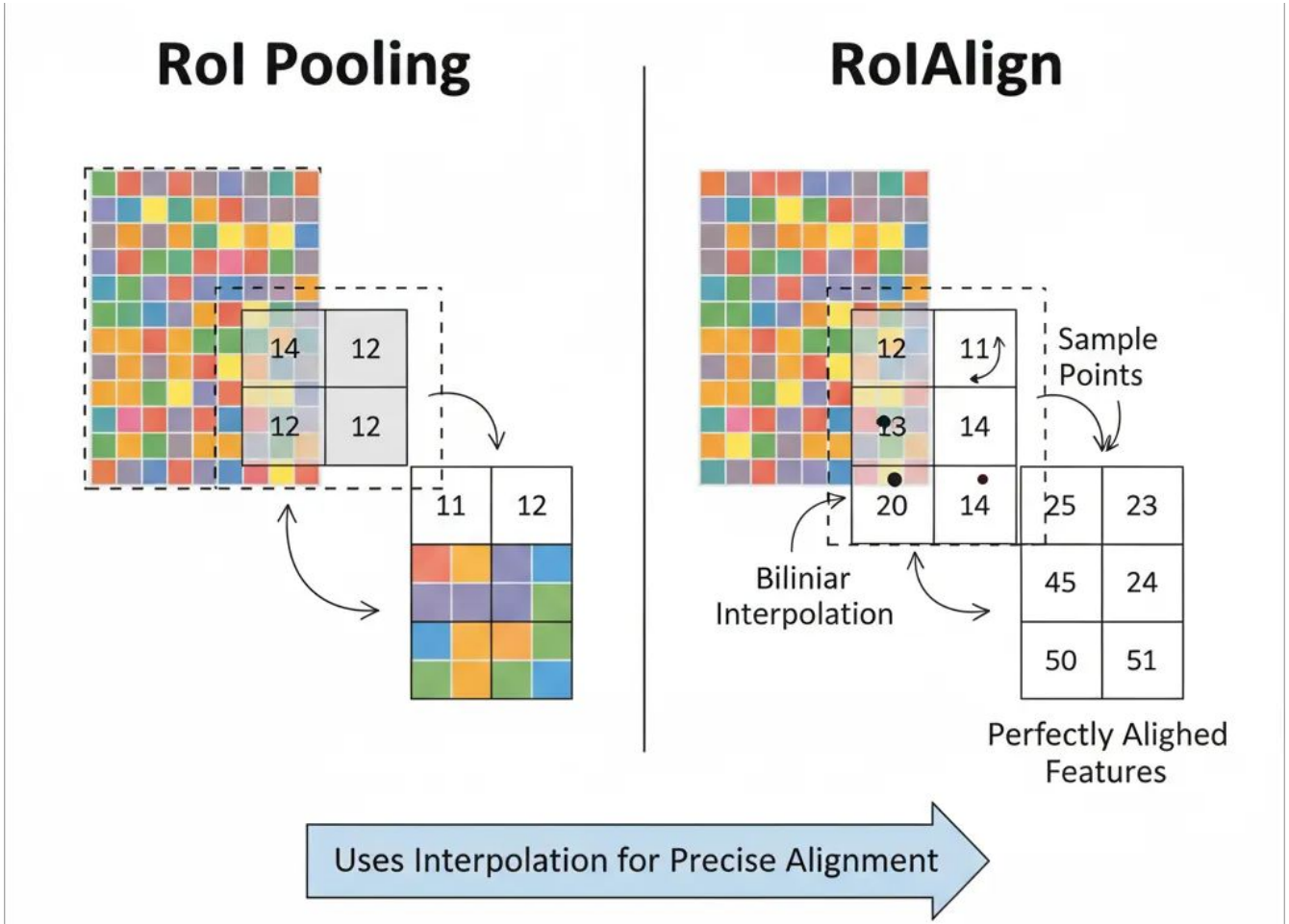


ROI Align Layer

- In Faster R-CNN, RoI Pooling was used to crop RoIs from feature maps and resize them into fixed sizes.
- RoI Pooling caused misalignments due to rounding coordinates.
- RoI Pooling is replaced by **RoI Align** in Mask R-CNN to ensure better precision in spatial alignment, especially for pixel-wise tasks like segmentation.

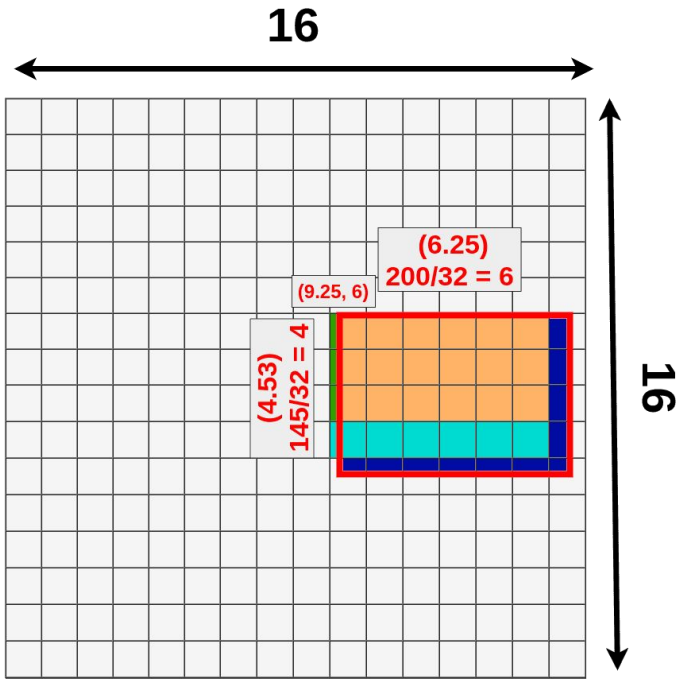
- Mask R-CNN solves this with RoIAlign, which uses bilinear interpolation to align features accurately without losing spatial precision.
- This ensures that object boundaries and mask edges are perfectly aligned.




RoI Pooling
14
12
12
12
11
12
RoIAlign
12
11
13
14
20
14
25
23
45
24
50
51
Sample Points
Biliniar Interpolation
Perfectly Alighed Features
Uses Interpolation for Precise Alignment

ROI Pooling

- The main difference between **RoI Pooling** and **RoI Align** is quantization. RoI Align is not using quantization for data pooling.



- RoI Pooling losses in quantization (dark and light blue), data gain (green)
- Every time it does that, part of the information about the object is missing.
- That lowers the precision of the whole model and a lot of really smart people thought about it.

RoI Pooling

- One of the drawbacks of **RoI Pooling** is that we are likely to **lose** certain **information** when we are performing the RoI pooling operation.
- Here, the **region proposal** is **5 x 7** in shape and we have to convert it into a **2 x 2** shape.
- While converting it into a 2 x 2 shape (**quantization**), one part of the region has **less** representation compared to other parts of the region.

This results in **information loss** since certain parts of the region have more weight than others.

Input

0.4	0.32	0.01	0.42	0.68	0.67	0.01	0.07
0.54	0.66	0.81	0.18	0.01	0.42	0.46	0.19
0	0.38	0.31	1	0.06	0.1	0.48	0.91
0.63	0.91	0.31	0.41	0.4	0.85	0.56	0.39
0.09	0.42	0.5	0.22	0.8	0.98	0.13	0.88
0.99	0.51	0.16	0.68	0.9	0.23	0.57	0.93
0.8	0.4	0.06	0.48	0.52	0.88	0.2	0.17
0.64	0.92	0.07	0.73	0.24	0.2	0.79	0.16

Region proposal

0.4	0.32	0.01	0.42	0.68	0.67	0.01	0.07
0.54	0.66	0.81	0.18	0.01	0.42	0.46	0.19
0	0.38	0.31	1	0.06	0.1	0.48	0.91
0.63	0.91	0.31	0.41	0.4	0.85	0.56	0.39
0.09	0.42	0.5	0.22	0.8	0.98	0.13	0.88
0.99	0.51	0.16	0.68	0.9	0.23	0.57	0.93
0.8	0.4	0.06	0.48	0.52	0.88	0.2	0.17
0.64	0.92	0.07	0.73	0.24	0.2	0.79	0.16

Dividing the region into sections

0.4	0.32	0.01	0.42	0.68	0.67	0.01	0.07
0.54	0.66	0.81	0.18	0.01	0.42	0.46	0.19
0	0.38	0.31	1	0.06	0.1	0.48	0.91
0.63	0.91	0.31	0.41	0.4	0.85	0.56	0.39
0.09	0.42	0.5	0.22	0.8	0.98	0.13	0.88
0.99	0.51	0.16	0.68	0.9	0.23	0.57	0.93
0.8	0.4	0.06	0.48	0.52	0.88	0.2	0.17
0.64	0.92	0.07	0.73	0.24	0.2	0.79	0.16

Max pooling

0.91	0.98
0.99	0.88

ROI Align

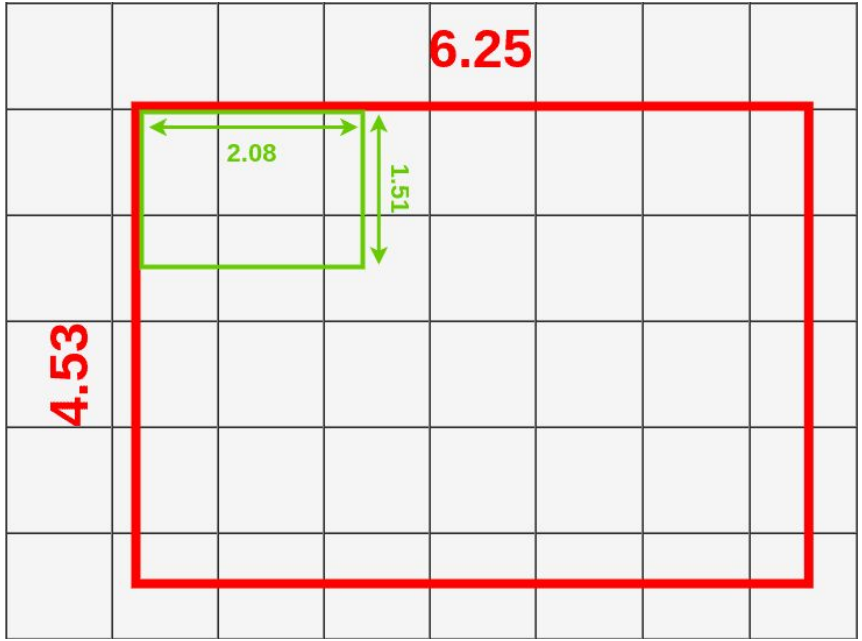
- We can skip quantization by dividing original RoI into **9 equal size boxes** and applying **bilinear interpolation** inside every one of them.

  (9.25, 6) - top left corner

  6.25 - width

  4.53 - height

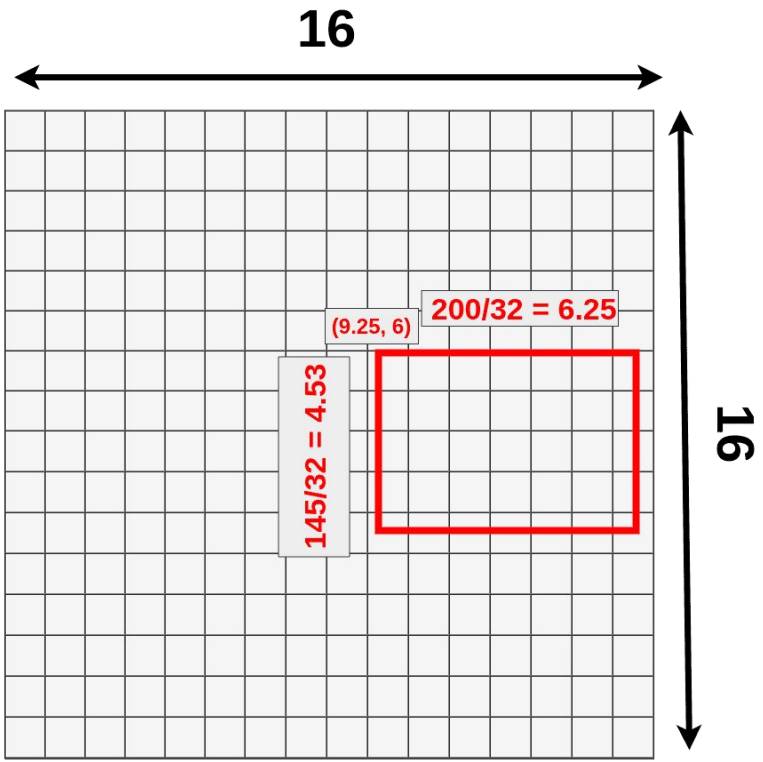
- Let's define boxes:

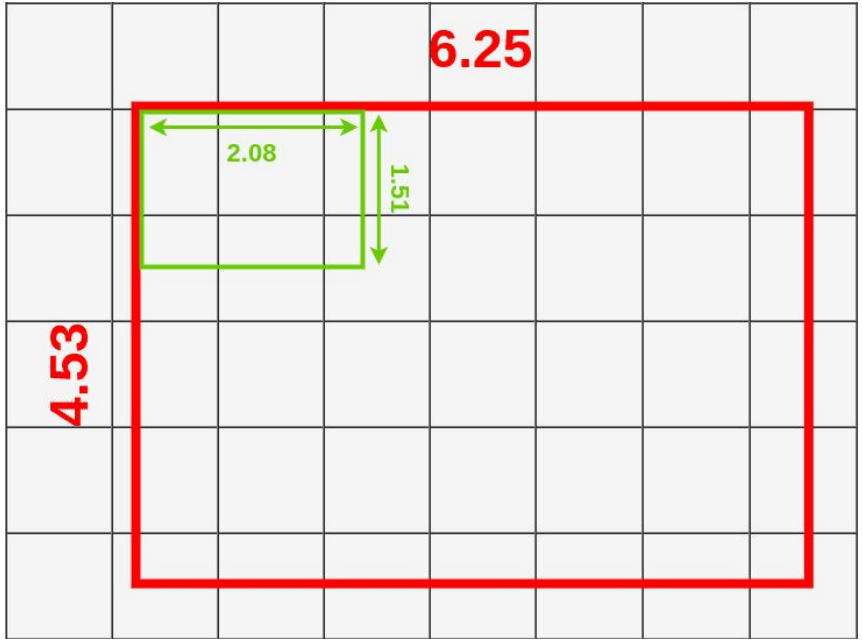


RoI Box size

source

RoI Align



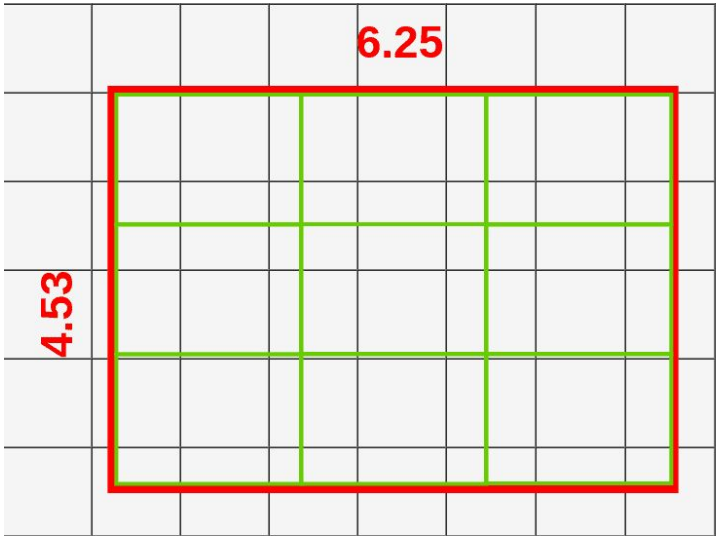


RoI Box size

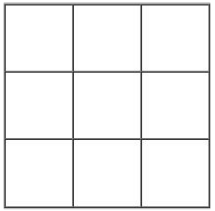
source

ROI Align

- Each box size is determined by the size of the mapped RoI and the size of the pooling layer. We're using a **3x3 pooling layer** so we have to divide mapped RoI **(6.25x4.53) by 3**.
- That gives us a box with a height of **1.51** and a width of **2.08** (rounding values). Now we can put our boxes into mapped RoI:



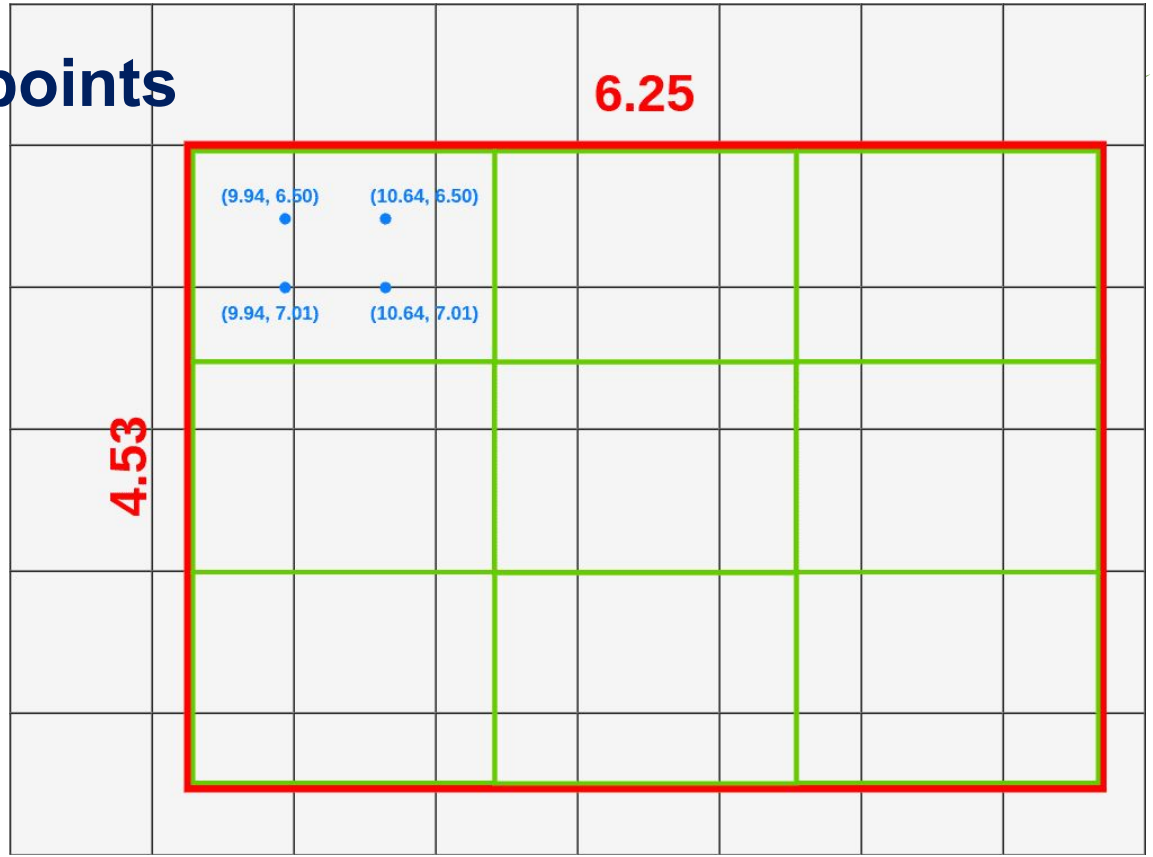
3x3 RoI Pooling



RoI divided into boxes

ROI Align- sampling points

- If you look at the first box (top left), you can notice that it covers **6 different grid cells**.
- To extract value for the pooling layer we have to sample some data from it.
- To sample data we have to create **4 sampling points** inside that box.

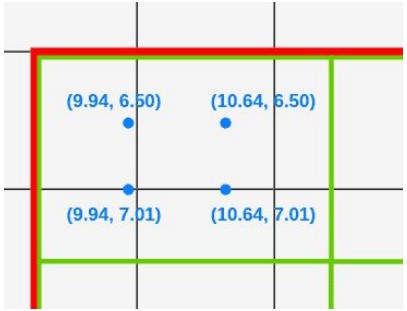


Sampling points distribution: They are **evenly spaced** within the bin.

ROI Align

X_box, Y_box = (9.25,6)- top-left of red box
width , height of first **grid = [2.08, 1.52]**

- You can calculate where each of those points should be by **dividing height and width of the box by 3**.
- FOr eg, here, we're calculating first point (top left) coordinates like this:
  - X = X_box + (width/3) * 1 = **9.94** => **9.25+(2.08/3)*1= 9.943**
  - Y = Y_box + (height/3) * 1 = **6.50** => **6+(1.51/3)*1= 6.503**
- To calculate the second point (bottom left) we have to change only the **Y**:
  - X = X_box + (width/3) * 1 = **9.94**
  - Y = Y_box + (height/3) * **2** = **7.01**



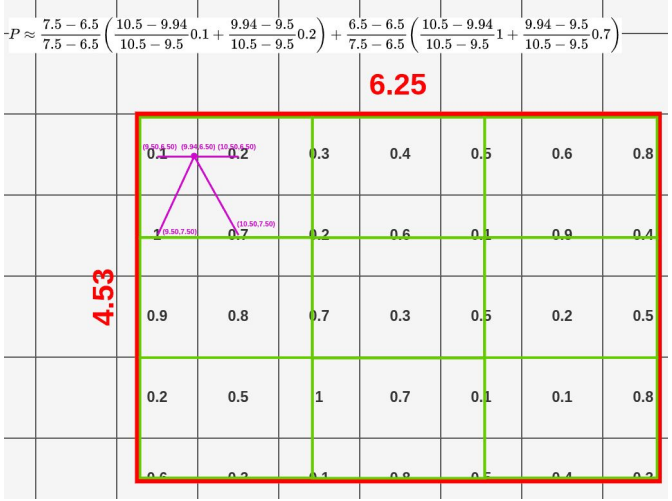
These 4 points are **not aligned with the original feature map grid**

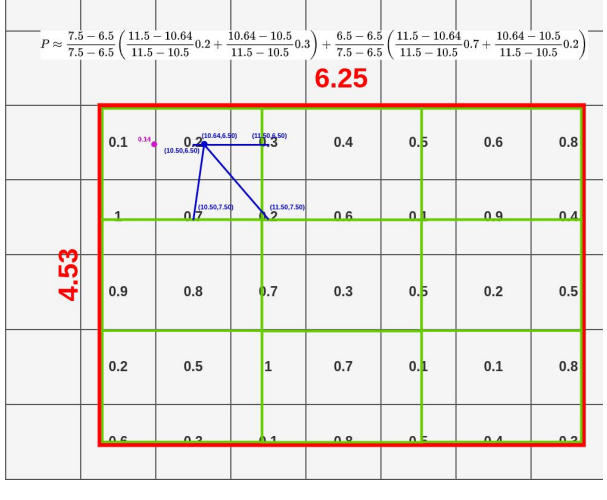
- Once we have all the points, we can apply **bilinear interpolation** to sample data for this box.
- Bilinear interpolation is usually used in image processing to sample colors and its equation looks like this:

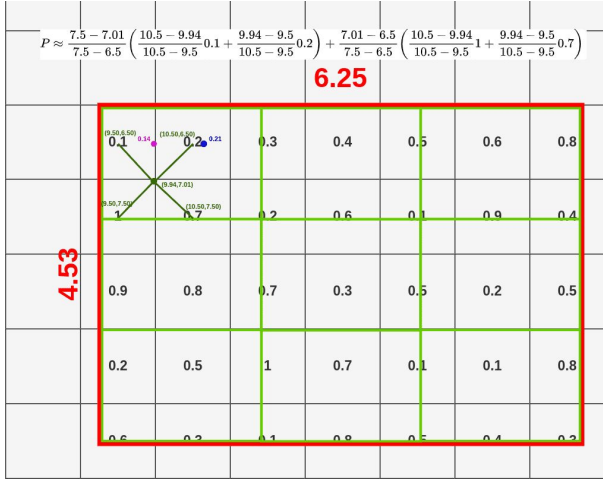
$$P \approx \frac{y_2 - y}{y_2 - y_1}\left(\frac{x_2 - x}{x_2 - x_1}Q_{11} + \frac{x - x_1}{x_2 - x_1}Q_{21}\right) + \frac{y - y_1}{y_2 - y_1}\left(\frac{x_2 - x}{x_2 - x_1}Q_{12} + \frac{x - x_1}{x_2 - x_1}Q_{22}\right)$$

Bilinear Interpolation equation

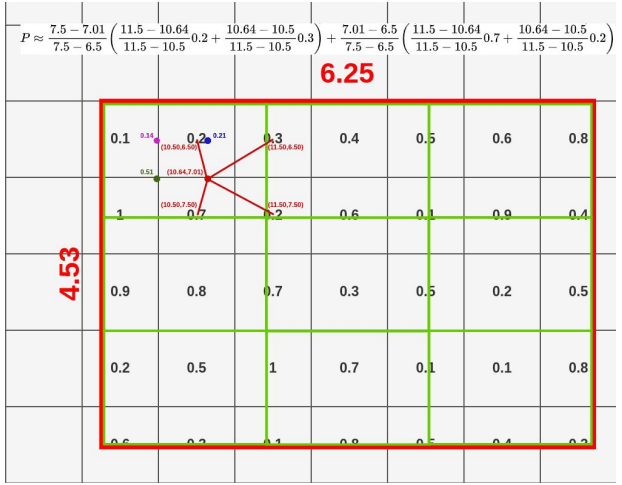
- Bilinear interpolation is needed because your sampling points (like 9.44, 6.50) usually don’t lie exactly on pixel locations of the feature map.
- So even after you pick 4 sample points in a bin, each of those points still falls between grid pixels, not on them.
- Bilinear interpolation is used to estimate the feature value at those fractional locations by smoothly combining the 4 nearest pixel values.
- it converts “points between pixels” into valid feature values without snapping or losing spatial precision.







Bilinear Interpolation for the first point, second point, third point and fourth point

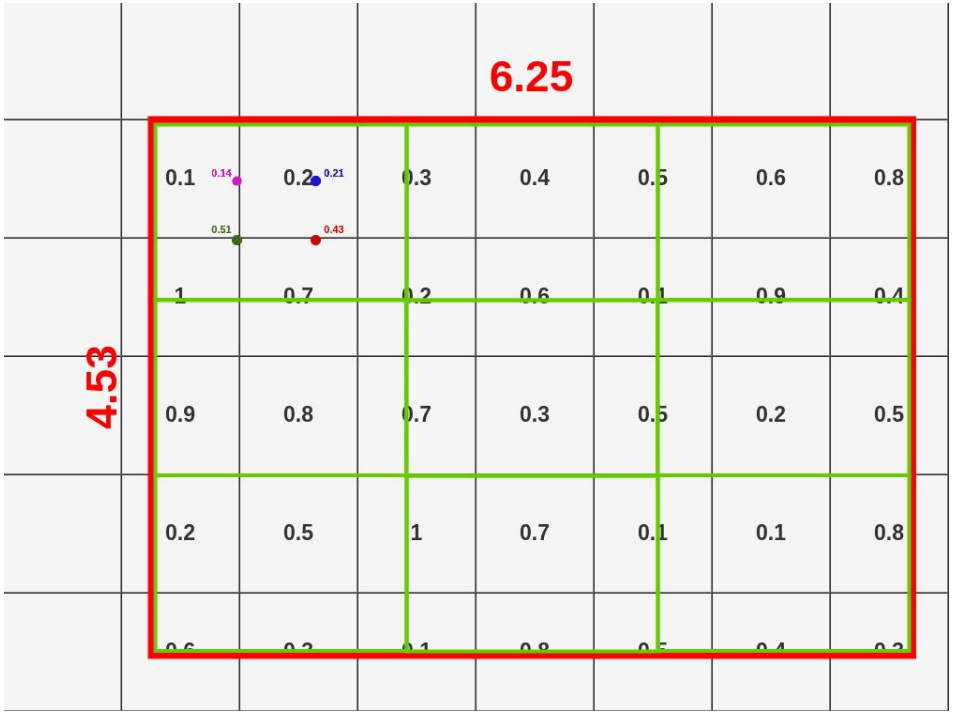


Because the 4 sample points (inside each bin) are usually **not aligned** with the integer grid of the **feature map**, they lie at **fractional coordinates** like `(9.44, 6.50)` instead of exactly `(9, 6)` or `(10, 7)`.

We use **bilinear interpolation** to **estimate** the value at that point using the 4 surrounding grid points.

It computes what the value would be at that exact fractional location

- Now we have all the points calculated and can apply Max Pooling on them (it could be Avg Pooling if you want):



1x1 = MAX(0.14, 0.21, 0.51, 0.43) = 0.51

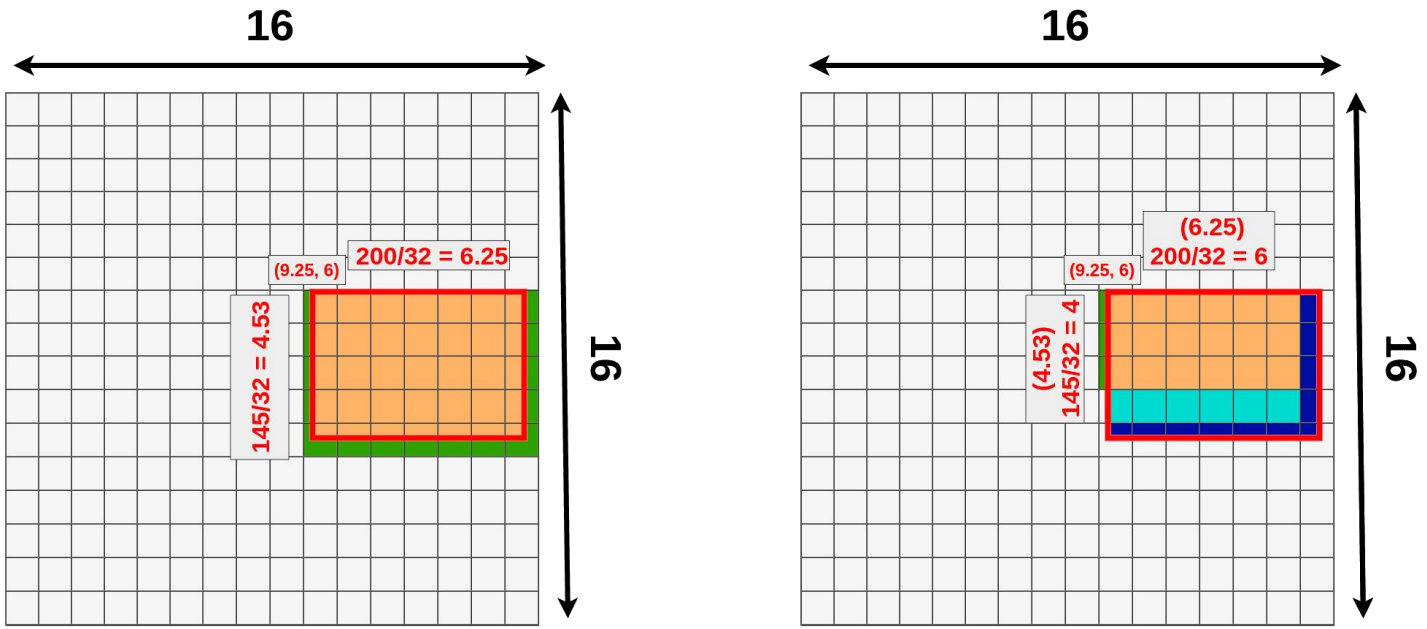
3x3 RoIAlign

0.51		

First Box Pooling



RoIAlign pooling process



*Comparing **RoIAlign**(left) and **RoIPooling**(right) data sources.*

- green means additional data used for pooling.
- blue (both shades) means **data lost** when pooling.

Source

Mask Head- Two Parallel Output Branches

- After RoIAlign, the network splits into two parallel branches:

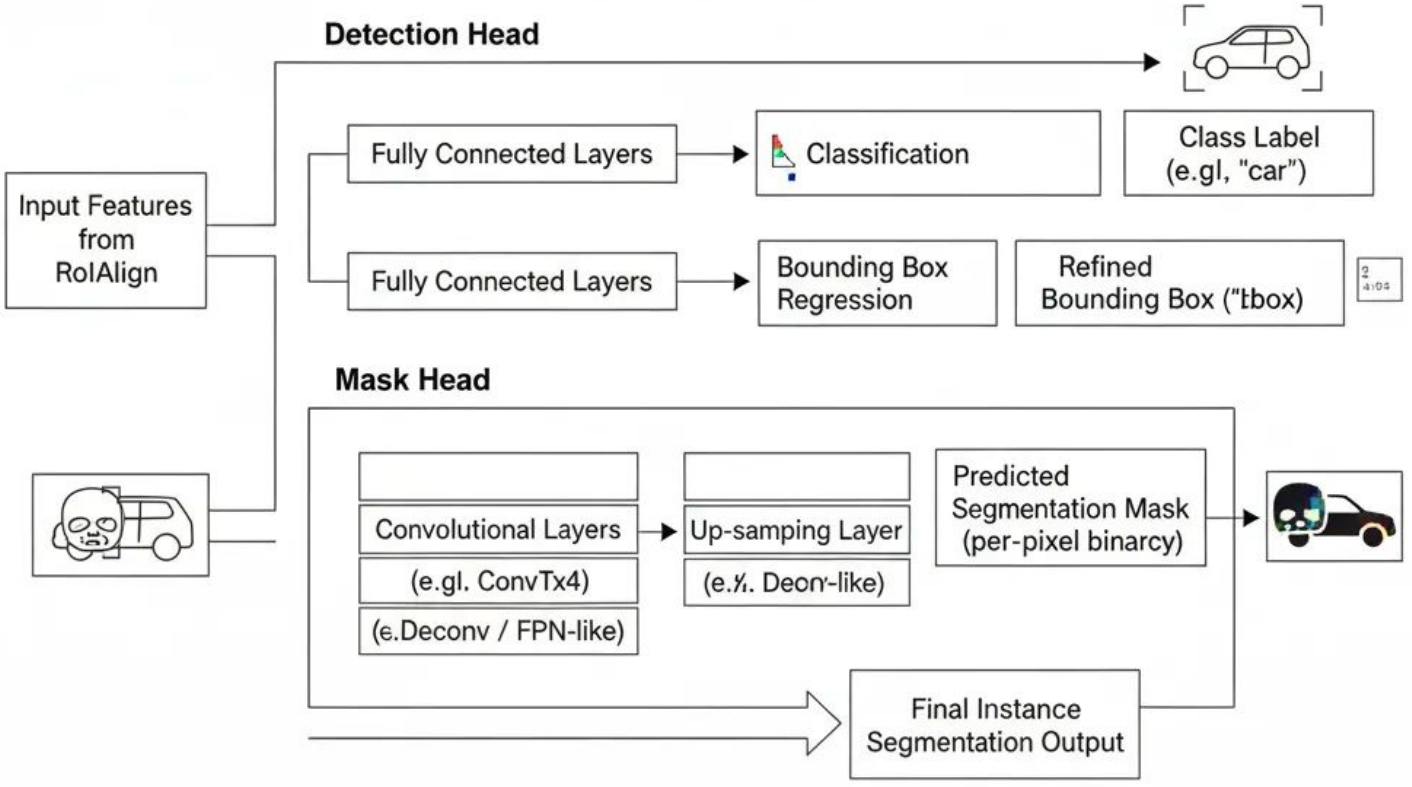
a) Detection Branch

- Performs classification (e.g., car, person, cat)
- Performs bounding box regression (exact box coordinates)

b) Mask Branch

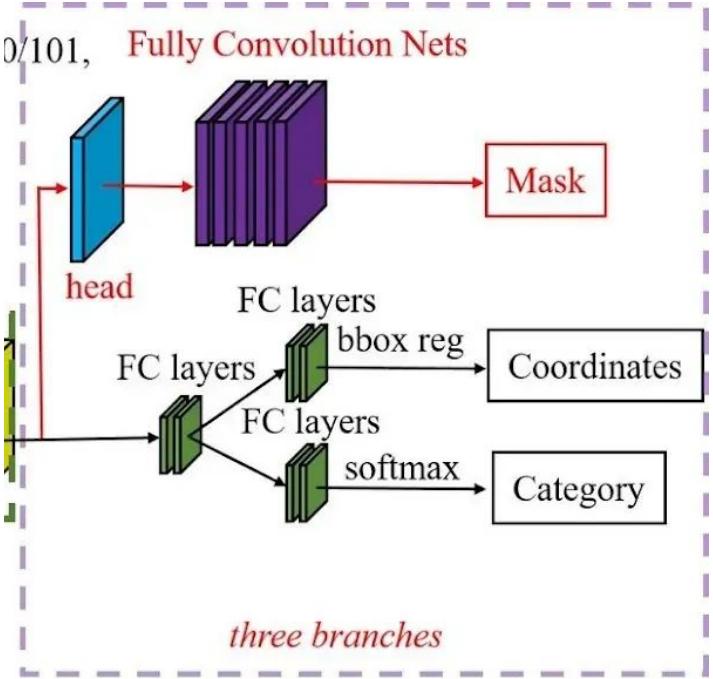
- A fully convolutional network (FCN) that predicts a binary mask for each object.
- Each mask is typically 28×28 pixels and corresponds to one class.
- Later resized and placed back into the original image.
- Press enter or click to view image in full size

Mask Head- Two Parallel Output Branches



Mask Head

- The **Mask Head** is an additional branch in Mask R-CNN, responsible for **generating segmentation masks** for **each region proposal**.
- The head uses the aligned features obtained through ROI Align to predict a binary mask for each object, delineating the pixel-wise boundaries of the instances.
- The Mask Head is typically composed of **several convolutional layers followed by upsample layers (deconvolution or transposed convolution layers).**



We are fetching the **class and bounding box** information from **one layer** and the **mask** information from **another layer**.

Mask R-CNN - Loss Function

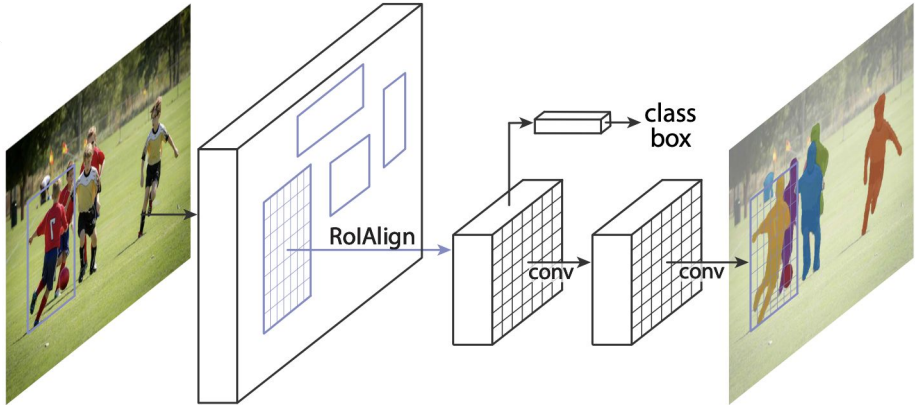
- The total loss is a combination of three components:

  Total Loss = Classification Loss + Bounding Box Loss + Mask Loss

- Classification Loss: Measures if the model predicted the correct class.
- Bounding Box Loss: Measures how well the predicted box fits the object.
- Mask Loss: Measures how accurately the mask matches the object pixels.

This multi-task loss ensures that all three objectives improve together.

Mask R-CNN



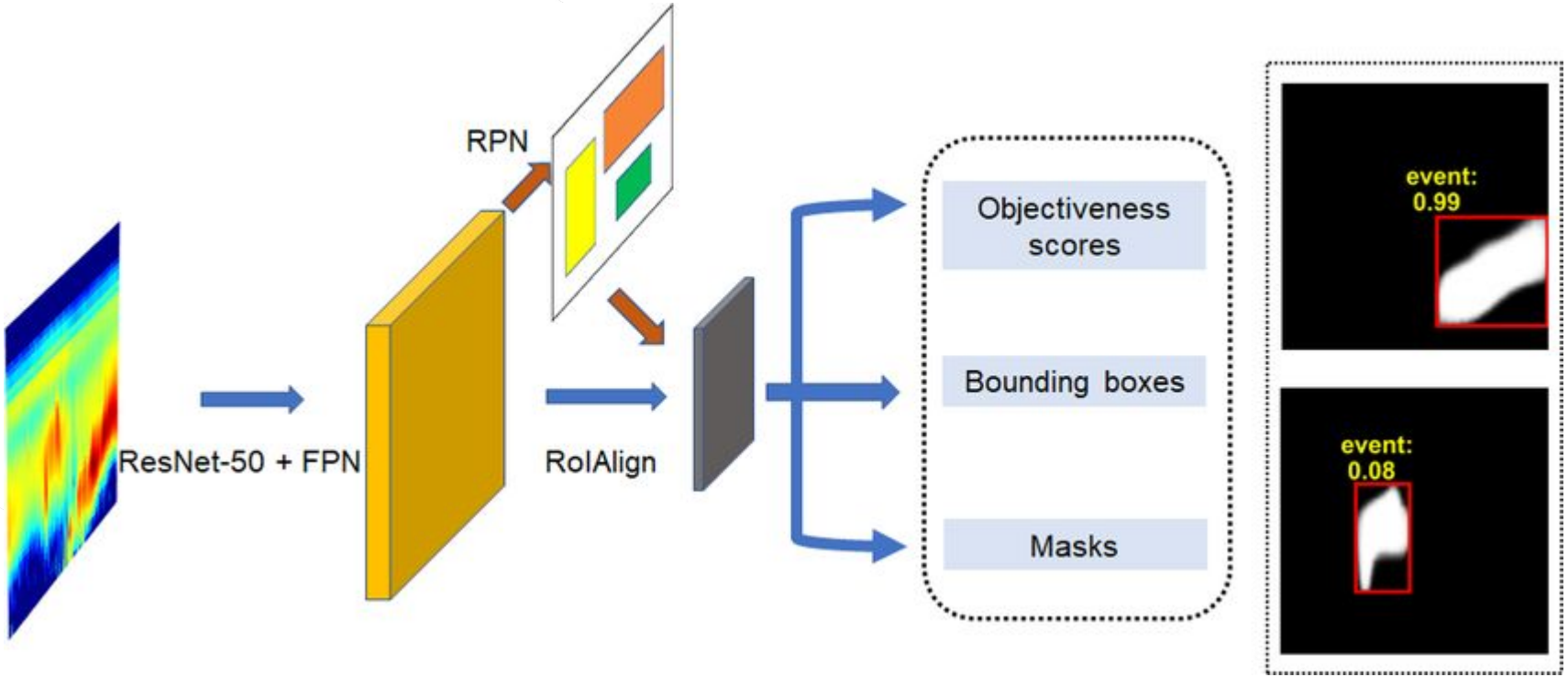
- During training, the model is jointly optimized using a combination of c**lassification loss**, **bounding box regression loss**, and **mask segmentation loss**.
- This allows the model to learn to simultaneously **detect objects**, refine their **bounding boxes**, and produce precise **segmentation masks**.

Output

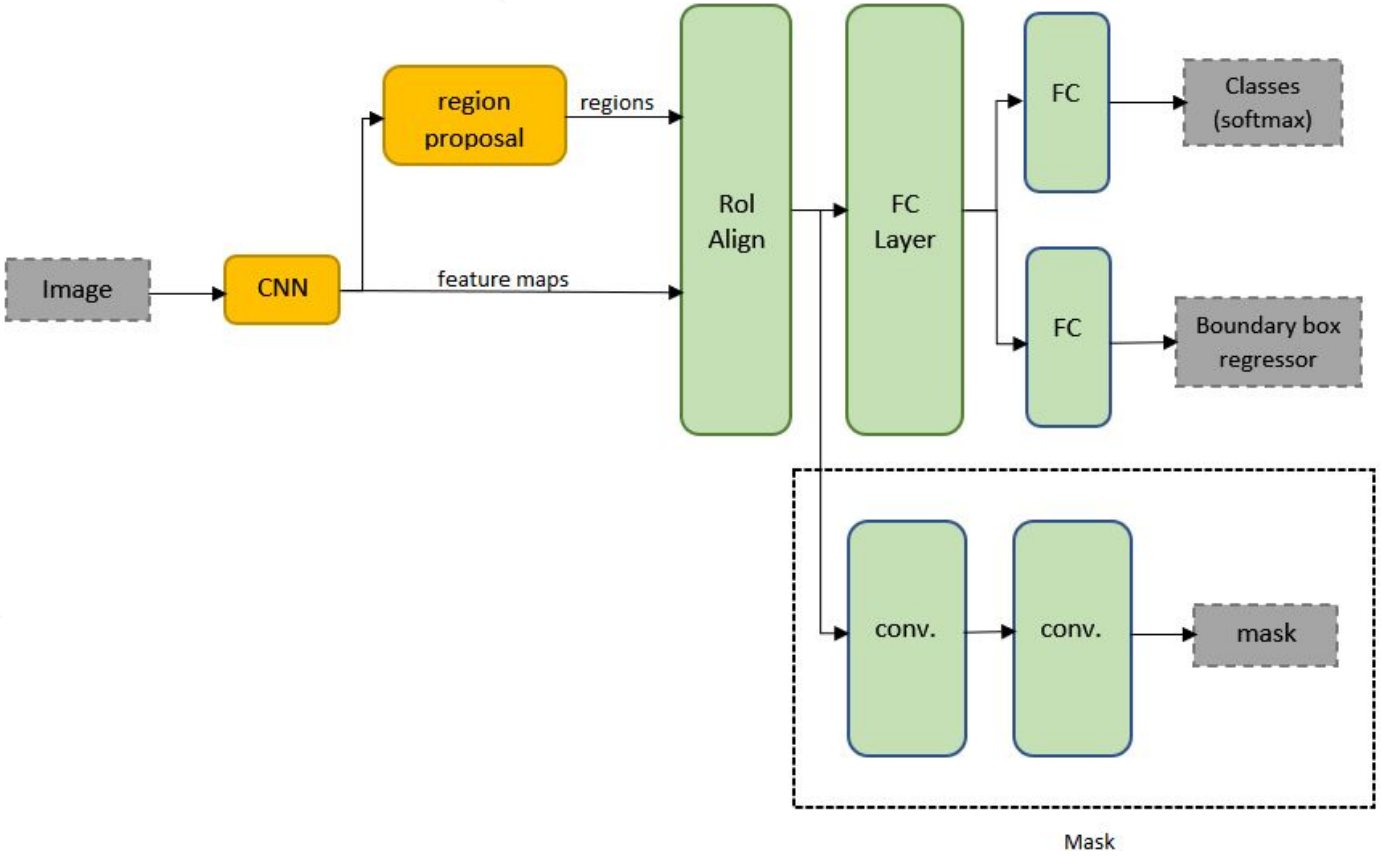
For each object detected, Mask R-CNN produces:

- Class Label : What the object is.
- Bounding Box: Where the object is.
- Segmentation Mask: Exact shape of the object.
- Together, they give a complete, pixel-level understanding of the scene.

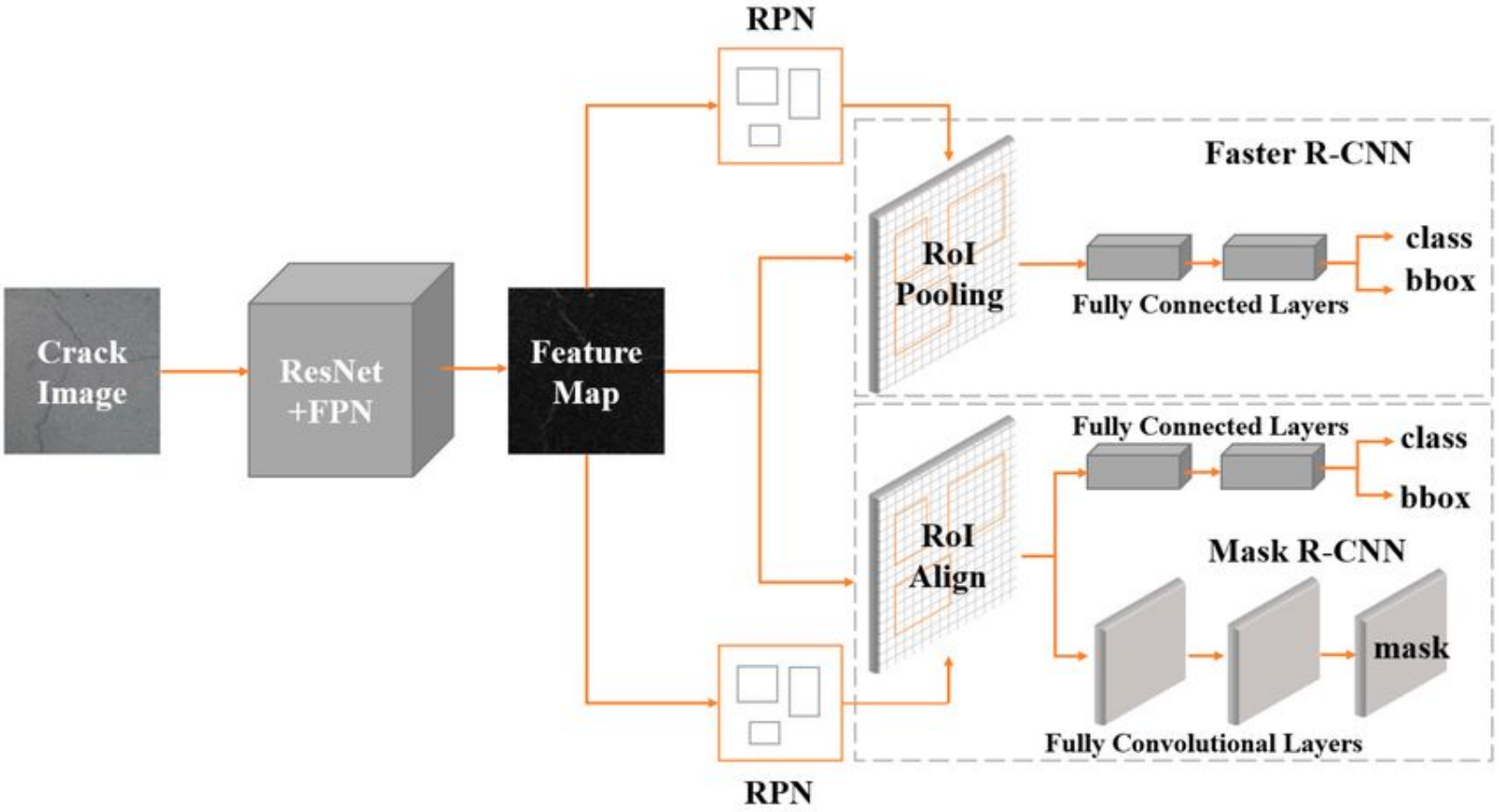
Mask RCNN


RPN
ResNet-50 + FPN
RoIAlign
Objectiveness scores
Bounding boxes
Masks
event: 0.99
event: 0.08

Mask R-CNN in nutshell

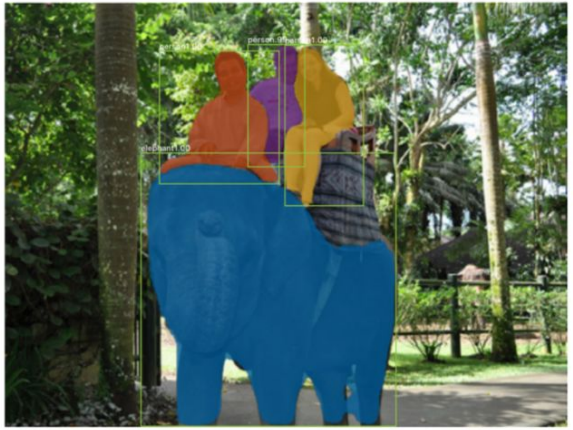


Faster RCNN vs Mask RCNN



Mask R-CNN: Results





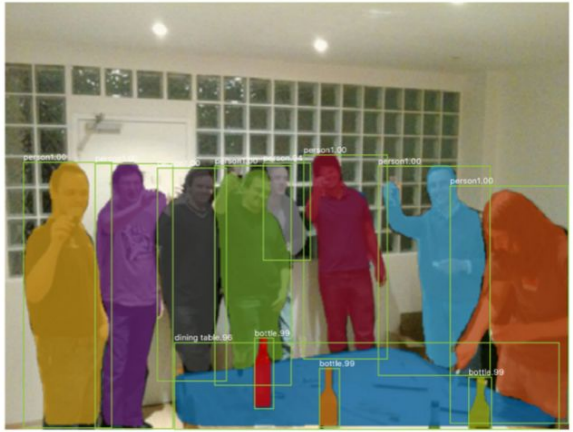




Figure 4. More results of **Mask R-CNN** on COCO test images, using ResNet-101-FPN and running at 5 fps, with 35.7 mask AP (Table 1)

Mask R-CNN performance

	backbone	AP	AP_{50}	AP_{75}	AP_S	AP_M	AP_L
MNC [10]	ResNet-101-C4	24.6	44.3	24.8	4.7	25.9	43.6
FCIS [26] +OHEM	ResNet-101-C5-dilated	29.2	49.5	-	7.1	31.3	50.0
FCIS+++ [26] +OHEM	ResNet-101-C5-dilated	33.6	54.5	-	-	-	-
Mask R-CNN	ResNet-101-C4	33.1	54.9	34.8	12.1	35.6	51.1
Mask R-CNN	ResNet-101-FPN	35.7	58.0	37.8	15.5	38.1	52.4
Mask R-CNN	ResNeXt-101-FPN	**37.1**	**60.0**	**39.4**	**16.9**	**39.9**	**53.5**

Table 1. **Instance segmentation** *mask* AP on COCO `test-dev`. MNC [10] and FCIS [26] are the winners of the COCO 2015 and 2016 segmentation challenges, respectively. Without bells and whistles, Mask R-CNN outperforms the more complex FCIS+++, which includes multi-scale train/test, horizontal flip test, and OHEM [38]. All entries are *single-model* results.

- AP_S: Average precision for small objects.
- AP_M: Average precision for medium objects.
- AP_L: Average precision for large objects
- AP50: Average Precision calculated at an IoU threshold of **0.50**. It is a standard benchmark that evaluates the model's performance when the predicted bounding boxes or masks must have an overlap of at least **50%** with the ground truth bounding boxes or masks.

Mask R-CNN performance

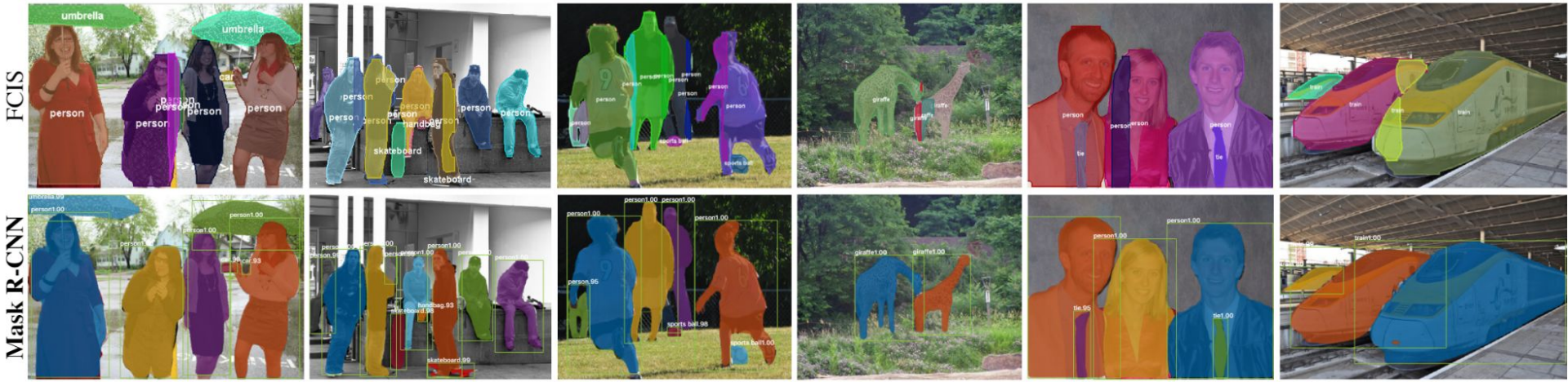


Figure 6. FCIS+++ [26] (top) *vs*. Mask R-CNN (bottom, ResNet-101-FPN). FCIS exhibits systematic artifacts on overlapping objects.

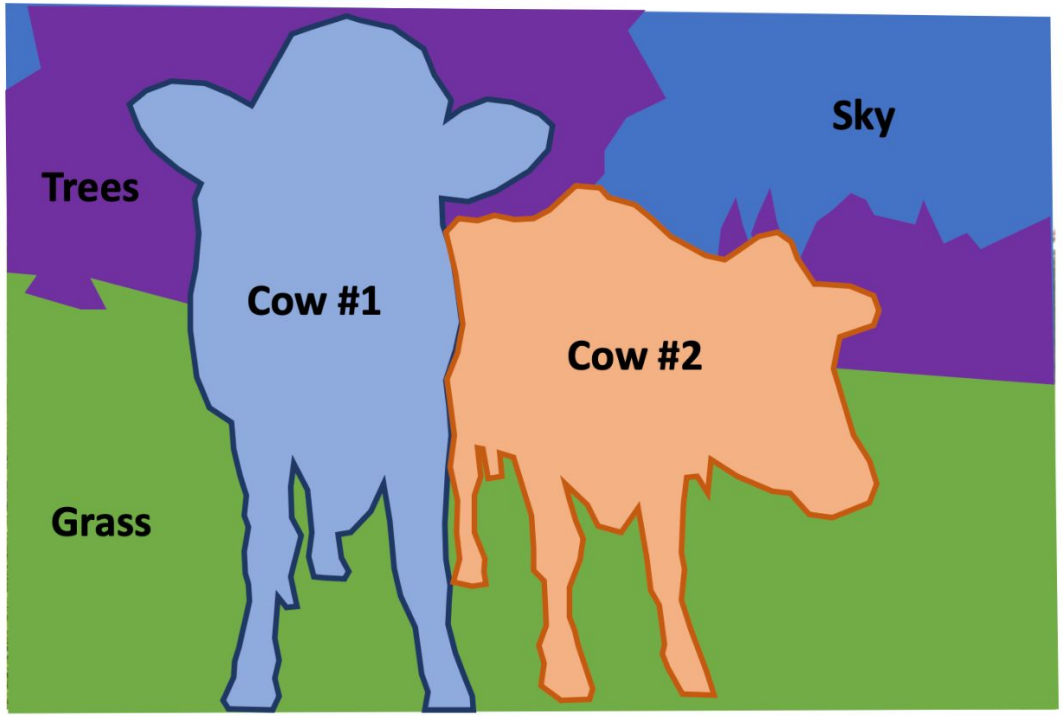
Mask R-CNN: Limitations

- **Computational Complexity:** Training and inference can be computationally intensive, requiring substantial resources, especially for high-resolution images or large datasets.
- **Small and overlapping Object Segmentation**: Mask R-CNN may struggle to accurately segment very small objects due to limited pixel information.
- **Data Requirements**: Training Mask R-CNN effectively requires a large amount of annotated data, which can be time-consuming and expensive to acquire.
- **Limited Generalization to Unseen Categories**: The model's ability to generalize to unseen object categories is limited, especially when data is scarce.

Beyond Instance Segmentation: Panoptic Segmentation

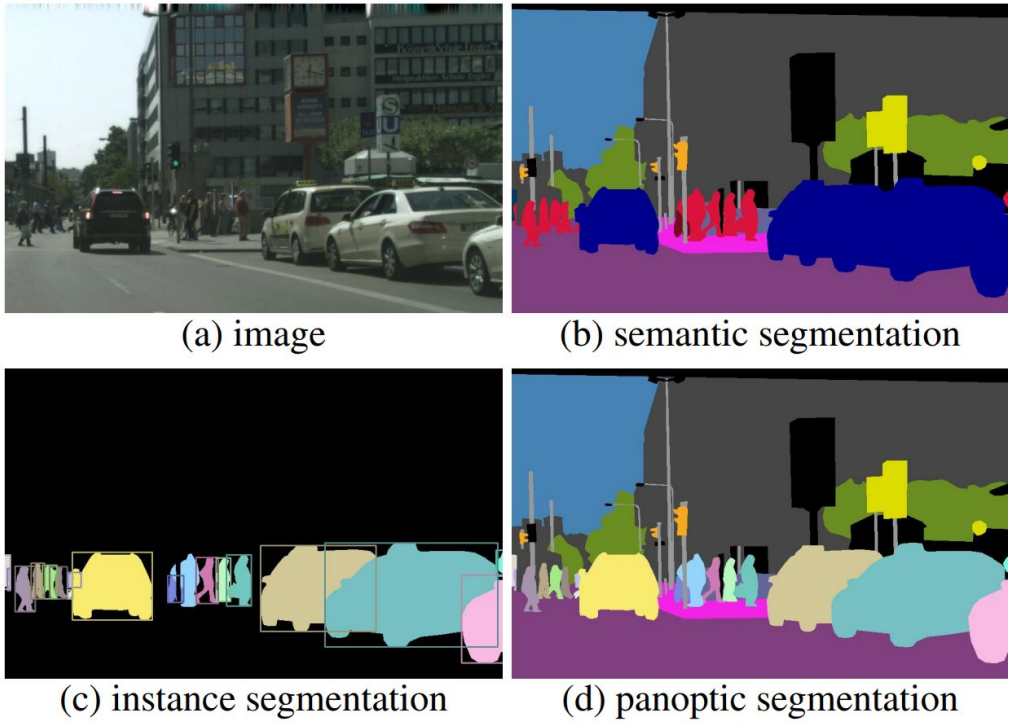
Label all pixels in the image (both things and stuff)

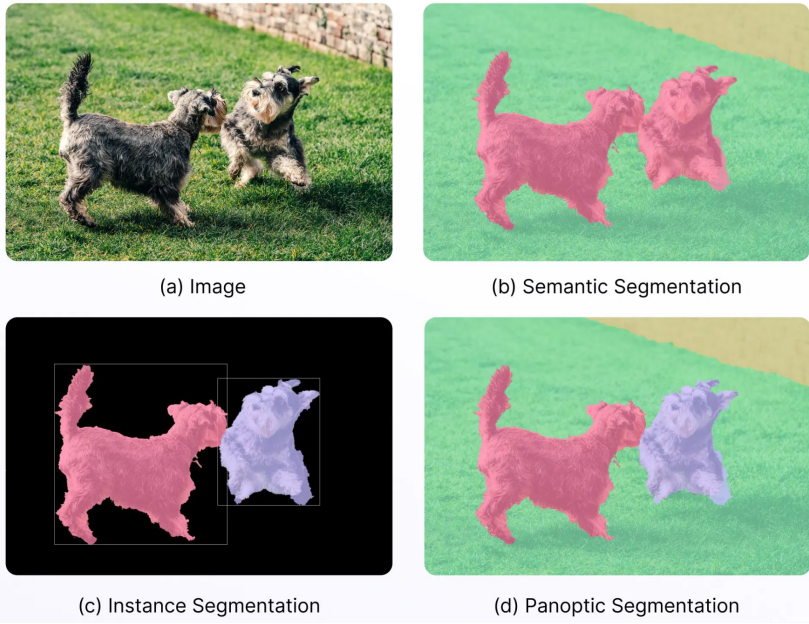
For “thing” categories also separate into instances



Kirillov et al, “Panoptic Segmentation”, CVPR 2019
Kirillov et al, “Panoptic Feature Pyramid Networks”, CVPR 2019

Panoptic Segmentation





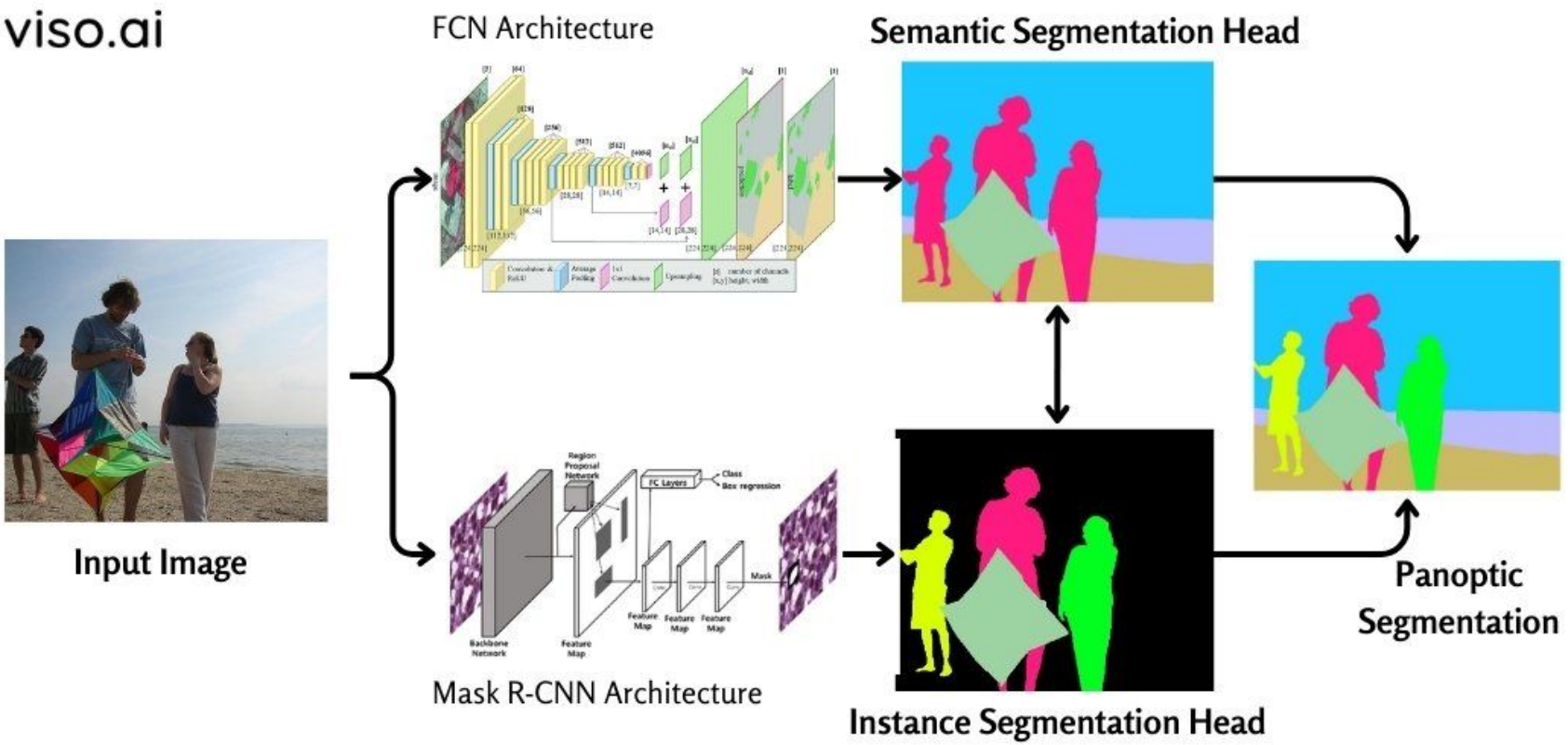
source

Beyond Instance Segmentation: Panoptic Segmentation



Kirillov et al, "Panoptic Feature Pyramid Networks", CVPR 2019

Traditional Panoptic Segmentation



Beyond Instance Segmentation: Human Keypoints

Represent the pose of a human by locating a set of **keypoints**

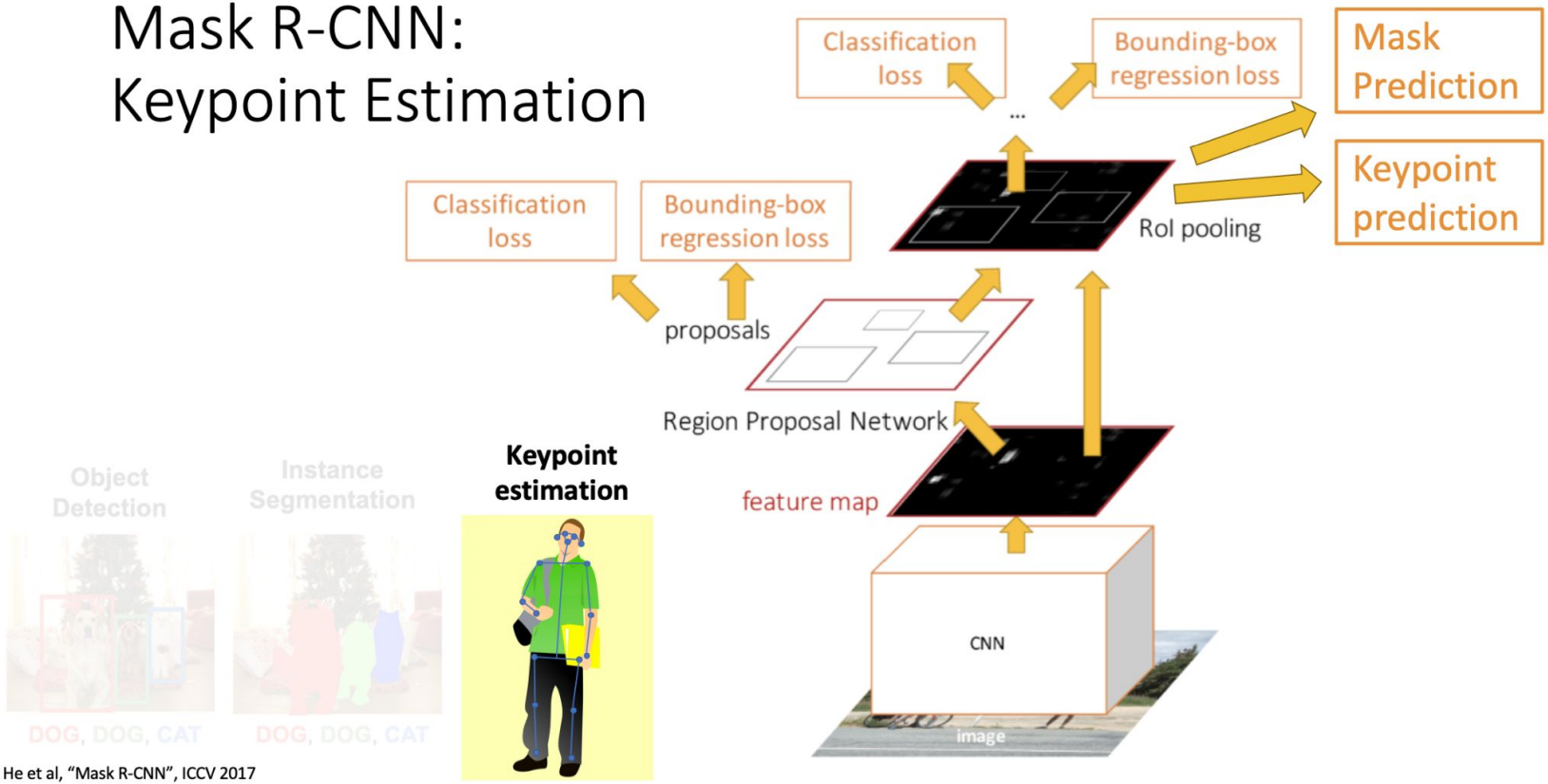
e.g. 17 keypoints:
- Nose
- Left / Right eye
- Left / Right ear
- Left / Right shoulder
- Left / Right elbow
- Left / Right wrist
- Left / Right hip
- Left / Right knee
- Left / Right ankle





[Keypoint detection paper]

Mask R-CNN: Keypoint Estimation



He et al, "Mask R-CNN", ICCV 2017

Joint Instance Segmentation and Pose Estimation



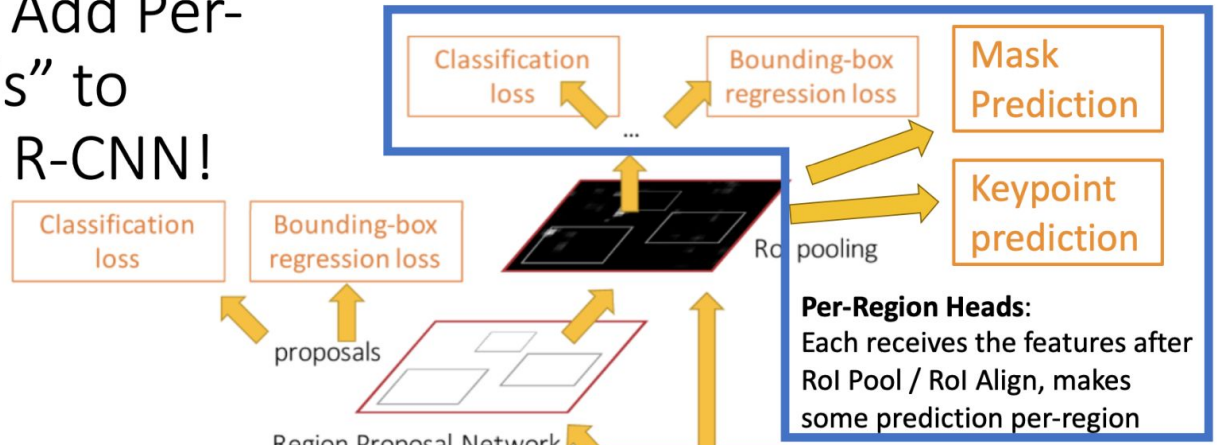
Figure 7. Keypoint detection results on COCO `test` using Mask R-CNN (ResNet-50-FPN), with person segmentation masks predicted from the same model. This model has a keypoint AP of 63.1 and runs at 5 fps.

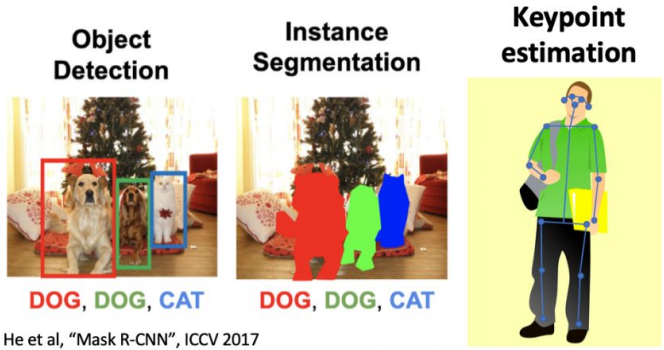
Animate a 3D avatar with your own body movements



Joint Instance Segmentation and Pose Estimation

General Idea: Add Per-Region "Heads" to Faster / Mask R-CNN!

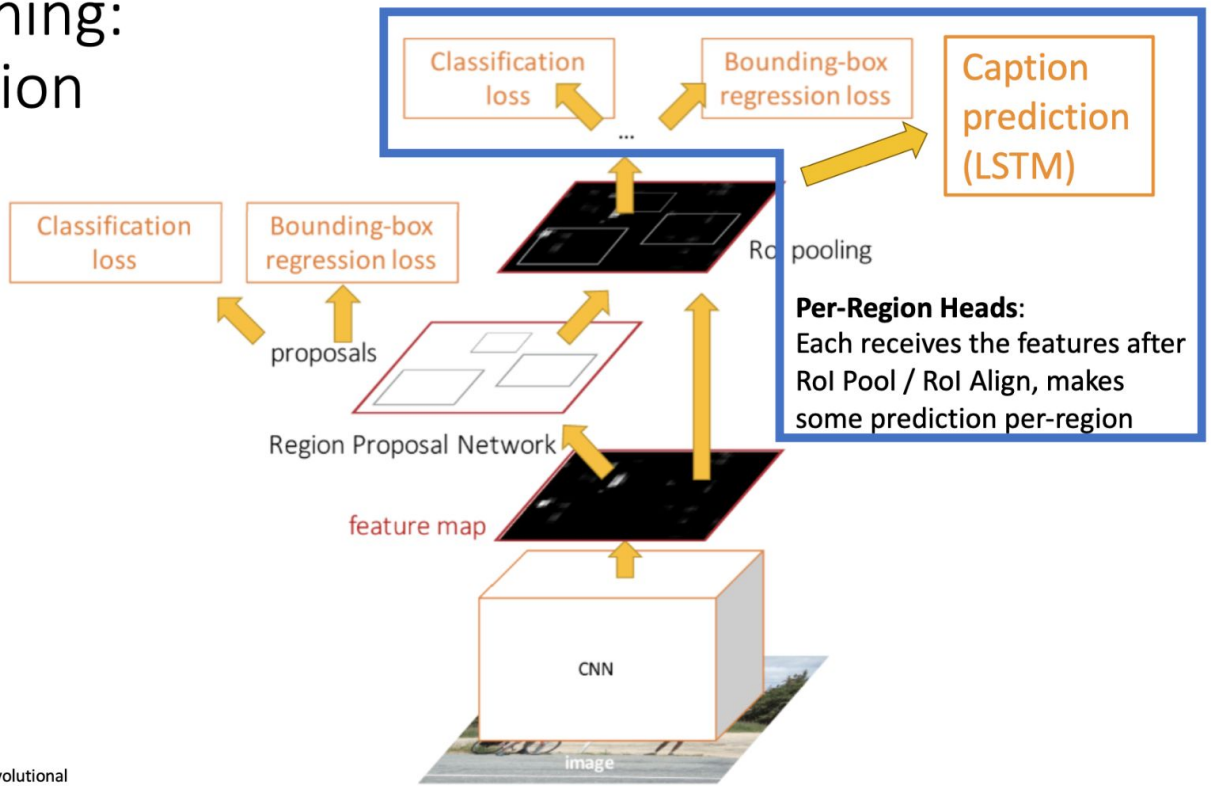




He et al, "Mask R-CNN", ICCV 2017

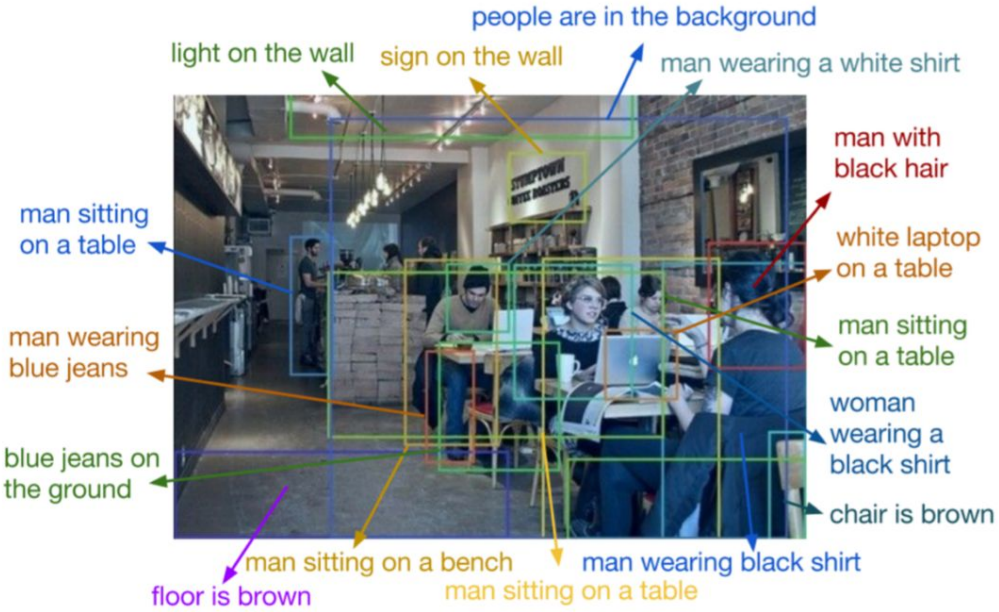
Dense Captioning

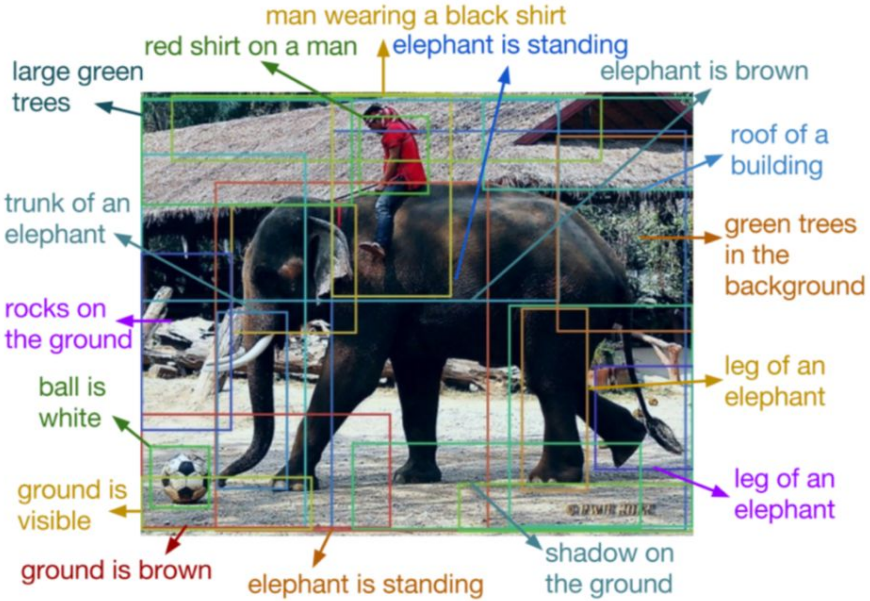
Dense Captioning:
Predict a caption
per region!



Johnson, Karpathy, and Fei-Fei, "DenseCap: Fully Convolutional Localization Networks for Dense Captioning", CVPR 2016

Dense Captioning





Johnson, Karpathy, and Fei-Fei, "DenseCap: Fully Convolutional Localization Networks for Dense Captioning", CVPR 2016



Thank You

Compiled by Ugyen Choden