From Dense Predictions to Sparse Realities: Decoding the Future of Object Detection

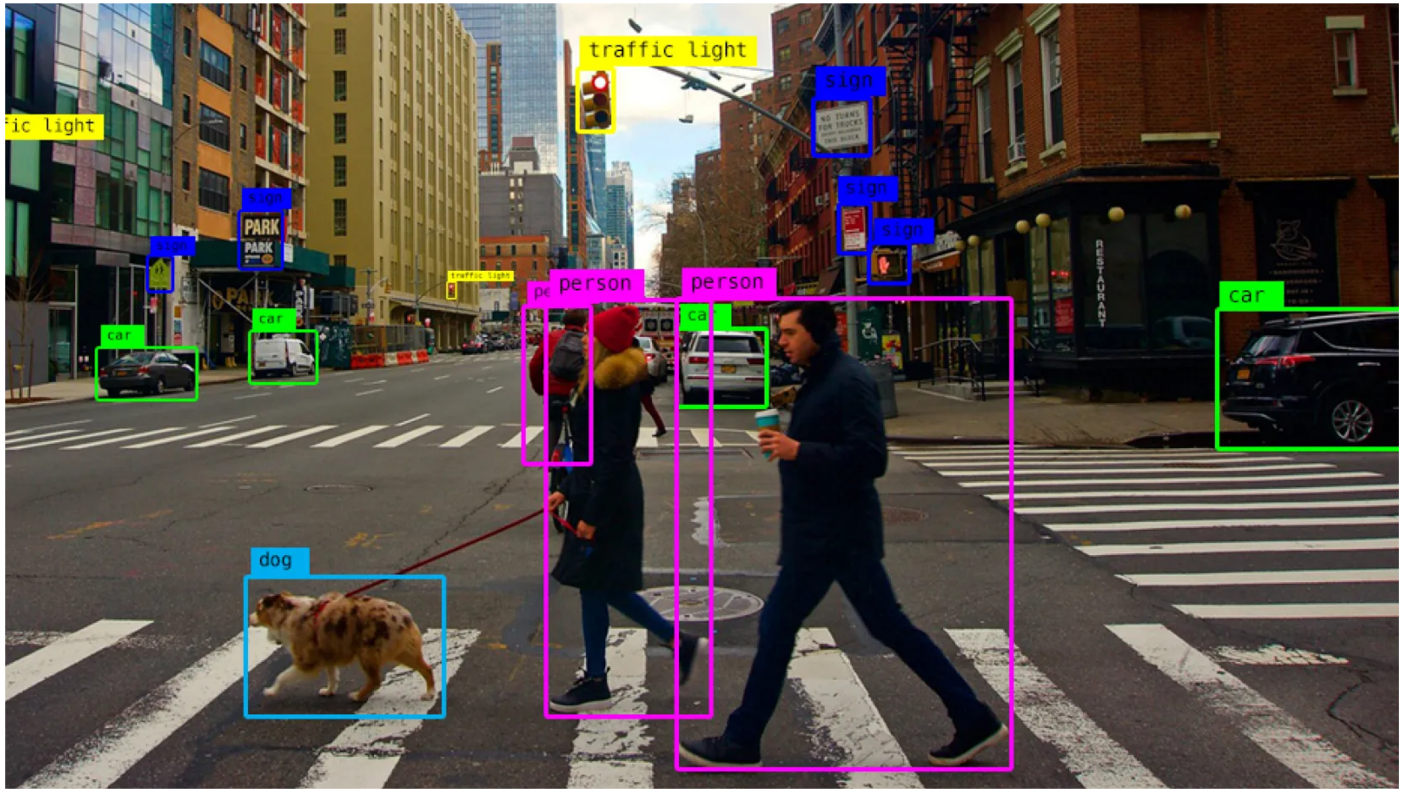
Dragan Alexandru-Samuel

Contents

1. What is Object Detection?
2. How did we do it before end-to-end?
3. Why we want it end-to-end?
4. How do we make it end-to-end?
5. End-to-end architectures

Object Detection

- A method to determine the position and class of objects in an image
- Real-world => Multiple objects, multiple classes
- Multiple types of doing it
  - Bounding box
  - Mask


traffic light
sign
sign
sign
person
person
car
car
car
dog
RESTAURANT
PARK


car:1.00
car:1.00
car:1.00
car:1.00
car:0.89
bus:0.75
car:1.00
person:0.82
car:1.00
person:1.00
person:1.00
person:1.00
person:1.00

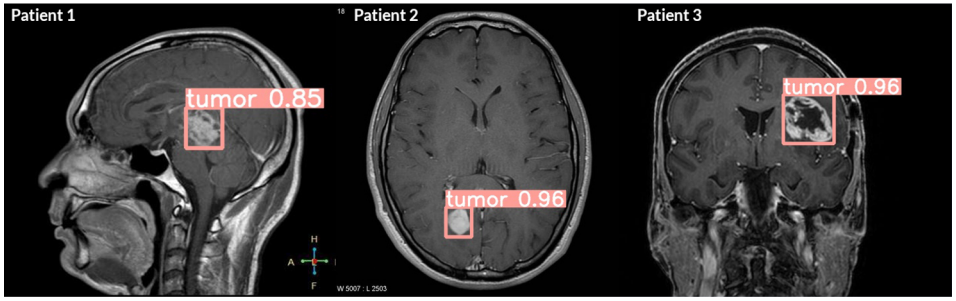
What's so hard about it?

- Objects can be (partially) occluded
- Image may contain a large number of them

Why do we care about it?

- Tumor detection
- Fall detection in elderly people
- Obstacle & pedestrian detection in autonomous driving

and we’re only getting started...

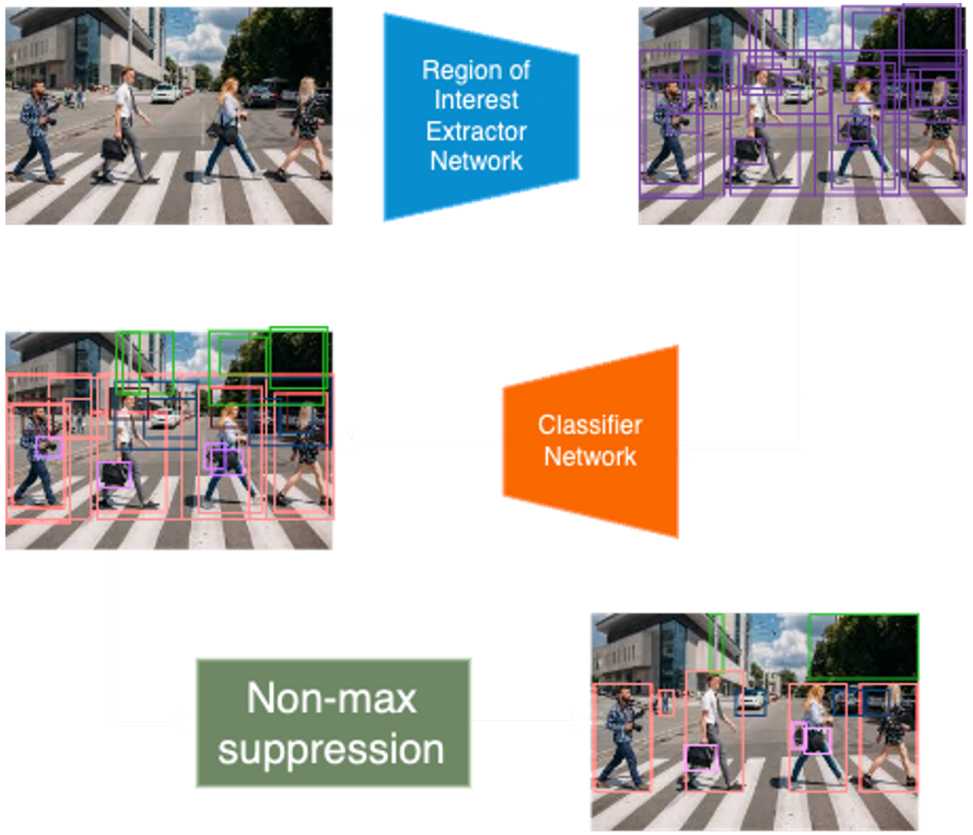


How did we do object detection?

Two types of algorithms

- Two stage object detectors
  - First model => extracts regions of interest
  - Second model => refines and classifies those regions into objects
- One stage object detectors
  - One model to do it all

Two-stage Detector



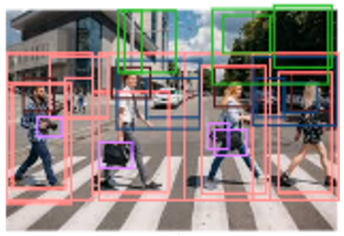
Some Two-Stage Detector Algorithms

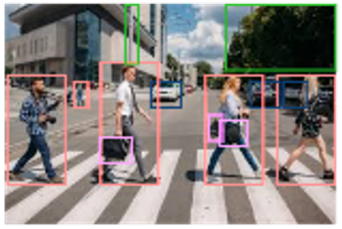
- R-CNN (2013, Ross Girshick et al.)
- Fast R-CNN (2015, Ross Girshick)
- Faster R-CNN (2015, Ren et al.)
- Mask R-CNN (based on Faster R-CNN, 2017, He et al.)

Two-Stage Object Detector

- More accurate
- Better at localization
- More complex models
- Bigger models

One Stage Detector





Non-max
suppression

Some One-Stage Detector Algorithms

- SSD (2015, Liu et al.)
- YOLO Family
  - YOLO (2015, Redmon et al.)
  - YOLO 9000 (2016, Redmon et al.)
  - YOLO V3 (2018, Redmon et al.)
- FCOS (2019, Tian et al.)

One-Stage Object Detector

- Simpler models
- Suitable for real-time detection
- Less accurate (esp. for smaller objects)
- Worse at localization

The problem with current models

- Create lots of object candidates
- Additional algorithms are needed for image pre/post-processing
- E.g.
  - YOLO uses K-Means to find optimal bounding box sizes
  - Most algorithms require NMS as a post-processing step

The Solution? Go End-To-End..

Why end-to-end?

- Model is simpler
- The model learns better ways to solve a task

E.g: in NLP it achieved state-of the art performance

  - Neural Machine Translation
  - Text to Speech

How do we go end-to-end?

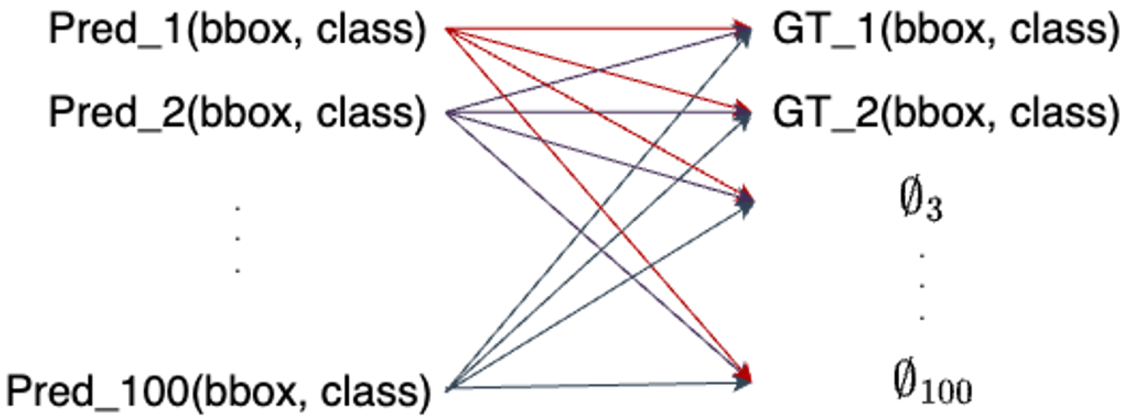
- Flow in the “Classic” way - during training
  - 1. Pass image through model
  - 2. Create lots of RoIs
  - 3. Reduce (classify them)
  - 4. Reduce (NMS)
  - 5. final bounding boxes
  - 6. Compute localization (bbox) and classification error
  - 7. Backprop

How do we go end-to-end?

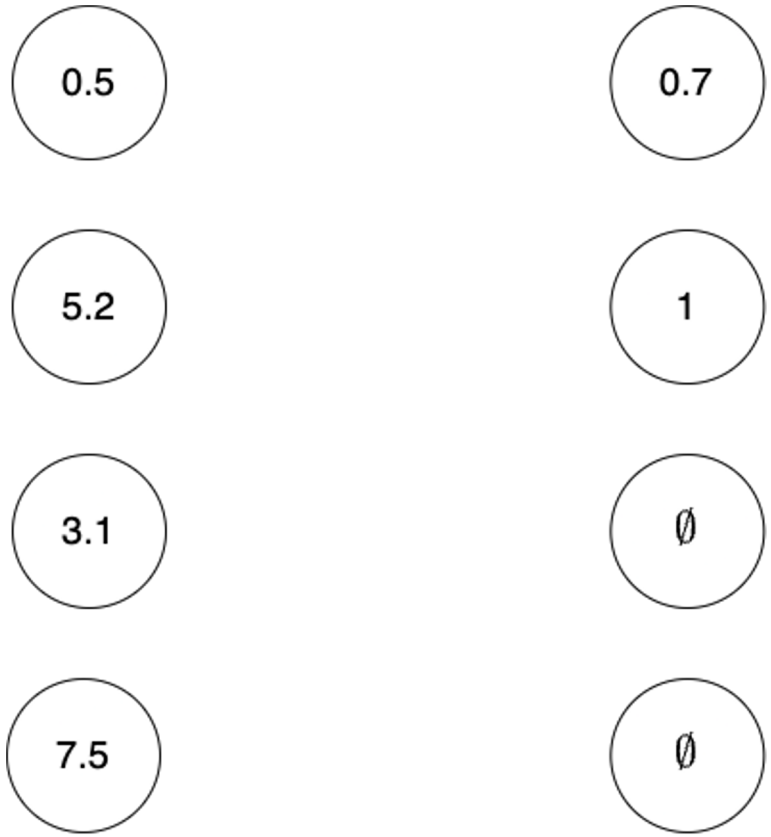
- The “New” way - during training
  - 1. Pass image through model
  - 2. Extract a fixed set of bbox candidates (e.g 100) and classify them
  - 3. Final set of predictions
  - 4. **!!!** Compute loss between **set of** predictions and **set of** ground truths
  - 5. Backprop

Enter bipartite matching loss

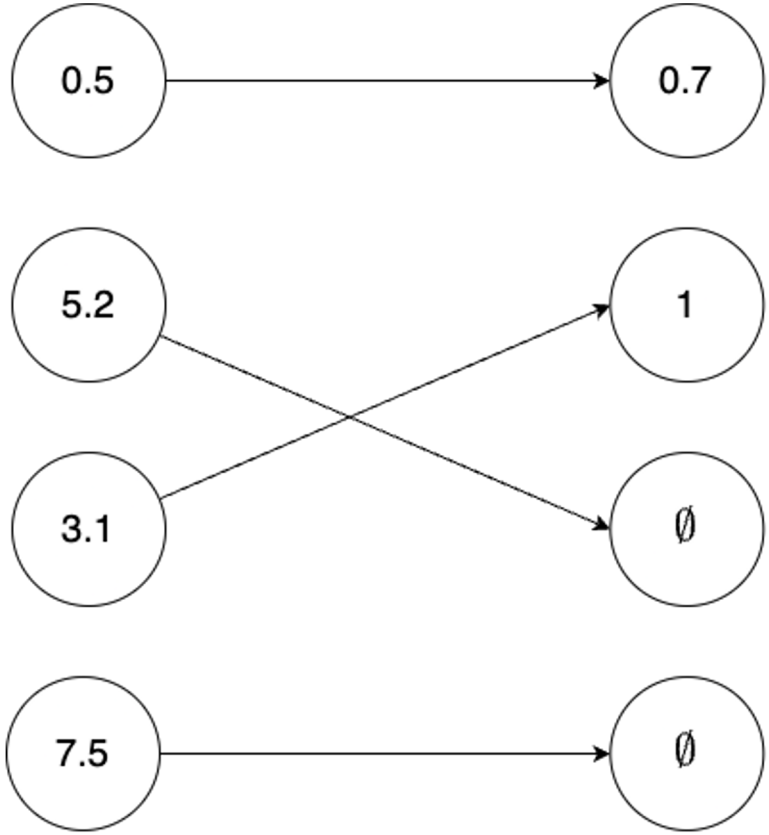
- What it does?
  - 1. Take set of predictions
  - 2. Take set of ground truths
  - 3. Computes “distance” between each (pred., truth) pair
  - => Optimally assign **a** Prediction to **a** Ground Truth (Hungarian algorithm)



Bipartite matching loss - An example



Bipartite matching loss - An example



Enter bipartite matching loss

How do we compute the “distance”?

1. Distance between the BBoxes (IoU & L1 distance)
2. Distance between the classes (cross-entropy)
3. Combine them linearly
   (e.g: α * L_class(pred_c, gt_c) + β * L_bbox(pred_bbox, gt_bbox)

End to end architectures

- DETR (DEtection TRansformer) (2020, Carion et al.)
- Sparse-RCNN (2021, Sun et al.)
- DeFCN (2020, Wang et al.)

DEtection TRansformer (DETR)

- Transformer based encoder-decoder network
- Loss = Hungarian loss
- Fixed number of Object queries - guide candidate creation

DEtection TRansformer (DETR)

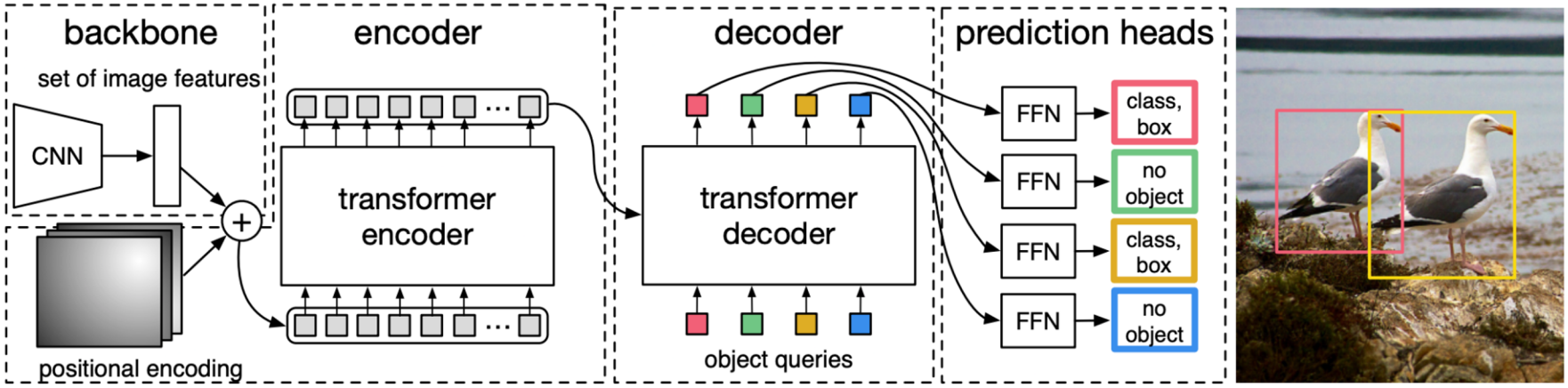


Image by authors

Sparse R-CNN

- Based on **Faster R-CNN**
- Uses a fixed number of objects
  - Sparse boxes
  - Sparse candidates
- Dynamic Instance Interactive Head
  - Iterative design
  - Combines (box, candidate) pair to extract prediction

Sparse R-CNN

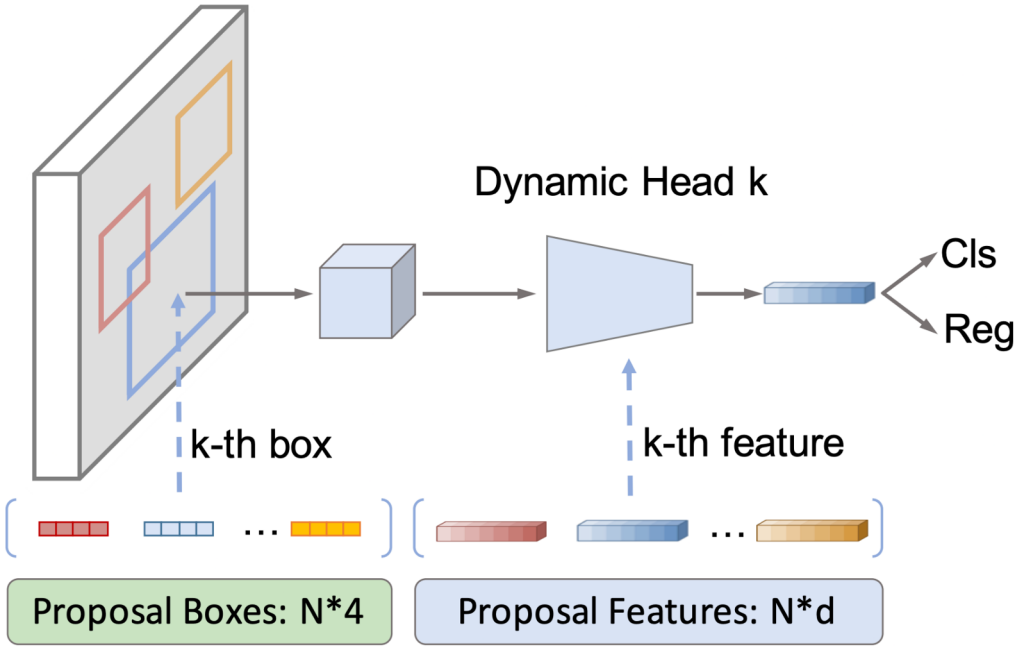


Image by authors

DeFCN (Detection Fully Convolutional Network)

- Based on FCOS
- Improved Loss - Prediction aware one-to-one assignment (POTO)
  - Geometric mean to build the loss
- 3D Max Filter (3DMF) to act as NMS

DeFCN (Detection Fully Convolutional Network)

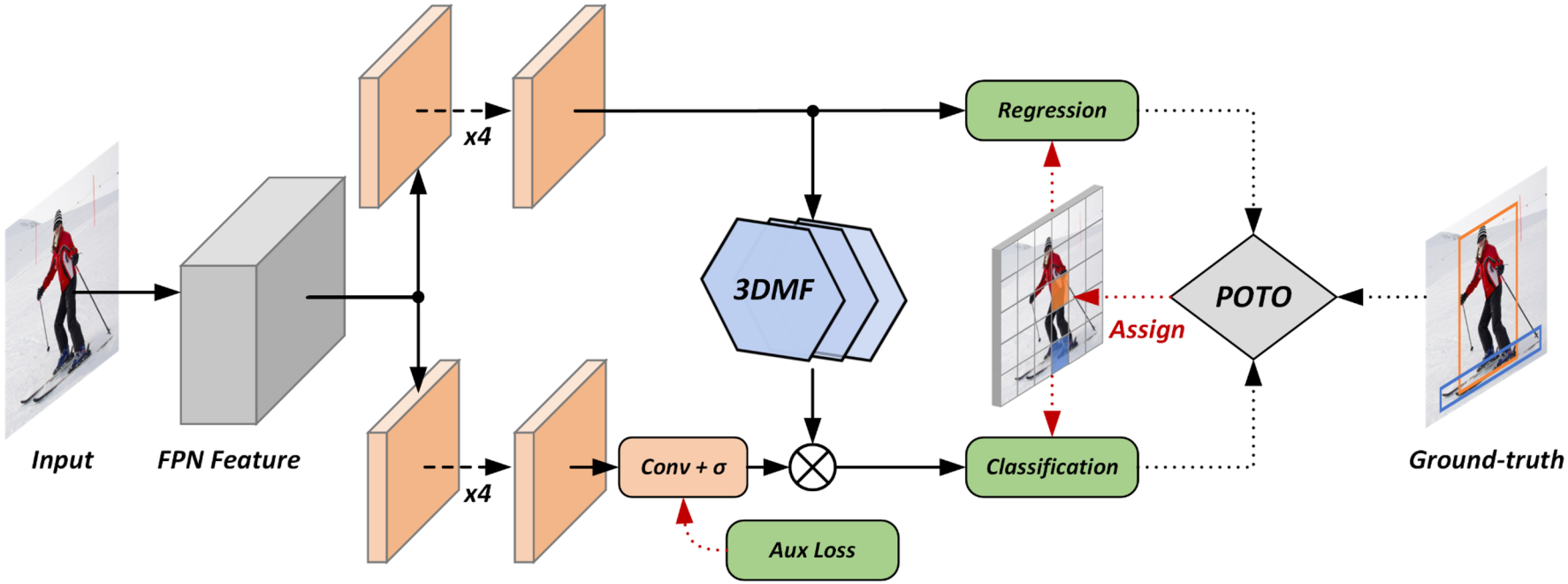


Image by authors

Thank you!