Object detection with Deep Neural Networks

Perumadura De Silva, Kolli Abhiram

Object detection with Deep Neural Networks

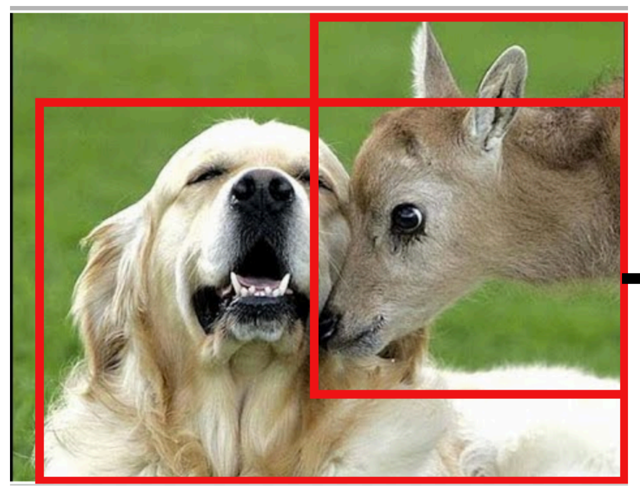
Content

- Introduction to object detection
- Required tools
- Multi Stage detectors
- Evaluation metrics for object detection
- Practical example

Object detection with Deep Neural Networks

Introduction

- Object detection is an extension of image classification task
- Multi-Class classifiers: Focuses on only one object per image
- Multi-Label classifiers: Focuses on multiple objects per image
- Object detection: Focuses on multiple objects and their locations per image



Detection model(s) and/or mechanisms (Feature Extraction)

Bounding Box Coordinates and classes (Classifier and Regressor heads)

Highlevel overview of Object detection

Detection model acts as a feature extractor

Classification head performs class classification

Regression Head performs bounding box coordinate regression

Object detectors are a combination of classifiers and regressors

Object detection with Deep Neural Networks

Introduction

- Datasets in object detection:
  - Unlike in the classification case, a dataset for object detection requires the bounding box coordinates for each class
  - The bounding box coordinates can be defined with different formats
    - Pascal VOC Format: XML
    - COCO Format: JSON object
    - The bounding box coordinates can be in $[min, max]$ format or $[center, hight, width]$ format
  - Both these formats can represent labels for other tasks such as:
    - Image segmentation
    - Key Point detection

```xml
<annotation>
        <folder>Kangaroo</folder>
        <filename>00001.jpg</filename>
        <path>./Kangaroo/stock-12.jpg</path>
        <source>
                <database>Kangaroo</database>
        </source>
        <size>
                <width>450</width>
                <height>319</height>
                <depth>3</depth>
        </size>
        <segmented>0</segmented>
        <object>
                <name>kangaroo</name>
                <pose>Unspecified</pose>
                <truncated>0</truncated>
                <difficult>0</difficult>
                <bndbox>
                        <xmin>233</xmin>
                        <ymin>89</ymin>
                        <xmax>386</xmax>
                        <ymax>262</ymax>
                </bndbox>
        </object>
</annotation>
```

Ref.1: Pascal VOC format

```
{
    "info": {...},
    "licenses": [...],
    "images": [...],
    "annotations": [...],
    "categories": [...], <-- Not in Captions annotations
    "segment_info": [...] <-- Only in Panoptic annotations
}
```

Ref.2: COCO format

Object detection with Deep Neural Networks

Required tools

- Dataset annotator:
  - Performs the annotation of the bounding boxes
  - In most cases the bounding boxes are orthogonal
  - Rotates bounding boxes that encodes the rotation angle is possible
    - In this case one has to use a specific type of detection models
      - Check refs for some resources
  - Most data annotation tools can perform other tasks such as: Keypoint annotation, Image segmentation

Object detection with Deep Neural Networks

Required tools

- Object detection frameworks:
  - Since the object detection task is more complex than the image classification, the underline models can be complex
  - Therefore popular frameworks such as TensorFlow and PyTorch has object detection libraries which simplifies the training and model inference process
    - Model Inference: Once a model is trained, The model graph + trained parameters can be extracted to deploy in production.
    - The goal of graph extraction is to prune all the nodes that were used for training (optimiser ops, gradient ops) and keep only a forward model
    - Advance inference techniques: Model weight quantisations; Sacrifice the classification performance for resource saving

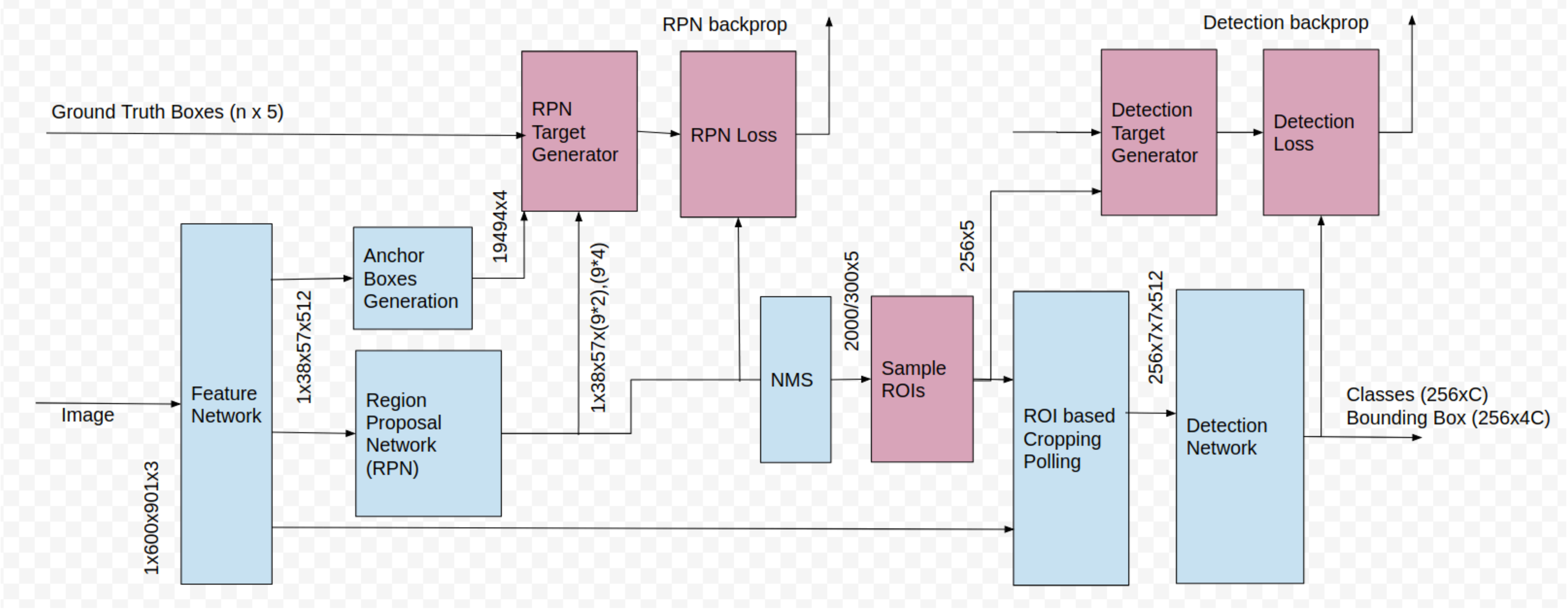
Object detection with Deep Neural Networks

Multi Stage detectors

- The multistage detectors:
  - object proposal stage
    - Selective search for potential object proposals, Region proposal networks (RPN) that learns to find proposals
    - Suggest the objectness of a proposal and predict the required coordinate shift from a reference position
  - Proposal classification and regression stage
    - This stage further fine tune the suggested proposals by:
      - classifying them in to the respective classes
      - regress the required coordinate shift from current coordinate position

Object detection with Deep Neural Networks

Multi Stage detectors (Faster RCNN)



Ref.3: Faster RCNN model

Object detection with Deep Neural Networks

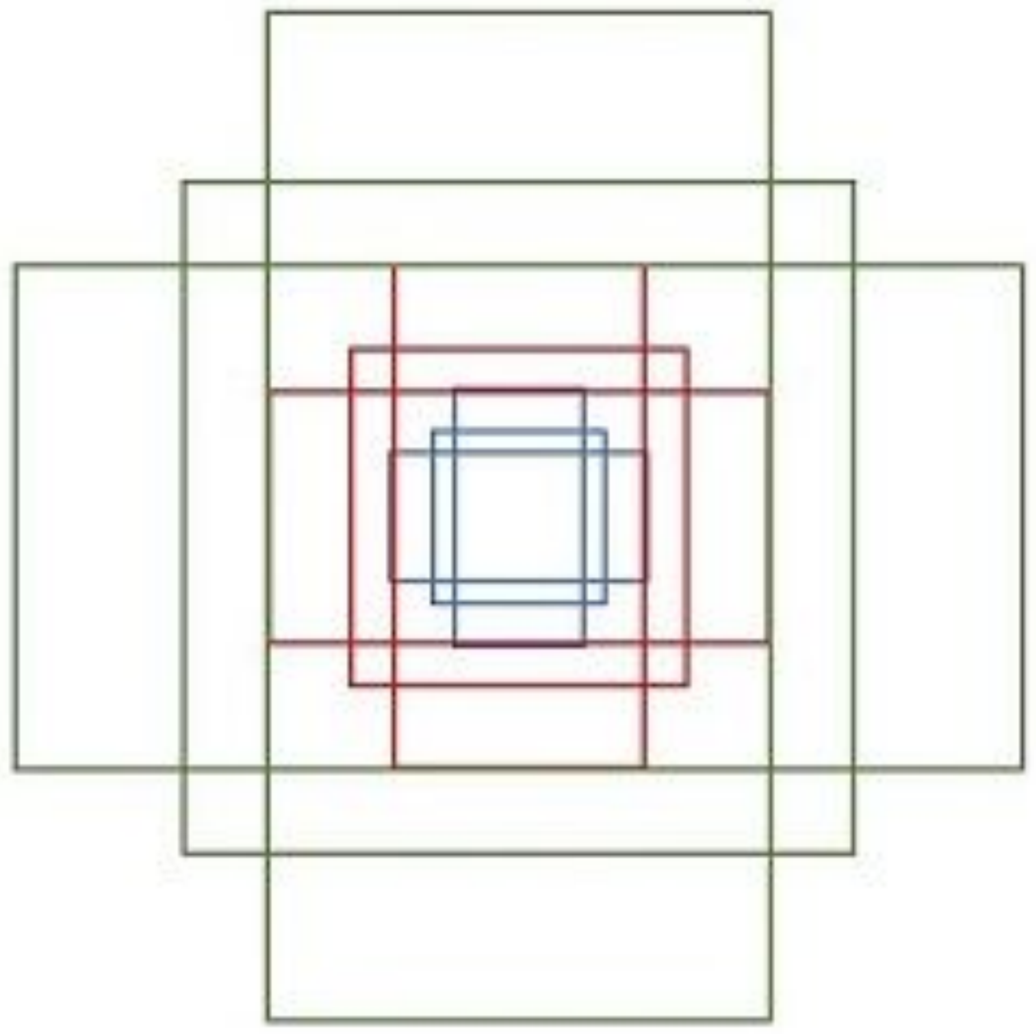
Multi Stage detectors (Faster RCNN Stage-1)

- **Feature Network:**
  - A typical Convolutional Neural Network (CNN) such as VGG or ResNet can be used
  - For Fast model inference a MobileNet (Ref.4) can be used
  - Take VGG16 architecture: This model reduces the input by a ratio 32
    - Input (800x800) — VGG —> output (25x25)

Object detection with Deep Neural Networks

Multi Stage detectors (Faster RCNN Stage-1)

- **Anchor box generation:**
  - Anchors are used as the reference frames to find object proposals
  - It is an application specific hyper parameter
  - For each point in feature map:
    - Generate N number of anchors from different scales and aspect ratios (M)
  - So in the VGG16 case 25x25*9 = 5625 anchor boxes per feature map
  - Each anchor box is represented as 4 coordinates $[x_{min}, x_{max}, y_{min}, y_{max}]$
  - For 9 anchor boxes, $(M,4)$ matrix for each anchor location



Ref.5: 9 anchor boxes from 3 different scales and aspect ratios

Object detection with Deep Neural Networks

Multi Stage detectors (Faster RCNN Stage-1)

- **RPN Target generator**
    - Find target proposals by calculating the Intersection of Union (IoU) between an anchor and a ground truth (GT) object
    - One GT can have multiple anchors
    - $IoU = \frac{Area_{Overlap}}{Area_{Union}}$
    - If $IoU > 0.7$ label the anchor as positive (object)
    - If $0.0 < IoU < 0.3$ label the anchor as Negative (background)
    - Discard the anchors that are neither positive nor negative from training
    - So for each anchor box there will be $(N,2)$ classes and $(N,4)$ coordinates

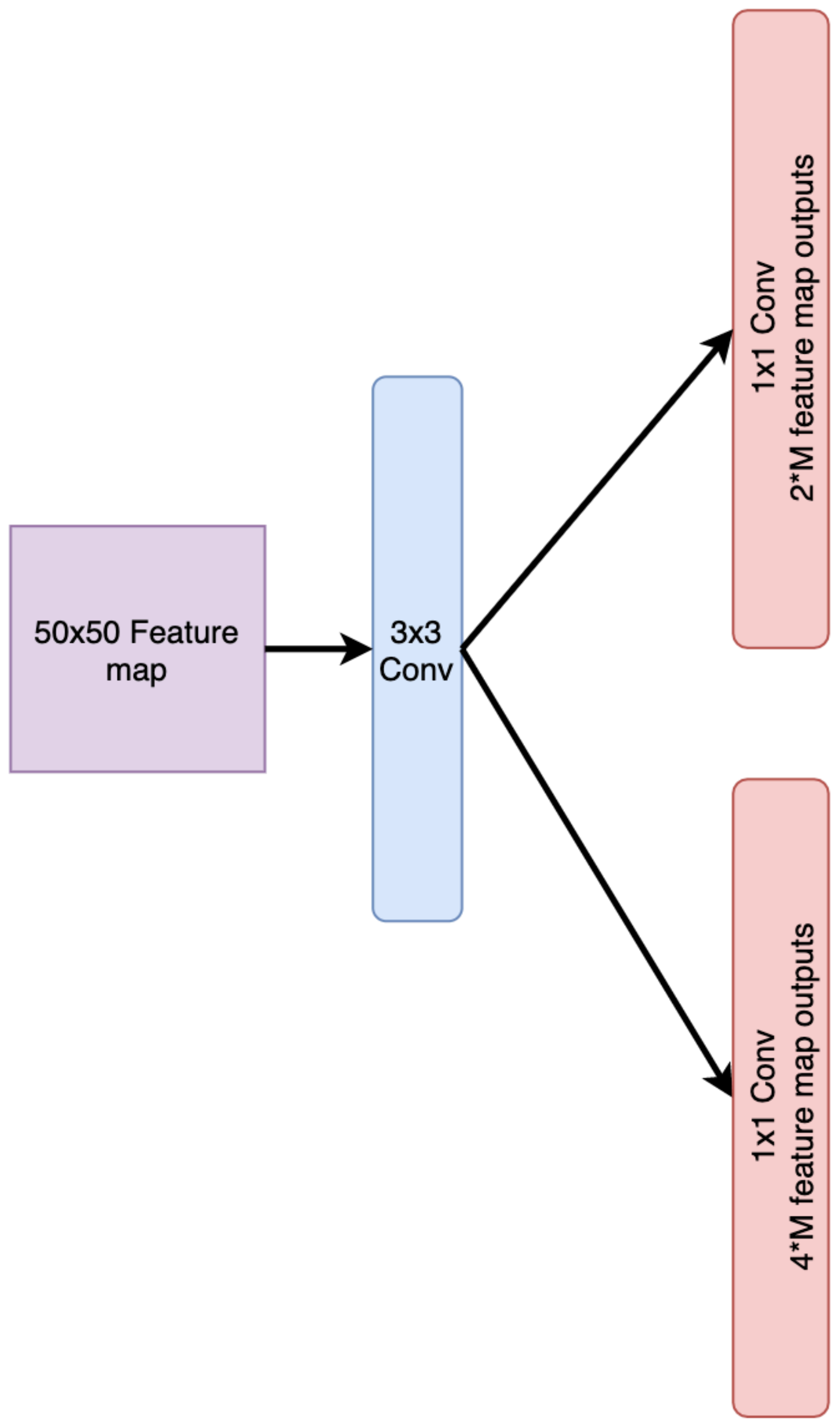
Ref.6: Overlap

Ref.6: Union

Object detection with Deep Neural Networks

Multi Stage detectors (Faster RCNN Stage-1)

- **Region Proposal Network (RPN)**
  - Take the feature extractor output as the input
  - Classify for each anchor location:
    - Class logit (classification layer), 1 for object and 0 for background
    - Center shift (x, y) and scale variation (w, h)
  - These regressed values show how to shift the centre of the anchor and change the h, w
    - $t_x = \frac{x - x_a}{w_a}, t_y = \frac{y - y_a}{h_a}$; Scale invariant translation of the centre Eq.1
    - $t_w = log\frac{w}{w_a}, t_h = log\frac{h}{h_a}$; Log-space translation of the bounding box height width Eq.2
    - x, y predicted centers, x_a, y_a anchor centers, w, h predicted width height, w_a, h_a anchor width height
  - The Regression layer learns the $t_{x,y}, t_{w,h}$ functions and the predicted translation can be found from the equations
- **RPN Loss:**
  - $L(p_i, t_i) = \frac{1}{N_{cls}} \sum_i L_{cls}(p_i, p_i^*) + \lambda \frac{1}{N_{reg}} \sum_i p_i^* L_{reg}(t_i, t_i^*)$ (Ref.7)



Region Proposal Network

Object detection with Deep Neural Networks
Multi Stage detectors (Faster RCNN Stage-1)

- Non-Max Suppression (NMS):
  - Since one GT box can have multiple anchors, we get multiple RoIs with similar overlap
  - NMS is used to remove all the RoIs except for the best RoI that represent the GT
  - Algorithm:
    1. Take the highest score for the objectiveness
    2. Remove all: IoU > Threshold (0.5 for e.g) boxes
    3. Move to the next highest objectiveness
    4. Repeat from 1
  - Refer to Ref.9 for more information on implementation

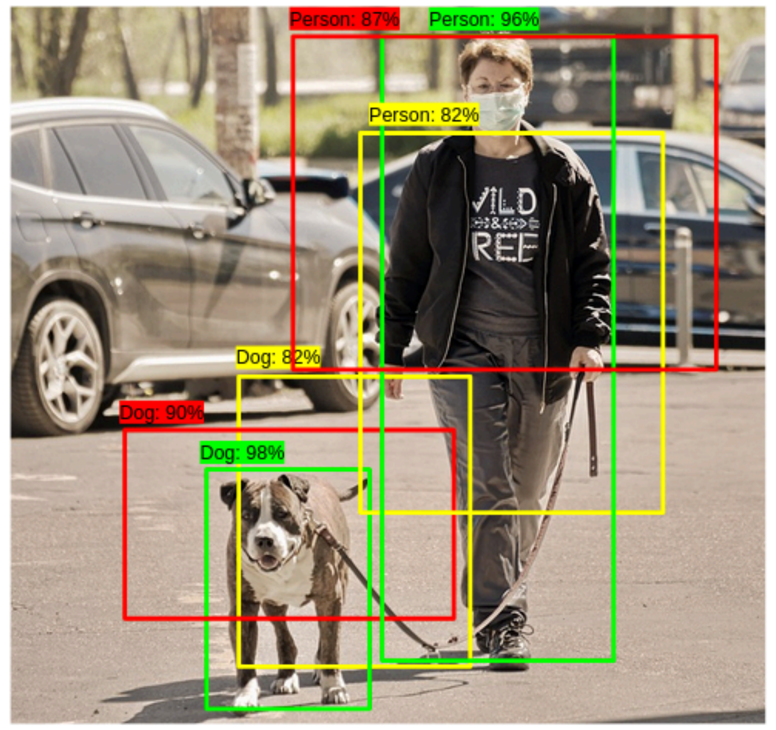


Fig.1

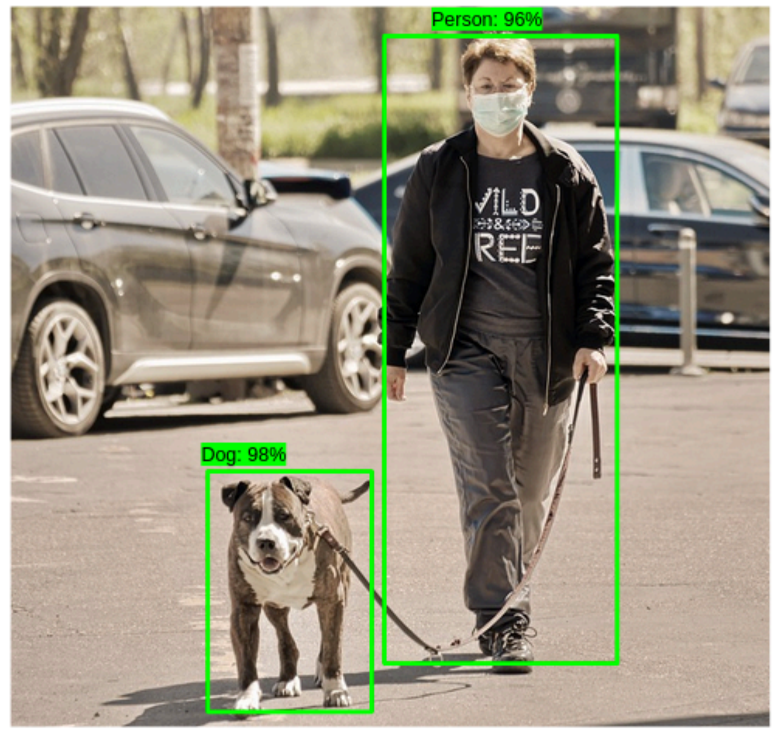


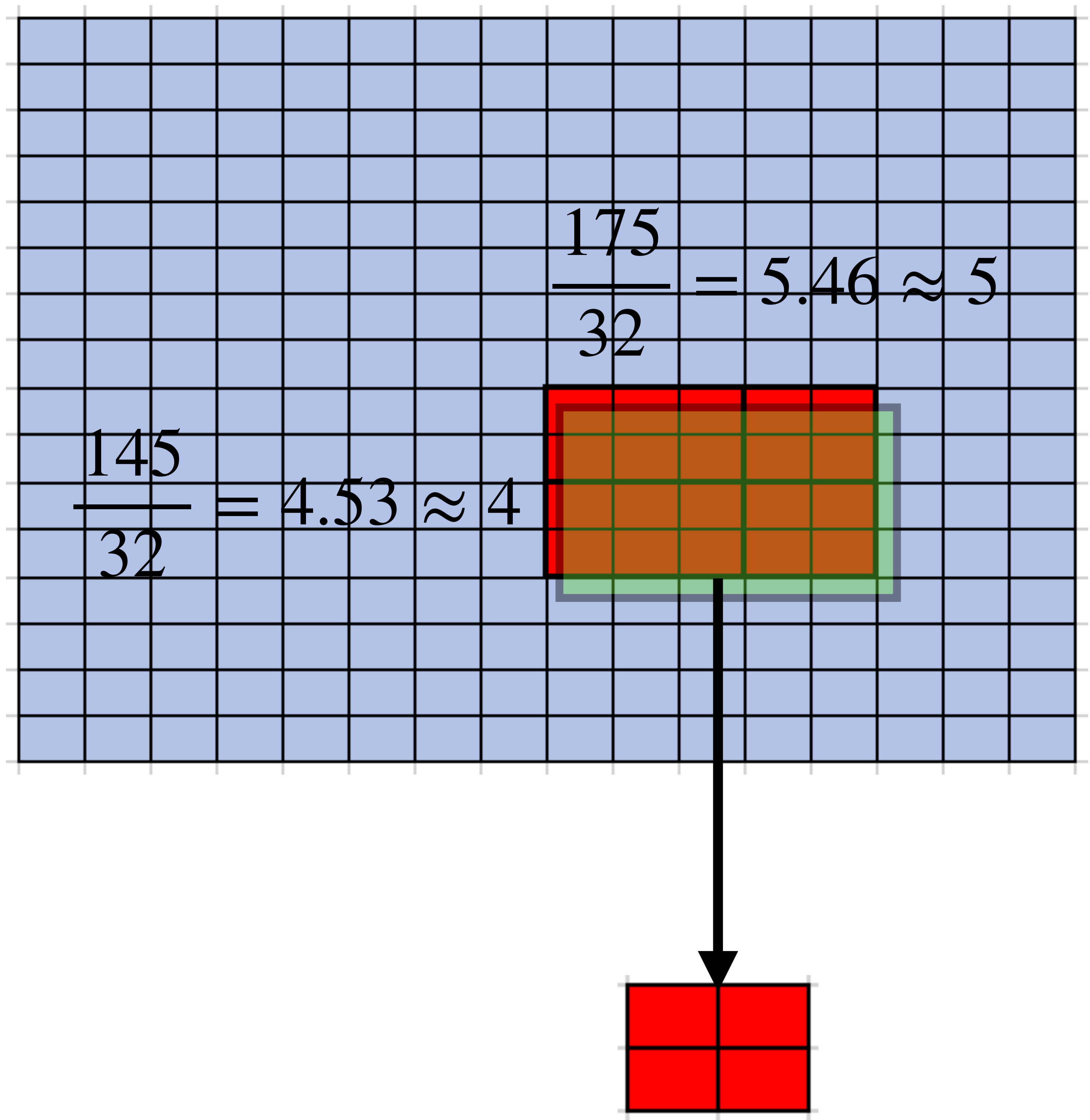
Fig.2

Ref.9: Fig.1 Two objects with multiple detections, Fig.2 Boxes after NSM

Object detection with Deep Neural Networks

Multi Stage detectors (Faster RCNN Stage-2)

- Region of interest pooling:
  - The second stage of the model (Fully connected layers) requires equal size samples
  - Therefore the RoI proposals must be pooled to a fix size
  - The pooling window size is a hyper parameter and pooled output is independent of the input size
  - E.g: take a RoI at an input with w:175, h:145.
    - The feature extractor scale this RoI to 5.46, 4.53
    - By rounding, some information are lost and some are gained
    - The new scaled region is again divided by the pool window size
    - This involves ceiling and flooring the numbers (np.ceil, np.floor)
    - Finally the max values within the divided region are taken
  - More advanced pooling technique such as RoIAlign exists to address the issues of vanilla RoI method (check Ref 8)



Object detection with Deep Neural Networks

Multi Stage detectors (Faster RCNN Stage-2)

- Detection Target Generator:
  - Similar to the target generation in the RPN model;
    - For the labels:
      - Compute the IoU between a sampled RoI proposal and GTs,
      - If IoU $>$ 0.5 assign a positive label (the class label)
      - If 0.1 $<$ IoU $<$ 0.5 assign a negative sample (background)
      - Discard all the other RoIs since they do not contribute to training
    - For the box coordinates:
      - Use equations 1 and 2 to find the box regression GTs
      - $t_x = \frac{x - x_{GT}}{w_{GT}}; t_y = \frac{y - y_{GT}}{h_{GT}}; t_w = log\frac{w}{w_{GT}}; t_h = log\frac{h}{h_{GT}}$

Object detection with Deep Neural Networks

Multi Stage detectors (Faster RCNN Stage-2)

- **Detection Network:**
  - Refine the selected Region of Interest (RoI) coordinates and classify them into the respective classes
  - The network is a fully connected model with one hidden layer and one output layer
  - Inputs are K x $len_{Pool_{flatten}}$ feature matrix from the RoI pooling layer
    - K is the number of RoIs (both positive and negative) after the NMS
  - The classification layer outputs K x N_{cls} (class for each RoI)
  - The regression layer outputs a K x N_{reg} coordinate shifts
- **Detection Loss:**
  - For classifier, the cross entropy loss and regressor Smooth L1 loss

Object detection with Deep Neural Networks

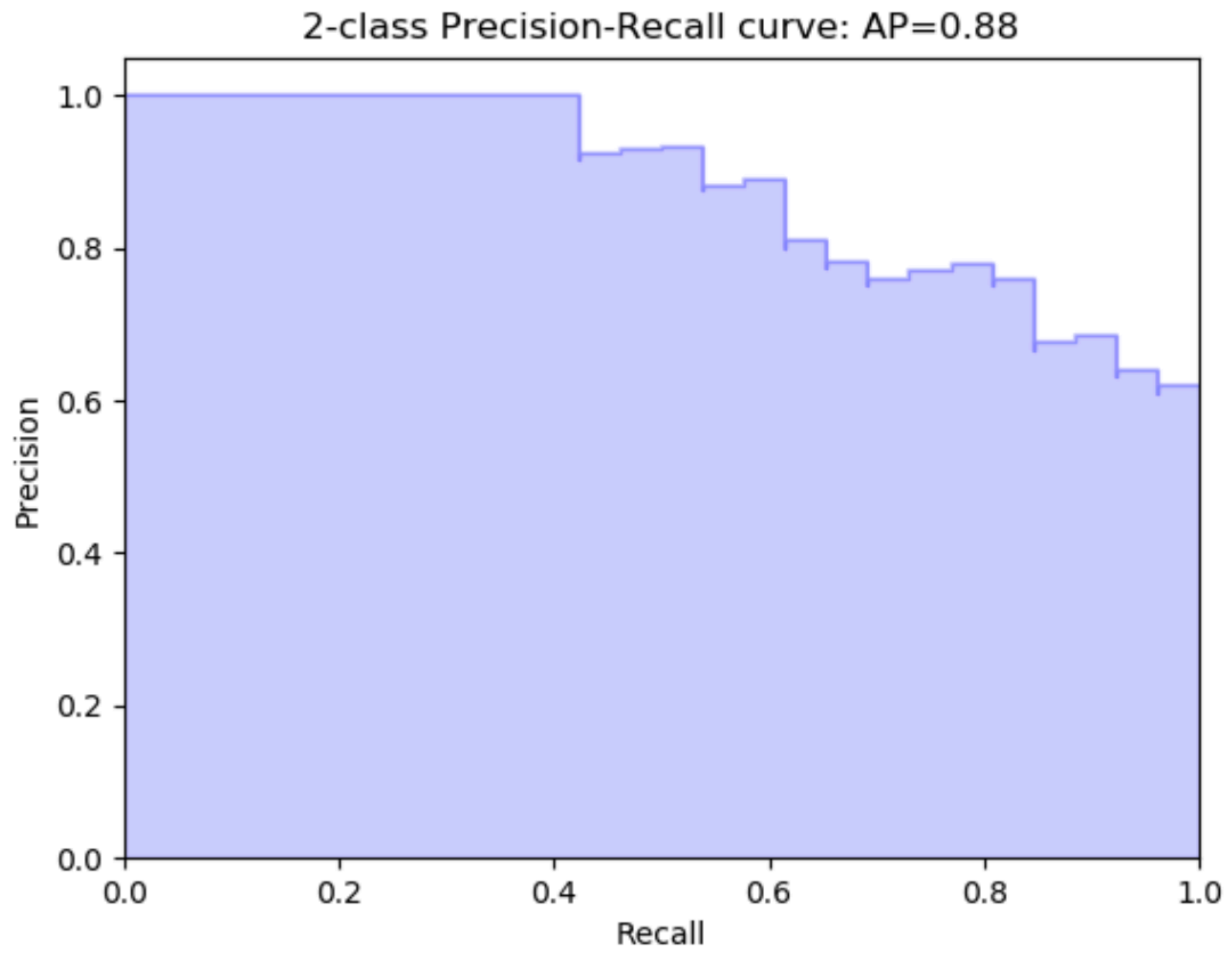
Evaluation metrics for object detection

- Several object detection competitions (e.g Pascal VOC and COCO) defines different evaluation metrics to evaluate the performance of an object detector
- The detection metric for object detection is Average Precision (AP) and mean AP (mAP)
- The AP is the sum of weighted precision at different recalls in a precision recall curve for a given class or IoU Threshold
- mAP is calculated by either averaging over only the classes or classes and/or different IoU thresholds

Object detection with Deep Neural Networks

Evaluation metrics for object detection

- In detection the TP, FP, FN and TN are as follows:
    - For every detection, calculate the IoU between detected box and GT box:
        - If IoU > Threshold & predicted class = GT class for class threshold; TP
        - If IoU < Threshold; FP
        - If a GT not detected count a FN
        - No TN since everything that is not a TP is a TN and there are a large number of TN in a given input, so this metric is not considered
    - Vary the detection confidence threshold (like in classification) to Precision-Recall curve
- So the mAP at a given IoU threshold reflects the performance of the detector (higher the better) in respect to the class classification
- However finding mAP by averaging across the thresholds (eg 0.5:0.05:0.95) indicates the quality of the localisation



Ref.10: Precision recall curve and its AP

- Ref.1: https://towardsdatascience.com/coco-data-format-for-object-detection-a4c5eaf518c5
- Ref.2: https://www.immersivelimit.com/tutorials/create-coco-annotations-from-scratch#:~:text=The%20COCO%20bounding%20box%20format,other%20annotations%20in%20the%20dataset).
- Ref.3: https://whatdhack.medium.com/a-deeper-look-at-how-faster-rcnn-works-84081284e1cd
- Ref.4: https://arxiv.org/pdf/1704.04861.pdf
- Ref.5: https://www.programmersought.com/article/336611383/
- Ref.6: https://www.pyimagesearch.com/2016/11/07/intersection-over-union-iou-for-object-detection/
- Ref.7: https://arxiv.org/pdf/1506.01497.pdf, https://arxiv.org/pdf/1311.2524.pdf
- Ref.8: https://deepsense.io/region-of-interest-pooling-explained/, https://erdem.pl/2020/02/understanding-region-of-interest-ro-i-pooling, https://erdem.pl/2020/02/understanding-region-of-interest-part-2-ro-i-align
- Ref.9: https://www.analyticsvidhya.com/blog/2020/08/selecting-the-right-bounding-box-using-non-max-suppression-with-implementation/, https://www.pyimagesearch.com/2014/11/17/non-maximum-suppression-object-detection-python/, https://www.pyimagesearch.com/2015/02/16/faster-non-maximum-suppression-python/, https://www.coursera.org/lecture/convolutional-neural-networks/non-max-suppression-dvrjH
- Ref.10: http://cs230.stanford.edu/section/7/,
- Rotated bounding box detection: https://developer.nvidia.com/blog/detecting-rotated-objects-using-the-odtk/, https://github.com/NVIDIA/retinanet-examples