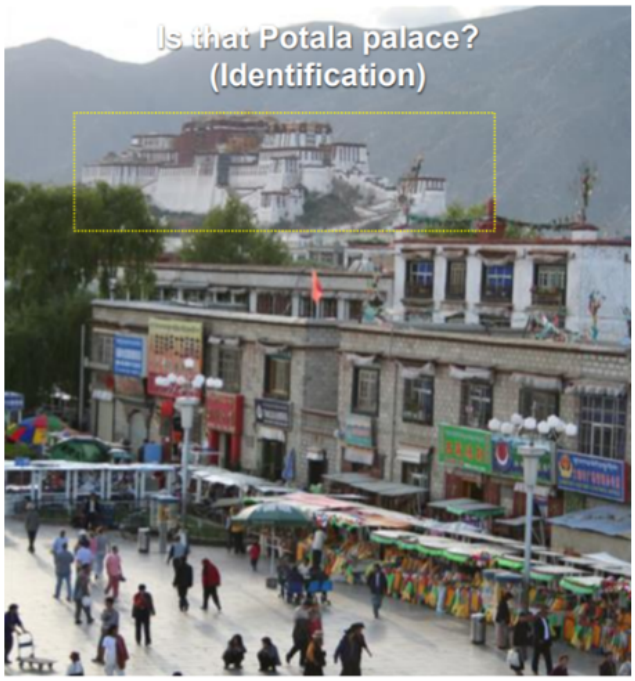
Project 5: Object detection using neural networks

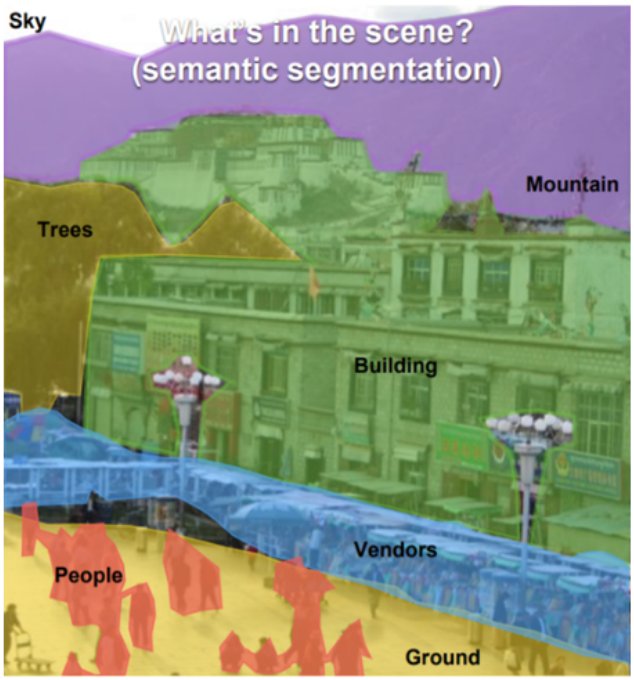
Week 1-2/3

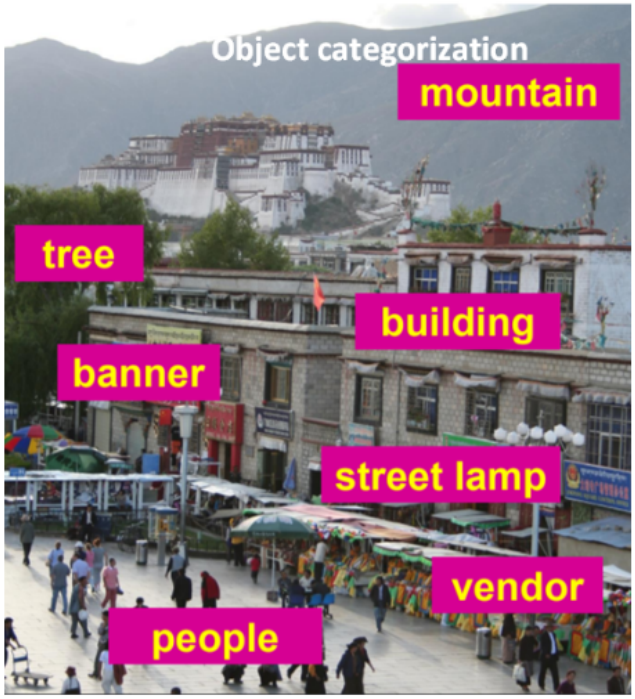
Semantic vision

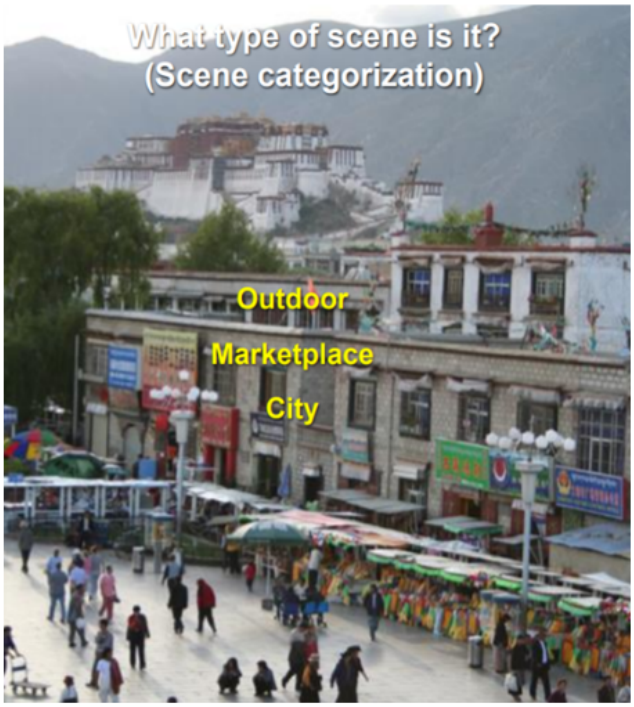
In general, **semantics** concerns the extraction of meaning from data. **Semantic vision** seeks to understand not only what objects are present in an image but, perhaps even more importantly, the relationship between those objects. In semantic vision, an image is typically segmented into regions of interest. Contextual relationships provide important cues for understanding a scene. Spatial context can be formulated in terms of the relationship between an object and its neighbouring objects. For example, a car is likely to appear above a road, but is unlikely to be surrounded by sky. Contextual relationships can also be inferred from the relationship between items in an image and the camera. For instance, if a tree is occluded by a car, then the car is closer to the camera than the tree. The ability to attribute relationships between objects demonstrates reasoning, an important step towards true "cognition".

What do we mean by 'semantic vision'?


Is that Potala palace?
(Identification)


Sky
What's in the scene?
(semantic segmentation)
Mountain
Trees
Building
Vendors
People
Ground

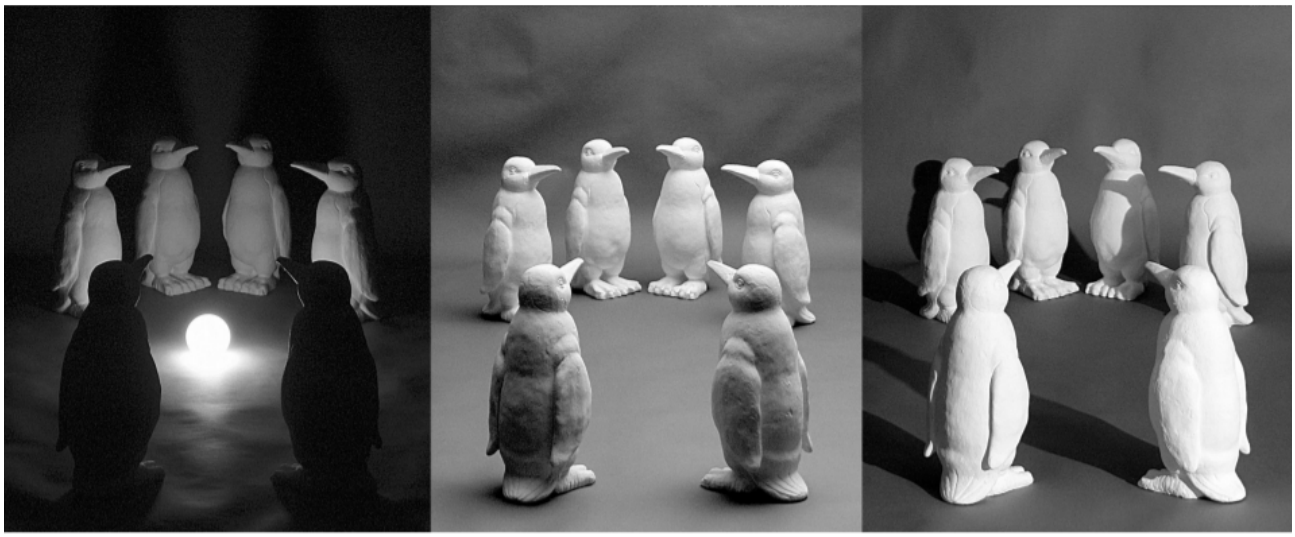

Object categorization
mountain
tree
building
banner
street lamp
vendor
people


What type of scene is it?
(Scene categorization)
Outdoor
Marketplace
City

Applications for **semantic vision** include *autonomous driving*, *medical imaging analysis*, *industrial inspection*, *classification of terrain from satellite imagery*, *data management*, etc.

Challenges

1. Variable viewpoint



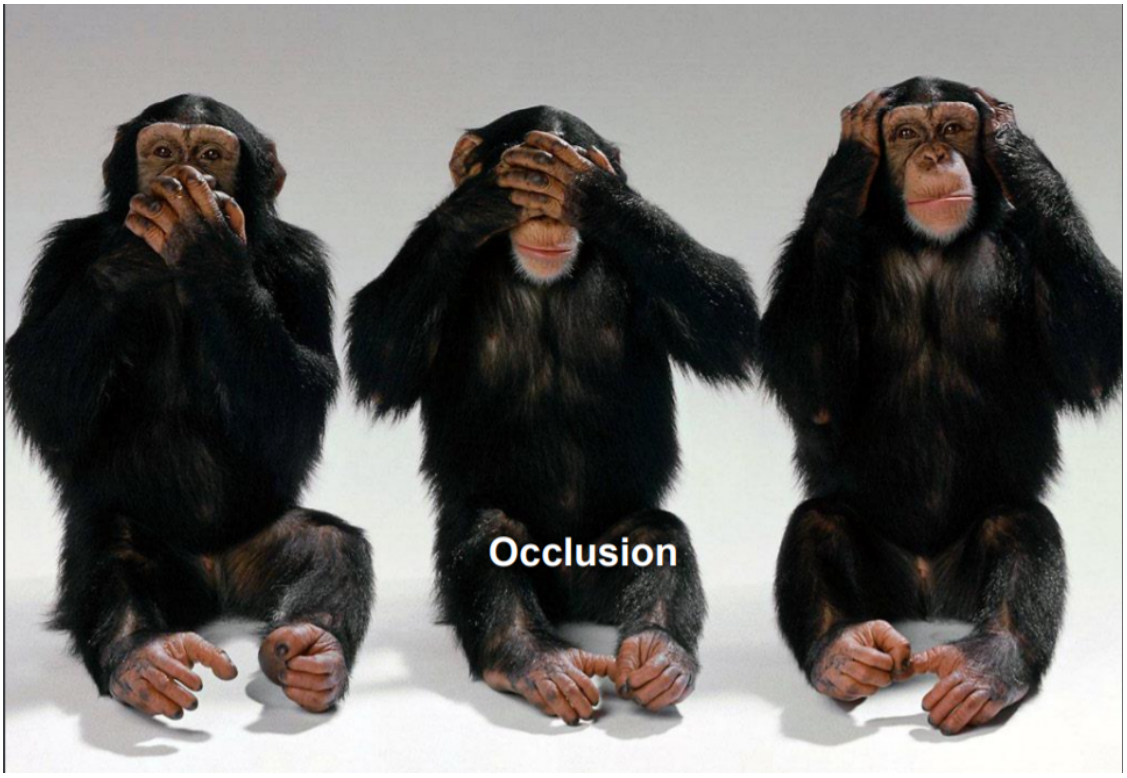
2. Variable illumination (J. Koenderink)

3. Scale

4a. Deformation

4b. Deformation

5a. Occlusion (Magritte,1957**)**



5b. Occlusion

6a. Background clutter (Kilmeny Niland. 1995**)**

6b. Background clutter

7. Intra-class variations

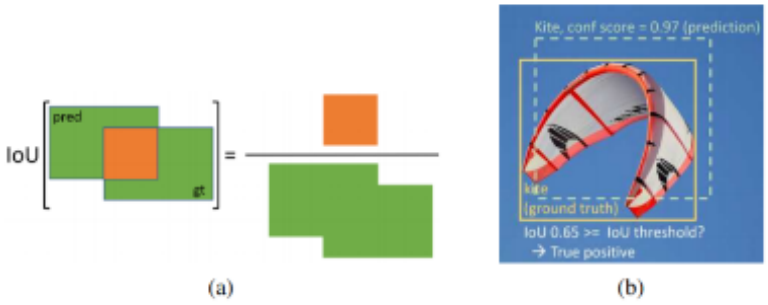
Of all the visual tasks we might ask a computer to perform, analyzing a scene and recognizing all of the constituent objects remains the most challenging. While computers excel at accurately reconstructing the 3D shape of a scene from images taken from different views, they cannot name all the objects and animals present in a picture, even at the level of a two year-old child. There is not even any consensus among researchers on when this level of performance might be achieved.

If we are given an image to analyze, we could try to apply a recognition algorithm to every possible sub-window in this image. Such algorithms are likely to be both slow and

error-prone. Instead, it is more effective to construct special purpose *detectors*, whose job it is to rapidly find likely regions where particular objects might occur.

While face and pedestrian detection algorithms were the earliest to be extensively studied, computer vision has always been interested in solving the **general object detection** and **labeling problem**, in addition to whole-image classification.

Before we describe the elements of modern object detectors, we should first discuss what metrics they are trying to optimize. The main task in object detection, is to put accurate bounding boxes around all the objects of interest and to correctly label such objects. To measure the accuracy of each bounding box (not to small and not too big), the common metric is intersection over union (IOU), which is also known as the Jaccard index or Jaccard similarity coefficient. The IOU is computed by taking the predicted and ground truth bounding boxes and for an object and computing the ratio of their area of intersection and their area of union.



Intersection over union (IOU): (a) schematic formula, (b) real-world example © 2020 Ross Girshick

As we will shortly see, object detectors operate by first proposing a number of plausible rectangular regions (detections) and then classifying each detection while also producing a confidence score. These regions are then run through some kind of non-maximal suppression (NMS) stage, which removes weaker detections that have too much overlap with stronger detections, using a greedy most-confident-first algorithm. To evaluate the performance of an object detector, we run through all of the detections, from most confident to least, and classify them a true positive **TP** (correct label and sufficiently high IOU) or false positive **FP** (incorrect label or ground truth object already matched).

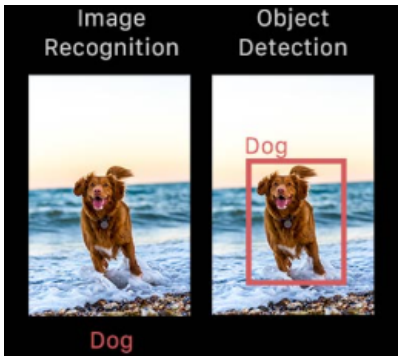
Remember: Evaluation Metrics from Project 1

What is object detection?

Object detection is a computer vision technique that works to identify and locate objects within an image or video. Specifically, object detection draws bounding boxes around these detected objects, which allow us to locate where said objects are in (or how they move through) a given scene. Object detection is commonly confused with image recognition, so before we proceed, it's important that we clarify the distinctions between them. Image recognition assigns a label to an image. A picture of a dog receives the label "dog". A picture of two dogs still receives the label "dog". Object detection, on the other

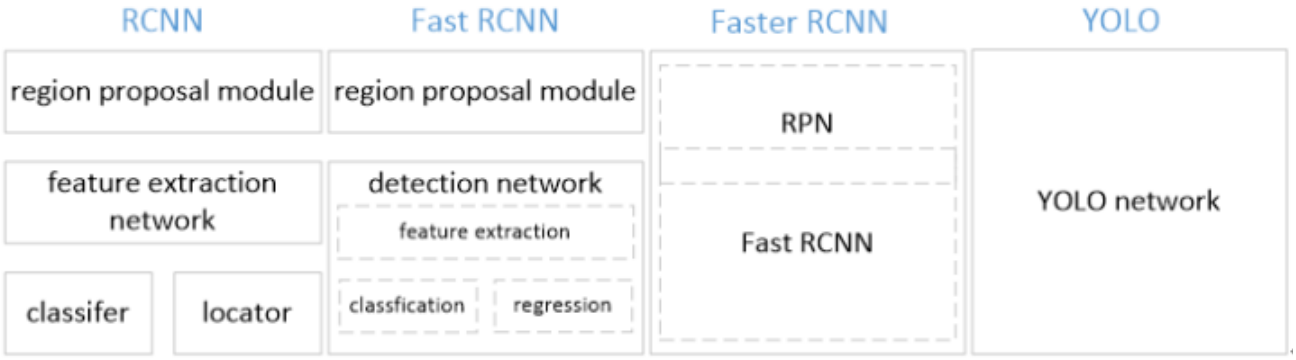
hand, draws a box around each dog and labels the box “dog”. The model predicts where each object is and what label should be applied. In that way, object detection provides more information about an image than recognition.

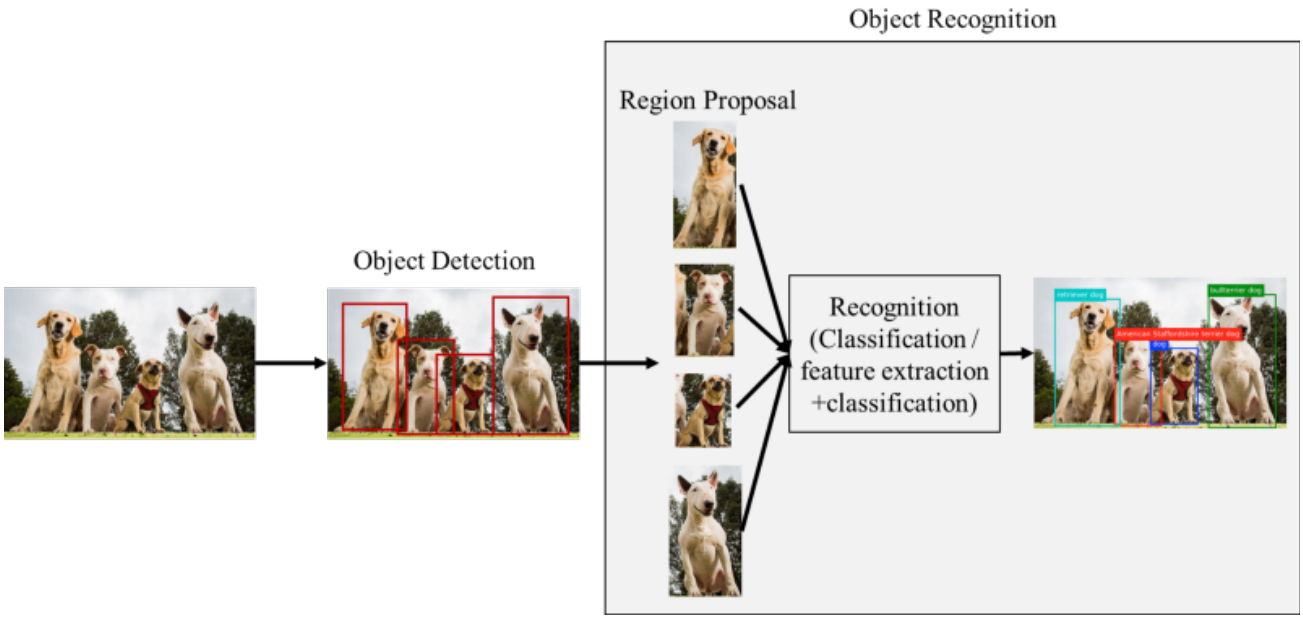
Here’s an example of how this distinction looks in practice:



Object detection can be broken down into *machine learning*-based approaches and *deep learning*-based approaches. In more traditional ML-based approaches, computer vision techniques are used to look at various features of an image, such as the color histogram or edges, to identify groups of pixels that may belong to an object. These features are then fed into a regression model that predicts the location of the object along with its label. On the other hand, deep learning-based approaches employ convolutional neural networks (CNNs) to perform end-to-end, unsupervised object detection, in which features don’t need to be defined and extracted separately.

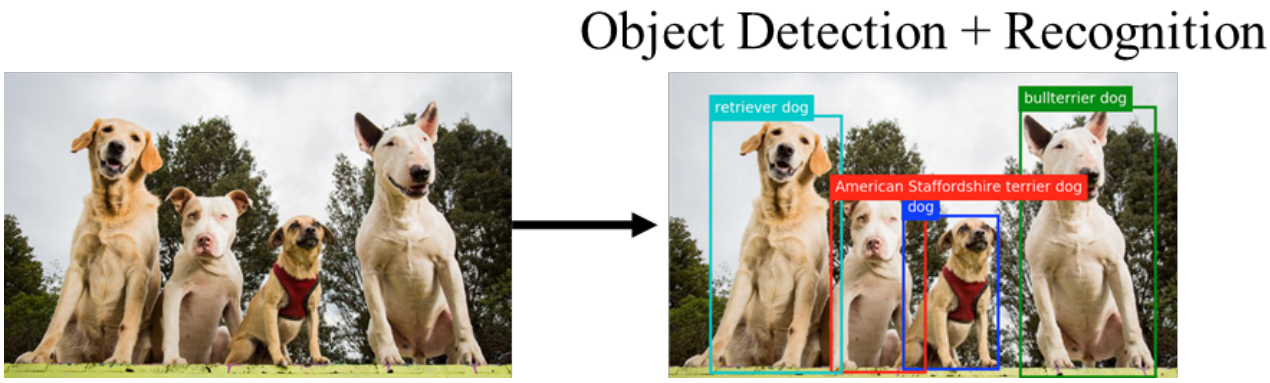
Two-stage vs One-stage Detectors





Two-stage detectors

First, the model proposes a set of regions of interest by select search or regional proposal network. The proposed regions are sparse as the potential bounding box candidates can be infinite. Then a classifier only processes the region candidates.

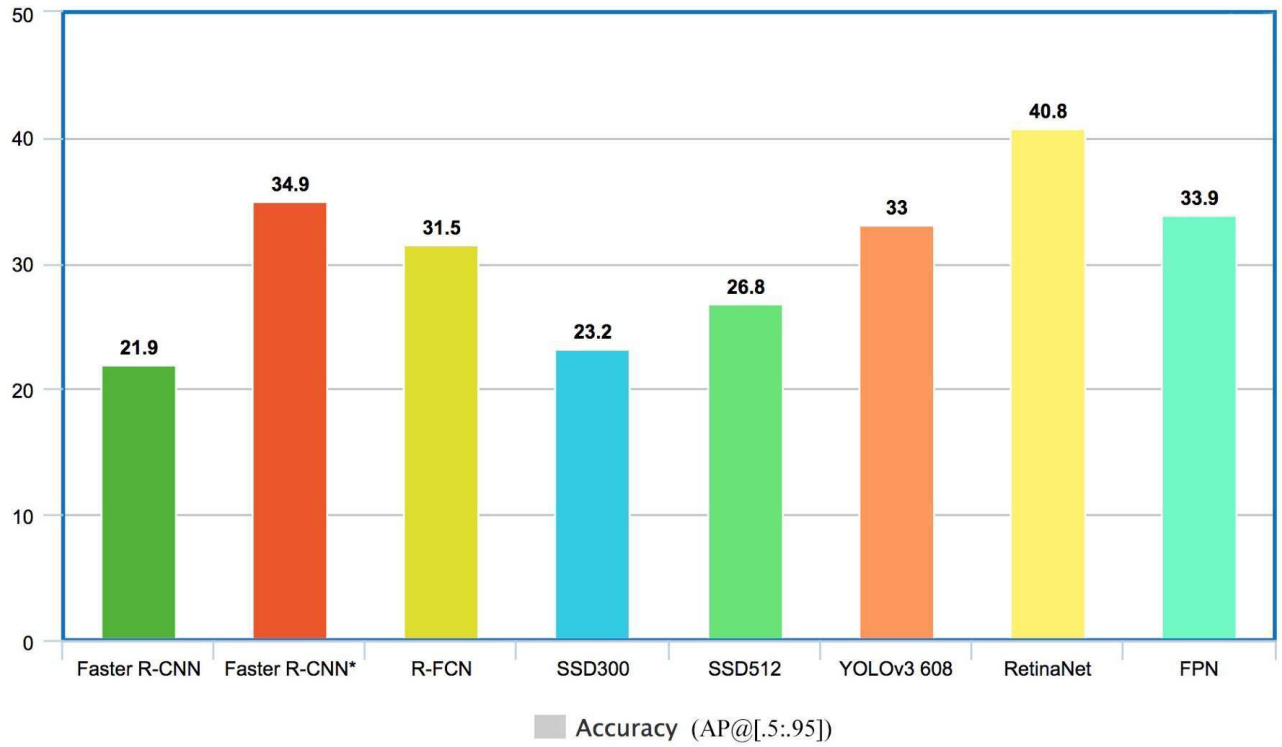


One-stage detectors

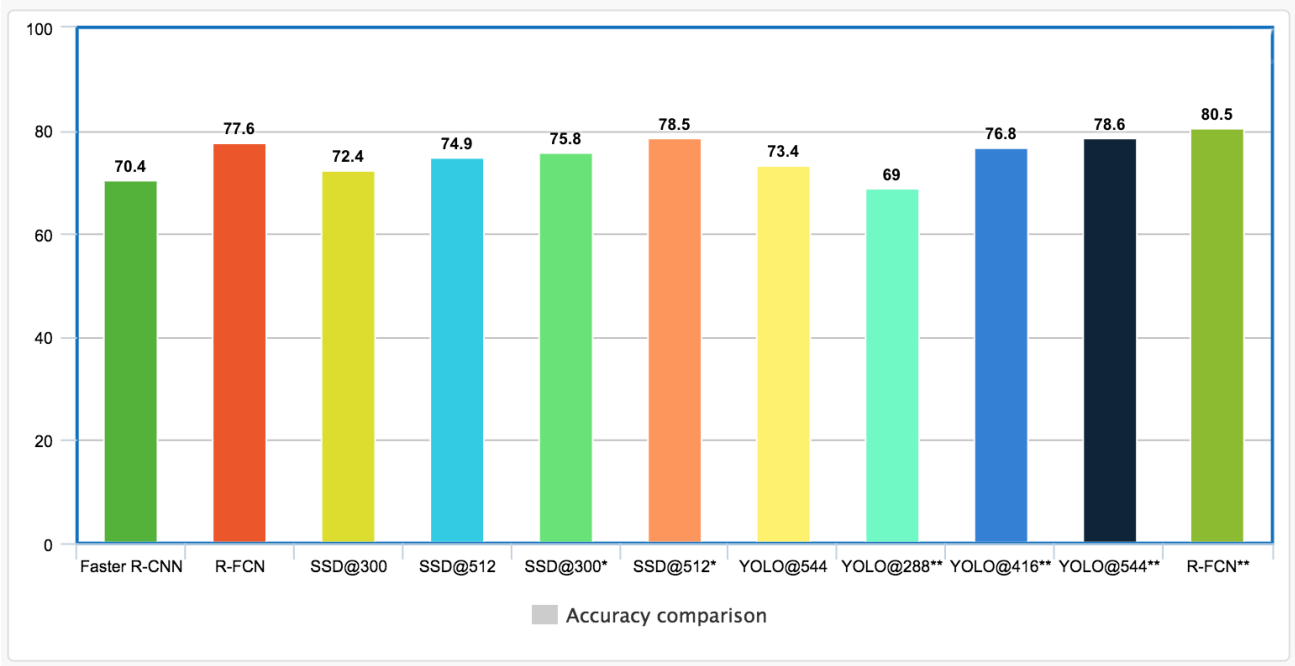
Single convolutional network predicts the bounding boxes and the class probabilities for these boxes.

Results on MS-COCO and Pascal VOC dataset

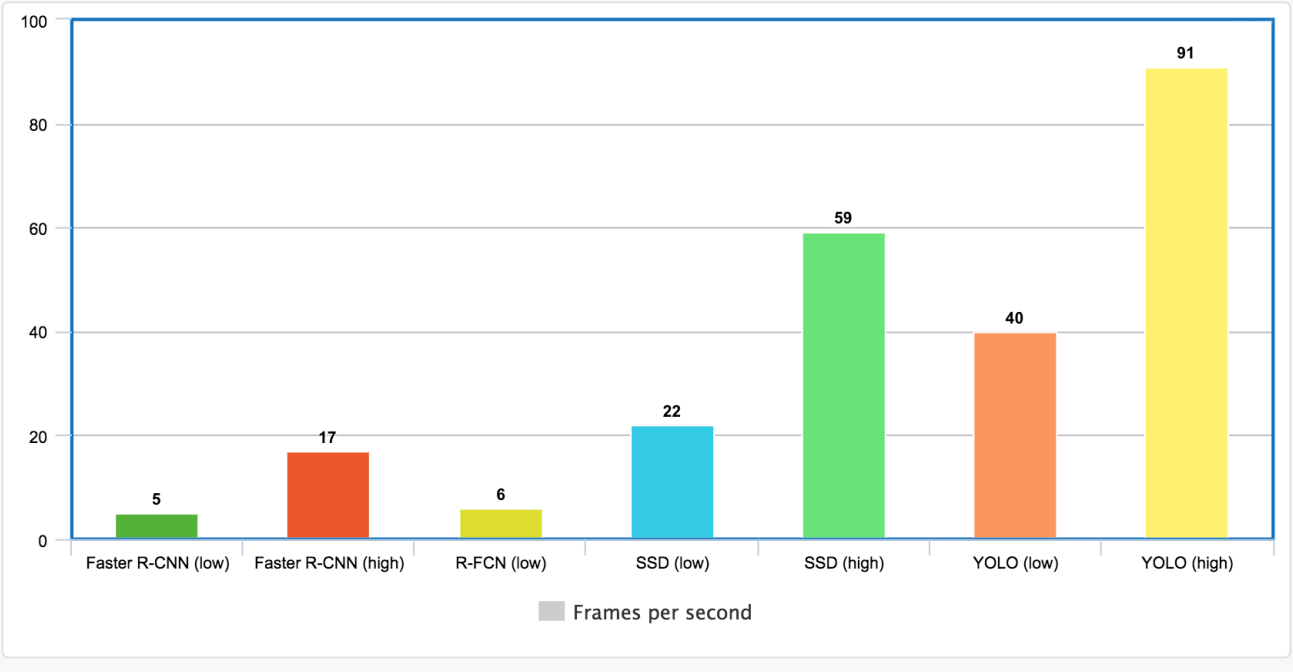
For the last couple years, many results are exclusively measured with the COCO object detection dataset. COCO dataset is harder for object detection and usually detectors achieve much lower mAP. Here is the comparison for some key detectors.



For the result presented below, the model is trained with both PASCAL VOC 2007 and 2012 data. The mAP is measured with the PASCAL VOC 2012 testing set. For SSD, the chart shows results for 300 × 300 and 512 × 512 input images. For YOLO, it has results for 288 × 288, 416 ×461 and 544 × 544 images. Higher resolution images for the same model have better mAP but slower to process.



Input image resolutions and feature extractors impact speed. Below is the highest and lowest FPS reported by the corresponding papers. Yet, the result below can be highly biased in particular they are measured at different mAP.



How does the YOLO family work?

YOLO uses a totally different approach. YOLO is a clever convolutional neural network (CNN) for doing object detection in real-time. The algorithm applies a single neural network to the full image, and then divides the image into regions and predicts bounding boxes and probabilities for each region. These bounding boxes are weighted by the predicted probabilities.

YOLO is popular because it achieves high accuracy while also being able to run in real-time. The algorithm “only looks once” at the image in the sense that it requires only one forward propagation pass through the neural network to make predictions. After non-max suppression (which makes sure the object detection algorithm only detects each object once), it then outputs recognized objects together with the bounding boxes.

With YOLO, a single CNN simultaneously predicts multiple bounding boxes and class probabilities for those boxes. YOLO trains on full images and directly optimizes detection performance. This model has a number of benefits over other object detection methods:

- is extremely fast
- sees the entire image during training and test time so it implicitly encodes contextual information about classes as well as their appearance.
- learns generalizable representations of objects so that when trained on natural images and tested on artwork, the algorithm outperforms other top detection methods.

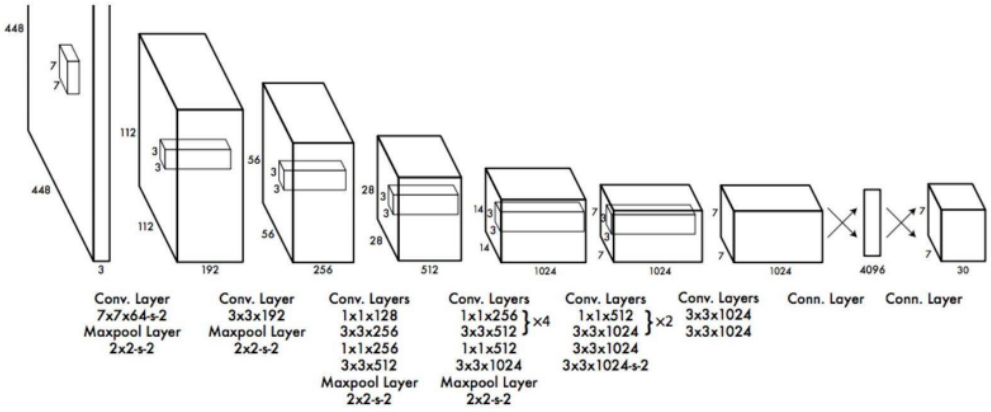


Figure 3: The Architecture. Our detection network has 24 convolutional layers followed by 2 fully connected layers. Alternating 1×1 convolutional layers reduce the features space from preceding layers. We pretrain the convolutional layers on the ImageNet classification task at half the resolution (224×224 input image) and then double the resolution for detection.

Dataset assignation: ~100 images/student * (2/3 students/dataset) => 200/300 annotated images
https://docs.google.com/spreadsheets/d/1qwWiXjat1LCvhPUtRgb2P8Dj7ElxBNQWwHKKmXZ1BBI/edit?usp=sharing

Datasets available at:
https://drive.google.com/drive/folders/1omvW9dfDZ98jEG72c-PBp-B0eiQHk37q?usp=sharing

Results to be uploaded at:
https://drive.google.com/drive/folders/1NiGGDlTnka-LS0OT25WL3RY7do3rXEl7?usp=sharing

Project 5 TODOs (https://colab.research.google.com/github/roboflow-ai/yolov5-custom-training-tutorial/blob/main/yolov5-custom-training.ipynb#scrollTo=kTvDNSILZoN9):

1. Prepare your dataset (each student has a different dataset)
    a. See annotation procedure at https://docs.ultralytics.com/yolov5/tutorials/train_custom_data/#1-create-dataset
    b. **In each image from train & test subsets annotate all objects of the most relevant 4 object classes form the dataset**

        Examples:

Livingroom: couch, TV, dining table, chair, etc.
Dining room: table, chair, wine glass, plate, etc
House: door, window, lawn, wall, etc.

2. Train your model (in colab)
3. Run the model on test subset

Project 5 outcomes:

1. Upload annotations (for both \train & \test) to **P5_outcomes\Nume_Prenume_grupa\my-dataset_annotated\test & …\train** folder on Drive (**classes.txt & .txt files for all images**)
2. Run detector on images from the test subset and upload results to **P5_outcomes\Nume_Prenume_grupa\results_on_test_subset** folder on Drive
3. After the training is done add **train-runs/my-dataset/exp** folder to **P5_outcomes\Nume_Prenume_grupa\training_output** on Drive
4. Inspect and compare **train-runs\my-dataset\exp\test_batch_pred.jpg & test_batch_labels.jpg** and check evaluation metrics from **train-runs\my-dataset\exp\results.png**

Further study at:

1. Chapter 14 - Recognition, 1st Edition, Richard Szeliski
2. Chapter 6 - Recognition, 2nd Edition, Richard Szeliski
3. DeepStack v1.2.1
4. Joseph Redmon, Santosh Divvala, Ross Girshick, Ali Farhadi. ***You Only Look Once: Unified, Real-Time Object Detection***. CVPR 2016
5. Joseph Redmon and Ali Farhadi. ***YOLO9000: Better, Faster, Stronger***. CVPR 2017
6. Joseph Redmon and Ali Farhadi. ***YOLOv3: An Incremental Improvement***. arXiv:1904.07850 2018