# Concept Review

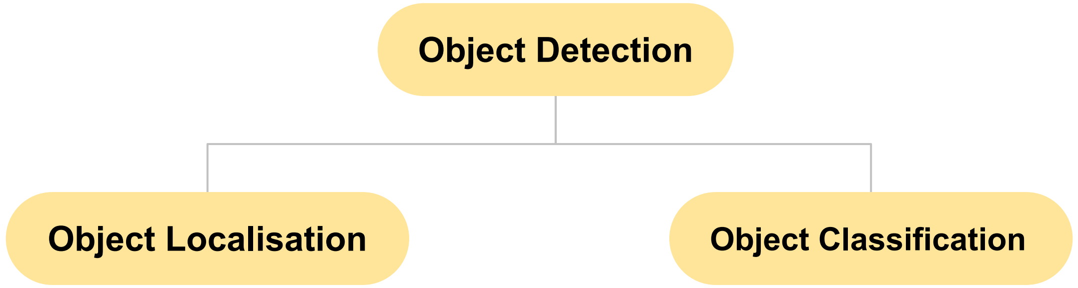
## Object Detection Systems using CNNs

Ananya K R

12th December 2020

## Problem Statement

The problem definition of **object detection** is to determine where objects are located in a given image (**object localization**) and which category each object belongs to (**object classification**).

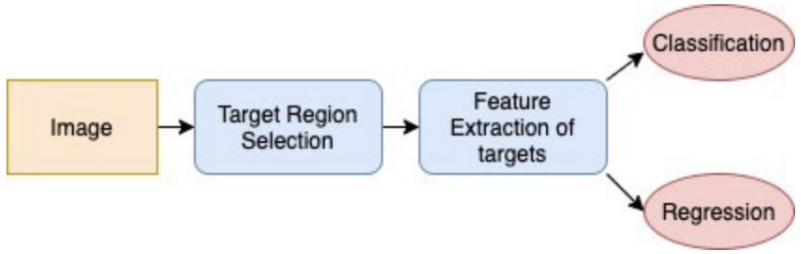


# Traditional Methods

**Informative region selection.** As different objects may appear in any positions of the image and have different aspect ratios or sizes, it is a natural choice to scan the whole image with a multi-scale sliding window. **Disadvantages** - computationally expensive, numerous redundant windows. https://miro.medium.com/max/1200/1*IFSe3gAZncOxGVKI7h5b-Q.gif

**Feature extraction.** To recognize different objects, we need to extract visual features which can provide a semantic and robust representation. **Disadvantages** - Due to the diversity of appearances, illumination conditions and backgrounds, it’s difficult to manually design a robust feature descriptor to perfectly describe all kinds of objects.

**Classification.** A classifier is needed to classify the existing crops using the obtained feature descriptor values and assign a crop to a given class. Simultaneously, drawing a bounding box around an object in a given image. **Disadvantages** - These models need far more information about a class and so, tedious tweaking is needed to get good results.



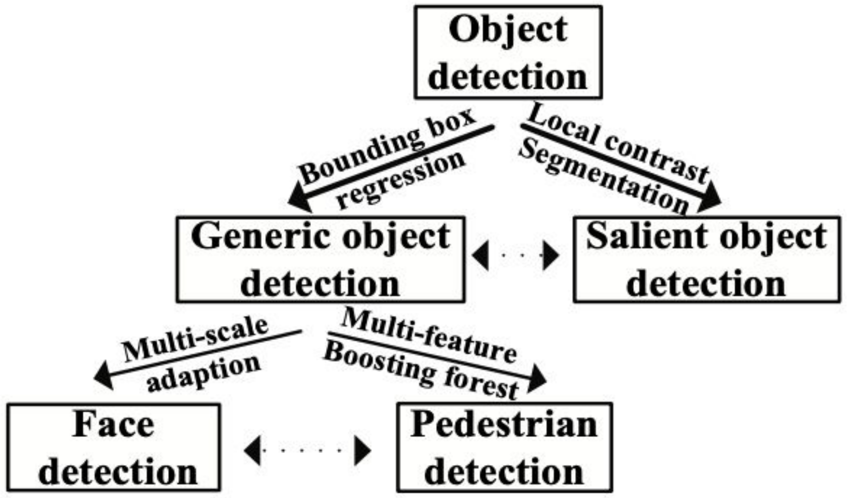
**Traditional Object Detection**

1. Informative Region Selection
2. Feature Extraction (SIFT, HOG, Haar-like)
3. Classification (SVM, AdaBoost, DPM)

$\longrightarrow$

DEEP LEARNING using CNNs

# Deep Learning using CNNs



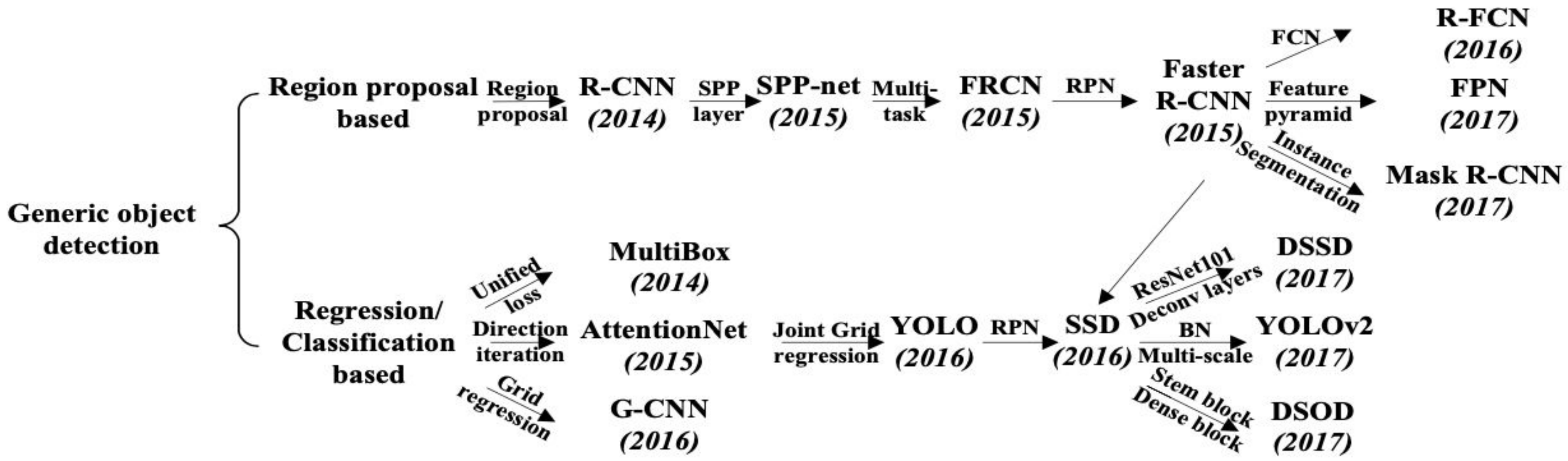
**Practical Implementation**

1. Bounding Box Regression using CALTECH-101 dataset (Keras) https://www.pyimagesearch.com/2020/10/05/object-detection-bounding-box-regression-with-keras-tensorflow-and-deep-learning/

2. Saliency Detection using OpenCV https://www.pyimagesearch.com/2018/07/16/opencv-saliency-detection/

3. Face Recognition using OpenCV https://www.pyimagesearch.com/2018/09/24/opencv-face-recognition/

4. Pedestrian Detection using SVMs https://www.pyimagesearch.com/2015/11/09/pedestrian-detection-opencv/



Areas of Focus : R-CNNs, SSD, YOLO

# R-CNNs (Region-Based CNNs)

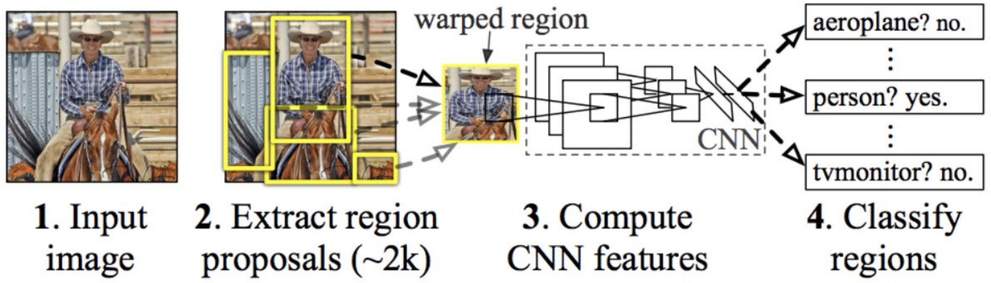
One of the first large and successful application of convolutional neural networks to the problem of object localization, detection, and segmentation. The approach was demonstrated on benchmark datasets, achieving then state-of-the-art results on the **VOC-2012 dataset and the 200-class ILSVRC-2013 object detection dataset**.

Their proposed R-CNN model is comprised of three modules; they are:

**Module 1: Region Proposal**. Generate and extract category independent region proposals, e.g. candidate bounding boxes.

**Module 2: Feature Extractor**. Extract feature from each candidate region, e.g. using a deep convolutional neural network.

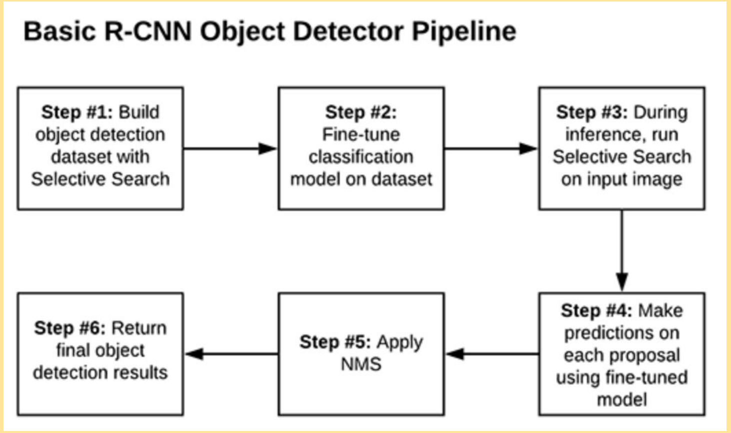
**Module 3: Classifier**. Classify features as one of the known class, e.g. linear SVM classifier model.



## Practical Implementation

Using the Raccoon object detection dataset curated by **Dat Tran**, **R-CNN object detection with Keras, TensorFlow, and Deep Learning**

https://www.pyimagesearch.com/2020/07/13/r-cnn-object-detection-with-keras-tensorflow-and-deep-learning/

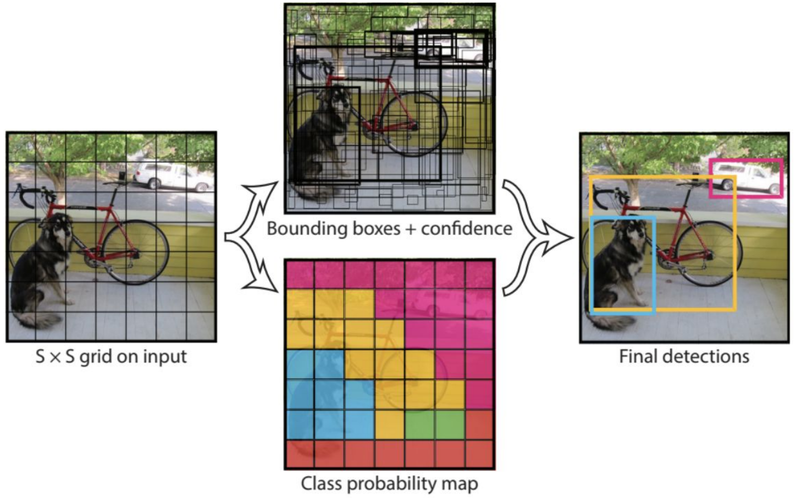

Basic R-CNN Object Detector Pipeline
Step #1: Build object detection dataset with Selective Search
Step #2: Fine-tune classification model on dataset
Step #3: During inference, run Selective Search on input image
Step #4: Make predictions on each proposal using fine-tuned model
Step #5: Apply NMS
Step #6: Return final object detection results

# YOLO (You only look once)

The R-CNN models may be generally more accurate, yet the YOLO family of models are fast, much faster than R-CNN, achieving object detection in real-time.

The approach involves a single neural network trained end to end that takes a photograph as input and predicts bounding boxes and class labels for each bounding box directly. The technique offers lower predictive accuracy (e.g. more localization errors), although operates at 45 frames per second and up to 155 frames per second for a speed-optimized version of the model.

The model works by first splitting the input image into a grid of cells, where each cell is responsible for predicting a bounding box if the center of a bounding box falls within it. A class prediction is also based on each cell.



## Practical Implementation

Using the Microsoft COCO dataset, **Object detection using YOLOv3**

https://www.pyimagesearch.com/2018/11/12/yolo-object-detection-with-opencv/

## SSD (Single Shot MultiBox Detector)

SSD reached new records in terms of performance and precision for object detection tasks, scoring over 74% mAP (*mean Average Precision*) at 59 frames per second on standard datasets such as PascalVOC and COCO.

**Single Shot:** the tasks of object localization and classification are done in a *single forward pass* of the network

**MultiBox:** this is the name of a technique for bounding box regression developed by Szegedy et al.

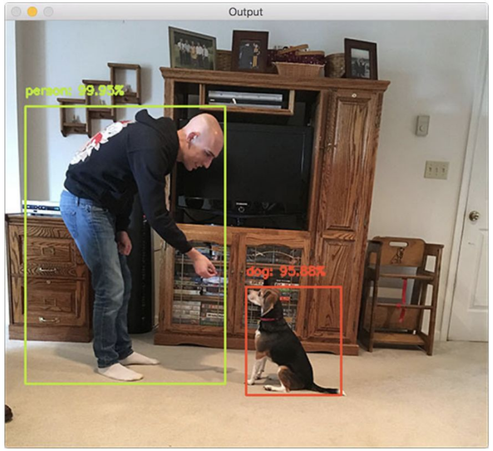
**Detector:** The network is an object detector that also classifies those detected objects

The MobileNet SSD was first trained on the **COCO dataset** (Common Objects in Context) and was then fine-tuned on PASCAL VOC reaching 72.7% mAP (mean average precision).

**Practical Implementation**

Using the Microsoft COCO dataset, **Object detection using SSD and MultiNets**

https://www.pyimagesearch.com/2017/09/11/object-detection-with-deep-learning-and-opencv/



## R-CNNs vs. YOLO vs. SSD for Object Detection

Faster R-CNNs are likely the most “heard of” method for object detection using deep learning; however, the technique can be difficult to understand (especially for beginners in deep learning), hard to implement, and challenging to train. Furthermore, even with the “faster” implementation R-CNNs (where the “R” stands for “Region Proposal”) the algorithm can be quite slow, on the order of 7 FPS.

If we are looking for pure speed then we tend to use YOLO as this algorithm is much faster, capable of processing 40-90 FPS on a Titan X GPU. The super fast variant of YOLO can even get up to 155 FPS. The problem with YOLO is that it leaves much accuracy to be desired.

SSDs, originally developed by Google, are a balance between the two. The algorithm is more straightforward than Faster R-CNNs. We can also enjoy a much faster FPS throughput than **Girshick et al**. at 22-46 FPS depending on which variant of the network we use. SSDs also tend to be more accurate than YOLO. To learn more about SSDs, please refer to **Liu et al**.

# Thank you