# Technology selection

It is very hard to have a fair comparison among different object detectors. There is no straight answer on which model is the best. For real-life applications, we make choices to balance accuracy and speed.

In this project, two models are tried: YOLO and Mask R-CNN.

## 1. YOLO: Real-Time Object Detection

YOLOv3 is the latest variant of a popular object detection algorithm YOLO – You Only Look Once. The published model recognizes 80 different objects in images and videos, but most importantly it is super fast and nearly as accurate as Single Shot MultiBox (SSD).

### 1.1 How does YOLO work?

We can think of an object detector as a combination of a object locator and an object recognizer.

In traditional computer vision approaches, a sliding window was used to look for objects at different locations and scales. Because this was such an expensive operation, the aspect ratio of the object was usually assumed to be fixed.

Early Deep Learning based object detection algorithms like the R-CNN and Fast R-CNN used a method called Selective Search to narrow down the number of bounding boxes that the algorithm had to test.

Another approach called Overfeat involved scanning the image at multiple scales using sliding windows-like mechanisms done convolutionally.

This was followed by Faster R-CNN that used a Region Proposal Network (RPN) for identifying bounding boxes that needed to be tested. By clever design the features extracted for recognizing objects, were also used by the RPN for proposing potential bounding boxes thus saving a lot of computation.

YOLO on the other hand approaches the object detection problem in a completely different way. It forwards the whole image only once through the network. SSD is another object detection algorithm that forwards the image once though a deep learning network, but YOLOv3 is much faster than SSD while achieving very comparable accuracy. YOLOv3 gives faster than real-time results on a M40, TitanX or 1080 Ti GPUs.

### 1.2 Detection process

First, it divides the image into a 13×13 grid of cells. The size of these 169 cells vary depending on the size of the input. For a 416×416 input size that we used in our experiments, the cell size was 32×32. Each cell is then responsible for predicting a number of boxes in the image.

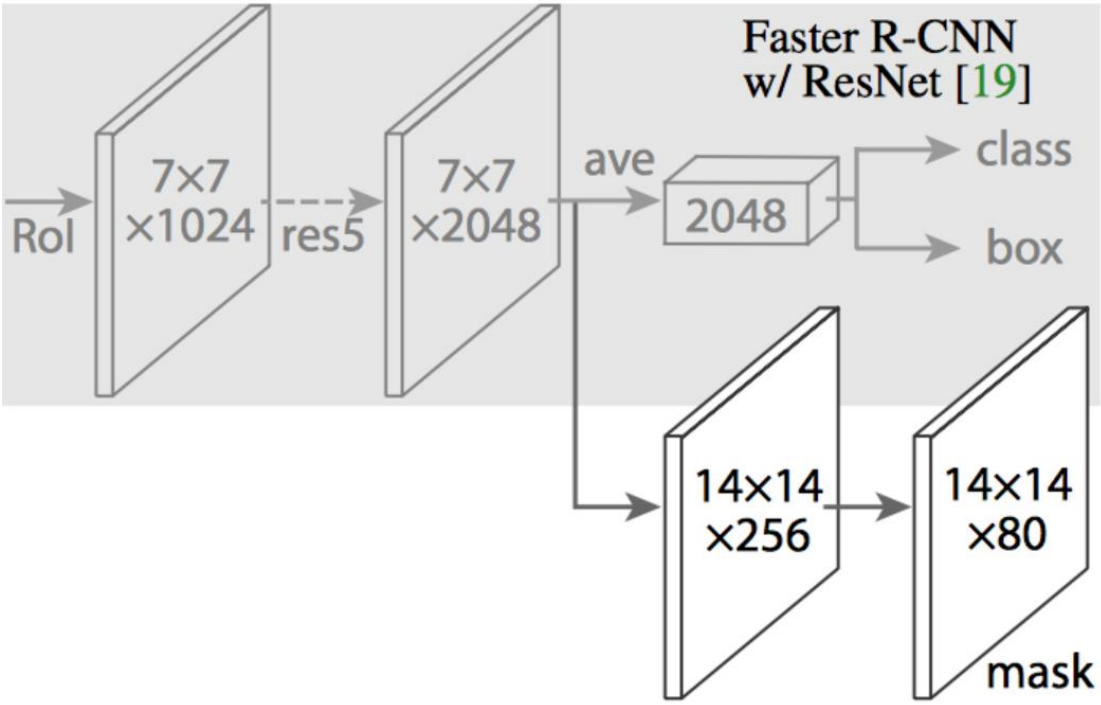
For each bounding box, the network also predicts the confidence that the bounding box actually encloses an object, and the probability of the enclosed object being a particular class.

Most of these bounding boxes are eliminated because their confidence is low or because they are enclosing the same object as another bounding box with very high

confidence score. This technique is called non-maximum suppression.

The authors of YOLOv3, Joseph Redmon and Ali Farhadi, have made YOLOv3 faster and more accurate than their previous work YOLOv2. YOLOv3 handles multiple scales better. They have also improved the network by making it bigger and taking it towards residual networks by adding shortcut connections.

## 2. Mask R-CNN




### 2.1 Evolve from fast RCNN

Much like Fast R-CNN, and Faster R-CNN, Mask R-CNN's underlying intuition is straight forward. Given that Faster R-CNN works so well for object detection, Mask R-CNN extends it to also carry out pixel level segmentation.

Mask R-CNN does this by adding a branch to Faster R-CNN that outputs a binary mask that says whether or not a given pixel is part of an object. The branch (in white in the above image), as before, is just a Fully Convolutional Network on top of a CNN based feature map. Here are its inputs and outputs:

Inputs: CNN Feature Map.
Outputs: Matrix with 1s on all locations where the pixel belongs to the object and 0s elsewhere (this is known as a binary mask).
The Mask R-CNN authors had to make one small adjustment to make this pipeline work as expected: RoiAlign.

### 2.2 RoiAlign - Realigning RoIPool to be More Accurate

When run without modifications on the original Faster R-CNN architecture, the Mask R-CNN

authors realized that the regions of the feature map selected by RoIPool were slightly misaligned from the regions of the original image. Since image segmentation requires pixel level specificity, unlike bounding boxes, this naturally led to inaccuracies.

The authors were able to solve this problem by cleverly adjusting RoIPool to be more precisely aligned using a method known as RoIAlign.

Imagine we have an image of size 128x128 and a feature map of size 25x25. Let's imagine we want features the region corresponding to the top-left 15x15 pixels in the original image (see above). How might we select these pixels from the feature map?

We know each pixel in the original image corresponds to ~ 25/128 pixels in the feature map. To select 15 pixels from the original image, we just select 15 * 25/128 ~= 2.93 pixels.

In RoIPool, we would round this down and select 2 pixels causing a slight misalignment. However, in RoIAlign, we avoid such rounding. Instead, we use bilinear interpolation to get a precise idea of what would be at pixel 2.93. This, at a high level, is what allows us to avoid the misalignments caused by RoIPool.

Once these masks are generated, Mask R-CNN combines them with the classifications and bounding boxes from Faster R-CNN to generate such wonderfully precise segmentations.

**Reference**

[1] https://arxiv.org/pdf/1506.02640.pdf
[2] https://arxiv.org/pdf/1703.06870.pdf