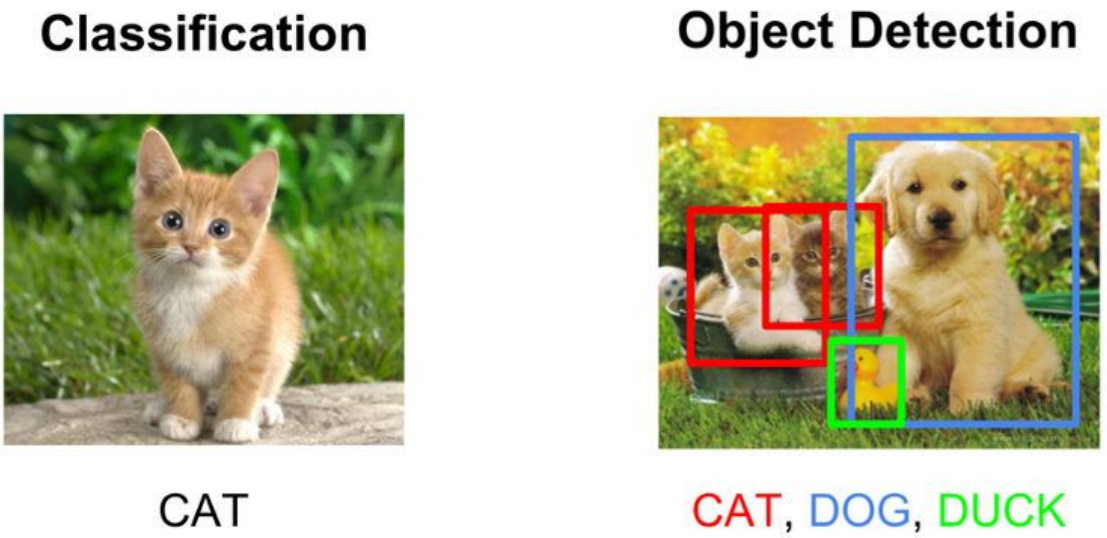
2D Object Detection

Table of contents:

1. Introduction to object detection

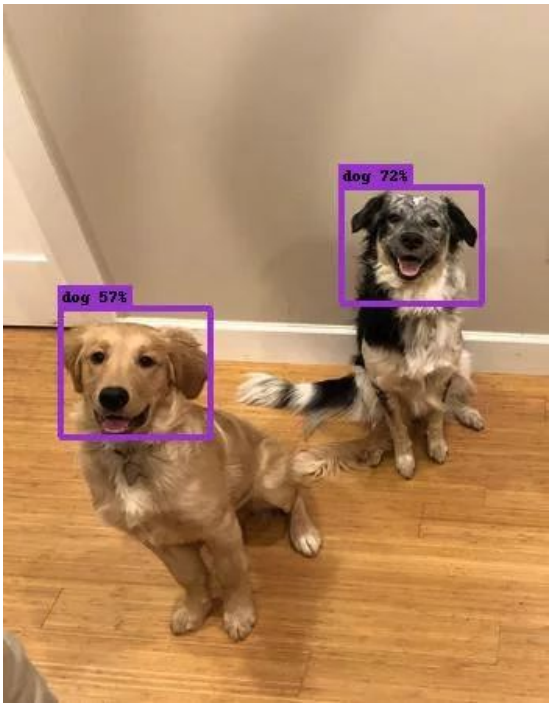
With recent advancements in deep learning based computer vision models , object detection applications are easier to develop than ever before. Besides significant performance improvements, these techniques have also been leveraging massive image datasets to reduce the need for large datasets. In addition, with current approaches focussing on full end-to-end pipelines, performance has also improved significantly, enabling real-time use cases.

People often confuse image classification and object detection scenarios. In general, if you want to classify an image into a certain category, you use image classification. On the other hand, if you aim to identify the location of objects in an image, and, for example, count the number of instances of an object, you can use object detection.



Data requirements

In order to train a custom model, you need **labelled data**. Labelled data in the context of object detection are images with corresponding **bounding box coordinates and labels**. That is, the bottom left and top right (x,y) coordinates + the class .



A question that is always asked is the following: in order to do object detection on problem X, how many pictures do I need? Instead, it is more important to properly understand in which scenarios the model will be deployed. It is crucial that a large number (for example, > 100 and potentially >1000) **representative** images are available per class. Representative in this context means that they should corresponds with the range of scenarios in which the model will be used. If you are building a traffic sign detection model that will run in a car, you have to use images taken under different weather, lighting and camera conditions in their appropriate context. Object detection models are not magic and actually rather dumb. If the model does not have enough data to learn general patterns, it won't perform well in production.

While the image on the left is clear and easy to detect, ultimately, you should train on data which better reflects the use case.

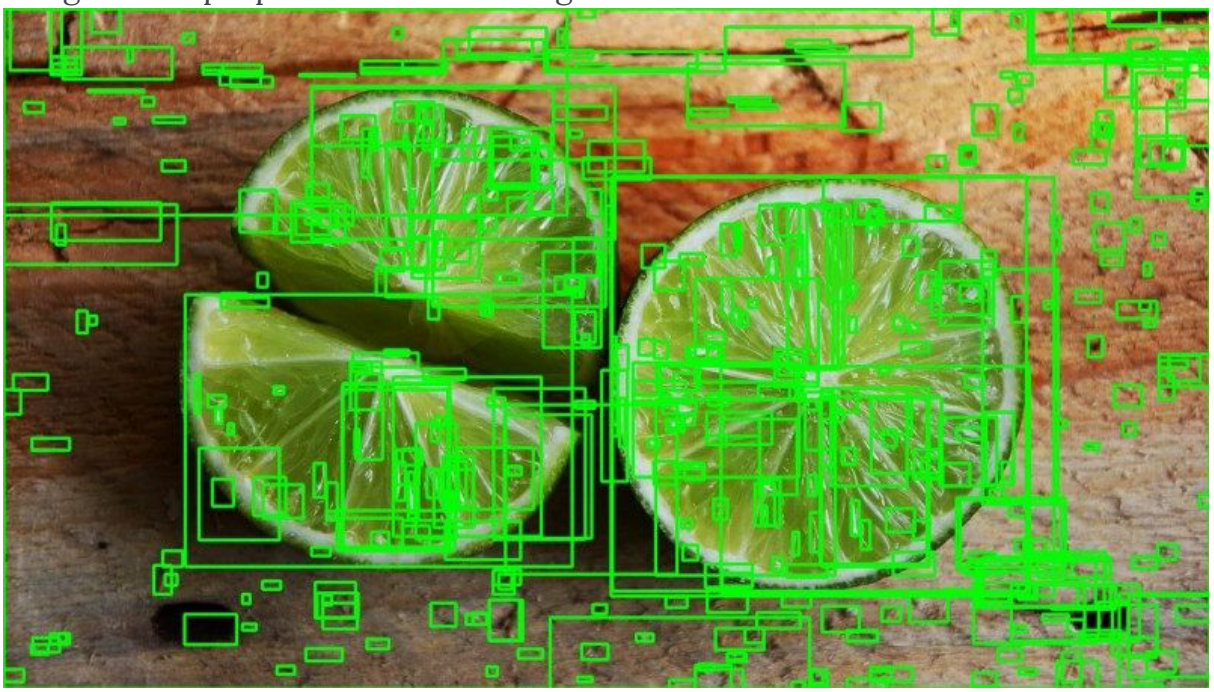
General object detection framework

Typically, there are **three steps** in an object detection framework.

1. First, a model or algorithm is used to generate **regions of interest or region proposals**. These region proposals are a large set of bounding boxes spanning the full image (that is, an object localisation component).
2. In the second step, **visual features** are extracted for each of the bounding boxes, they are evaluated and it is determined whether and which objects are present in the proposals based on visual features (i.e. an **object classification component**).
3. In the final post-processing step, overlapping boxes are combined into a single bounding box (that is, **non maximum suppression)**.

Region proposals

Several different approaches exist to generate region proposals. Originally, the 'selective search' algorithm was used to generate object proposals. In short, selective search is a clustering based approach which attempts to group pixels and generate proposals based on the generated clusters.



An example of selective search applied to an image. A threshold can be tuned in the SS algorithm to generate more or fewer proposals.

An important trade-off that is made with region proposal generation is the number of regions vs. the computational complexity. The more regions you generate, the more likely you will be able to find the object. On the flip-side, if you exhaustively generate all possible proposals, it won't be possible to run the object detector in real-time, for example. In some cases, it is possible to use

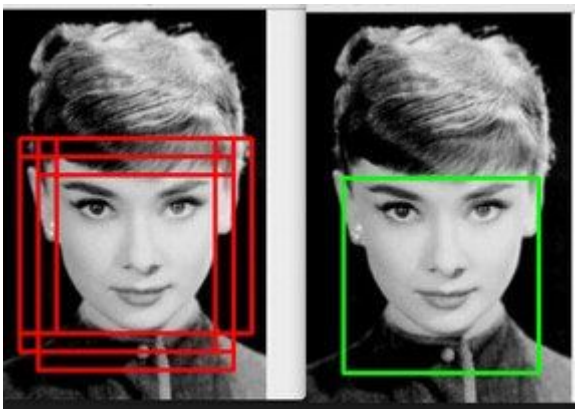
problem specific information to reduce the number of ROI's. For example, pedestrians typically have a ratio of approximately 1.5, so it is not useful to generate ROI's with a ratio of 0.25.

Feature extraction

The goal of feature extraction is to reduce a variable sized image to a fixed set of visual features. Image classification models are typically constructed using strong visual feature extraction methods. Whether they are based on traditional computer vision approaches, such as for example, filter based approached, histogram methods, etc., or deep learning methods, they all have the exact same objective: extract features from the input image that are representative for the task at hands and use these features to determine the class of the image. In object detection frameworks, people typically use pretrained image classification models to extract visual features, as these tend to generalise fairly well. For example, a model trained on the MS CoCo dataset is able to extract fairly generic features.

Non-maximum suppression

The general idea of non-maximum suppression is to reduce the number of detections in a frame to the actual number of objects present. If the object in the frame is fairly large and more than 2000 object proposals have been generated, it is quite likely that some of these will have significant overlap with each other and the object. NMS techniques are typically standard across the different detection frameworks, but it is an important step that might require hyperparameter tweaking based on the scenario.



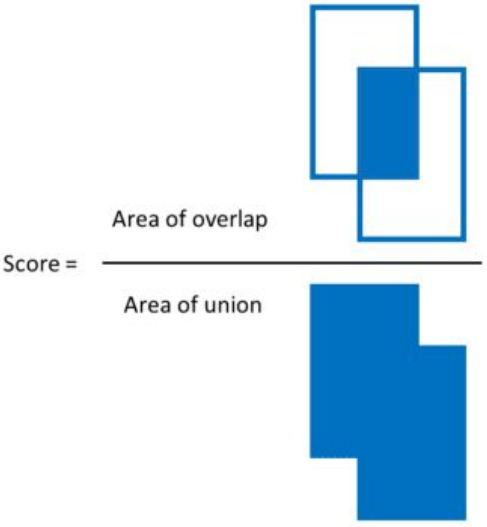
An example of NMS in the context of face detection.

Evaluation metric

The most common evaluation metric that is used in object recognition tasks is 'mAP', which stands for **'mean average precision'**. It is a number from 0 to 100 and higher values are typically better, but it's value is different from the accuracy metric in classification.

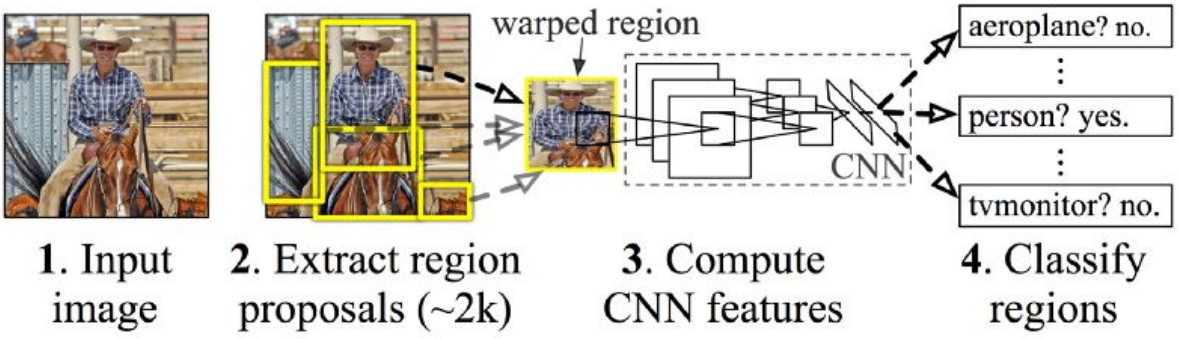
Each bounding box will have a score associated (likelihood of the box containing an object). Based on the predictions a precision-recall curve (PR curve) is computed for each class by varying the score threshold. The average precision (AP) is the area under the PR curve. First the AP is computed for each class, and then averaged over the different classes. The end result is the mAP.

Note that a detection is a true positive if it has an **'intersection over union'** (IoU or overlap) with the ground-truth box greater than some threshold (usually 0.5). Instead of using mAP we typically use mAP@0.5 or mAP@0.25 to refer to the IoU that was used.

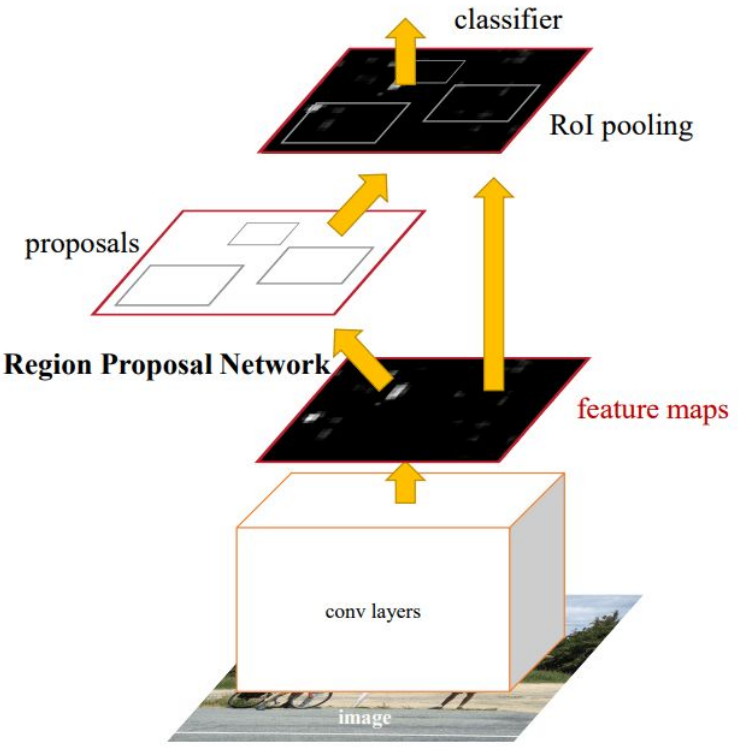


Faster R-CNN

Faster R-CNN was developed by researchers at **Microsoft**. It is based on R-CNN which used a multi-phased approach to object detection. R-CNN used **Selective search** to determine region proposals, pushed these through a classification network and then used an SVM to classify the different regions.



Instead of using default bounding boxes, Faster R-CNN has a Region Proposal Network (RPN) to generate a fixed set of regions. The RPN uses the convolutional features from the the image classification network, enabling nearly cost-free region proposals. The RPN is implemented as a fully convolutional network that predicts object bounds and objectness scores at each position.

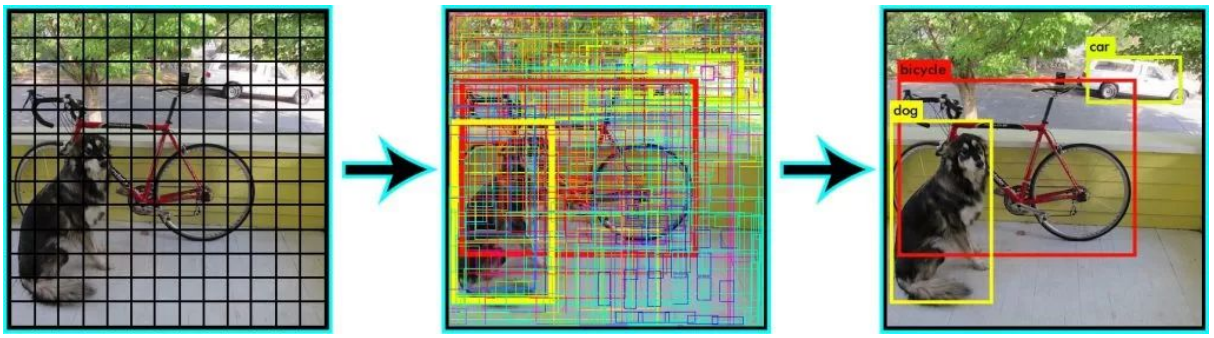


YOLO

For YOLO, detection is a simple **regression problem** which takes an input image and learns the class probabilities and bounding box coordinates.

YOLO divides each image into a **grid of S x S** and each grid predicts **N bounding boxes and confidence**. The confidence reflects the accuracy of the bounding box and whether the bounding box actually contains an object(regardless of class). YOLO also predicts the classification score for each box for every class in training. You can combine both the classes to calculate the probability of each class being present in a predicted box.

So, total SxSxN boxes are predicted. However, most of these boxes have low confidence scores and if we set a threshold say 30% confidence, we can remove most of them as shown in the example below.



Notice that at runtime, we have run our image on CNN only once. Hence, YOLO is super fast and can be run real time. Another key difference is that YOLO sees the complete image at once as opposed to looking at only a generated region proposals in the previous methods. So, this contextual information helps in avoiding false positives. However, one limitation for YOLO is that it only predicts 1 type of class in one grid hence, it struggles with very small objects.

Demo of the application

The application runs two detection models (Faster RCNN and YOLO models) on a given image and compares the results. As expected, YOLO proves to be much faster but on the other hand, less accurate.

YOLO

Faster RCNN

Number of objects detected with Faster RCNN : 11
Number of objects detected with YOLO : 9
Mean accuracy Faster RCNN : 0.99
Mean accuracy YOLO : 0.88