Recent Advances in Object Detection in the Age of Deep Convolutional Neural Networks

Shivang Agarwal(∗,1), Jean Ogier du Terrain(∗,1,2), Frédéric Jurie(1)

(∗) equal contribution

(1)Normandie Univ, UNICAEN, ENSICAEN, CNRS
(2)Safran Electronics and Defense

August 16, 2019

Abstract

Object detection, the computer vision task dealing with detecting instances of objects of a certain class (e.g., ‘car’, ‘plane’, etc.) in images, attracted a lot of attention from the community during the last six years. This strong interest can be explained not only by the importance this task has for many applications but also by the phenomenal advances in this area since the arrival of deep convolutional neural networks (DCNNs). This article reviews the recent literature on object detection with deep CNN, in a comprehensive way. This study covers not only the design decisions made in modern deep (CNN) object detectors, but also provides an in-depth perspective on the set of challenges currently faced by the computer vision community, as well as some complementary and new directions on how to overcome them. In its last part it goes on to show how object detection can be extended to other modalities and conducted under different constraints. This survey also reviews in its appendix the public datasets and associated state-of-the-art algorithms.
## Contents

1 Introduction ................................................. 4
   1.1 What is object detection in images? How to evaluate detector performance? .......... 6
      1.1.1 Problem definition ..................................... 6
      1.1.2 Performance evaluation .................................. 6
      1.1.3 Other detection tasks .................................. 8
   1.2 From Hand-crafted to Data Driven Detectors ............................................. 9
   1.3 Overview of Recent Detectors ............................................. 10

2 On the Design of Modern Deep Detectors ............................................. 11
   2.1 Architecture of the Networks ............................................. 11
      2.1.1 Backbone Networks ......................................... 11
      2.1.2 Single Stage Detectors ................................... 14
      2.1.3 Double Stage Detectors .................................. 17
      2.1.4 Cascades .................................................. 21
      2.1.5 Parts-Based Models ....................................... 23
   2.2 Model Training .............................................. 24
      2.2.1 Losses ................................................... 24
      2.2.2 Hyper-Parameters ....................................... 26
      2.2.3 Pre-Training ............................................. 27
      2.2.4 Data Augmentation ....................................... 27
   2.3 Inference .................................................. 29
   2.4 Concluding Remarks ........................................... 30

3 Going Forward in Object Detection ............................................. 31
   3.1 Major Challenges ............................................. 31
      3.1.1 Scale Variance ........................................... 32
      3.1.2 Rotational Variance ...................................... 33
      3.1.3 Domain Adaptation ....................................... 34
      3.1.4 Object Localization ...................................... 36
      3.1.5 Occlusions ............................................... 37
      3.1.6 Detecting Small Objects .................................. 38
   3.2 Complementary New Ideas in Object Detection ........................................... 39
      3.2.1 Graph Networks ........................................... 39
      3.2.2 Adversarial Trainings .................................... 41
      3.2.3 Use of Contextual Information ............................. 42
   3.3 Concluding Remarks ............................................. 43

4 Extending Object Detection ............................................. 43
   4.1 Detecting Objects in Other Modalities ........................................... 43
      4.1.1 Object Detection in Videos ................................ 44
      4.1.2 Object Detection in 3D Point Clouds .......................... 45
   4.2 Detecting Objects Under Constraints ............................................... 46
      4.2.1 Weakly Supervised Detection ................................ 46
      4.2.2 Few-shot Detection ....................................... 48
      4.2.3 Zero-shot Detection ....................................... 49
      4.2.4 Fast and Low Power Detection ................................ 49
   4.3 Towards Versatile Object Detectors ................................................. 50
      4.3.1 Interpretability and Robustness ................................ 52
      4.3.2 Universal Detector, Lifelong Learning .......................... 52
   4.4 Concluding Remarks .............................................. 53
### Appendix A Datasets and Results

#### A.1 Classical Datasets with Common Objects
- A.1.1 Pascal-VOC
- A.1.2 MS COCO
- A.1.3 ImageNet Detection Task
- A.1.4 VisualGenome
- A.1.5 OpenImages

#### A.2 Specialized datasets
- A.2.1 Aerial Imagery
- A.2.2 Text Detection in Images
- A.2.3 Face Detection
- A.2.4 Pedestrian Detection
- A.2.5 Logo Detection
- A.2.6 Traffic Signs Detection
- A.2.7 Other Datasets

#### A.3 3D Datasets

#### A.4 Video Datasets

#### A.5 Concluding Remarks
1 Introduction

The task of automatically recognizing and locating objects in images and videos is important in order to make computers able to understand or interact with their surroundings. For humans, it is one of the primary tasks, in the paradigm of visual intelligence, in order to survive, work and communicate. If one wants machines to work for us or with us, they will need to make sense of their environment as good as humans or in some cases even better than humans. Solving the problem of object detection with all the challenges it presents has been identified as a major precursor to solving the problem of semantic understanding of the surrounding environment.

A large number of academics as well as industry researchers have already shown their interest in it by focusing on applications, such as autonomous driving, surveillance, relief and rescue operations, deploying robots in factories, pedestrian and face detection, brand recognition, visual effects in images, digitizing texts, understanding aerial images, etc. which have object detection as a major challenge at their core.

The Semantic Gap, defined by Smeulders et al. [349] as the lack of coincidence between the information one can extract from some visual data and its interpretation by a user in a given situation, is one of the main challenges object detection must deal with. There is indeed a difference of nature between raw pixel intensities contained in images and semantic information depicting objects.

Object detection is a natural extension of the classification problem. The added challenge is to correctly detect the presence and accurately locate the object instance(s) in the image (Figure 1). It is (usually) a supervised learning problem in which, given a set of training images, one has to design an algorithm which can accurately locate and correctly classify as many object instances as possible in a rectangle box while avoiding false detections of background or multiple detections of the same instance. The images can have object instances from same classes, different classes or no instances at all. The object categories in training and testing set are then supposed to be statistically similar. The instance can occupy very few pixels, 0.01% to 0.25%, as well as the majority of the pixels, 80% to 90%, in an image. Apart from the variation in size the variation can be in lighting, rotation, appearance, background, etc. There may not be enough data to accurately cover all the variations well enough. Small objects, particularly, give low performance at being detected because the available information to detect them is present but compressed and hard to decode without some prior knowledge or context. Some object instances can also be occluded.

An additional difficulty is that real world applications like video object detection demand this problem to be solved in real time. With the current state of the art detectors that is often not the case. Fastest detectors are usually worse than the best performing ones (e.g. heavy ensembles).

We present this review to connect the dots between various deep learning and data driven techniques proposed in recent years, as they have brought about huge improvements in the performance, even though the recently introduced object detection datasets are much more challenging. We intend to study what makes them work and what are their shortcomings. We discuss the seminal works in the field and the incremental works which are more application oriented. We also see their approach on trying to overcome each of the challenges. The earlier methods which were based on hand-crafted features are outside the scope of this review. The problems that are related to object detection such as semantic segmentation are also outside the scope of this review, except when used to bring contextual information to detectors. Salient object detection being related to semantic segmentation will also not be treated in this survey.

Several surveys related to object detection have been written in the past, addressing specific tasks such as pedestrian detection [84], moving objects in surveillance systems [161], object detection in remote sensing images [53], face detection [126, 453], facial landmark detection [420], to cite only some illustrative examples. In contrast with this article, the aforementioned surveys do not cover the latest advances obtained with deep neural net-
works. Recently four non peer reviewed surveys appeared on arXiv that also treat the subject of Object Detection using Deep Learning methods. This article shares the same motivations as [470] and [35], but covers the topic more comprehensively and extensively as these two surveys which only cover the backbones and flagship articles associated with modern object detection. This work investigates more thoroughly papers that one would not necessarily call mainstream, like boosting methods or true cascades, and study related topics like weakly supervised learning and approaches that carry promises but that have yet to become widely used by the community (graph networks and generative methods). Concurrently to this article, the paper by [220] goes into many details about the modern object detectors. We wanted this survey to be more than just an inventory of existing methods but to provide the reader with a complete tool-set to be able to understand fully how the state of the art came to be and what are the potential leads to advance it further, by studying surrounding topics such as interpretability, lifelong detectors, few-shot learning or domain adaptation (in addition to delving into non mainstream methods already mentioned).

The following subsections give an overview of the problem, some of the seminal works in the field (hand-crafted as well as data driven) and describe the task and evaluation methodology. Section 2 goes into the detail of the design of the current state-of-the-art models. Section 3 presents recent methodological advances as well as the main challenge modern detectors have to face. Section 4 shows how to extend the presented detectors to different detection tasks (video, 3D) or perform under different constraints (energy efficiency, training data, etc.). Finally, Section 5 concludes the review. We also list a wide variety of datasets and the associated state of the art performances in the Appendix.
1.1 What is object detection in images? How to evaluate detector performance?

1.1.1 Problem definition

Object detection is one of the various tasks related to the inference of high-level information from images. Even if there is no universally accepted definition of it in the literature, it is usually defined as the task of locating all the instances of a given category (e.g. ‘car’ instances in the case of car detection) while avoiding raising alarms when/where no instances are present. The localization can be provided as the center of the object on the image, as a bounding box containing the object, or even as the list of the pixels belonging to the object. In some rare cases, only the presence/absence of at least one instance of the category is sought, without any localization.

Object detection is always defined with respect to a dataset containing images associated with a list containing some descriptions (position, scale, etc.) of the objects each image contains. Let’s denote by $\mathcal{I}$ an image and $O(I)$ the set of $N^*_I$ object descriptions, with:

$$O(I) = \{(Y_1^*, Z_1^*), \ldots, (Y_{N^*_I}^*, Z_{N^*_I}^*)\}$$

where $Y_i^* \in \mathcal{Y}$ represent the category of the $i$-th object and $Z_{N^*_I}^* \in \mathcal{Z}$ a representation of its location/scale/geometry in the image. $\mathcal{Y}$ is the set of possible categories, which can be hierarchical or not. $\mathcal{Y}$ is the space of possible locations/scales/geometries of objects in images. It can be the position of the center of the object $(x_c, y_c) \in \mathbb{R}^2$, a bounding box $(x_{\min}, y_{\min}, x_{\max}, y_{\max}) \in \mathbb{R}^4$ encompassing the object, a mask, etc.

Using these notations, object detection can be defined as the function associating an image with a set of detections

$$D(I, \lambda) = \{(Y_1, Z_1), \ldots, (Y_i, Z_i), \ldots, (Y_{N(\lambda)}, Z_{N(\lambda)})\}.$$  

The operating point $\lambda$ allows to fix a tradeoff between false alarms and missed detections.

Object detection is related but different from object segmentation, which aims to group pixels from the same object into a single region, or semantic segmentation which is similar to object segmentation except that the classes may also refer to varied backgrounds or ‘stuff’ (e.g. ‘sky’, ‘grass’, ‘water’ etc., categories). It is also different from Object Recognition which is usually defined as recognizing (i.e. giving the name of the category) of an object contained in an image or a bounding box, assuming there is only one object in the image. For some authors Object Recognition involves detecting all the objects in an image. Instance object detection is more restricted than object detection as the detector is focused on a single object (e.g. a particular car model) and not any object of a given category. In case of videos, object detection task is to detect the objects on each frame of the video.

1.1.2 Performance evaluation

Evaluating the function of detection for a given image $I$ is done by comparing the actual list of objects locations $O(I)$ (so-called the ground truth) of a given category with the detections $D(I, \lambda)$ provided by the detector. Such a comparison is possible only once the two following definitions are given:

1. A geometric compatibility function $G : (Z, Z^*) \in \mathcal{Z} \times \mathcal{Z} \rightarrow \{0, 1\}$ defining the conditions that must be met for considering two locations as equivalent.

2. An association matrix $A \in \{0, 1\}^{N(I, \lambda) \times N^*(I)}$ defining a bipartite graph between the detected objects $\{Z_1, \ldots, Z_{N(I, \lambda)}\}$ and the $A(i, j) \leq 1$ ground truth objects $\{Z_i^*, \ldots, Z_{N^*(I)}^*\}$, with:

$$\sum_{j=1}^{N(I, \lambda)} A(i, j) \leq 1$$
$$\sum_{i=1}^{N^*(I)} A(i, j) \leq 1$$

$$G(Z_i^*, Z_j) = 0 \implies A(i, j) = 0$$
With such definitions, the number of correct detections is given by

\[ TP(I, \lambda) = \sum_{i,j} A(i,j) \]

If several association matrices \( A \) satisfy the previous constraints, the one maximizing the number of correct detections is chosen. An illustration of TP and False Positives (FP) in an image is shown in Figure 2.

Such a definition can be viewed as the size of the maximal matching in a bipartite graph. Nodes are locations (ground truth, on the one hand, detections on the other hand). Edges are based on the acceptance criterion \( G \) and the constraints stating that ground truth object and detected objects can be associated only once each.

It is possible to average the correct detections at the level of a test set \( T \) through the two following ratios:

\[
\text{Precision}(\lambda) = \frac{\sum_{I} TP(I, \lambda)}{\sum_{I \in T} N(I, \lambda)}
\]

\[
\text{Recall}(\lambda) = \frac{\sum_{I} TP(I, \lambda)}{\sum_{I \in T} N^*(I)}
\]

Precision/Recall curve is obtained by varying the operational point \( \lambda \). The Mean Average Precision can be computed by averaging the Precision for several Recall values (typically 11 equally spaced values).

The definition of \( G \) can vary from data sets to data sets. However, only a few definitions reflect most of the current research. One of the most common one comes the Pascal VOC challenge [88]. It assumes ground truths are defined by non-rotated rectangular bounding boxes containing object instances, associated with class labels. The diversity of the methods to be evaluated prevents the use of ROC (Receiver Operating Characteristic) or DET (Detection Error Trade-off), commonly used for face detection, as it would assume all the methods use the same window extraction scheme (such as the sliding window mechanism), which is not always the case. In the Pascal VOC challenge, object detection is evaluated by one separate AP score per category. For a given category, the Precision/Recall curve is computed from the ranked outputs (bounding boxes) of the method to be evaluated. Recall is the proportion of positive examples ranked above a given rank, while precision is the number of positive boxes above that rank. The AP summarizes the Precision/Recall curve and is defined as the mean (interpolated) precision of the set of eleven equally spaced recall levels. Output bounding boxes are judged as true positives (correct detections) if the overlap ratio (intersection over union or IOU) exceeds 0.50. Detection outputs are assigned to ground truth in the order given by decreasing confidence scores. Duplicated detections of the same object are considered as false detections. The performance over the whole dataset is computed by averaging the APs across all the categories.

The recent and popular MSCOCO challenge [214] relies on the same principles. The main difference is that the overall performance (mAP) is obtained by averaging the AP obtained with 10 different IOU thresholds between 0.50 and 0.95. The ImageNet Large Scale Visual Recognition Challenge (ILSVRC) also has a detection task in which algorithms have to produce triplets of class labels,
bounding boxes and confidence scores. Each image has mostly one dominant object in it. Missing object detections are penalized in the same way as a duplicate detection and the winner of the detection challenge is the one who achieves first place AP on most of the object categories. The challenge also has the Object Localization task, with a slightly different definition. The motivation is not to penalize algorithms if one of the detected objects is actually present while not included in the ground-truth annotations, which is not rare due to the size of the dataset and the number of categories (1000). Algorithms are expected to produce 5 class labels (in decreasing order of confidence) and 5 bounding boxes (one for each class label). The error of an algorithm on an image is 0 if one of the 5 bounding boxes is a true positive (correct class label and correct localization according to IOU), 1 otherwise. The error is averaged on all the images of the test set.

Some recent datasets, like DOTA [421], proposed two tasks named as detection on horizontal bounding boxes and detection on oriented bounding boxes, corresponding to two different kinds of ground truths (with or without target orientations), no matter how those methods were trained. In some other datasets, the scale of the detection is not important and a detection is counted as a True Positive if its coordinates are close enough to the center of the object. This is the case for the VeDAI dataset [302]. In the particular case of object detection in 3D point clouds, such as in the KITTI object detection benchmark [98], the criteria is similar to Pascal VOC, except that the boxes are in 3D and the overlap is measured in terms of volume intersection.

Object detection in videos Regarding the detection of objects in videos, the most common practice is to evaluate the performance by considering each frame of the video as being an independent image and averaging the performance over all the frames, as done in the ImageNet VID challenge [319].

It is also possible to move away from the 2D case and to evaluate video-mAP based on tubelets IoU, where a tubelet is detected if and only if the mean per frame IoU for every frame in the video is greater than a threshold, $\sigma$, and the tube label is correctly predicted. We take this definition directly from [107], where they used it to compute mAP and ROC curves at a video-level.

1.1.3 Other detection tasks

This survey only covers the methodologies for performance evaluation found in the recent literature. But, beside these common evaluation measures, there are a lot of more specific ones, as object detection can be combined with other complex tasks, e.g., 3D orientation and layout inference in [423]. The reader can refer to the review by Mariano et al. [232] to explore this topic. It is also worth mentioning the very recent work of Oksuz et al. [260] which proposes a novel metric providing richer and more discriminative information than AP, especially with respect to the localization error.

We have decided to orient this survey mainly on bounding boxes tasks even if there is a tendency to move away from this task considering the performances of the modern deep learning methods that already approach human accuracy on some datasets. The reason of this choice are numerous. First of all, historically speaking bounding boxes were one of the first object detection task and thus there is already a body of literature on this topic that is immense. Secondly, not all the datasets provide annotation down to the level of pixels. In aerial imagery for instance most of the datasets are only bounding boxes. It is also the case for some pedestrian detection datasets. Instance segmentation level annotations are still costly for the moment, even with the recent development of annotator friendly algorithms (e.g., [32, 230]) that offer pixel level annotations at the expense of a few user clicks. Maybe in the future all datasets will contain annotations down to the level of pixels but it is not yet the case. Even when one has pixel-level annotations for tasks like instance segmentation, which is becoming the standard, bounding boxes are needed from the detector to distinguish between two in-
stances of the same class, which explains that most modern instance segmentation pipelines like [118] have a bounding box branch. Therefore, metrics evaluating the bounding boxes from the models are still relevant in that case. One could also make the argument that bounding boxes are more robust annotations because they are less sensitive to the annotator noise but it is debatable. For all of these reasons the rest of this survey will tackle mainly bounding boxes and associated tasks.

1.2 From Hand-crafted to Data Driven Detectors

While the first object detectors initially relied on mechanisms to align a 2D/3D model of the object on the image using simple features, such as edges [217], key-points [224] or templates [278], the arrival of Machine Learning (ML) was the first revolution which had shaken up the area. One of the most popular ML algorithms used for object detection was boosting, e.g., [326]) or Support Vector Machines, e.g. [64]. This first wave of ML-based detectors were all based on hand-crafted (engineered) visual features processed by classifiers or regressors. These hand-crafted features were as diverse as Haar Wavelets [398], edgelets [418], shapelets [320], histograms of oriented gradient [64], bags-of-visual-words [187], integral histograms [287], color histograms [399], covariance descriptors [388], linear binary patterns Wang et al. [408], or their combinations [85]. One of the most popular detectors before the DCNN revolution was the Deformable Part Model of Felzenszwalb et al. [90] and its variants, e.g. [322].

This very rich literature on visual descriptors has been wiped out in less than five years by Deep Convolutinal Neural Networks, which is a class of deep, feed-forward artificial neural networks. DCNNs are inspired by the connectivity patterns between neurons of the human visual cortex and use no pre-processing as the network learns itself the filters previously hand-engineered by traditional algorithms, making them independent from prior knowledge and human effort. They are said to be end-to-end trainable and solely rely on the training data. This leads to their major disadvantage of requiring copious amounts of data. The first use of ConvNets for detection and localization goes back to the early 1990s for faces [392], hands [258] and multi-character strings [237]. Then in 2000s they were used in text [65], face [96, 263] and pedestrians [328] detection.

However, the merits of DCNN for object detection was generated in the community only after the seminal work of Krizhevsky et al. [181] and Sermanet et al. [327] on the challenging ImageNet dataset. Krizhevsky et al. [181] were the first to demonstrate localization through DCNN in the ILSVRC 2012 localization and detection tasks. Just one year later Sermanet et al. [327] were able to describe how the DCNN can be used to locate and detect objects instances. They won the ILSVRC 2013 localization and detection competition and also showed that combining the classification, localization and detection tasks can simultaneously boost the performance of all tasks.

The first DCNN-based object detectors applied a fine-tuned classifier on each possible location of the image in a sliding window manner [262], or on some specific regions of interest [105], through a region proposal mechanism. Girshick et al. [105] treated each region proposal as a separate classification and localization task. Therefore, given an arbitrary region proposal, they deformed it to a warped region of fixed dimensions. DCNN are used to extract a fixed-length feature vector from each proposal respectively and then category-specific linear SVMs were used to classify them. Since it was a region based CNN they called it R-CNN. Another important contribution was to show the usability of transfer learning in DCNN. Since data is scarce, supervised pre-training on an auxiliary task can lead to a significant boost to the performance of domain specific fine-tuning. Sermanet et al. [327], Girshick et al. [105] and Oquab et al. [262] were among the first authors to show that DCNN can lead to dramatically higher object detection performance on ImageNet detection challenge [66] and PASCAL VOC [88] respectively as compared to previous state-of-the-art systems based on HOG [64] or SIFT [225].
Since most prevalent DCNN had to use a fixed size input, because of the fully connected layers at the end of the network, they had to either warp or crop the image to make it fit into that size. He et al. [116] came up with the idea of aggregating feature maps of the final convolutional layer. Thus, the fully connected layer at the end of the network gets a fixed size input even if the input images in the dataset are of varying sizes and aspect ratios. This helped reduce overfitting, increased robustness and improved the generalizability of the existing models. Compared to R-CNN which used one forward pass per proposal to generate the feature map, the methodology proposed by [116] allowed to share computation among all the proposals and do just one forward pass for the whole image and then select the region from the final feature map according to the regions proposed. This naturally increased the speed of the network by over one hundred times.

All the previous approaches train the network in multistage pipelines are complex, slow and inelegant. They include extracting features through CNNs, classifying through SVMs and finally fitting bounding box regressors. Since, each task is handled separately, convolutional layers cannot take advantage of end-to-end learning and bounding box regression. Girshick [104] helped alleviate this problem by streamlining all the tasks in a single model using a multitask loss. As we will explain later, this not only improved upon the accuracy but also made the network run faster at test time.

### 1.3 Overview of Recent Detectors

The foundations of the DCNN based object detection, having been laid out, it allowed the field to mature and move further away from classical methods. The fully-convolutional paradigm glimpsed in [327] gained more traction every day in the community.

When Ren et al. [309] successfully replaced the only component of Fast R-CNN that still relied on non-learned heuristics by inventing RPN (Region Proposal Networks), it put the last nail in the coffin of traditional object detection and started the age of completely end-to-end architectures. Specifically, the anchor mechanism, developed for the RPN, was here to stay. This grid of fixed a-priori (or anchors), not necessarily corresponding to the receptive field of the feature map pixel they lied on, created a framework for fully-convolutional classification and regression and is used nowadays by most pipelines like [221] or [216], to cite a few.

These conceptual changes make the detection pipelines far more elegant and efficient than their counterparts when dealing with big training sets. However, it comes at a cost. The resulting detectors become complete black boxes, and, because they are more prone to overfitting, they require more data than ever.

[309] and its other double stage variants are now the go-to methods for objects detection and will be thoroughly explored in Sec. 2.1.3. Although this line of work is now prominent, other choices were explored all based on fully-convolutional architectures.

Single-stage algorithms that were completely abandoned since Viola et al. [398] have now become reasonable alternatives thanks to the discriminative power of the CNN features. Redmon et al. [308] first showed that the simplest architectural design could bring unfathomable speed with acceptable performances. Liu et al. [221] sophisticated the pipeline by using anchors at different layers while making it faster and more accurate than Redmon et al. [308]. These two seminal works gave birth to a considerable amount of literature on single stage methods that we will cover in Sec. 2.1.2. Boosting and Deformable part-based models, that were once the norm, have yet to make their comebacks into the mainstream. However, some recent popular works used close ideas like Dai et al. [63] and thus these approaches will also be discussed in the survey sections 2.1.4 and 2.1.5.

The fully-convolutional nature of these new dominant architectures allows all kinds of implementation tricks during training and at inference time that will be discussed at the end of the next section. However, it makes the subtle design choices of the different architectures something of a dark art to the newcomers.
The goal of the rest of the survey is to provide a complete view of this new landscape while giving the keys to understand the underlying principles that guide interesting new architectural ideas. Before diving into the subject, the survey starts by reminding the readers about the object detection task and the metrics associated with it.

After introducing the topic and touching upon some general information, next section will get right into the heart of object detection by presenting the designs of recent deep learning based object detectors.

2 On the Design of Modern Deep Detectors

Here we analyze, investigate and dissect the current state-of-the-art models and the intuition behind their approaches. We can divide the whole detection pipeline into three major parts. The first part focuses on the arrangement of convolutional layers to get proposals (if required) and box predictions. The second part is about setting various training hyper-parameters, deciding upon the losses, etc. to make the model converge faster. The third part’s center of attention will be to know various approaches to refine the predictions from the converged model(s) at test time and therefore get better detection performances. The first part has been able to get the attention of most of the researchers and second and third part not so much. To give a clear overview of all the major components and popular options available in them, we present a map of object detection pipeline in Figure 3.

Most of the ideas from the following sub-sections have achieved top accuracy on the challenging MS COCO [214] object detection challenge and PASCAL VOC [88] detection challenge or on some other very challenging datasets.

2.1 Architecture of the Networks

The architecture of the DCNN object detectors follows a Lego-like construction pattern based on chaining different building blocks. The first part of this Section will focus on what researchers call the backbone of the DCNN, meaning the feature extractor from which the detector draws its discriminative power. We will then tackle diverse arrangements of increasing complexity found in DCNN detectors: from single stage to multiple stages methods. Finally, we will talk about the Deformable Part Models and their place in the deep learning landscape.

2.1.1 Backbone Networks

A lot of deep neural networks originally designed for classification tasks have been adopted for the detection task as well. And a lot of modifications have been done on them to adapt for the additional difficulties encountered. The following discussion is about these networks and the modifications in question.

Backbones: Backbone networks play a major role in object detection models. Huang et al. [140] partially confirmed the common observation that, as the classification performance of the backbone increases on ImageNet classification task [319], so does the performance of object detectors based on those backbones. It is the case at least for popular double-stage detectors like Faster-RCNN [309] and R-FCN [62] although for SSD [221] the object detection performance remains around the same (see the following Sections for details about these 3 architectures).

However, as the size of the network increases, the inference and the training become slower and require more data. The most popular architectures in increasing order of inference time are MobileNet [134], VGG [343], Inception [151, 364, 365], ResNet [117], Inception-ResNet [366], etc. All of the above architectures were first borrowed from the classification problem with little or no modification.
Figure 3: A map of object detection pipeline along with various options available in different parts. Images are taken from a Dataset and then fed to a Network. They use one of the Single-Stage (see Figure 6) or Double-Stage Framework (see Figure 8) to make Predictions of the probabilities of each class and their bounding boxes at each spatial location of the feature map. During training these predictions are fed to Losses and during testing these are fed to Inference for pruning. Based on the final predictions, a Performance is evaluated against the ground-truths. All the ideas are referenced in the text. Best viewed in color.
Some other backbones used in object detectors which were not included in the analysis of [140] but have given state-of-the-art performances on ImageNet [66], or COCO [214] detection tasks are Xception [57], DarkNet [306], Hourglass [256], Wide-Residual Net [193, 445], ResNeXt [426], DenseNet [139], Dual Path Networks [50] and Squeeze-and-Excitation Net [136]. The recent DetNet [208], proposed a backbone network, is designed specifically for high performance detection. It avoided large down-sampling factors present in classification networks. Dilated Residual Networks [439] also worked with similar motivations to extract features with fewer strides. SqueezeNet [148] and ShuffleNet [461] choose instead to focus on speed. More information for networks focusing on speed can be found in Section 4.2.4.

Adapting the mentioned backbones to the inherent multi-scale nature of object detection is a challenge, we will give in the following paragraph examples of commonly used strategies.

Multi-scale detections: Papers [28, 200, 431] made independent predictions on multiple feature maps to take into account objects of different scales. The lower layers with finer resolution have generally been found better for detecting small objects than the coarser top layers. Similarly, coarser layers are better for the bigger objects. Liu et al. [221] were the first to use multiple feature maps for detecting objects. Their method has been widely adopted by the community. Since final feature maps of the networks may not be coarse enough to detect sizable objects in large images, additional layers are also usually added. These layers have a wider receptive field.

Fusion of layers: In object detection, it is also helpful to make use of the context pixels of the object [102, 446, 451]. One interesting argument in favor of fusing different layers is it integrates information from different feature maps with different receptive fields, thus it can take help of surrounding
local context to disambiguate some of the object instances.

Some papers [51, 92, 156, 192, 471] have experimented with fusing different feature layers of these backbones so that the finer layers can make use of the context learned in the coarser layers. Lin et al. [215, 216], Shrivastava et al. [338], Woo et al. [415] took one step ahead and proposed a whole additional top-down network in addition to standard bottom-up network connected through lateral connections. The bottom-top network used can be any one of the above mentioned. While Shrivastava et al. [338] used only the finest layer of top-down architecture for detection, Feature Pyramid Network (FPN) [216] and RetinaNet [215] used all the layers of top-down architecture for detection. FPN used the feature maps thus generated in a two-stage detector fashion while RetinaNet used them in a single-stage detector fashion (See Section 2.1.3 and Section 2.1.2 for more details). FPN [215] has been a part of the top entries in MS COCO 2017 challenge. An illustration of multiple scales and fusion of layers is shown in Figure 4.

Now that we have seen how to best use the feature maps of the object detectors backbones we can explore the architectural details of the different major players in DCNN object detection, starting with the most immediate methods: single-stage detectors.

2.1.2 Single Stage Detectors

The two most popular approaches in single stage detection category are YOLO [308] and SSD [221]. In this Section we will go through their basic functioning, some upsides and downsides of using these two approaches and further improvements proposed on them.

YOLO: Redmon et al. [308] presented for the first time a single stage method for object detection where raw image pixels were converted to bounding box coordinates and class probabilities and can be optimized end-to-end directly. This allowed to directly predict boxes in a single feed-forward pass without reusing any component of the neural network or generating proposals of any kind, thus speeding up the detector.

They started by dividing the image into a $S \times S$ grid and assuming $B$ bounding boxes per grid. Each cell containing the center of an object instance is responsible for the detection of that object. Each bounding box predicts 4 coordinates, objectness and class probabilities. This reframed the object detection as a regression problem. To have a receptive field cover that covers the whole image they included a fully connected layer in their design towards the end of the network.

SSD: Liu et al. [221], inspired by the Faster-RCNN architecture, used reference boxes of various sizes and aspect ratios to predict object instances (Figure 5) but they completely got rid of the region proposal stage (discussed in the following Section). They were able to do this by making the whole network work as a regressor as well as a classifier. During training, thousands of default boxes corresponding to different anchors on different feature maps learned to discriminate between objects and background. They also learned to directly localize and predict class probabilities for the object instances. This was achieved with the help of a multitask loss. Since, during inference time a lot of boxes try to localize the objects, generally a post-processing step like Greedy NMS is required to suppress duplicate detections.

In order to accommodate objects of all the sizes they added additional convolutional layers to the backbone and used them, instead of a single feature map, to improve the performance. This method was later applied to approaches related to two-stage detectors too [215].

Pros and Cons: Oftentimes single stage detectors do not give as good performance as the double-stage ones, but they are a lot faster [140] although some double-stages detectors can be faster than single-stages due to architectural tricks and modern single-stage detectors outperform the older multi-stages pipelines.
The various advantages of YOLO strategy are that it is extremely fast, with 45 to 150 frames per second. It sees the entire image as opposed to region proposal based strategies which is helpful for encoding contextual information and it learns generalizable representations of objects. But it also has some obvious disadvantages. Since each grid cell has only two bounding boxes, it can only predict at most two objects in a grid cell. This is particularly inefficient strategy for small objects. It struggles to precisely localize some objects as compared to two stages. Another drawback of YOLO is that it uses coarse feature map at a single scale only.

To address these issues, SSD used a dense set of boxes and considered predictions from various feature maps instead of one. It improved upon the performance of YOLO. But since it has to sample from these dense set of detections at test time it gives lower performance on MS COCO dataset as compared to two-stage detectors. The two-stage object detectors get a sparse set of proposals on which they have to perform predictions.

**Further improvements:** Redmon and Farhadi [306] and Redmon and Farhadi [307] suggested a lot of small changes in versions 2 and 3 of the YOLO method. The changes like applying batch normalization, using higher resolution input images, removing the fully connected layer and making it fully convolutional, clustering box dimensions, location prediction and multi-scale training helped to improve performance while a custom network (DarkNet) helped to improve speed.

Many further developments by many researchers have been proposed on Single Shot MultiBox Detector. The major advancements over the years have been illustrated in Figure 6. Deconvolutional Single Shot Detector (DSSD) [92], instead of the element-wise sum, used a deconvolutional module to increase the resolution of top layers and added each layer, through element-wise products to previous layer. Rainbow SSD [156] proposed to concatenate features of shallow layers to top layers by max-pooling as well as features of top layers to shallow layers through deconvolution operation. The final fused information increased from few hundreds to 2,816 channels per feature map. RUN [192] proposed a 3-way residual block to combine adjacent layers before final prediction. Cao et al. [29] used concatenation modules and element-sum modules to add contextual information in a slightly different manner. Zheng et al. [471] slightly tweak DSSD by fusing lesser number of layers and adding extra ConvNets to improve speed as well as performance.

They all improved upon the performance of conventional SSD and they lie within a small range among themselves on Pascal VOC 2012 test set [88], but they added considerable amount of computational costs, thus making it little slower. WeaveNet [51] aimed at reducing computational costs by gradually sharing the information from adjacent scales in an iterative manner. They hypothesized that by weaving the information iteratively, sufficient multi-scale context information can be transferred and integrated to current scale.

Recently three strong candidates have emerged
Figure 6: Evolution of single stage detectors over the years. Major advancements in chronological order are YOLO [308], SSD [221], DSSD [92], RetinaNet [216], RefineDet [460] and CornerNet [189]. Cuboid boxes, solid rectangular box, dashed rectangular boxes and arrows represent convolutional layer, fully connected layer, predictions and flow of features respectively. obj, cls and reg stand for objectness, classification and regression losses. Best viewed in color.
for replacing the undying YOLO and SSD variants:

- RetinaNet [216] borrowed the FPN structure but in a single stage setting. It is similar in spirit to SSD but it deserves its own paragraph given its growing popularity based on its speed and performance. The main new advance of this pipeline is the focal loss, which we will discuss in Section 2.2.1.

- RefineDet [460] tried to combine the advantages of double-staged methods and single-stage methods by incorporating two new modules in the single stage classic architecture. The first one, the ARM (Anchor Refinement modules), is used in multiple staged detectors’ fashion to reduce the search space and also to iteratively refine the localization of the detections. The ODM (Object Detection Module) took the output of the ARM to output fine-grained classification and further improve the localization.

- CornerNet [189] offered a new approach for object detection by predicting bounding boxes as paired top-left and bottom right keypoints. They also demonstrated that one can get rid of the prominent anchors step while gaining accuracy and precision. They used fully convolutional networks to produce independent score heat maps for both corners for each class in addition to learning an embedding for each corner. The embedding similarities were then used to group them into multiple bounding boxes. It beat its two (less original) competing rivals on COCO.

However, most methods used in competitions until now are predominantly double-staged methods because their structure is better suited for fine-grained classification. It is what we are going to see in the next Section.

2.1.3 Double Stage Detectors

The process of detecting objects can be split into two parts: proposing regions & classifying and regressing bounding boxes. The purpose of the proposal generator is to present the classifier with class-agnostic rectangular boxes which try to locate the ground-truth instances. The classifier, then, tries to assign a class to each of the proposals and further fine-tune the coordinates of the boxes.

**Region proposal:** Hosang et al. [131] presented an in-depth review of ten "non-data driven" object proposal methods including Objectness [2, 3], CPMC [30, 31], Endres and Hoiem [81, 82], Selective Search [391, 393], Rahtu et al. [293], Randomized Prim [229], Bing [56], MCG [286], Rantanenila et al. [298], Humayun et al. [145] and EdgeBoxes [491] and evaluated their effect on the detector’s performance. Also, Xiao et al. [425] developed a novel distance metric for grouping two super-pixels in high-complexity scenarios. Out of all these approaches Selective Search and EdgeBoxes gave the best recall and speed. The former is an order of magnitude slower than Fast R-CNN while the latter, which is not as efficient, took as much time as a detector. The bottleneck lied in the region proposal part of the pipeline.

Deep learning based approaches [86, 363] had also been used to propose regions but they were not end-to-end trainable for detection and required input images to be of fixed size. In order to address strong localization bias [46] proposed a box-refinement method based on the super-pixel tightness distribution. DeepMask [282] and SharpMask [284] proposed segmentation based object proposals with very deep networks. [162] estimated the objectness of image patches by comparing them with exemplar regions from prior data and finding the ones that are most similar to it.

The next obvious question became apparent. How can deep learning methods be streamlined into existing approaches to give an elegant, simple, end-to-end trainable and fully convolutional model? In the discussion that follows we will discuss two widely adopted approaches in two-stage detectors, pros and cons of using such approaches and further improvements made on them.
**R-CNN and Fast R-CNN:** The first modern ubiquitous double-staged deep learning detection method is certainly [105]. Although it is has now been abandoned due to faster alternatives it is worth mentioning to better understand the next paragraphs. Closer to the traditional non deep-learning methods the first stage of the method is the detection of objects in pictures to reduce the number of false positive of the subsequent stage. It is done using a hierarchical pixel grouping algorithm widely popular in the 2000s called selective search [393]. Once the search space has been properly narrowed, all regions above a certain score are warped to a fixed size so that a classifier can be applied on top of it. Further fine-tuning on the last layers of the classifier is necessary on the classes of the dataset used (they replace the last layer so that it has the right number of classes) and an SVM is used on top of the fixed fine-tuned features to further refine the localization of the detection. This method was the precursor of all the modern deep learning double-staged methods, in spite of the fact that this first iteration of the method was far from the elegant paradigm used nowadays. Fast R-CNN Girshick [104] from the same author started to refine R-CNN by being one of the first researcher with He et al. [116] to come-up with his own deep-learning detection building block. This differentiable mechanism called RoI-pooling (Region of Interest Pooling) was used for resizing fixed regions (also extracted with selective-search) coming not from the image directly but from the feature computed on the full image, which kept the spatial layout of the original image. Not only did that bring speed-up to the slow R-CNN method (x200 in inference) but it also came with a net gain in performances (around 6 points in mAP).

**Faster-RCNN:** The seminal Faster-RCNN paper [309] showed that the same backbone architecture used in Fast R-CNN for classification can be used to generate proposals as well. They proposed an efficient fully convolutional data driven based approach for proposing regions called Region Proposal Network (RPN). RPN learned the "objectness" of all instances and accumulated the proposals to be used by the detector part of the backbone. The detector further classified and refined bounding boxes around those proposals. RPN and detector can be trained separately as well as in a combined manner. When sharing convolutional layers with the detector they result in very little extra cost for region proposals. Since it has two parts for generating proposals and detection, it comes under the category of two-stage detectors.

Faster-RCNN used thousands of reference boxes, commonly known as anchors. Anchors formed a grid of boxes that act as starting points for regressing bounding boxes. These anchors were then trained end-to-end to regress to the ground truth and an objectness score was calculated per anchor. The density, size and aspect ratio of anchors are decided according to the general range of size of object instances expected in the dataset and the receptive field of the associated neuron in the feature map.

RoI Pooling, introduced in [104], warped the proposals generated by the RPN to fixed size vectors for feeding to the detection sub-network as its inputs. The quantization and rounding operation defining the pooling cells introduced misalignments and actually hurt localization.

**R-FCN:** To avoid running the costly RoI-wise subnetwork in Faster-RCNN hundreds of times, i.e., once per proposal, Dai et al. [62] got rid of it and shared the convolutional network end to end. To achieve this they proposed the idea of position sensitive feature maps. In this approach each feature map was responsible for outputting score for a specific part, like top-left, center, bottom right, etc., of the target class. The parts were identified with RoI-Pooling cells which were distributed alongside each part-specific feature map. Final scores were obtained by average voting every part of the RoI from the respective filter. This implementation trick introduced some more translational variance to structures that were essentially translation-invariant by construction. Translational variance
Figure 7: Graphical explanation of RoIPooling, RoIWarping and RoIAlign (actual dimensions of pooled feature map differ). The red box is the predicted output of Region Proposal Network (RPN) and the dashed blue grid is the feature map from which proposals are extracted. (a) RoI Pooling first aligns the proposal to the feature map (first quantization) and then max-pools or average-pools the features (second quantization). Note that some information is lost because the quantized proposal is not an integral multiple of the final map’s dimensions. (b) RoI Warping retains the first quantization but deals with second one through bilinear interpolation, calculated from four nearest features, through sampling \( N \) points (black dots) for each cell of final map. (c) RoI Align removed both the quantizations by directly sampling \( N \) points on the original proposal. \( N \) is set to four generally. Best viewed in color.

in object detection can be beneficial for learning localization representations. Although this pipeline seems to be more precise, it is not always better performance-wise than its Faster R-CNN counterpart.

From an engineering point of view, this method of Position sensitive RoI-Pooling (PS Pooling) also prevented the loss of information at RoI Pooling stage in Faster-RCNN. It improved the overall inference time speed of two-stage detectors but performed slightly worse.

**Pros and Cons:** RPNs are generally configured to generate nearly 300 proposals to get state-of-the-art performances. Since each of the proposal passed through a head of convolutional layers and fully connected layers to classify the objects and fine tune the bounding boxes, it decreased the overall speed. Although they are slow and not suited to real-time applications, the ideas based on these approaches give one the best performances in the challenging COCO detection task. Another drawback is that Ren et al. [309] and Dai et al. [62] used coarse feature maps at a single scale only. This is not sufficient when objects of diverse sizes are present in the dataset.

**Further improvements:** Many improvements have been suggested on the above methodologies concerning speed, performance and computational efficiency.

DeepBox [184] proposed a light weight generic objectness system by capturing semantic properties. It helped in reducing the burden of localization on the detector as the number of classes increased. Light-head R-CNN [206] proposed a smaller detection head and thin feature maps to speed up two-stage detectors. Singh et al. [346] brought R-FCN to 30 fps by sharing position sensitive feature maps across classes. Using slight architectural changes, they were also able to bring the number of classes predicted by R-FCN to 3000
Figure 8: Evolution of double stage detectors over the years. Major advancements in chronological order are R-CNN [105], SPPNet [116], Fast-RCNN [104], Faster RCNN [309], RFCN [62], FPN [215], Mask RCNN [118], Deformable CNN [63] (only the modification is shown and not the entire network). The main idea is marked in dashed blue rectangle wherever possible. Other information is same as in Figure 6.
without losing too much speed.

Several improvements have been made to RoI-Pooling. The spatial transformer of [154] used a differentiable re-sampling grid using bilinear interpolation and can be used in any detection pipeline. Chen et al. [37] used this for Face detection, where faces were warped to fit canonical poses. Dai et al. [61] proposed another type of pooling called RoI Warping based on bilinear interpolation. Ma et al. [228] were the first to introduce a rotated RoI-Pooling working with oriented regions (More on oriented RoI-Pooling can be found in Section 3.1.2). Mask R-CNN [118] proposed RoI Align to address the problem of misalignment in RoI Pooling which used bilinear interpolation to calculate the value of four regularly sampled locations on each cell. It allowed to fix to some extents the alignment between the computed features and the regions they were extracted from. It brought consistent improvements to all Faster R-CNN baselines on COCO. A comparison is shown in Figure 7. Recently, Jiang et al. [158] introduced a Precise RoI Pooling based on interpolating not just 4 spatial locations but a dense region, which allowed full differentiability with no misalignments.

Li et al. [196], Yu et al. [442] also used contextual information and aspect ratios while StuffNet [24] trained for segmenting amorphous categories such as ground and water for the same purpose. Chen and Gupta [48] made use of memory to take advantage of context in detecting objects. Li et al. [207] incorporated Global Context Module (GCM) to utilize contextual information and Row-Column Max Pooling (RCM Pooling) to better extract scores from the final feature map as compared to the R-FCN method.

Deformable R-FCN [63] brought flexibility to the fixed geometric transformations at the Position sensitive RoI-Pooling stage of R-FCN by learning additional offsets for each spatial sampling location using a different network branch in addition to other tricks discussed in Section 2.1.5. Lin et al. [215] proposed to use a network with multiple final feature maps with different coarseness to adapt to objects of various sizes. Zagoruyko et al. [446] used skip connections with the same motivation. Mask-RCNN [118] in addition to RoI-align added a branch in parallel to the classification and bounding box regression for optimizing the segmentation loss. Additional training for segmentation led to an improvement in the performance of object detection task as well. The advancements of two stage detectors over the years is illustrated in Figure 8.

The double-staged methods have now by far attained supremacy over best performing object detection DCNNs. However, for certain applications two-stage methods are not enough to get rid of all the false positives.

2.1.4 Cascades

Traditional one-class object detection pipelines resorted to boosting like approaches for improving the performance where uncorrelated weak classifiers (better than random chance but not too correlated with the true predictions) are combined to form a strong classifier. With modern CNNs, as the classifiers are quite strong, the attractiveness of those methods has plummeted. However, for some specific problems where there are still too many false positives, researchers still find it useful. Furthermore, if the weak CNNs used are very shallow it can also sometimes increase the overall speed of the method.

One of the first ideas that were developed was to cascade multiple CNNs. Li et al. [198] and Yang and Nevatia [434] both used a three-staged approach by chaining three CNNs for face detection. The former approach scanned the image using a $12 \times 12$ patch CNN to reject 90% of the non-face regions in a coarse manner. The remaining detections were offset by a second CNN and given as input to a $24 \times 24$ CNN that continued rejecting false positives and refining regressions. The final candidates were then passed on to a $48 \times 48$ classification network which output the final score. The latter approach created separate score maps for different resolutions using the same FCN on different scales of the test image (image pyramid). These score maps were then up-sampled to the same resolution and added to create a final score map, which
was then used to select proposals. Proposals were then passed to the second stage where two different verification CNNs, trained on hard examples, eradicated the remaining false positives. The first one being a four-layer FCN trained from scratch and the second one an AlexNet [181] pre-trained on ImageNet.

All the approaches mentioned in the last paragraph are ad hoc: the CNNs are independent of each other, there is no overall design, therefore, they could benefit from integrating the elegant zooming module that is the RoI-Pooling. The RoI-Pooling can act like a glue to pass the detections from one network to the other, while doing the down-sampling operation locally. Dai et al. [61] used a Mask R-CNN like structure that first proposed bounding boxes, then predicted a mask and used a third stage to perform fine grained discrimination on masked regions that are RoI-Pooled a second time.

Ouyang et al. [268], Wang et al. [404] optimized in an end-to-end manner a Faster R-CNN with multiple stages of RoI-Pooling. Each stage accepted only the highest scored proposals from the previous stage and added more context and/or localized the detection better. Then additional information about context was used to do fine grained discrimination between hard negatives and true positives in [268], for example. On the contrary, Zhang et al. [455] showed that for pedestrian detection RoI-Pooling, too coarse a feature map actually hurts the result. This problem has been alleviated by the use of feature pyramid networks with higher resolution feature maps. Therefore, they used the RPN proposals of a Faster R-CNN in a boosting pipeline involving a forest (Tang et al. [373] acted similarly for small vehicle detection).

Yang et al. [431], aware of the problem raised by Zhang et al. [455], used RoI-Pooling on multiple scaled feature maps of all the layers of the network. The classification function on each layer was learned using the weak classifiers of AdaBoost and then approximated using a fully connected neural network. While all the mentioned pipelines are hard cascades where the different classifiers are independent, it is sometimes possible to use a soft cascade where the final score is a linear weighted combination of the scores given by the different weak classifiers like in Angelova et al. [6]. They used 200 stages (instead of 2000 stages in their baseline with AdaBoost [16]) to keep recall high enough while improving precision. To save computations that would be otherwise unmanageable, they terminated the computations of the weighted sum whenever the score for a certain number of classifiers fell under a specified threshold (there are, therefore, as many thresholds to learn as there are classifiers). These thresholds are then really important because they control the trade-off between speed, recall and precision.

All the previous works in this Section involved a small fixed number of localization refinement steps, which might cause proposals to be not perfectly aligned with the ground truth, which in turn might impact the accuracy. That is why lots of work proposed iterative bounding box regression (while loop on localization refinement until condition is reached). Najibi et al. [254], Rajaram et al. [295] started with a regularly spaced grid of sparse pyramid boxes (only 200 non-overlapping in Najibi et al. [254] whereas, Rajaram et al. [295] used all Faster R-CNN anchors on the grid) that were iteratively pushed towards the ground truth according to the feature representation obtained from RoI-Pooling the current region. An interesting finding was that even if the goal was to use as many refinement steps as necessary if the seed boxes or anchors span the space appropriately, regressing the boxes only twice can in fact be sufficient [254]. Approaches proposed by Gidaris and Komodakis [101] and Li et al. [199] can also be viewed, internally, as iterative regression based methods proposing regions for detectors, such as Fast R-CNN.

Recently, Cai and Vasconcelos [27] noticed that when increasing the IoU threshold for a window to be considered positive in the training (to get better quality hypothesis for the next stages), one loses a lot of positive windows. Thus one has to keep using the low 0.5 threshold to prevent overfitting and thus one gets bad quality hypothesis in the next stages. This is true for all the works mentioned in this section that are based on Faster R-
CNN (e.g. [268, 404]). To combat this effect, they slowly increase the IoU threshold over the stages to get different sets of detectors using the latest stage proposals as input distribution for the next one. With only 3 to 4 stages they consistently improve the quality of a wide range of detectors with an average of 3 points gained w.r.t. the non-cascaded version. This algorithmic advance is used in most of the winning entries of the 2018 COCO challenge (used at least by the first three teams).

Orthogonal to this approach Jiang et al. [158] frames the regression of the multi-stage cascade as an optimization problem thus introducing a proxy for a smooth measure of confidence of the bounding box localization. This article among others will be discussed in more details in the Section 2.2.1.

Boosting and multistage (> 2) methods, we have seen previously, exhibit very different possible combinations of DCNNs. But we thought it would be interesting to still have a Section for a special kind of method that was hinted at in the previous Sections, namely the part-based models, if not for their performances at least for their historical importance.

### 2.1.5 Parts-Based Models

Before the reign of CNN methods, the algorithms based on Deformable Parts-based Model (DPM) and HoG features used to win all the object detection competitions. In this algorithm latent (not supervised) object parts were discovered for each class and optimized by minimizing the deformations of the full objects (connections were modeled by springs forces). The whole thing was built on a HoG image pyramid.

When Region based DCNNs started to beat the former champion, researchers began to wonder if it was only a matter of using better features. If this was the case then the region based approach would not necessarily be a more powerful algorithm. The DPM was flexible enough to integrate the newer more discriminative CNN features. Therefore, some research works focused in this research direction.

In 2014, Savalle and Tsogkas [325] tried to get the best of both worlds: they replaced the HoG feature pyramids used in the DPM with the CNN layers. Surprisingly, the performance they obtained, even if far superior to the DPM+HoG baseline, was considerably worse than the R-CNN method. The authors suspected the reason for it was the fixed size aspect ratios used in the DPM together with the training strategy. Girshick et al. [106] put more thought on how to mix CNN and DPM by coming up with the distance transform pooling thus bringing the new DPM (DeepPyramidDPM) to the level of R-CNN (even slightly better). Ranjan et al. [297] built on it and introduced a normalization layer that forced each scale-specific feature map to have the same activation intensities. They also implemented a new procedure of sampling optimal targets by using the closest root filter in the pyramid in terms of dimensions. This allowed them to further mimic the HOG-DPM strengths. Simultaneously, Wan et al. [401] also improved the DeepPyramidDPM but failed short compared to the newest version of R-CNN, fine-tuned (R-CNN FT). Therefore, in 2015 it seemed that the DPM based approaches have hit a dead end and that the community should focus on R-CNN type methods.

However, the flexibility of the RoI-Pooling of Fast R-CNN was going to help making the two approaches come together. Ouyang et al. [267] combined Fast R-CNN to get rid of most backgrounds and a DeepID-Net, which introduced a max-pooling penalized by the deformation of the parts called def-pooling. The combination improved over the state-of-the-art. As we mentioned in Section 2.1.3, Dai et al. [63] built on R-FCN and added deformations in the Position Sensitive RoI-Pooling: an offset is learned from the class Position Sensitive pooled tensor with a fully connected network for each cell of the RoI-Pooling thus creating ”parts” like features. This trick of moving RoI cells around is also present in [247], although slightly different because it is closer to the original DPM. Dai et al. [63] even added offsets to convolutional filters cells on Conv-5, which became doable thanks to bilinear interpolation. It, thus, became a truly deformable fully convolutional...
network. However, Mordan et al. [247] got better performances on VOC without it. Several works used deformable R-FCN like [429] for aerial imagery that used a different training strategy. However, even if it is still present in famous competitions like COCO, it is less used than its counterparts with fixed RoI-Pooling. It might come back though thanks to recent best performing models like [345] that used [63] as their baseline and selectively back-propagated gradients according to the object size.

2.2 Model Training

The next important aspect of the detection model’s design is the losses being used to converge the huge number of weights and the hyper-parameters that must be conducive to this convergence. Optimizing for a wrongfully crafted loss may actually lead the model to diverge instead. Choosing incorrect hyper-parameters, on the one hand, can stagnate the model, trap it in a local optima or, on the other hand, over-fit the training data (causing poor generalizations). Since DCNNs are mostly trained with mini-batch SGD (see for instance [190]), we focus the following discussion on losses and on the optimization tricks necessary to attain convergence. We also review the contribution of pre-training on some other dataset and data augmentation techniques which bring about an excellent initialization point and good generalizations respectively.

2.2.1 Losses

Multi-variate cross entropy loss, or log loss, is generally used throughout the literature to classify images or regions in the context of detectors. However, detecting objects in large images comes with its own set of specific challenges: regress bounding boxes to get precise localization, which is a hard problem that is not present at all in classification and an imbalance between target object regions and background regions.

A binary cross entropy loss is formulated as shown in Eq. 1. It is used for learning the combined objectness. All instances, \( y \), are marked as positive labels with a value one. This equation constrains the network to output the predicted confidence score, \( p \), to be 1 if it thinks there is an object and 0 otherwise.

\[
CE(p, y) = \begin{cases} 
-\log(p) & \text{if } y = 1 \\
-\log(1 - p) & \text{otherwise} 
\end{cases} 
\] (1)

A multi-variate version of the log loss is used for classification (Eq. 2). \( p_{o,c} \) predicts the probability of observation \( o \) being class \( c \) where \( c \in \{1,\ldots,C\} \). \( y_{o,c} \) is 1 if observation \( o \) belongs to class \( c \) and 0 otherwise. \( c = 0 \) is accounted for the special case of background class.

\[
MCE(p, y) = -\sum_{c=0}^{C} y_{o,c} \log(p_{o,c}) 
\] (2)

Fast-RCNN [104] used a multitask loss (Eq. 3) which is the de-facto equation used for classifying as well as regressing. The losses are summed over all the regions proposals or default reference boxes, \( i \). The ground-truth label, \( p^*_i \), is 1 if the proposal box is positive, otherwise 0. Regularization is learned only for positive proposal boxes.

\[
L\left(\{p_i\}, \{t_i\}\right) = \frac{1}{N_{cls}} \sum_i L_{cls}(p_i, p^*_{i}) + \frac{\lambda}{N_{reg}} \sum_i p^*_i L_{reg}(t_i, t^*_{i}) 
\] (3)

where \( t_i \) is a vector representing the 4 coordinates of the predicted bounding box and similarly \( t^*_i \) represents the 4 coordinates of the ground truth. Eq. 4 presents the equation for exact parameterized coordinates. \( \{x_a, y_a, w_a, h_a\} \) are the center x and y coordinates, width and height of the default anchor box respectively. Similarly \( \{x^*_a, y^*_a, w^*_a, h^*_a\} \) are ground truths and \( \{x, y, w, h\} \) are the coordinates to be predicted. The two terms are normalized by mini-batch size, \( N_{cls} \), and number of proposals/default reference boxes, \( N_{reg} \), and weighted by
Since in recent detection papers also use \( L \) where \( \{ \)\( i \)\( , r , t , h \}\) came up with a binary logistic type regression loss. After dividing the image patch into \( M \) columns and \( M \) rows, they computed the probability of each row and column being inside or outside \( \{ \)the predicted observation or not \( \) (Eq. 7). \( L_{\text{border}} = \sum_{s \in \{ l , r , t , h \}} \sum_{i = 1}^{M} \lambda^{+} T_{s,i} \log(p_{s,i}) + \lambda^{-} (1 - T_{s,i}) \log(1 - p_{s,i}) \)) (7)

\( \forall i \in \{ 1 , ..., M \} , \) \( T_{s,i} = \begin{cases} 1 , & \text{if } i = B_{s}^{gt} \\ 0 , & \text{otherwise} \end{cases} \)

where \( \lambda^{-} = \frac{M^{+}}{M} , \) \( \lambda^{+} = (M - 1) \lambda^{-} \). The notations can be inferred from Eq. 6. In the second paper [101], related to the same topic, applied the regression losses iteratively at the region proposal stage in a class agnostic manner. They used final convolutional features and predictions from last iteration to further refine the proposals.

It was also found out to be beneficial to optimize the loss directly over Intersection over Union (IoU) which is the standard practice to evaluate a bounding box or segmentation algorithm. Yu et al. [441] presented Eq. 8 for regression loss.

\( L_{\text{unit-box}} = - \ln(\text{IoU}(\text{gt}, \text{pred})) \) (8)

The terms are self-explanatory. Jiang et al. [158] also learned to predict IoU between predicted box and ground truth. They made a case to use localization confidence instead of classification confidence to suppress boxes at NMS stage. It gave higher recall on MS COCO dataset. This loss is however very unstable and has a number of regions where the IoU has zero-gradient and thus it is undefined. Tychsen-Smith and Petersson [390] adapt this loss to make it more stable by adding hard bounds, which prevent the function from diverging. They also factorize the score function by adding a fitness term representing the IoU of the box w.r.t. the ground truth.

**Losses for class imbalance:** Since in recent detectors there are a lot of anchors which most of the time cover background, there is a class imbalance between positive and negative anchors. An alternative is Online Hard Example Mining (OHEM). Shrivastava et al. [337] performed to select only
worst performing examples (so-called hard examples) for calculating gradients. Even if by fixing the ratio between positive and negative instances, generally 1:3, one can partly solve this imbalance. Lin et al. [216] proposed a tweak to the cross entropy loss, called focal loss, which took into account all the anchors but penalized easy examples less and hard examples more. Focal loss (Eq. 9) was found to increase the performance by 3.2 mAP points on MS COCO, in comparison to OHEM on a ResNet-50-FPN backbone and 600 pixel image scale.

\[
FL(p, y) = \begin{cases} 
-\alpha_t(1-p)^\gamma \log(p) & \text{if } y = 1 \\
-\alpha_t p^\gamma \log(1-p) & \text{otherwise}
\end{cases}
\] (9)

One can also adopt simpler strategies like rebalancing the cross-entropy by putting more weights on the minority class [259].

**Supplementary losses:** In addition to classification and regression losses, some papers also optimized extra losses in parallel. Dai et al. [61] proposed a three-stage cascade for differentiating instances, estimating masks and categorizing objects. Because of this they achieved competitive performance on object detection task too. They further experimented with a five-stage cascade also. Uber-Net [173] trained on as many as six other tasks in parallel with object detection. He et al. [118] have shown that using an additional segmentation loss by adding an extra branch to the Faster R-CNN detection sub-network can also improve detection performance. Li et al. [203] introduced position-sensitive inside/outside score maps to train for detection and segmentation simultaneously. Wang et al. [409] proposed an additional repulsion loss between predicted bounding boxes in order to have one final prediction per ground truth. Generally, it can be observed, *instance segmentation* in particular, aids the object detection task.

### 2.2.2 Hyper-Parameters

The detection problem is a highly non-convex problem in hundreds of thousands of dimensions. Even for classification where the structure is much simpler, no general strategy has emerged yet on how to use mini-batch gradient descent correctly. Different popular versions of mini-batch Stochastic Gradient Descent (SGD) [318] have been proposed based on a combination of momentum, to accelerate convergence, and using the history of the past gradients, to dampen the oscillations when reaching a minimum: AdaDelta [447], RMSProp [378] and the unavoidable ADAM [171, 304] are only the most well-known. However, in object detection literature authors, use either plain SGD or ADAM, without putting too much thought into it. The most important hyper-parameters remain the learning rate and the batch size.

**Learning rate:** There is no concrete way to decide the learning rate policy over the period of the training. It depends on a myriad of factors like optimizer, number of training examples, model, batch size, etc. We cannot quantify the effect of each factor; Therefore, the current way to determine the policy is by hit-and-trial. What works for one setting may or may not work for other settings. If the policy is incorrect then the model might fail to converge at all. Nevertheless, some papers have studied it and have established general guidelines that have been found to work better than others. A large learning rate might never converge while a small learning rate gives sub-optimal results. Since, in the initial stage of training the change in weights is dramatic, Goyal et al. [111] have proposed a *Linear Gradient Warmup* strategy in which learning rate is increased every iteration during this period. Then starting from a point (for e.g. $10^{-3}$) the policy was to decrease learning rate over many epochs. Krizhevsky [180] and Goyal et al. [111] also used a *Linear Scaling Rule* which linearly scaled the learning rate according to the mini-batch size.

**Batch size:** The object detection literature doesn’t generally focus on the effects of using a bigger or smaller batch size during training. Training modern detectors requires working on full images and therefore on large tensors which can be trou-
blesome to store on the GPU RAM. It has forced the community to use small batches, of 1 to 16 images, for training (16 in RetinaNet [216] and Mask R-CNN [118] with the latest GPUs).

One obvious advantage of increasing the batch size is that it reduces the training time but since the memory constraint restricts the number of images, more GPUs have to be employed. However, using extra large batches have been shown to potentially lead to big improvements in performances or speed. For instance, batch normalization [152] needs many images to provide meaningful statistics. Originally batch size effects were studied by [111] on ImageNet dataset. They were able to show that by increasing the batch size from 256 to 8192, train time can be reduced from 29 hours to just 1 hour while maintaining the same accuracy. Further, You et al. [437] and Akiba et al. [1] brought down the training time to below 15 minutes by increasing the batch size to 32k.

Very recently, MegDet [274], inspired from [111], have shown that by averaging gradients on many GPUs to get an equivalent batch size of 256 and adjusting the learning rates could lead to some performance gains. It is hard to say now which strategy will eventually win in the long term but they have shown that it is worth exploring.

2.2.3 Pre-Training

Transfer learning was first shown to be useful in a supervised learning approach by Girshick et al. [105]. The idea is to fine-tune from a model already trained on a dataset that is similar to the target dataset. This is usually a better starting point for training instead of randomly initializing weights. For e.g. model pre-trained on ImageNet being used for training on MS COCO. And since, COCO dataset’s classes is a superset of PASCAL VOC’s classes most of the state-of-the-art approaches pre-train on COCO before training it on PASCAL VOC. If the dataset at hand is completely unrelated to dataset used for pre-training, it might not be useful. For e.g. model pre-trained on ImageNet being used for detecting cars in aerial images.

Singh and Davis [345] made a compelling case for the minimum difference in scales of object instances between classification dataset used for pre-training and detection dataset to minimize domain shift while fine-tuning. They asked should we pre-train CNNs on low resolution classification dataset or restrict the scale of object instances in a range by training on an image pyramid? By minimizing scale variance they were able to get better results than the methods that employed scale invariant detector. The problem with the second approach is some instances are so small that in order to bring them in the scale range, the images have to be up-scaled so much that they might not fit in the memory or they will not be used for training at all. Using a pyramid of images and using each for inference is also slower than methods that use input image exactly once.

Section 3.1.3 covers pre-training and other aspects of it like fine-tuning and beyond in great detail. There are only, to the best of our knowledge, two articles that tried to match the performances of ImageNet pre-training by training detectors from scratch. The first one being [331] that used deep supervision (dense access to gradients for all layers) and very recently [332] that adaptively recalibrated supervision intensities based on input object sizes.

2.2.4 Data Augmentation

The aim of augmenting the train set images is to create diversity, avoid overfitting, increase amount of data, improve generalizability and overcome different kinds of variances. This can easily be achieved without any extra annotations efforts by manually designing many augmentation strategies. The general practices that are followed include and are not limited to scale, resize, translation, rotation, vertical and horizontal flipping, elastic distortions, random cropping and contrast, color, hue, brightness, saturation and sharpness adjustments etc. The two recent and promising but not widely adapted techniques are Cutout [70] and Sample Pairing [149]. Taylor and Nitschke [376] benchmarked various popular data augmentation schemes to know the ones that are most appro-
The modification done for each image is mentioned below the figure. Best viewed in color.

Although there are many techniques available and each one of them is easy to implement, it is difficult to know in advance, without expert knowledge, which techniques assist the performance for a target dataset. For example, vertical flipping in case of traffic signs dataset is not helpful because one is not likely to encounter inverted signs in the test set. It is not trivial to select the approaches for each target dataset and test all of them before deploying a model. Therefore, Cubuk et al. [60] proposed a search algorithm based on reinforcement learning to find the best augmentation policy. Their approach tried to find the best suitable augmentation operations along with their magnitude and probability of happening. Smart Augmentation [194] worked by creating a network that learned how to automatically generate augmented data during the training process of a target network in a way that reduced the loss. Tran et al. [382] proposed a Bayesian approach, where new annotated training points are treated as missing variables and generated based on the distribution learned from the training set. Devries and Taylor [69] applied simple transformations such as adding noise, interpolating, or extrapolating between data points. They performed the transformation, not in input space, but in a learned feature space. All the above approaches are implemented in the domain of classification only but they might be beneficial for the detection task as well and it would be interesting to test them.

Generative adversarial networks (GANs) have also been used to generate the augmented data directly for classification without searching for the best policies explicitly [7, 251, 280, 348]. Ratner et al. [300] used GANs to describe data augmentation strategies. GAN approaches may not be as effective for detection scenarios yet because generating an image with many object instances placed in a relevant background is much more difficult than generating an image with just one dominant object. This is also an interesting problem which might be addressed in the near future and is explored in Section 3.2.2.
Figure 10: An illustration of the inference stage. In this example, bounding boxes around horses (blue) and persons (pink) are obtained from the detector (along with the confidence scores mentioned on top of each box). (a) NMS chooses the most confident box and suppresses all other boxes having an IoU greater than a threshold. Note, it sometimes leads to suppression of boxes around other occluded objects. (b) Soft-NMS deals with this situation by reducing the confidence scores of boxes instead of completely suppressing them.

2.3 Inference

The behavior of the modern detectors is to pick up pixels of target objects, propose as many windows as possible surrounding those pixels and estimate confidence scores for each of the window. It does not aim to suggest one box exactly per object. Since all the reference boxes act independently during test time and similar input pixels are picked up by many neighboring anchors, each positive prediction in the prediction set highly overlaps with other boxes. If the best ones out of these are not selected, it will lead to many double detections and thus false positives. The ideal result would be to predict exactly one prediction box per ground-truth object that has high overlap with it. To reach near this ideal state, some sort of post-processing needs to be done.

Greedy Non-maximum suppression (NMS) [64] is the most frequent technique used for inference modules to suppress double detections through hard thresholding. In this approach, the prediction box with the highest confidence was chosen and all the boxes having an Intersection over Union (IoU) higher than a threshold, $N_t$, were suppressed or rescored to zero. This step was made iteratively till all the boxes were covered. Because of its nature there is no single threshold that works best for all datasets. Datasets with just one object per image will trivially apply NMS by choosing only the highest-ranking box. Generally, datasets with sparse and fewer number of objects per image (2 to 3 objects) require a lower threshold. While datasets with cramped and higher numbers of objects per image (7 and above) give better results with a higher threshold. The problems that arose with this naive and hand-crafted approach was that it may completely suppress nearby or occluded true positive detections, choose top scoring box which might not be the best localized one and its inability to suppress false positives with insufficient overlap.

To improve upon it, many approaches have been proposed but most of them work for a very special case such as pedestrians in highly occluded scenarios. We discuss the various directions they take and the approaches that work better than the greedy NMS in the general scenario. Most of the following discussion is based on [132] and [21], who, in their papers, provided us with an in-depth view of the alternate strategies being used.

Many clustering approaches for predicted boxes have been proposed. A few of them are mean shift clustering [64, 413], agglomerative clustering [22], affinity propagation clustering [250], heuristic variants [327], etc. Rothe et al. [315] presented a learn-
ing based method which "passes messages between windows" or clustered the final detections to finally select exemplars from each cluster. Mrowca et al. [250] deployed a multi-class version of this paper. Clustering formulations with globally optimal solutions have been proposed in [371]. All of them worked for special cases but are less consistent than Greedy NMS, generally.

Some papers learn NMS in a convolutional network. Henderson and Ferrari [121] and Wan et al. [401] tried to incorporate NMS procedure at training time. Stewart et al. [357] generated a sparse set of detections by training an LSTM. Hosang et al. [130] trained the network to find different optimal cutoff thresholds \(N_t\) locally. Hosang et al. [132] took one step further and got rid of the NMS step completely by taking into account double detections in the loss and jointly processed neighboring detections. The former inclined the network to predict one detection per object and the latter provided with the information if an object was detected multiple times. Their approach worked better than greedy NMS and they obtained a performance gain of 0.8 mAP on COCO dataset.

Most recently, greedy NMS was improved upon by Bodla et al. [21]. Instead of setting the score of neighboring detections as zero they decreased the detection confidence as an increasing function of overlap. It improved the performance by 1.1 mAP for COCO dataset. There was no extra training required and since it is hand-crafted it can be easily integrated in object detection pipeline. It is used in current top entries for MS COCO object detection challenge.

Jiang et al. [158] performed NMS based on separately predicted localization confidence instead of usually accepted classification confidence. Other papers rescored detections locally [38, 386] or globally [397]. Some others detected objects in pairs in order to handle occlusions [266, 321, 370]. Rodriguez et al. [312] made use of the crowd density. Quadratic unconstrained binary optimization (QUBO) [317] used detection scores as a unary potential and overlap between detections as a pairwise potential to obtain the optimal subset of detection boxes. Niepert et al. [257] saw overlapping windows as edges in a graph.

As a bonus, in the end, we also throw some light on the inference "tricks" that are generally known to the experts participating in the competitions. The tricks that are used to further improve the evaluation metrics are: Doing multi-scale inference on an image pyramid (see Section 3.1.1 for training); Doing inference on the original image and on its horizontal flip (or on different rotated versions of the image if the application domain does not have a fixed direction) and aggregating results with NMS; Doing bounding box voting as in [102] using the score of each box as its weight; Using heavy backbones, as observed in the backbone section; Finally, averaging the predictions of different models in ensembles. For the last trick often it is better to not necessarily use the top-N best performing models but to prefer instead uncorrelated models so that they can correct each other’s weaknesses. Ensembles of models are outperforming single models by often a large margin and one can average as many as a dozen models to outrank its competitors. Furthermore, with DCNNs generally one does not need to put too much thought on normalizing the models as each one gives bounded probabilities (because of the softmax operator in the last layer).

### 2.4 Concluding Remarks

This concludes a general overview of the landscape of the mainstream object detection halfway through 2018. Although the methods presented are all different, it has been shown that in fact most papers have converged towards the same crucial design choices. All pipelines are now fully convolutional, which brings structure (regularization), simplicity, speed and elegance to the detectors. The anchors mechanism of Ren et al. [309] has now also been widely adopted and has not really been challenged yet, although iteratively regressing a set of seed boxes show some promise [101, 254]. The need to use multi-scale information from different layers of the CNN is now apparent [174, 215, 216]. The RoI-Pooling module and its cousins can also be cited as one of the main architectural advances of recent years but might not ultimately be used
Figure 11: An illustration of challenges in object detection. To detect all instances of the class "fork" (yellow bounding boxes) from the COCO dataset [214], a detector should be able to handle small objects (lower middle picture) as well as big objects (third column photograph). It needs to be scale invariant as well as being rotation invariant (all forks have different orientation in the pictures). It should also manage occlusions as in the left-hand side photograph. After being trained on the pictures in the first three columns, detection algorithms are expected to generalize to the “cartoon” image on the right (domain adaptation).

3 Going Forward in Object Detection

While we demonstrated that object detection has already been turned upside-down by CNN architectures and that nowadays most methods revolve around the same architectural ideas, the field has not yet reached a status quo, far from it. Completely new ideas and paradigms are being developed and explored as we write this survey, shaping the future of object detection. This section lists the major challenges that remain mostly unsolved and the attempts to get around them using such ideas. To have an idea of number of papers being published targeting each challenge, we ran a corresponding query on advanced search of Google Scholar. The exact query is mentioned below each figure respectively (see Figure 12, 13, 14, 15 and 16). We report these numbers from year 2011 to 2018. We note that this method doesn’t give the exact number of papers targeting each challenge but still gives us a rough idea of the interest of the community in each challenge. We couldn’t use this for the localization challenge because almost all object detection papers mention localization even if they are not targeting to solve it.

3.1 Major Challenges

There are some walls that the current models cannot overcome without heavy structural changes, we list these challenges in Figure 11.

Often, when we hear that object recognition is solved, we argue that the existence of these walls are solid proof that it is not. Although we have advanced the field, we cannot rely indefinitely on the current DCNNs. This section shows how the recent literature addressed these topics.
In the past three years a lot of approaches have been proposed to deal with the challenge of scale variance. On the one hand, object instances in the image may fill only 0.01% to 0.25% of the pixels, and, on the other hand, the instance may fill 80% to 90% of the whole image. It is tough to make a single feature map predict all the objects, with this huge variance, because of the limited receptive field that it’s neurons have. Particularly small objects (discussed in Section 3.1.6) are difficult to classify and localize. In this section we will discuss three main approaches that are used to tackle the challenge of scale variance.

First, is to make image pyramids [90, 104, 116, 327]. This helps enlarge small objects and shrink the large objects. Although the variance is reduced to an extent but each image has to be pass forwarded multiple times thus, making it computationally expensive and slower than the approaches discussed in the following discussion. This approach is different from data augmentation techniques [60] where an image is randomly cropped, zoomed in or out, rotated etc. and used exactly once for inference. Ren et al. [310] extracted feature maps from a frozen network at different image scales and merged them using maxout [110]. Singh and Davis [345] selectively back-propagated the gradients of object instances if they fall in a predetermined size range. This way, small objects must be scaled up to be considered for training. They named their technique Scale Normalization for Image Pyramids (SNIP).

Second, a set of default reference boxes, with varied size and aspect ratios that cover the whole image uniformly, were used. Ren et al. [309] proposed a set of reference boxes at each sliding window location which are trained to regress and classify. If an anchor box has a significant overlap with the ground truth it is treated as positive otherwise, it is ignored or treated as negative. Due to the huge density of anchors most of them are negative. This leads to an imbalance in the positive and negative examples. To overcome it OHEM [337] or Focal Loss [216] are generally applied at training time. One more downside of anchors is that their design has to be adapted according to the object sizes in the dataset. If large anchors are used with too many small objects then, and vice versa, then they won’t be able to train as efficiently. Default reference boxes are an important design feature in double stage [62] as well as single-stage methods [221, 306]. Most of the top winning entries [63, 118, 215, 216] use them in their models. Bodla et al. [21] helped by improving the suppression technique of double detections, generated from the dense set of reference boxes, at inference time.

Third, multiple convolutional layers were used for bounding box predictions. Since a single feature map was not enough to predict objects of varied sizes, SSD [221] added more feature maps to the original classification backbones. Cai et al. [28] proposed regions as well as performed detections on multiple scales in a two-stage detector. Najibi et al. [255] used this method to achieve state-of-the-art on a face dataset [433] and Li et al. [200] on pedestrian dataset [87]. Yang et al. [431] used all the layers to reject easy negatives and then performed scale-dependent pooling on the remaining proposals. Shallower or finer layers are deemed...
to be better for detecting small objects while top or coarser layers are better at detecting bigger objects. In the original design, all the layers predict the boxes independently and no information from other layers is combined or merged. Many papers, then, tried to fuse different layers [51, 192] or added additional top-down network [338, 415]. They have already been discussed in Section 2.1.1.

Fourth, Dilated Convolutions (a.k.a. atrous convolutions) [438] were deployed to increase the filter’s stride. This helped increase the receptive field size and, thus, incorporate larger context without additional computations. Obviously smaller receptive fields are also needed if the objects are small and thus only a clever combination of larger receptive field with atrous convolutions and smaller ones like in ASPP [42] (Atrous Spatial Pyramid Pooling) can lead to a successful scale invariance in detection. It has been successfully applied in the context of object detection [62] and semantic segmentation [42]. Dai et al. [63] presented a generalized version of it by learning the deformation offsets additionally.

3.1.2 Rotational Variance

In the real world object instances are not necessarily present in an upright manner but can be found at an angle or even inverted. While it is hard to define rotation for flexible objects like a cat, a pose definition would be more appropriate, it is much easier to define it for texts or objects in aerial images which have an expected rigid shape. It is well known that CNNs as they are now do not have the ability to deal with the rotational variance of the data. More often than not, this problem is circumvented by using data augmentation: showing the network slightly rotated versions of each patch. When training on full images with multiple annotations it becomes less practical. Furthermore, like for occlusions, this might work but it is disappointing as one could imagine incorporating rotational invariance into the structure of the network.

Building rotational invariance can be simply done by using oriented bounding boxes in the region proposal step of modern detectors. Jiang et al. [159] used Faster R-CNN features to predict oriented bounding boxes, their straightened versions were then passed on to the classifier. More elegantly, few works like [26, 119, 228] proposed to construct different kinds of RoI-pooling module for oriented bounding boxes. Ma et al. [228] transformed the RoI-Pooling layer of Faster R-CNN by rotating the region inside the detector to make it fit the usual horizontal grid, which brought an astonishing increase of performances from the 38.7% of regular Faster R-CNN to 71.8% with additional tricks on MSRA. Similarly, He et al. [119] used a rotated version of the recently introduced RoI-Align to pool oriented proposals to get more discriminative features (better aligned with the text direction) that will be used in the text recognition parts. Busta et al. [26] also used rotated pooling by bilinear interpolation to extract oriented features to recognize text after having rendered YOLO to be able to predict rotated bounding boxes. Shi et al. [333] detected in the same way, oriented bounding boxes (called segments) with a similar architecture but differ from [26, 119, 228] because it also learned to merge the oriented segments appropriately, if they cover the same word or sentence, which allowed greater flexibility.

Liu and Jin [222] needed slightly more complicated anchors: quadrangles anchors, and regressed

Figure 13: Number of papers published each year for challenge of rotational variance. Query used in Google Scholar: (“rotational variance” OR “rotational invariance” OR “rotational invariant”) AND “object detection”.
compact text zones in a single-stage architecture similar to Faster R-CNN’s RPN. This system being more flexible than the previous ones, necessitated more parameters. They used Monte-Carlo simulations to compute overlaps between quadrangles. Liao et al. [212] directly rotated convolution filters inside the SSD framework, which effectively rendered the network rotation-invariant for a finite set of rotations (which is generalized in the recent [410] for segmentation). However, in the case of text detection even oriented bounding boxes can be insufficient to cover text with a layout with too much curvature and one often sees the same failure cases in different articles (circle-shaped texts for instance).

A different kind of approach for translation invariance was taken by the two following works of Cheng et al. [52] and Laptev et al. [188] that made use of metric-learning. Former proposed an original approach of using metric learning to force features of an image and its rotated versions to be close to each other hence, somehow invariant to rotations. In a somewhat related approach the latter found a canonical pose for different rotated versions of an image and used a differentiable transformation to make every example canonical and to pool the same features.

The difficulty of predicting oriented bounding boxes is alleviated if one resorts to semantic segmentation like in [466]. They learned to output semantic segmentation then oriented bounding boxes were found based on the output score map. However, it shares the same downsizes as other approaches [26, 119, 212, 222, 228] for text detection because in the end one still has to fit oriented rectangles to evaluate the performances.

Other applications than text detection also require rotation invariance. In the domain of aerial imagery, the recently released DOTA [421] is one of the first datasets of its kind expecting oriented bounding boxes for predictions. One can anticipate an avalanche of papers trying to use text detection techniques like [372], where the SSD framework is used to regress bounding box angles or the former metric learning technique from Cheng et al. [52] and Cheng et al. [54]. For face detection, paper like [335] relied on oriented proposals too. The diversity of the methods show that no real standard has emerged yet. Even the most sophisticated detection pipelines are only rotation invariant to a certain extent.

The detectors presented in this section do not yet have the same popularity as the vertical ones because all the main datasets like COCO do not present rotated images. One could define a rotated-COCO or rotated-VOC to evaluate the benefit these pipelines could bring over their vertical versions but it is obviously difficult and would not be accepted as is by the community without a strong, well-thought-evaluation protocol.

### 3.1.3 Domain Adaptation

It is often needed to repurpose a detector trained on domain A to function on domain B. In most cases this is because the dataset in domain A has lots of training examples and the categories in it are generic whereas the dataset in domain B has less training examples and objects that are very specific or distinct from A. There are surprisingly very few recent articles that tackled explicit domain adaptation in the context of object detection –[361, 368, 428] did it for HOG based features – even though the literature for domain adaptation...
for classification is dense, as shown by the recent survey of Csurka [59]. For instance when one trains a Faster R-CNN on COCO and want to test it off-the-shelf on the car images of KITTI Geiger et al. [98] (‘car’ is one of the 80 classes of COCO) one gets only 56.1% AP w.r.t. 83.7% using more similar images because of the differences between the domains (see [383]).

Most works adapt the features learned in another domain (mostly classification) by simply fine-tuning the weights on the task at hand. Since [105], literally every state-of-the-art detectors are pre-trained on ImageNet or on an even bigger dataset. This is the case even for relatively large object detection datasets like COCO. There is no fundamental reason for it to be a requirement. The objects of the target domains have to be similar and of the same scales as the objects on which the network was pre-trained as pointed out by Singh and Davis [345], that detected small cars in aerial imagery by first pre-training on ImageNet. The seminal work of Hoffman et al. [127], already evoked in the weakly supervised Section 4.2.1, showed how to transfer a good classifier trained on large scale image datasets to a good detector trained on few images by fine-tuning the first layers of a convnet trained on classification and adapting the final layer using nearest neighbor classes. Hinterstoisser et al. [125] demonstrated another example of transfer learning where they froze the first layers of detectors trained on synthetic data and fine-tuned only the last layers on the target task.

We discuss below all the articles we found that go farther than simple transfer learning for domain adaptation for object detection. Raj et al. [294] aligned features subspace from different domains for each class using Principal Component Analysis (PCA). Chen et al. [49] used H-divergence theory and adversarial training to bridge the distribution mismatches. All the mentioned articles worked on adapting the features. Thanks to GANs some of them are trying to adapt directly to the image [150], which used CycleGAN from [480] to convert images directly from one domain to the other. The object detection community needs to evolve if we want to move beyond transfer-learning.

One of the end goals of domain adaptation would be to be able to learn a model on synthetic data, which is available (almost) for free and to have it performing well on real images. Pepik et al. [279] was, to the best of our knowledge, the first to point out that, even though CNNs are texture sensitive, wire-framed and CAD models used in addition to real data can improve the performances of detectors. Peng et al. [277] augmented PASCAL-VOC data with 3D CAD models of the objects found in PASCAL-VOC (planes, horses, potted plants, etc.) and then rendered them in backgrounds where they are likely to be found and improved overall detection performances. Following this line, several authors introduced synthetic data for various tasks such as i) persons: Varol et al. [395] ii) furniture: Massa et al. [236] created rendered CAD furnitures on real backgrounds by using grayscale images to avoid color artifacts and improved the detection performances on the IKEA dataset. iii) text: Gupta et al. [112] created an oriented text detection benchmark by superimposing synthetic text to existing scenes while respecting geometric and uniformity constraints and showed better results on ICDAR iv) logos: Su et al. [359] did the same without any constraints by superimposing transparent logos to existing images.

Georgakis et al. [99] synthesized new instances of 3D CAD models by copy pasting rendered objects on surface normals, very close to [296], which used Blender to put instances of objects inside a refrigerator. Later Dwibedi et al. [79] with the same approach but without respecting any global consistency shown promise. For them only local consistency is important for modern object detectors. Similar to [99], they used different kinds of blending to make the detector robust to the pasting artifacts (more details can be found in [78]). More recently, Dvornik et al. [77] extended [79] by first finding locations in images with high likelihood of object presence before pasting objects. Another recent approach [383] found that domain randomization when creating synthetic data is vital to train detectors: training on Virtual KITTI Gaidon et al. [94], a dataset that was built to be close to KITTI (in terms of aspects, textures, vehicles and bounding
boxes statistics), is not sufficient to be state-of-the-art on KITTI. One can gain almost one point of AP when building his own version of Virtual KITTI by introducing more randomness than was present in the original in the form of random textures and backgrounds, random camera angles and random flying distractor objects. Randomness was apparently absent from KITTI but is beneficial for the detector to gain generalization capabilities.

Several authors have shown interest in proposing tools for generating artificial images at a large scale. Qiu and Yuille [291] created the open-source plug-in UnrealCV for a popular game engine Unreal Engine 4 and showed applications to deep network algorithms. Tian et al. [377] used the graphical model CityEngine to generate a synthetic city according to the layout of existing cities and added cars, trucks and buses to it using a game engine (Unity3D). The detectors trained on KITTI and this dataset are again better than just with KITTI. Alhaija et al. [4] pushed Blender to its limits to generate almost real-looking 3D CAD cars with environment maps and pasted them inside different 2D/3D environments including KITTI, Virtual KITTI (and even Flickr). It is worth noting that some datasets included real images to better simulate the scene viewed by a robot in active vision settings, as in [5].

Another strategy is to render simple artificial images and increase the realism of the images in a second iteration, using Generative Adversarial Networks [339]. RenderGAN was used to directly generate realistic training images [348]. We refer the reader to the section on GANs (Section 3.2.2) for more information on the use of GANs for style transfer.

We have seen that for the time being synthetic datasets can augment existing ones but not totally replace them for object detection, however, the domain shift between synthetic data and the target distribution is still too large to rely on synthetic data only.

### 3.1.4 Object Localization

Accurate localization remains one of the two biggest sources of error [129] in fully supervised object detection. It mainly originates from small objects and more stringent evaluation protocol applied in the latest datasets. The predicted boxes are required to have an IoU of up to 0.95 with the ground-truth boxes. Generally, localization is dealt by using smooth L1 or L2 losses along with classification loss. Some papers proposed a more detailed methodology to overcome this issue. Also, annotating bounding boxes for each and every object is expensive. We will also look into some methods that localize objects using only weakly annotated images.

Kong et al. [174] overcame the poor localization because of coarseness of the feature maps by aggregating hierarchical feature maps and then compressing them into a uniform space. It provided an efficient combination framework for deep but semantic, intermediate but complementary, and shallow but high-resolution CNN features. Chen et al. [46] proposed multi-thresholding straddling expansion (MTSE) to reduce localization bias and refine boxes during proposal time which is based on super-pixel tightness as opposed to objectness based models. Zhang et al. [465] addressed the localization problem by using a search algorithm based on Bayesian optimization that sequentially proposed candidate regions for an object bounding box. Hosang et al. [132] tried to integrate NMS in the convolutional network which in the end improved localization.

Many papers [101, 158] also try to adapt the loss function to address the localization problem. Gidaris and Komodakis [103] proposed to assign conditional probabilities to each row and column of a sample region, using a neural convolutional network adapted for this task. These probabilities allow more accurate inference of the object bounding box under a simple probabilistic framework. Since Intersection over Union (IoU) is used in the evaluation strategies of many detection challenges, Yu et al. [441] and Jiang et al. [158] optimized over IoU directly. The loss-based papers have been dis-
3.1.5 Occlusions

The occlusions lead to partial missing information from object instances. They may be occluded due to the background or other object instances. Less information naturally leads to harder examples and inaccurate localizations. The occlusions happen all the time in real-life images. However, since deep learning is based on convoluting filters and that occlusions by definition introduce parasite patterns most modern methods are not robust to it by construction.

Training with occluded objects help for sure [244] but it is often not doable because of a lack of data and furthermore, it cannot be bulletproof. Wu et al. [419] managed to learn an And-Or model for cars by dynamic programming, where the And stood for the decomposition of the objects into parts and the Or for all different configurations of parts (including occluded configurations). The learning was only possible thanks to the heavy use of synthetic data to model every possible type of occlusion. Another way to generate examples of occlusions is to directly learn to mask the proposals of Fast R-CNN [407].

For dense pedestrians crowds deformable models and parts can help improve detection accuracy (see 2.1.5) e.g. if some parts are masked some others will not be, therefore, the average score is diminished but not made zero like in [106, 265, 325]. Parts are also useful for occlusion handling in face detection where different CNNs can be trained on different facial parts [432]. The survey already tackled Deformable RoI-Pooling (RoI-Pooling with parts) [247]. Another way of re-introducing parts in modern pipelines is the deformable kernels of [63]. They presented a way to alleviate the occlusion problems by giving more flexibility to the usually fixed geometric structures.

Building special kinds of regression losses for bounding boxes acknowledging the proximity of each detection (which is reminiscent of the springs in the old part-based models) was done in [409]. They, in addition, to the attraction term in the traditional regression loss that pushes predictions towards their assigned ground truth added a repulsion term that pushed predictions away from each other.

Traditional non-maximum suppression causes a lot of problems with occlusions because overlapping boxes are suppressed. Hence, if one object is in front of another only one is detected. To address this, Hosang et al. [132] offered to learn non-maximum suppression making it continuous (and differentiable) and Bodla et al. [21] used a soft version that only degraded the score of the overlapping objects (more details can be found about various...
3.1.6 Detecting Small Objects

Detecting small objects is harder than detecting medium sized and large sized objects because of less information associated with them, easier possibility of confusion with the background, higher precision requirement for localization, large image size, etc. In COCO metrics evaluation, objects occupying areas lesser than and equal to $32 \times 32$ pixels come under this category and this size threshold is generally accepted within the community for datasets related to common objects. Datasets related to aerial images [421], traffic signs [490], faces [253], pedestrians [84] or logos [358] are generally abundant with small object instances.

In case of objects like logos or traffic signs, objects have an expected shape, size and aspect ratio of the objects to be detected, and this information can be embedded to bias the deep learning model. This strategy is much harder and not feasible for common objects as they are a lot more diverse. As an illustration, the winner of the COCO challenge 2017 [274], which used many of the latest techniques and ensemble of four detectors reported a performance of 34.5\% mAP on small objects and 64.9\% mAP on large objects. The following entries reported even a greater dip for smaller objects than the larger ones. Pham et al. [281] have presented an evaluation, focusing on real-time small object detection, of three state-of-the-art models, YOLO, SSD and Faster R-CNN with related trade-off between accuracy, execution time and resource constraints.

There are different ways to tackle this problem, such as: i) up-scaling the images ii) shallow networks, iii) contextual information, iv) super-resolution. These four directions are discussed in the following.

The first – and most trivial direction – consists in up-scaling the image before detection. But a naive upscaling is not efficient as the large images become too large to fit into a GPU for training. Gao et al. [95], first, down-sampled the image and then used reinforcement learning to train attention-based models to dynamically search for the interesting regions in the image. The selected regions are then studied at higher resolution and can be used to predict smaller objects. This avoided the need of analyzing each pixel of the image with equal attention and saved some computational costs. Some papers [62, 63, 345] used image pyramids during training time in the context of object detection while [310] used it during inference time.

The second direction is to use shallow networks. Small objects are easier to predict by detectors which have smaller receptive field. The deeper networks with their large receptive field tend to lose
some information about the small objects in their coarser layers. Sommer et al. [259, 351] proposed very shallow networks with less than 5 convolutional layers and three fully connected layers for the purpose of detecting objects in aerial imagery. Such type of detectors are useful when the expected instances are only of type small. But if expected instances are of diverse size it is more beneficial to use finer feature maps of very deep networks for small objects and coarser feature maps for larger objects. We have already discussed this approach in Section 3.1.1. Please refer to Section 4.2.4 for more low power and shallow detectors.

The third direction is to make use of context surrounding the small object instances. Gidaris and Komodakis [102], Zhu et al. [489] used context to improve the performance but Chen et al. [36] used context specifically for improving the performance for small objects. They augmented the R-CNN with the context patch in parallel to the proposal patch generated from region proposal network. Zagoruyko et al. [446] combined their approach of making the information flow through multiple paths with DeepMask object proposals [282, 284] to gain a massive improvement in the performance for small objects. Context can also be used by fusing coarser layers of the network with finer layers [215, 216, 338]. Context related literature has been covered in Section 3.2.3 in detail.

Finally, the last direction is to use Generative Adversarial Networks to selectively increase the resolution of small objects, as proposed by Li et al. [201]. Its generator learned to enhance the poor representations of the small objects to super-resolved ones that are similar enough to real large objects to fool a competing discriminator. Table 1 summarizes the past subsection by grouping the articles by main idea and target challenge. We find it very useful to see which ideas have been thoroughly investigated by the literature and which are under-explored.

3.2 Complementary New Ideas in Object Detection

In this subsection we review ideas which haven’t quite matured yet but we feel could bring major breakthroughs in the near future. If we want the field to advance, we should embrace new grand ideas like these, even if that means completely rethinking all the architectural ideas evoked in Section 2.

3.2.1 Graph Networks

The dramatic failings of state-of-the-art detectors on perturbed versions of the COCO validation sets, spotted by Rosenfeld et al. [314], are raising questions for better understanding of compositionality, context and relationships in detectors.

Battaglia et al. [11] recently wrote a position article arguing about the need to introduce more representational power into Deep Learning using graph networks. It means finding new ways to enforce the learning of graph structures of connected

Figure 17: Good detections of persons are marked in green and bad detections in red. Helpful context in blue (the presence of mirror frames) can help lower the score of a box. The relationships (in Fuchsia) between bounding boxes can also help: a person cannot be present twice in a picture. Enhancing parts of the picture using SR (Figure 18) is yet another way to better make a decision. All those "reasoning" modules are not included in the mainstream detectors.
Table 1: Summary of the main ideas found in the literature to account for the limitations of the current deep learning architectures. For each idea we list the papers that implement it and the challenges they (sometimes only partially) address.
entities instead of outputting independent predictions. Convolutions are too local and translation equivariant to reflect the intricate structure of objects in their context.

One embodiment of this idea in the realm of detection can be found in the work of Wang et al. [406], where long-distance dependencies were introduced in deep-learning architectures. These combined local and non-local interactions are reminiscent of the CRF [185], which sparked a renewed interest for graphical models in 2001. Dot products between features determine their influences on each other, the closest they are in the feature space, the stronger their interactions will be (using a Gaussian kernel for instance). This seems to go against the very principles of DCNNs, which are, by nature, local. However this kind of layer can be integrated seamlessly in any DCNN to its benefit, it is very similar to self-attention [55]. It is not clear yet if these new networks will replace their local counterparts in the long-term but they are definitely suitable candidates.

Graph structures also emerge when one needs to incorporate a priori (or inductive biases) on the spatial relationships of the objects to detect (relational reasoning) [135]. The relation module uses attention to learn object dependencies, also using dot products of features. Similarly, Wang et al. [406] incorporated geometrical features to further disambiguate relationships between objects. One of the advantages of this pipeline is the last relation module, which is used to remove duplicates similarly to the usual NMS step but adaptively. We mention this article in particular because although relationships between detected objects have been used in the literature before, it was the first attempt to have it as a differentiable module inside a CNN architecture.

### 3.2.2 Adversarial Trainings

No one in the computer vision community was spared by the amazing successes of the Generative Adversarial Networks [109]. By pitting a con-artist (a CNN) against a judge (another CNN) one can learn to generate images from a target distribution up to an impressive degree of realism. This new tool keeps the flexibility of the regular CNN architectures as it is implemented using the same bricks and therefore, it can be added in any detection pipeline.

Even if [407] does not belong to the GAN family per se, the adversarial training it uses: dropping pixels in examples to make them harder to classify and hence, render the network robust to occlusions, obviously drew its inspiration from GANs. Ouyang et al. [269] went a step further and used the GAN formalism to learn to generate pedestrians from white noise in large images and showed how those created examples were beneficial for the training of object detectors. There are numerous recent papers, e.g., [23, 276], proposing approaches for converting synthetic data towards more realistic images for classification. Inoue et al. [150] used the latest CycleGAN [480] to convert real images to cartoons and by doing so gained free annotations to train detectors on weakly labeled images and became the first work to use GANs to create full images for detectors. As stated in the introduction, GANs can also be used, not in a standalone manner but, directly embedded inside a detector too: Li et al. [201] operated at the feature level by adapting the features of small objects to match features obtained with well resolved objects. Bai et al. [9] trained a generator directly for super-resolution of small objects patches using traditional GAN loss in addition to classification losses and MSE loss per pixel. Integrating the module in modern pipelines brought improvement to the original mAP on COCO, this very simple pipeline is summarized Figure 18. These two articles addressed the detection of small objects, which will be tackled in more details in Section 3.1.6.

Shen et al. [330] used GANs to completely replace the Multiple Instance Learning paradigm (see Section 4.2.1) using the GAN framework to generate candidate boxes following the real distribution of the training images boxes and built a state-of-the-art detector that is faster than all the others by two orders of magnitude.

Thus, this extraordinary breakthrough is starting to produce interesting results in object detec-
tion and its importance is growing. Considering the latest result in the generation of synthetic data using GANs for instance the high resolution examples of [166] or the infinite image generators, BiCycleGAN from Zhu et al. [480] and MUNIT from Huang et al. [142], it seems the tsunami that started in 2014 will only get bigger in the years to come.

3.2.3 Use of Contextual Information

We will see in this section that the word context can mean a lot of different things but taking it into account gives rise to many new methods in object detection. Most of them (like spatial relationships or using stuff to find things) are often overlooked in competitions, arguably for bad reasons (too complex to implement in the time frame of the challenge).

Methods have evolved a lot since Heitz and Koller [120] used clustering of stuff/backgrounds to help detect objects in aerial imagery. Now, thanks to the CNN architectures, it is possible to do detection of things and stuff segmentation in parallel, both tasks helping the other [24].

Of course, this finding is not surprising. Certain objects are more likely to appear in certain stuff or environments (or context): thanks to our knowledge of the world, we find it weird to have a flying train: Katti et al. [167] showed that adding this human knowledge helps existing pipelines. The environments of the visual objects also comprise of other objects that they are present with, which advocates for learning spatial relationships between objects. Mrowca et al. [250] and Gupta et al. [113] independently used spatial relationships between proposals and classes (using WordNet hierarchy) to post-process detections. This is also the case in [492] where RNNs were used to model those relationships at different scales and in [48] where an external memory module was keeping track of the likelihood of objects being together. Hu et al. [135], that we mentioned in Section 3.2.1, went even further with a trainable relation module inside the structure of the network. In a different but not unrelated manner Gonzalez-Garcia et al. [108] improved the detection of parts of objects by associating parts with their root objects.

All multi-scale architectures use different sized context, as we saw in Section 2.1.1. Zeng et al. [450] used features from different sized regions (different contexts) in different layers of the CNN with message-passing in between features related to different context. Kong et al. [175] used skip connections and concatenation directly in the CNN architecture to extract multi-level and multi-scale information.

Sometimes, even the simplest local context surrounding a region of interest can help (see, for instance, the methods presented in Section 2.1.4, where the amount of context varies in between the classifiers). Extracted proposals can include variable amounts of pixels (context means size of the proposal) to help the classifiers such as in [268] or in [36, 101, 103]. Li et al. [196] included global image context in addition to regional context. Some approaches went as far as integrating all the image context: it was done for the first time in YOLO [308] with the addition of a fully connected layer on the last feature map. Wang et al. [406] modified the convolutional operator to put weights on every part of the image, helping the network use context outside the object to infer their existence. This use of global context is also found with the Global Context Module of the recent detection pipeline from Megvii [275]. Li et al. [207] proposed a fully connected layer on all the feature maps (similar to
Table 2: Summary of the approaches taken to exploit different types of context.

<table>
<thead>
<tr>
<th>Article references</th>
<th>Ideas</th>
<th>Type of Context</th>
</tr>
</thead>
<tbody>
<tr>
<td>24, 169</td>
<td>Segmenting Stuff, Using background cues</td>
<td>Background context (Environment)</td>
</tr>
<tr>
<td>[48, 113, 135, 250, 442, 492]</td>
<td>Likelihood of Objects being together/Inferring Relationships/Memory Modules</td>
<td>Other Objects Context</td>
</tr>
<tr>
<td>[108]</td>
<td>Finding Root to find parts</td>
<td>Parts-Objects Context</td>
</tr>
<tr>
<td>[175, 450]</td>
<td>Using different feature scales</td>
<td>Multi-scale Context</td>
</tr>
<tr>
<td>[36, 101, 163, 196, 208]</td>
<td>Adding variable sized context/Adding Borders of RoI</td>
<td>Surrounding Pixels</td>
</tr>
<tr>
<td>[207, 308, 406]</td>
<td>Adding connections to all pixels</td>
<td>Full Image Context</td>
</tr>
</tbody>
</table>

Redmon et al. [308]) with dilated kernels.

Other kinds of context can also be put to work. Yu et al. [442] used latent variables to decide on which context cues to use to predict the bounding boxes. It is not clear yet which method is the best to take context into account, another question is: do we want to? Even if the presence of an object in a context is unlikely, do we actually want to blind our detectors to unlikely situations? All the types of context that can be leveraged, have been summarized in Table 2.

3.3 Concluding Remarks

This section finished the tour of all the principal CNN based approaches past, present and future that treat general object detection in the traditional settings. It has allowed to peer through the armor of the CNN detectors and see them for what they are: impressive machines having amazing generalization capabilities but still powerless in a variety of cases, in which a trained human would have no problem (domain adaptation, occlusions, rotations, small objects) for an example of a difficult test case even for so-called robust detectors see Figure 17. Potential ideas to go past these obstacles have also been mentioned among them the use of adversarial training and context are the most prominent. The following section will go into more specific set-ups, less traditional problems or environments that will frame the detector abilities even further.

4 Extending Object Detection

Object detection may still feel like a narrow problem: one has a big training set of 2D images, huge resources (GPUs, TPUs, etc.) and wants to output 2D bounding boxes on a similar set of 2D images. However, these basic assumptions are often not present in practical scenarios. Firstly, because there exists many other modalities where one can perform object detection. These require conceptual changes in architectures to perform equally well. Secondly, sometimes one might be constrained to learn from exceedingly few fully annotated images, therefore, training a regular detector is either irrelevant or not an optimal choice because of overfitting. Also detectors are not built to be run in research labs alone but to be integrated into industrial products, which often come with an upper bound on energy consumption and speed requirements to satisfy the customer. The aim of the following discussion will be to know more about the research work done to extend the deep learning based object detection into new modalities and with tough constraints. It ends with reflections on what other interesting functionalities a strong detector in the future might possess.

4.1 Detecting Objects in Other Modalities

There are several modalities other than 2D images that can be interesting: videos, 3D point clouds, medical imaging, hyper-spectral imagery, etc. We will be discussing in this survey the former two. We did not treat for instance the volumetric images from the medical domain (MRI, etc.) or hyper-
spectral imagery, which are outside of the scope of this article and would deserve their own survey.

4.1.1 Object Detection in Videos

The upside of detecting objects in videos is that it provides additional temporal information but it also has unique challenges associated with it: motion blur, appearance changes, video defocus, pose variations, computational efficiency etc. It is a recent research domain due to the lack of large scale public datasets. One of the first video datasets is the ImageNet VID [319], proposed in 2015. This dataset as well as the recent datasets for object detection in video are mentioned in Section A.4.

One of the simplest ways to use temporal information for detecting object is the detection by tracking paradigm. As an example, Ray et al. [301] proposed a spatio-temporal detector of motion blobs, associated into tracks by a tracking algorithm. Each track is then interpreted as a moving object. Despite its simplicity, this type of algorithm is marginal in the literature as it is only interesting when the appearances of the objects are not available.

The most widely used approaches in the literature are those relying on tubelets. Tubelets have been introduced in the T-CNN approach of Kang et al. [163, 164]. T-CNN relied on 4 steps. First, still-image object detection (with Faster R-CNN like detectors) was performed. Second, multi-context suppression removed detection hypotheses having the lowest scores: highly ranked detection scores were treated as high-confidence classes and the rest were suppressed. Third, motion-guided propagation transferred detection results to adjacent frames to reduce false negatives. Fourth, temporal tubelet rescoring used a tracking algorithm to obtain sequences of bounding boxes, classified into positive and negative samples. Positive samples were mapped to a higher range, thus, increasing the score margins. T-CNN has several follow ups. The first was Seq-NMS [115] which constructed sequences along nearby high-confidence bounding boxes from consecutive frames, rescoring to the average confidence. Other boxes close to this sequence were suppressed. Another one was MCMOT [191] in which a post-processing stage, under the form of a multi-object tracker, was introduced, relying on hand-crafted rules (e.g., detector confidences, color/motion clues, changing point detection and forward-backward validation) to determine whether bounding boxes belonged to the tracked objects, and to further refine the tracking results. Tripathi et al. [385] exploited temporal information by training a recurrent neural network that took as input, sequences with predicted bounding boxes, and optimized an objective enforcing consistency across frames.

The most advanced pipeline for object detection in videos is certainly the approach of Feichtenhofer et al. [89], borrowing ideas from tubelets as well as from feature aggregation. The approach relies on a multitask objective loss, for frame-based object detection and across-frame track regression, correlating features that represented object co-occurrences across time and linking the frame level detections based on across-frame tracklets to produce the detections.

The literature on object detection in videos also addressed the question of computing time, since applying a detector on each frame can be time consuming. In general, it is non-trivial to transfer the state-of-the-art object detection networks to videos, as per-frame evaluation is slow. Deep feature flow [485, 486] ran the convolutional subnetwork only on sparse key frames, propagated deep feature maps to other frames via a flow field.
It led to significant speedup as flow computation is relatively fast. In the impression network [123] proposed to iteratively absorb sparsely extracted frame features, impression features being propagated all the way down the video which helped enhance features of low-quality frames. In the same way, the light flow of [487] is a very small network designed to aggregate features on key frames. For non-key frames, sparse feature propagation was performed, reaching a speed of 25.6 fps. Fast YOLO [329] came up with an optimized architecture that has 2.8X fewer parameters with just a 2% IOU drop, by applying a motion-adaptive inference method. Finally, [41] proposed to reallocate computational resources over a scale-time space: while expensive detection is done sparsely and propagated across both scales and time. Cheaper networks did the temporal propagation over a scale-time lattice.

An interesting question is "What can we expect from using temporal information?" The improvement of the mAP due to the direct use of temporal information can vary from +2.9% [484] to +5.6% [89]. Table 3 does a recap of this sub-subsection.

### 4.1.2 Object Detection in 3D Point Clouds

This section addresses the literature about object detection in 3D data, whether it is true 3D point clouds or 2D images augmented with depth data (RGBD images). These problems raise novel challenges, especially in the case of 3D point clouds for which the nature of the data is totally different (both in terms of structure and contained information). We can distinguish 4 main types of approaches depending on i) the use of 2D images and geometry, ii) the detections made in raw 3D point clouds, iii) the detections made in a 3D voxel grid and iv) the detections made in 2D after projecting the point cloud on a 2D plane. Most of the presented methods are evaluated on the KITTI benchmark [98]. Section A.3 introduces the datasets used for 3D object detection and quantitatively compares best methods on these datasets.

The methods belonging to the first category, monocular, start by the processing of RGB images and then add shape and geometric prior or occlusion patterns to infer 3D bounding boxes, as proposed by Chen et al. [44], Mousavian et al. [249] and Xiang et al. [424]. Deng and Latecki [68] revisited the amodal 3D detection by directly relating 2.5D visual appearance to 3D objects and proposed a 3D object detection system that simultaneously predicted 3D locations and orientations of objects in indoor scenes. Li et al. [197] represented the data in a 2D point map and used a single 2D end-to-end fully convolutional network to detect objects and predicted full 3D bounding boxes even while using a 2D convolutional network. Deep MANTA [34] is a robust convolutional network introduced for simultaneous vehicle detection, part localization, visibility characterization and 3D dimension estimation, from 2D images.

Among the methods using 3D point clouds directly, we can mention the series of papers relying on PointNet [288] and PointNet++ [290] networks, which are capable of dealing with the irregular format of point clouds without having to transform them into 3D voxel grids. F-PointNet [289] is a 3D detector operating on raw point clouds (RGBD scans). It leveraged mature 2D object detector to propose 2D object regions in RGB images and then collected all points within the frustum to form...
a frustum point cloud.

Voxel based methods such as VoxelNet [478] represented the irregular format of point clouds by fixed size 3D Voxel grids on which standard 3D convolution can be applied. Li [195] discretized the point cloud on square grids, and represented discretized data by a 4D array of fixed dimensions. Vote3Deep [83] examined the trade-off between accuracy and speed for different architectures applied on a voxelized representation of input data.

Regarding approaches based on birds eye view, MV3D [47] projected LiDAR point cloud to a birds eye view on which a 2D region proposal network is applied, allowing the generation of 3D bounding box proposals. In a similar way, LMNet [240] addressed the question of real-time object detection using 3D LiDAR by projecting the point cloud onto 5 different frontal planes. More recently, BirdNet [15] proposed an original cell encoding mechanisms for birds eye view, which is invariant to distance and differences on LiDAR devices resolution, as well as a detector taking this representation as input. One of the fastest methods (50 fps) is ComplexYOLO [342], which expanded YOLOv2 by a specific complex regression strategy to estimate multi-class 3D boxes in Cartesian space, after building a birds eye view of the data.

Some recent methods, such as [182], combined different sources of information (eg., birds eye view, RGB images, 3D voxels, etc.) and proposed an architecture performing multimodal feature fusion on high resolution feature maps. Ku et al. [182] is one of the top performing methods on KITTI benchmark [98]. Finally, it is worth mentioning the super-pixel based method by Srivastava et al. [356] allowed to discover novel objects in 3D point clouds. Using the same fashion as in the previous section we display a recap in Table 4.

### 4.2 Detecting Objects Under Constraints

In object detection, challenges arise not only because of the naturally expected problems (scale, rotation, localization, occlusions, etc.) but also due to the ones that are created artificially. The first motivation for the following discussion is to know and understand the research works that deal with the inadequacy of annotations in certain datasets. This inadequacy could be due to weak (image-level) labels, scarce bounding box annotations or no annotations at all for certain classes. The second motivation is to discuss the approaches dealing with hardware and application constraints, real-world detectors might encounter.

#### 4.2.1 Weakly Supervised Detection

Research teams want to include as many images as possible in their proposed datasets. Due to budget constraints or to save costs or for some other reasons, sometimes, they chose not to annotate precise bounding boxes around objects and include only image level annotations or captions. The object detection community has proven that it is still possible with enough weakly annotated data to train good object detectors.

The most obvious way to address Weakly Supervised Object Detection (WSOD) is to use the Multiple Instance Learning (MIL) framework [233]. The image is considered as being a bag of regions extracted by conventional object proposals:
at least one of these candidate regions is positive if the image has the appropriate weak label, if not, no region is positive. The classical formulation of the problem at hand (before CNNs) then becomes a latent-SVM on the region’s features where the latent part is the assignment of each proposal (that is weakly constrained by the image label). This problem being highly non-convex is heavily dependent on the quality of the initialization.

Song et al. [353, 354] thus focused on the initialization of the boxes by starting from selective-search proposals. They used for each proposal, its K-nearest neighbors in other images to construct a bipartite graph. The boxes were then pruned by taking only the patches that occur in most positive images (covering) while not belonging to the set of neighbors of regions found in negative images. They also applied Nesterov smoothing on the SVM objective to make the optimization easier. Of course, if proposals do not spin enough of the image some objects will not be detected and thus the performance will be bad as there is no re-localization.

The work of Sun et al. [362] also belongs to this category. Bilen et al. [19] added regularization to the smoothened optimization problem of Song et al. [353] using prior knowledge, but followed the same general directions. In another related research direction Wang et al. [402] learned to cluster the regions extracted with selective search into K-categories using unsupervised learning (pLSA) and then learned category selection using bag of words to determine the most discriminative clusters per class.

However, it is not always a requirement to explicitly solve the latent-SVM problem. Thanks to the fully convolutional structure of most CNNs it is sometimes possible to get a rough idea where an object might be while training for classification. For example, the arg-max of the produced spatial heat maps before global max-pooling is often located inside a bounding box as shown in [261, 262]. It is also possible to learn to detect objects without using any ground truth bounding boxes for training by masking regions of the image and see how the global classification score is impacted, as proposed by Bazzani et al. [12].

This free localization information can be improved through the use of different pooling strategies. For instance: producing a spatial heat map and using a global average pooling instead of global max pooling to train in classification. This strategy was used in [474] where the heat maps per class were thresholded to obtain bounding boxes. In this line of work, Pinheiro and Collobert [283] went a step further by producing pixel-level label segmentation maps using Log-Sum-Exp pooling in conjunction with some image and smoothing prior. Other pooling strategies involved aggregating minimum and maximum evidences to get a more precise idea where the object is and isn’t, e.g., as in the line developed in Durand et al. [74, 75, 76]. Bilen and Vedaldi [18] used the spatial pyramid pooling module to take MIL to the modern-age by incorporating it into a Fast R-CNN like architecture with a two-stream Fast R-CNN proposal classification part: one with classification score and the other with relative rankings of proposals that are merged together using hadamard products. Thus, producing region level labels predictions like in classic detection settings. They then aggregated all labels per image by taking the sum. They trained it end-to-end using image level labels thanks to their aggregation module while adding a spatial-regularization constraint on the features obtained by the SPP module.

Another idea, which can be combined with MIL is to draw the supervision from elsewhere. Tracked object proposals were used by Kumar Singh et al. [183] to extract pseudo-groundtruth to train detectors. This idea was further explored by Chen et al. [40] where the keywords extracted from the subtitles of documentaries allowed to further ground and cluster the generated annotations. In a similar way, Yuan et al. [443] used action description supervision via LSTMs. Cheap supervision can also be gained by involving user feedback [270], where the users iteratively improved the pseudo-ground truth by saying if the objects were missed or partly included in the detections. Click supervision by users, far less demanding than full annotations, also improved the performance of detectors [271]. [316] used active learning to select the
right images to annotate and thus get the same performance by using far fewer images. One can also leverage strong annotations for other classes to improve the performance of weakly supervised classes. This was done in [374] by using the powerful LSDA framework [127]. This was also the case in [128, 146, 311].

This year, a lot of interesting new works continued to develop the MIL+CNN framework using diverse approaches [97, 369, 400, 462–464]. These articles will not be treated in detail because the focus of this survey is object detection in general and not WSOD.

As of this writing, the state-of-the-art mAP on VOC2007 in WSOD is 47.6% [463]. The gap is being reduced at an exhilarating pace but we are still far from the 83.1% state-of-the-art with full annotations [247] (without COCO pre-training). We present a recap in Table 5.

### 4.2.2 Few-shot Detection

The cost of annotating thousands of boxes over hundreds of classes is too high. Although some large scale datasets are created, but it is not practical to do it for every single target domain. Collecting and annotating training examples in the case of video is even costlier than still images, making few shot detection more interesting. For this purpose, researchers have come up with ways to train the detectors with as low as three to five bounding boxes per target class and get lower but competitive performance as compared to the fully supervised approach on a large scale dataset. Few shot learning usually relies on semi-supervised learning mechanisms.

Dong et al. [73] took up an iterative approach to simultaneously train the model and generate new samples which are used in the following iterations for training. They observed that as the model becomes more discriminative it is able to sample harder as well as more number of instances. Iterating between multiple kinds of detectors was found to outperform the single detector approach. One interesting aspect of the paper is that their approach with only three to four annotations per class gives results comparable to weakly annotated approaches with image level annotations on the whole PASCAL VOC dataset. A similar approach was used by Keren et al. [168], who proposed a model which can be trained with as few as one single exemplar of an unseen class and a larger target example that may or may not contain an instance of the same class as the exemplar (weakly supervised learning). This model was able to simultaneously identify and localize instances of classes unseen at training time.

Another way to deal with few-shot detection is to fine-tune a detector trained on a sourced domain to a target domain for which only few samples are available. This is what Chen et al. [39] did, by introducing a novel regularization method, involving, depressing the background and transferring the knowledge from the source domain to the target domain to enhance the fine-tuned detector.

For videos, Misra et al. [243] proposed a semi-supervised framework in which some initial labeled boxes allowed to iteratively learn and label hundreds of thousands of object instances automatically. Criteria for reliable object detection and tracking constrained the semi-supervised learning process and minimized semantic drift.

<table>
<thead>
<tr>
<th>Article references</th>
<th>Implementation</th>
<th>Paradigm</th>
</tr>
</thead>
<tbody>
<tr>
<td>[19, 353, 362, 402]</td>
<td>Optimization Tricks (smoothing, EM, etc.)</td>
<td>Full MIL</td>
</tr>
<tr>
<td>[212]</td>
<td>Monitor score change</td>
<td>Masking</td>
</tr>
<tr>
<td>[18, 261, 262, 283, 474]</td>
<td>Global Pooling / GAPooling / LogSumExp Pooling</td>
<td>Refining Pooling</td>
</tr>
<tr>
<td>[74–76]</td>
<td>top-k max/min</td>
<td>Contradictory Evidence Pooling</td>
</tr>
<tr>
<td>[40, 128, 146, 183, 276, 311, 374, 443]</td>
<td>Subtitles / Motion Cues / User clicks / Strong Annotations</td>
<td>Auxiliary Supervision</td>
</tr>
</tbody>
</table>

Table 5: Summary of the weakly supervised approaches.
4.2.3 Zero-shot Detection

Zero-shot detection is useful for a system where a large number of classes are to be detected. It’s hard to annotate a large number of classes as the cost of annotation gets higher with more classes. This is a unique type of problem in the object detection domain as the aim is to classify and localize new categories, without any training examples, during test time with the constraint that the new categories are semantically related to the objects in the training classes. Therefore, in practice the semantic attributes are available for the unseen classes.

The challenges that come with this problem are:

First, zero-shot learning techniques are restricted to recognize a single dominant object and not all the object instances present in the image. Second, the background class during fully supervised training may contain objects from unseen classes. The detector will be trained to discriminatively treat these classes as background.

While there is a comparably large literature present for zero shot classification, well covered in the survey [93], zero shot detection has only a few papers to the best of our knowledge. Zhu et al. [482] proposed a method where semantic features are utilized during training but it is agnostic to semantic information during test time. This means they incorporated semantic attribute information in addition to seen classes during training and generated proposals only, but no identification label. for seen and unseen objects at test time. Rahman et al. [292] proposed a multitask loss that combines max-margin, useful for separating individual classes, and semantic clustering, useful for reducing noise in semantic vectors by positioning similar classes together and dissimilar classes far apart. They used ILSVRC [66] which contains an average of only three objects per image. They also proposed another method for a more general case when unseen classes are not predefined during training. Bansal et al. [10] proposed two background-aware approaches, statically assigning the background image regions into a single background class embedding and latent assignment based alternating algorithms which associated background to different classes belonging to a large open vocabulary, for this task. They used MSCOCO [214] and VisualGenome [179] which contain an average of 7.7 and 35 objects per image respectively. They also set number of unseen classes to be higher, making their task more complex than previous two papers.

Since, it is quite a new problem there is no well-defined experimental protocol for this approach. They vary in number and nature of unseen classes, use of semantic attribute information of unseen classes during training, complexity of the visual scene, etc.

4.2.4 Fast and Low Power Detection

There is generally a trade-off between performance and speed (we refer to the comprehensive study of [140] for instance). When one needs real time detectors, like for video object detection, one loses some precision. However, researchers have been constantly working on improving the precision of fast methods and making precise methods faster. Furthermore, not every setup can have powerful GPUs, so for most industrial applications the detectors have to run on CPUs or on different low power embedded devices like Raspberry-Pie.

Most real-time methods are single stage because they need to perform inference in a quasi fully constitutional manner. The most iconic methods have already been discussed in detail in the rest of the paper [216, 221, 306–308]. Zhou et al. [475] designed a scale transfer module to replace the feature pyramid and thus got a detection network more accurate and faster than YOLOv2. Iandola et al. [147] provided a framework to efficiently compute multi-scale features. Redmon and Angelova [305] used a YOLO-like architecture to provide oriented bounding boxes symbolizing grasps in real time. Shafiee et al. [329] built a faster version of YOLOv2 that runs on embedded devices other than GPUs. Li and Zhou [210] managed to speed-up the SSD detector, bringing it to almost 70 fps, using a more lightweight architecture.

In single stage methods most of the computations are found in the backbone networks so researchers started to design new backbones for
detection in order to have fewer operations like PVANet [170] that built a deep and thin networks with fewer channels than its classification counterparts, or SqueezeDet [416] that is similar to YOLO but with more anchors and fewer parameters.

Iandola et al. [148] built an AlexNet backbone with 50 times fewer parameters. Howard et al. [134] used depth-wise-separable convolutions and point-wise convolutions to build an efficient backbone called MobileNets for image classification and detection. Sandler et al. [324] improved upon it by adding residual connections and removing nonlinearities. Very recently, Tan et al. [367] used architecture search to come up with an even more efficient network (1.5 times faster than Sandler et al. [324] and with lower latency). ShuffleNet [461] attained impressive performance on ARM devices. They can sustain only that many computations (40MFlops). Their backbone is 13 times faster than AlexNet.

Finally, Wang et al. [405] proposed PeleeNet, a light network that is 66% of the model size of MobileNet, achieving 76.4% mAP on PASCAL VOC2007 and 22.4% mAP on MS COCO at a speed of 17.1 fps on iPhone 6s and 23.6 fps on iPhone 8. [205] is also very efficient, achieving 72.1% mAP on PASCAL VOC2007 with 0.95M parameters and 1.06B FLOPs.

Fast double-staged methods exist, although the NMS part becomes generally the bottleneck. Among them one can also mention for the second time Singh et al. [346], which is one of the double-staged methods that researchers have brought to 30 fps by using superclass (sets of similar classes) specific detection. Using a mask obtained by a fast and coarse face detection method the authors of [37] reduced the computational complexity of their double stage detector by a great amount at test time by only computing convolutions on non-masked regions. Singh et al. [346] sped up R-FCN by using detection heads super classes (sets of similar classes) specific and thus decouple detection from classification. SNIPER [347] can train on 512x512 images using an adaptive sampling of the region of interests. Therefore, it’s training can use larger batch size and therefore, be way faster but it needed 30% more pixels than original images at inference time making it slower.

There have also been lots of work done on pruning and/or quantifying the weights of CNNs for image classification [114, 138, 141, 143, 144, 218, 273, 299, 454, 476], but much fewer in detection yet. Although, one can find some detection articles that used pruning. Girshick [104] used SVD on the weights of the fully connected layers in Fast R-CNN. Masana et al. [234], who pruned near-zero weights in detection networks and extended the compression to be domain-adaptive in Masana et al. [235].

To help the reader better encompass the different accuracy vs speed trade-offs present in the modern methods, we display some of the leading methods on PASCAL VOC 2007 [88] with their inference speed on one image (batch size of 1) in Figure 20.

It is not only necessary to respect available material constraints (data and machines) but detectors have to be reliable too. They must be robust to perturbations and they can make mistakes but the mistakes also need to be interpretable, which is a challenge in itself with the millions of weights and the architectural complexity of modern pipelines. It is a good sign to outperform all other methods on a benchmark, it is something else to perform accurately in the wild. That is why we dedicate the following sections to the exploration of such challenges.

4.3 Towards Versatile Object Detectors

So far in all this survey, detectors were tested on limited, well-defined benchmarks. It is mandatory to assess their performances. However, at the end we are really interested in their behaviors in the wild where no annotations are present. Detectors have to be robust to unusual situations and one would wish for detectors to be able to evolve themselves. This section will review the state of deep learning methods w.r.t. these expectations.
Figure 20: Performance on VOC07 with respect to Inference speed on a TitanX GPU. The vertical line represents the limit of Real-Time Speed (indistinguishable from continuous motion for the human eye). We also added in light gray some relevant work measured on similar devices (K40, TITAN Xp, Jetson TX2). Only RefineDet [460], DES [467] and STDN [475] are simultaneously real-time and above 80% in mAP although for some of them (DES, STDN) better hardware (TITAN Xp) must have helped.
4.3.1 Interpretability and Robustness

With the recent craze about self-driving cars, it has become a top priority to build detectors, that can be trusted with our lives. Hence, detectors should be robust to physical adversarial attacks [43, 226] and weather conditions, which was the reason for building KITTI [98] and DETRAC [411] back then and has now led to the creation of two amazingly huge datasets: ApolloScape from Baidu [440] and BDD100K from Berkeley [440] car detection datasets. The driving conditions of the real world are so complex: changing environments, reflections, different traffic signs and rules for different countries. So far, this open problem is largely unsolved even if some industry players seem to be confident enough to leave self-driving cars without safety nets in specific cities. It will surely involve at some point the heavy use of synthetic data otherwise it would take a lifetime to gather the data necessary to be confident enough. To finish on a positive note detectors in self-driving cars can benefit from multi-sensory inputs such as LiDAR point clouds [124], other lasers and multiple cameras so it can help disambiguate certain difficult situations (reflections on the cars in front of it for instance).

But most of all the detectors should incorporate a certain level of interpretability so that if a dramatic failure happens it can be understood and fixed. It is also a need for legal matters. Very few works have done so because it requires delving into the feature maps of the backbone network. A few works proposed different approaches for classification only but no consensus has been reached yet. Among the popular methods one can cite the gradient map in the image space of Simonyan et al. [344], the occlusion analysis of Zeiler and Fergus [449], the guided back propagation of Springenberg et al. [355] and, recently, the perturbation approach of Fong and Vedaldi [91]. Figure 21 shows the insights gained by using two of the mentioned methods on a classifier.

No method exists yet for object detectors to the best of our knowledge. It would be a very interesting research direction for future works.

4.3.2 Universal Detector, Lifelong Learning

Having object detectors able to iteratively, and without any supervision, learn to detect novel object classes and improve their performance would
be one of the Holy Grails of computer vision. This can have the form of lifelong learning, where goal is to sequentially retrain learned knowledge and to selectively transfer the knowledge when learning a new task, as defined in [341]. Or never ending learning [245], where the system has sufficient self-reflection to avoid plateaus in performances and can decide how to progress by itself. However, one of the biggest issues with current detectors is they suffer from catastrophic forgetting, as say Castro et al. [33]. It means their performance decreases when new classes are added incrementally. Some authors tried to face this challenge. For example, the knowledge distillation loss introduced by Li and Hoiem [209] allows to forget old data while using previous models to constraint updated ones during learning. In the domain of object detection, the only recent contribution we are aware of is the incremental learning approach of Shmelkov et al. [336], relying on a distillation mechanism. Lifelong learning and never ending learning are domains where a lot still have to be discovered or developed.

4.4 Concluding Remarks

It seems that deep learning in its current form is not yet fully ready to be applied to other modalities than 2D images: in videos, temporal consistency is hard to take into account with DCNNs because 3D convolutions are expensive, tubelets and tracklets are interesting ideas but lack the elegance of DCNNs on still images. For point clouds the picture is even worse. The voxelisation of point clouds does not deal with their inherent sparsity and create memory issues and even the simplicity and originality of the PointNet articles Qi et al. [288, 290] that leaves the point clouds untouched has not matured enough yet to be widely adopted by the community. Hopefully, dealing with other constraints like weak supervision or few training images is starting to produce worthy results without too much change to the original DCNN architectures [76, 97, 369, 400, 462–464]. It seems to be only a matter of refining cost functions and coming-up with more building blocks than reinventing DCNNs entirely. However, the Achilles heel of deep-learning methods is their interpretability and trustworthiness. The object detection community seems focused on improving the performances on static benchmarks instead of finding ways to better understand the behavior of DCNNs. It is understandable but it shows that Deep Learning has not yet reached full maturity. Eventually, one can hope that the performances of new detectors will plateau and when it does, researchers will be forced to come back to the basics and focus instead on interpretability and robustness before the next paradigm washes off deep-learning entirely.

5 Conclusions

Object detection in images, a key topic attracting a substantial part of the computer vision community, has been revolutionized by the recent arrival of convolutional neural networks, which swept all the methods previously dominating the field. This article provides a comprehensive survey of what happened in the domain since 2012. It shows that, even if top-performing methods concentrate around two main alternatives – single stage methods such as SSD or YOLO, or two stages methods in the footsteps of Faster RCNN – the domain is still very active. Graph networks, GANs, context, small objects, domain adaptation, occlusions, etc. are the directions that are actively studied in the context of object detection. Extension of object detection to other modalities, such as videos or 3D point clouds, as well as constraints, such as weak supervision is also very active and has been addressed. The appendix of this survey also provides a very complete list of the public datasets available to the community and highlights top performing methods on these datasets. We believe this article will be useful to better understand the recent progress and the bigger picture of this constantly moving field.
References


[27] Zhaowei Cai and Nuno Vasconcelos. Cascade R-CNN: delving into high quality object detection. In Computer Vision and Pattern Recognition (CVPR), 2018 IEEE Con-


[52] G. Cheng, P. Zhou, and J. Han. RIFD-CNN: Rotation-Invariant and Fisher Discriminative Convolutional Neural Networks for Object


[172] Brendan F Klare, Ben Klein, Emma Taborsky, Austin Blanton, Jordan Cheney, Kristen Allen, Patrick Grother, Alan Mah, and Anil K Jain. Pushing the frontiers of unconstrained face detection and recognition: Iarpa janus benchmark a. In IEEE Conference on Computer Vision and Pattern Recogn-


[259] Jean Ogier Du Terrail and Frédéric Jurie. ON THE USE OF DEEP NEURAL NETWORKS FOR THE DETECTION OF SMALL VEHICLES IN ORTHO-IMAGES. In IEEE International Conference on Image Processing, Beijing, China, September 2017. URL https://hal.archives-ouvertes.fr/hal-01527906.


[328] Pierre Sermanet, Koray Kavukcuoglu, Soumith Chintala, and Yann LeCun. Pedestrian detection with unsupervised


[355] Jost Tobias Springenberg, Alexey Dosovitskiy, Thomas Brox, and Martin A. Ried-


Gül Varol, Javier Romero, Xavier Martin, Naureen Mahmood, Michael J. Black, Ivan


[488] Yousong Zhu, Chaoyang Zhao, Jingqiao Wang, Xu Zhao, Yi Wu, and Hanqing Lu. Couplenet: Coupling global structure with local parts for object detection. In IEEE
A Datasets and Results

Most of the object detection’s influential ideas, concepts and literature having been now reviewed, the rest of the article dives into the datasets used to train and evaluate these detectors.

Public datasets play an essential role as they not only allow to measure and compare the performance of object detectors but also provides resources allowing to learn object models from examples. In the area of deep learning, these resources play an essential role, as it has been clearly shown that deep convolutional neural networks are designed to benefit and learn from massive amount of data [473]. This section discusses the main datasets used in the recent literature on object detection and present state-of-the-art methods for each dataset.

A.1 Classical Datasets with Common Objects

We first start by presenting the datasets containing everyday life object taken from consumer cameras. This category contains the most important datasets for the domain, attracting the largest part of the community. We will discuss in a second section the datasets devoted to specific detection tasks (e.g., face detection, pedestrian detection, etc.).

A.1.1 Pascal-VOC

Pascal-VOC [88] is the most iconic object detection dataset. It has changed over the years but the format everyone is familiar with is the one that emerged in 2007 with 20 classes (Person: person; Animal: bird, cat, cow, dog, horse, sheep; Vehicle: aeroplane, bicycle, boat, bus, car, motorbike, train; Indoor: bottle, chair, dining table, potted plant, sofa, tv/monitor). It is now used as a test bed for most new algorithms. As it is quite small there have been claims that we are starting to overfit on the test set and therefore, MS-COCO (see next section) is preferred nowadays to demonstrate the quality of a new algorithm. The 0.5 IoU based metrics this dataset introduced has now become the de facto standard for every single
detection problem. Overall, this dataset’s impact on the development of innovative methods in object detection cannot be overstated. It is quite hard to find all relevant literature but we have tried to be as thorough as possible in terms of best performing methods. The URL of the dataset is http://host.robots.ox.ac.uk/pascal/VOC/.

Two versions of Pascal-VOC are commonly used in the literature, namely VOC2007 and VOC2012:

- VOC07, with 9,963 images containing 24,640 annotated objects, is small. For this reason, papers using VOC07 often train on the union of VOC07 and VOC12 trainval (VOC07+12). The Average Precision (AP) averaged across the 20 classes is saturating at around 80 points @0.5 IoU. Some methods got extra points but it seems one cannot go over around 85 points (without pre-training on MS COCO). Using MS COCO data in addition, one can get up to 86.3 AP (see [207]). We chose to display methods with mAP over 80 points only on Table 6. We do not distinguish between the methods that do multiple inference tricks or the methods that reports results as is. However for each method we reported for the highest published results we could get.

- VOC12 is a little bit harder than its 2007 counterpart, and we have just gone over the 80 point mark. As it is harder, this time, most literature uses the union of the whole VOC2007 data (trainval+test) and VOC2012 trainval; It is referred to as 07++12. Again better results are obtained with pre-training on COCO data (83.8 points in [117]). Results above 75 points are presented in Table 7.

On both splits all backbones used by the leaders of the board are heavy backbones with more than a 100 layers except for [467] that gets close to state of the art using only VGG-16.

#### A.1.2 MS COCO

MS COCO [214] is the most challenging object detection dataset available today. It consists of 118,000 training images, 5,000 validation images and 41,000 testing images. They have also released 120K unlabeled images that follow the same class distribution as the labeled images. They may be useful for semi-supervised learning on COCO. The MS COCO challenge has been ongoing since 2015. There are 80 object categories, over 4 times more than Pascal-VOC. MS COCO is a fine replacement for Pascal-VOC, that has arguably started to age a little. Like ImageNet in its time, MS-COCO has become the de facto standard for the object detection community and any method winning the state-of-the-art on it is assured to gain much traction and visibility. The AP is calculated similar to Pascal-VOC but averaged on multiple IoUs from

---

<table>
<thead>
<tr>
<th>Method</th>
<th>Backbone</th>
<th>mAP</th>
</tr>
</thead>
<tbody>
<tr>
<td>247</td>
<td>ResNeXt-101</td>
<td><strong>83.1</strong></td>
</tr>
<tr>
<td>427</td>
<td>ResNet-101</td>
<td><strong>83.1</strong></td>
</tr>
<tr>
<td>452</td>
<td>ResNet-101</td>
<td>82.9</td>
</tr>
<tr>
<td>63</td>
<td>ResNet-101</td>
<td>82.6</td>
</tr>
<tr>
<td>176</td>
<td>ResNet-101</td>
<td>82.4</td>
</tr>
<tr>
<td>207</td>
<td>ResNet-101</td>
<td>82.1</td>
</tr>
<tr>
<td>467</td>
<td>VGG-16</td>
<td>81.7</td>
</tr>
<tr>
<td>92</td>
<td>ResNet-101</td>
<td>81.5</td>
</tr>
<tr>
<td>475</td>
<td>DenseNet-169</td>
<td>80.9</td>
</tr>
<tr>
<td>469</td>
<td>ResNet-101</td>
<td>80.7</td>
</tr>
<tr>
<td>62</td>
<td>ResNet-101</td>
<td>80.5</td>
</tr>
</tbody>
</table>

Table 6: State-of-the-art methods on VOC07 test set (Using VOC07+12).

<table>
<thead>
<tr>
<th>Method</th>
<th>Backbone</th>
<th>mAP</th>
</tr>
</thead>
<tbody>
<tr>
<td>427</td>
<td>ResNet-101</td>
<td><strong>81.2</strong></td>
</tr>
<tr>
<td>176</td>
<td>ResNet-101</td>
<td>81.1</td>
</tr>
<tr>
<td>247</td>
<td>ResNeXt-101</td>
<td>80.9</td>
</tr>
<tr>
<td>207</td>
<td>ResNet-101</td>
<td>80.6</td>
</tr>
<tr>
<td>452</td>
<td>ResNet-101</td>
<td>80.5</td>
</tr>
<tr>
<td>467</td>
<td>VGG-16</td>
<td>80.3</td>
</tr>
<tr>
<td>92</td>
<td>ResNet-101</td>
<td>80.0</td>
</tr>
<tr>
<td>221</td>
<td>ResNet-101</td>
<td>78.5</td>
</tr>
<tr>
<td>62</td>
<td>ResNet-101</td>
<td>77.6</td>
</tr>
</tbody>
</table>

Table 7: State-of-the-art methods on VOC12 test set (Using VOC07++12).
Most available alternatives stemmed from Faster R-CNN [309], which in its first iteration won the first challenge with 37.3 mAP with a ResNet101 backbone. In the second iteration of the challenge the mAP went up to 41.5 with an ensemble of Faster R-CNN [309] that used a different implementation of RoI-Pooling. This maybe inspired the RoI-Align of Mask R-CNN [118]. Tao Kong claimed that a single Faster R-CNN with HyperNet features [174] can reach 42.0 mAP. The best published single model method [274] nowadays is around 50.5 (52.5 with an ensemble) and relied on different techniques already mentioned in this survey. Among them one can mention FPN [215], large batch training [274] and GCN [275]. Ensembling Mask R-CNNs [118] gave around the same performance as [274] at around 50.3 mAP. Deformable R-FCN [63] is not lagging too far behind with 48.5 mAP single model performance (50.4 mAP with an ensemble) using Soft NMS [21] and the "mandatory" FPN [215]. Other entries were based mostly on Mask R-CNN [118]. We display the current leaderboard (http://cocodataset.org/#detection-leaderboard) also visible at for all the past challenges with the main-ideas present in the winning entries Figure 22. The URL of the dataset is http://cocodataset.org. 

A.1.3 ImageNet Detection Task

ImageNet is a dataset organized according to the nouns of the WordNet hierarchy. Each node of the hierarchy is depicted by hundreds and thousands of images, with an average of over 5,000 images per node. Since 2010, the Large Scale Visual Recognition Challenge is organized each year and contains a detection challenge using ImageNet images. The detection task, in which each object instance has to be detected, has 200 categories. There is also a classification and localization task, with 1,000 categories in which algorithms have to produce 5 labels (and 5 bounding boxes) only, allowing not to penalize the detection of objects that are present, but not included in the ground truth. In the 2017 contest, the top detector was proposed by a team from Nanjing University of Information Science and Imperial College London. It ranked first on 85 categories with an overall AP of 73.13. As far as we know, there is no paper describing the approach precisely (but some slides are available at the workshop page). The 2nd ranked method was from Bae et al. [8], who observed that modern convolutional detectors behave differently for each object class. The authors consequently built an ensemble detector by finding the best detector for each object class. They obtained a AP of 59.30 points and won 10 categories. ImageNet is available at http://image-net.org.

A.1.4 Visual Genome

Visual Genome [179] is a very peculiar dataset focusing on object relationships. It contains over 100,000 images. Each image has bounding boxes but also complete scene graphs. Over 17,000 categories of objects are present. The first ones in terms of representativeness by far are man and woman followed by trees and sky. On average there are 21 objects per image. It is unclear if it qualifies as an object detection dataset as the paper does not include clear object detection metrics or evaluation as its focus is on scene graphs and visual relationships. However, it is undoubtedly an enormous source of strongly supervised images to train object detectors. The Visual Genome Dataset has huge number of classes, most of them being small and hard to detect. The mAP reported in the literature is therefore, much smaller compared to previous datasets. One of the best performing approaches is of Li et al. [204] which reached 7.43 mAP by linking object detection, scene graph generation and region captioning. Faster R-CNN [104] has a mAP of 6.72 points on this dataset. The URL of the dataset is https://visualgenome.org.

A.1.5 OpenImages

The challenge OpenImagesV4 [178] that will be organized for the first time at ECCV2018 offers the largest to date common objects detection dataset with up to 500 classes (including the familiar ones
Figure 22: This plot displays the performance advances in the bounding boxes detection COCO challenge over the years. For each year we present the main ideas behind the three best performing entries in terms of \textbf{mmAP}. In 2015 the main frameworks were Fast R-CNN [104], DeepMask [282] and Faster R-CNN [309] supported by the new Deep ResNets [117]. In 2016, the same pipelines won the competition with the addition of AttractioNet [101] and LocNet [103] for better proposals and localization accuracy. In 2017 Mask R-CNN [118], FPN [215] and MegDet [274] proved that more complex ideas could allow to go over the 50\% mark. In 2018 the same pipelines, as in 2017 (namely Mask R-CNN), were enriched with the multi-stages of Cascade R-CNN [27], a new RPN and backbones that were for the first time specifically designed for the detection task. The last entry of 2018 reached 53\% mmAP and we can extrapolate the two first entries to be around 55\% bbox mmAP based on their ranking for instance segmentation.
from Pascal-VOC) on 1,743,000 images and more than 12,000,000 bounding boxes with an average of 7 objects per image for training, and 125,436 images for tests (41,620 for validation). The object detection metric is the AP@0.5IoU averaged across classes taking into account the hierarchical structure of the classes with some technical subtleties on how to deal with groups of objects closely packed together. This is the first detection dataset to have so many classes and images and it will surely require some new breakthrough to get it right. At the time of writing there is no published or non-published results on it, although the results of an Inception ResNet Faster R-CNN baseline can be found on their site to have 37 mAP. The URL of the project is https://storage.googleapis.com/openimages/web/index.html.

For industrial applications, more often than not, the objects to detect does not come from the categories present in VOC or MS-COCO. Furthermore, they do not share the same variances; Rotation variance for instance, is a property of several applications domains but is not present in any classical common object dataset. That is why, pushed by the industry needs, several other object detection domains have appeared all with their respective literature. The most famous of them are listed in the following sections.

A.2 Specialized datasets

To find interesting domains one has to find interesting products or applications that drive them. The industry has given birth to many sub-fields in object detection: they wanted to have self-driving cars so we built pedestrian detection and traffic signs detection datasets; they wanted to monitor traffic so we had to have aerial imagery datasets; they wanted to be able to read text for blind persons or automatic translations of foreign languages so we constructed text detection datasets; some people wanted to do personalized advertising (arguably not a good idea) so we engineered logo datasets. They all have their place in this specialized dataset section.

### A.2.1 Aerial Imagery

The detection of small vehicles in aerial imagery is an old problem that has gained much attraction in recent times. However, it was only in the last years that large dataset have been made publicly available, making the topic even more popular. The following paragraphs take inventory of these datasets and of the best performing methods.

**Google Earth** [120] comprises 30 images of the city of Bruxelles with 1,319 small cars and vertical bounding boxes, its variability is not enormous but it is still widely used in the literature. There are 5 folds. The CNN best result is [52] with 94.6 AP. It was later augmented with angle annotations by Henriques and Vedaldi [122]. The data can be found on Geremy Heitz webpage (http://ai.stanford.edu/~gaheitz/Research/TAS/).

**OJRDS** [375], with only 180 vehicles this dataset, is not very much used by the community.

**DLR 3k Munich Dataset** [219] is one of the most used datasets in the small vehicle detection literature with 20 extra large images. 10 training images with up to 3,500 cars and 70 trucks and 10 test images with 5,800 cars 90 trucks. Other classes are also available like car or truck’s trails and dashed lines. The state-of-the-art seems to belong to [373] at 83% of F1 on both cars and trucks and [372] at 82%, which provide oriented boxes. Some relevant articles that compare on this dataset are [67, 350, 351]. The data can be downloaded by asking the provided contact on https://www.dlr.de/eoc/en/.. aspx/tabid-5431/9230_read-42467/.

**VeDAI** [302] is for vehicle detection is aerial images. The vehicles contained in the database, in addition to being small, exhibit different variability such as multiple orientations, lighting/shadowing changes, occlusions. etc. Furthermore, each image is available in several spectral bands and resolutions. They provide the same images in 2 resolutions 512x512 and 1024x1024. There are a total of 9 classes and 1,200 images with an average of 5.5 instances per image. It is one of the few datasets to have 10 folds and the metric is based on an ellipse based distance between the cen-
ter of the ground truth and the centers of the detections. The state-of-the-art is currently held by [259]. Although many recent articles used their own metrics, which makes them difficult to compare [323, 351, 352, 372, 373]. VeDAI is available at https://downloads.greyc.fr/vedai/.

COWC [252], introduced in ECCV2016, is a very large dataset with regions from all over the world and more than 32,000 cars. It also contains almost 60,000 hard negative patches hand-picked, which is a blessing when training detectors that do not include hard-example mining strategies. Unfortunately, no test data annotations are available so detection methods cannot yet be properly tested on it. COWC is available at https://gdo152.llnl.gov/cowc/.

DOTA [421], released this year at CVPR, is the first mainstream dataset to change its metric to incorporate rotated bounding boxes similar to the text detections datasets. The images are of very different resolutions and zoom factors. There are 2,800 images with almost 200,000 instances and 15 categories. This dataset will surely become one of the important ones in the near future. The leader board https://captain-whu.github.io/DOTA/results.html shows that Mask R-CNN structures are the best at this task for the moment with the winner culminating at 76.2 oriented mAP but no other published method apart from [421] yet. UCAS-AOD [479], NWPU VHR10 [54] and HRSC2016 [223] all provided oriented annotations also but they are hard to find and very few articles actually use them. DOTA is available at https://captain-whu.github.io/DOTA/dataset.html.

xView [186] is a very large scale dataset gathered by the pentagon, containing 60 classes and 1 million instances. It is split in three parts: train, val and test. xView is available at http://xviewdataset.org. First challenge will end in August 2018, no results are available yet.

VisDrone [481] is the most recent dataset including aerial images. Images, captured by different drones flying over 14 different cities separated by thousands of kilometers in China, in different scenarios under various weather and lighting conditions. The dataset consists of 263 video sequences formed by 179,264 frames and 10,209 static images and contains different objects such as pedestrian, vehicles, bicycles, etc. and density (sparse and crowded scenes). Frames are manually annotated with more than 2.5 million bounding boxes and some attributes, e.g., scene visibility, object class and occlusion, are provided. VisDrone is very recent and no results are available yet. VisDrone is available at http://www.aiskyeye.com.

A.2.2 Text Detection in Images

Text detection in images or videos is a common way to extract content from images and opens the door to image retrieval or automatic text translation applications. We inventory, in the following, the main datasets as well as the best practices to address this problem.

ICDAR 2003 [227] was one of the first public datasets for text detection. The dataset contains 509 scene images and the scene text is mostly centered and iconic. Delakis and Garcia [65] was one of the first to use CNN on this dataset.

Street View Text (SVT) [403]. Taken from Google StreetView, it is a dataset filled with business names mostly, from outdoor streets. There are 350 images and 725 instances. One of the best performing methods on SVT is [468] with a F-measure of 83%. SVT can be downloaded from http://tc11.cvc.uab.es/datasets/SVT_1.

MSRA-TD500 [387] contains 500 natural images, which are taken from indoor (office and mall) and outdoor (street) scenes. The resolutions of the images vary from 1296 × 864 to 1920 × 1280. There are Chinese and English texts and mixed too. The training set contains 300 images randomly selected from the original dataset and the remaining 200 images constitute the test set. Best performing method on MSRA-TD500 is [212] with a F-measure of 79%. Shi et al. [333], Yao et al. [435], Ma et al. [228] and Zhang et al. [466] also performed very well (F-measures of 77%, 76%, 75% and 75% respectively). The dataset is available at http://www.iapr-tc11.org/mediawiki/index.php/MSRA_Text_Detection_500_Database_(MSRA-TD500).
**IIIT 5k-word** [242] has 1,120 images and 5,000 words from both street scene texts and born-digital images. 380 images are used to train and the remaining to test. Each text has also a category label easy or hard. [212] is state-of-the-art, as for MSRA-TD500. IIIT 5k-word is available at [http://cvit.iiit.ac.in/projects/SceneTextUnderstanding/IIIT5K.html](http://cvit.iiit.ac.in/projects/SceneTextUnderstanding/IIIT5K.html).

**Synth90K** [153] is a completely generated grayscale text dataset with multiple fonts and vocabulary well blended into scenes with 9 million images from a 90,000 vocabulary. It can be found on the VGG page at [http://www.robots.ox.ac.uk/~vgg/data/text/](http://www.robots.ox.ac.uk/~vgg/data/text/).

**ICDAR 2015** [165] is another popular iteration of the ICDAR challenge, following ICDAR 2013. Busta et al. [26] got state-of-the-art 87% of F measure in comparison to the 83.8% of Liao et al. [212] and the 82.54% of Jiang et al. [159]. TextBoxes++ [211] reached 81.7% and Shi et al. [333] is at 75%.

**COCO Text** [396], based on MS COCO, is the biggest dataset for text detection. It has 63,000 images with 173,000 annotations. [212] is the only published result with [477] yet that differs from the baselines implemented in the dataset paper [396]. So there must still be room for improvement. The very recent [211] outperformed [477]. COCO Text is available at [https://bgshih.github.io/cocotext/](https://bgshih.github.io/cocotext/).

**RCTW-17 (ICDAR 2017)** [334] is the latest ICDAR database. It is a large line-based dataset with mostly Chinese text. Liao et al. [212] achieved SOTA on this one too with 67.0% of F measure. The dataset is available at [http://www.icdar2017chinese.site/dataset/](http://www.icdar2017chinese.site/dataset/).

### A.2.3 Face Detection

Face detection is one of the most widely addressed detection tasks. Even if the detection of frontal in high resolution images is an almost solved problem, there is room for improvement when the conditions are harder (non-frontal images, small faces, etc.). These harder conditions are reflected by the following recent datasets. The main characteristics of the different face datasets are proposed in Table 8.

**Face Detection Data Set and Benchmark (FDDB)** [155] is built using Yahoo!, with 2845 images and a total of 5171 faces; it has a wide range of difficulties such as occlusions, strong pose changes, low resolution and out-of-focus faces, with both grayscale and color images. Zhang et al. [458] obtained an AUR of 98.3% on this dataset and is currently state-of-the-art for this dataset. Najibi et al. [255] obtained 98.1%. The dataset can be downloaded at [http://vis-www.cs.umass.edu/fddb/index.html](http://vis-www.cs.umass.edu/fddb/index.html).

**Annotated Facial Landmarks in the Wild (AFLW)** [177] is made from a collection of images collected on Flickr, with a large variety in face appearance (pose, expression, ethnicity, age, gender) and environmental conditions. It has the particularity to not to be aimed at face detection only, but more oriented towards landmark detection and face alignment. In total 25,993 faces in 21,997 real-world images are annotated. Annotations come with rich facial landmark information (21 landmarks per faces). The dataset can be downloaded from [https://www.tugraz.at/institute/icg/research/team-bischof/irs/downloads/aflw/](https://www.tugraz.at/institute/icg/research/team-bischof/irs/downloads/aflw/).

**Annotated Face in-the-Wild (AFW)** [483] is a dataset containing faces in real conditions, with their associated annotations (bounding box, facial landmarks and pose angle labels). Each image contains multiple, non-frontal faces. The dataset contains 205 images with 468 faces. Zhang et al. [458] obtained an AP of 99.85% on this dataset and is currently state-of-the-art for this dataset.

**PASCAL Faces** [430] contains images selected from PASCAL VOC [88] in which the faces have been annotated. [458] obtained an AP of 98.49% on this dataset and is currently state-of-the-art for this dataset.

**Multi-Attribute Labeled Faces (MALF)** [20] incorporates richer semantic annotations such as pose, gender and occlusion information as well as expression information. It contains 5,250 images collected from the Internet and approximately 12,000 labeled faces. The dataset and up-to-date results of the evaluation can be found at [http://www.cbsr.ia.ac.cn/faceevaluation/](http://www.cbsr.ia.ac.cn/faceevaluation/).
Wider Face [433] is one of the largest datasets for face detection. Each annotation includes information such as scale, occlusion, pose, overall difficulty and events, which makes possible in-depth analyses. This dataset is very challenging especially for the 'hard set'. Najibi et al. [255] obtained an AP of 93.1% (easy), 92.1% (medium) and 84.5% (hard) on this dataset and is currently state-of-the-art for this dataset. Zhang et al. [458] are also very good with AP of 92.8% (easy), 91.3% (medium) and 84.0% (hard). Datasets and results can be downloaded at http://mmlab.ie.cuhk.edu.hk/projects/WIDERFace/.

IARPA Janus Benchmark A (IJ-A) [172] contains images and videos from 500 subjects captured from 'in the wild' environment, and contains annotations for both recognition and detection tasks. All labeled faces are localized with bounding boxes as well as with landmarks (center of the two eyes, base of the nose). IJB-B [412] extended this dataset with 1,845 subjects, for 21,798 still images and 55,026 frames from 7,011 videos. IJB-C [238], which is the new extended version of the IARPA Janus Benchmark A and B, adds 1,661 new subjects to the 1,870 subjects released in IJB-B. The NIST Face Challenges are at https://www.nist.gov/programs-projects/face-challenges.

Un-constrained Face Detection Dataset (UFDD) [253] was built after noting that in many challenges large variations in scale, pose, appearance are successfully addressed but there is a gap in the performance of state-of-the-art detectors and real-world requirements, not captured by existing methods or datasets. UFDD aimed at identifying the next set of challenges and collect a new dataset of face images that involve variations such as weather-based degradations, motion blur and focus blur. The authors also provide an in-depth analysis of the results and failure cases of these methods. This dataset is very recent and has not been used specifically yet. However, Nada et al. [253] reported the performances (in terms of AP) of Faster-RCNN [309] (52.1%), SSH [255] (69.5%), S3FD [458] (72.5%) and HR-ER [137] (74.2%). Dataset and results can be downloaded at http://www.ufdd.info/.

### Table 8: Datasets for face detection.

<table>
<thead>
<tr>
<th>Dataset</th>
<th>#Images</th>
<th>#Faces</th>
<th>Source</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>FDDB [159]</td>
<td>2,924</td>
<td>5,171</td>
<td>Yahoo! News</td>
<td>Images</td>
</tr>
<tr>
<td>AFW [177]</td>
<td>21,059</td>
<td>25,293</td>
<td>Flickr</td>
<td>Images</td>
</tr>
<tr>
<td>APR [430]</td>
<td>3,017</td>
<td>1,253</td>
<td>Flickr</td>
<td>Images</td>
</tr>
<tr>
<td>TASCAL Faces [443]</td>
<td>867</td>
<td>1,345</td>
<td>Flickr/VOE</td>
<td>Images</td>
</tr>
<tr>
<td>MALF [20]</td>
<td>6,290</td>
<td>11,081</td>
<td>Flickr, Banshi etc.</td>
<td>Images</td>
</tr>
<tr>
<td>IJB-A [172]</td>
<td>24,327</td>
<td>67,183</td>
<td>Google, Bing, etc.</td>
<td>Images/Videos</td>
</tr>
<tr>
<td>IJB-CBFW [234]</td>
<td>1,529</td>
<td>9,928</td>
<td>Google</td>
<td>Images</td>
</tr>
<tr>
<td>Wider Face [433]</td>
<td>32,203</td>
<td>203,703</td>
<td>Google, Bing</td>
<td>Images</td>
</tr>
<tr>
<td>IJB-B [412]</td>
<td>70,924</td>
<td>125,474</td>
<td>Freescale</td>
<td>Images/Videos</td>
</tr>
<tr>
<td>IJB-C [238]</td>
<td>148,596</td>
<td>340,561</td>
<td>Freescale</td>
<td>Images/Videos</td>
</tr>
<tr>
<td>Wildest Faces [444]</td>
<td>67,889</td>
<td>109,771</td>
<td>YouTube</td>
<td>Videos</td>
</tr>
<tr>
<td>UFDD [255]</td>
<td>6,424</td>
<td>10,995</td>
<td>Google, Bing, etc.</td>
<td>Images</td>
</tr>
</tbody>
</table>

### A.2.4 Pedestrian Detection

Pedestrian detection is one of the specific tasks abundantly studied in the literature, especially since research on autonomous vehicles has intensified.

MIT [272] is one of the first pedestrian datasets. It’s puny in size (509 training and 200 testing images). The images were extracted
Figure 23: Number of images vs number of faces in each dataset (Table 8) on a log scale. The size of the bubble indicates average number of faces per image which can be used as an estimate of complexity of the dataset.

from the LabelMe database. You can find it at http://cbcl.mit.edu/software-datasets/PedestrianData.html

INRIA [64] is currently one of the most popular static pedestrian detection datasets introduced in the seminal HOG paper [64]. It uses obviously the Caltech metric. Zhang et al. [459] gained state-of-the-art with 6.4% log average miss rate. Method at the second position is [455] with 6.9% using the RPN from Faster R-CNN and boosted forests on extracted features. The others are not CNN methods (the third one using pooling with HOG, LBP and covariance matrices). It can be found at http://pascal.inrialpes.fr/data/human/. Similarly, PASCAL Persons dataset is a subset of the aforementioned Pascal-VOC dataset.

CVC-ADAS [100] is a collection of datasets including videos acquired on board, virtual-world pedestrians and real pedestrians. It can be found at following http://adas.cvc.uab.es/site/.

USC [417] is an old small pedestrian dataset taken largely from surveillance videos. It is still downloadable at http://iris.usc.edu/Vision-Users/OldUsers/bowu/DatasetWebpage/dataset.html

ETH [87] was captured from a stroller. There are 490 training frames with 1578 annotations. There are three test sets. The first test set has 999 frames with 5193 annotations, the second one 450 and 2359 and the third one 354 and 1828 respectively. The stereo cues are available. It is a difficult dataset where the state-of-the-art from Zhang et al. [459] trained on CityPersons still remains at 24.5% log average miss rate. The boosted forest of Zhang et al. [455] gets 30.2% only. It is available at https://data.vision.ee.ethz.ch/cvl/aess/iccv2007/

Daimler DB [84] is an old dataset captured in an urban setting, builds on DaimlerChrysler datasets with only grayscale images. It has been recently extended with Cyclist annotations into the Tsinghua Daimler Cyclist (TDC) dataset [202] with color images. The dataset is available at http://www.gavrila.net/Datasets/datasets.html.

TUD-Brussels [414] is from the TU Darmstadt University and contains image pairs recorded in a crowded urban setting with an on-board camera from a car. There are 1092 image pairs with 1776 annotations in the training set. The test set contains 508 image pairs with 1326 pedestrians. The evaluation is measured from the recall at 90% precision, somehow reminiscent of KITTI dataset. TUD-Brussels is available at https://www.mpi-inf.mpg.de/departments/computer-vision-and-multimodal-computing/research/people-detection-pose-estimation-and-tracking/multi-cue-onboard-pedestrian-detection/.

Caltech USA [71] contains images are captured in the Greater Los Angeles area by an independent driver to simulate real-life conditions without any bias. 192,000 pedestrian instances are available for training, 155,000 for testing. The evaluation use Pascal-VOC criteria at 0.5 IoU. The performance measure is the log average miss rate as application wise one cannot have too many False Positive per Image (FPPI). It is computed by averaging miss rates at 9 FPPIs from $10^{-2}$ to 1 uniformly in log scale. State-of-the-art algorithms are at around 4% log average miss rate. Wang et al. [409] got 4.0% by using a novel bounding box regression loss. Following it, we have Zhang et al. [459] at 4.1% using
a novel RoI-Pooling of parts helping with occlusions and pre-training on CityPersons. Mao et al. [231] is lagging behind with 5.5%, using a Faster R-CNN with additional aggregated features. There also exists a CalTech Japan dataset. The benchmark is hosted at http://www.vision.caltech.edu/Image_Datasets/CaltechPedestrians/.

KITTI [98] is one of the most famous datasets in Computer Vision taken over the city of Karlsruhe in Germany. There are 100,000 instances of pedestrians. With around 6000 identities and one person in average per image. The preferred metric is the AP (Average Precision) on the moderate (persons who are less than 25 pixels tall are left behind for ranking) set. Li et al. [200] got 65.01 AP on moderate by using an adapted version of Fast R-CNN with different heads to deal with different scales. The state-of-the-art of Chen et al. [45] had to rely on stereo information to get good object proposals and 67.47 AP. All KITTI related datasets are found at http://www.cvlibs.net/datasets/kitti/index.php.

GM-ATCI [340] is a dataset captured from a fisheye-lens camera that uses CalTech evaluation system. We could not find any CNN detection results on it possibly because the state-of-the-art using multiple cues is already pretty good with 3.5% log average miss rate. The sequences can be downloaded here https://sites.google.com/site/rearviewpeds1/

CityPersons [456] is a relatively new dataset that builds upon CityScapes [58]. It is a semantic segmentation dataset recorded in 27 different cities in Germany. There are 19,744 persons in the training set and around 11,000 in the test set. There are way more identities present than in CalTech even though there are fewer instances (1300 in CalTech w.r.t. 10000 in CityPersons). Therefore, it is more diverse and thus, more challenging. The metric is the same as CalTech with some subsets like the Reasonable: the pedestrians that are more than 50 pixels tall and less than 35% occluded. Again Zhang et al. [459] and Wang et al. [409] take the lead with 11.32% and 11.48% respectively on the reasonable set w.r.t. the baseline on adapted Faster R-CNN that stands at 12.97% log average miss rate. The dataset is available at https://bitbucket.org/shanshanzhang/citypersons.

EuroCity [25] is the largest pedestrian detection dataset ever released with 238,300 instances in 47,300 images. Images are taken over 31 cities in 12 different European countries. The metric is the same as CalTech. Three baselines were tested (Faster R-CNN, R-FCN and YOLOv3). Faster R-CNN dominated on the reasonable set with 8.1%, followed by YOLOv3 with 8.5% and R-FCN lagging behind with 12.1%. On other subsets with heavily occluded or small pedestrians the ranking is not the same. We refer the reader to the dataset paper of [25].

### A.2.5 Logo Detection

Logo detection was attracting a lot of attention in the past, due to the specificity of the task. At the moment we write this survey, there are fewer papers on this topic and most of the logo detection pipelines are direct applications of Faster RCNN [309].

BelgaLogos [160] images come from the BELGA press agency. The dataset is composed of 10,000 images covering all aspects of life and current affairs: politics and economics, finance and social affairs, sports, culture and personalities. All images are in JPEG format and have been re-sized with a maximum value of height and width equal to 800 pixels, preserving aspect ratio. There are 26 different logos. Only a few images are annotated with bounding boxes. The dataset can be downloaded at https://www-sop.inria.fr/members/Alexis.Joly/BelgaLogos/BelgaLogos.html.

FlickrLogos [80, 313] consists of real-world images collected from Flickr, depicting company logos in various situations. The dataset comes in two versions: The original FlickrLogos-32 dataset and the FlickrLogos-47 [80] dataset. In FlickrLogos-32 the annotations for object detection were often incomplete, since only the most prominent logo instances were labeled. FlickrLogos-47 uses the same image corpus as FlickrLogos-32 but new classes were introduced (logo and text as separate classes) and missing object instances have been an-
notated. FlickrLogos-47 contains 833 training and 1402 testing images. The dataset can be downloaded at http://www.multimedia-computing.de/flickrlogos/.

Logo32plus [17] is an extension of the train set of FlickrLogos-32 [80]. It has the same classes of objects but much more training instances (12,312 instances). The dataset can be downloaded at http://www.ivl.disco.unimib.it/activities/logorecognition.

WebLogo-2M [358] is very large, but annotated at image level only and does not contain bounding boxes. It contains 194 logo classes and over 2 million logo images. Labels are noisy as the annotations are automatically generated. Therefore, this dataset is designed for large-scale logo detection model learning from noisy training data. For performance evaluation, the dataset includes 6,569 test images with manually labeled logo bounding boxes for all the 194 logo classes. The dataset can be downloaded at http://www.eecs.qmul.ac.uk/~hs308/WebLogo-2M.html/.

SportsLogo [213], in the absence of public video logo dataset, was collected on a set of tennis videos containing 20 different tennis video clips with camera motions (blurring) and occlusion. The logos can appear on the background as well as on players and staffs clothes. 20 logos are annotated, with about 100 images for each logo.

Logos in the Wild [389] contains images collected from the web with logo annotations provided in Pascal-VOC style. It contains large varieties of brands in-the-wild. The latest version (v2.0) of the dataset consists of 11,054 images with 32,850 annotated logo bounding boxes of 871 brands. It contains from 4 to 608 images per searched brand, and 238 brands occur at least 10 times. It has up to 118 logos in one image. Only the links to the images are released, which is problematic as numerous images have already disappeared, making exact comparisons impossible. The dataset can be downloaded from https://www.iosb.fraunhofer.de/servlet/is/78045/.

Open Logo Detection Challenge [360]. This dataset assumes that only a small proportion of logo classes are annotated whilst the remaining classes have no labeled training data. It contrasts with previous logo datasets which assumed all the logo classes are annotated. The OpenLogo challenge contains 27,189 images from 309 logo classes, built by aggregating/refining 7 existing datasets and establishing an open logo detection evaluation protocol. The dataset can be downloaded at https://qmul-openlogo.github.io.

<table>
<thead>
<tr>
<th>Dataset</th>
<th>#Classes</th>
<th>#Images</th>
</tr>
</thead>
<tbody>
<tr>
<td>BelgaLogos [160]</td>
<td>26</td>
<td>10,000</td>
</tr>
<tr>
<td>FlickrLogos-32 [313]</td>
<td>32</td>
<td>8,240</td>
</tr>
<tr>
<td>FlickrLogos-47 [80]</td>
<td>47</td>
<td>8,240</td>
</tr>
<tr>
<td>Logo32plus [17]</td>
<td>32</td>
<td>7,830</td>
</tr>
<tr>
<td>WebLogo-2M [358]</td>
<td>194</td>
<td>2,190,757</td>
</tr>
<tr>
<td>SportsLogo [213]</td>
<td>20</td>
<td>1,978</td>
</tr>
<tr>
<td>Logos in the Wild [389]</td>
<td>871</td>
<td>11,054</td>
</tr>
<tr>
<td>OpenLogos [360]</td>
<td>309</td>
<td>27,189</td>
</tr>
</tbody>
</table>

Table 9: Datasets for logo detection.

### A.2.6 Traffic Signs Detection

This section reviews the 4 main datasets and benchmarks for evaluating traffic sign detectors [133, 246, 379, 490], as well as the Bosch Small Traffic Lights [13]. The most challenging one is the Tsinghua Tencent 100k (TTK100) [490], on which Faster RCNN like detectors such as [285] have an overall precision/recall of 44%/68%, which shows the difficulty of the dataset.

**LISA Traffic Sign Dataset** [246] was among the first datasets for traffic sign detection. It contains 47 US signs and 7,855 annotations on 6,610 video frames. Sign sizes vary from 6x6 to 167x168 pixels. Each sign is annotated with sign type, position, size, occluded (yes/no), on side road (yes/no). The URL for this dataset is http://cvrr.ucsd.edu/LISA/lisa-traffic-sign-dataset.html

**The German Traffic Sign Detection Benchmark (GTSDB)** [133] is one of the most popular traffic signs detection benchmarks. It introduced a dataset with evaluation metrics, baseline results, and a web interface for comparing approaches. The dataset provides a total of 900 images with 1,206
traffic signs. The traffic sign sizes vary between 16 and 128 pixels w.r.t. the longest edge. The image resolution is $1360 \times 800$; images capture different scenarios (urban, rural, highway) during the daytime and dusk featuring various weather conditions. It can be found at http://benchmark.ini.rub.de/?section=gtsdb&subsection=news.

Belgian TSD [379] consists of 7,356 still images for training, with a total of 11,219 annotations, corresponding to 2,459 traffic signs visible at less than 50 meters in at least one view. The test set contains 4 sequences, captured by 8 roof-mounted cameras on the van, with a total of 121,632 frames and 269 different traffic signs for evaluating the detectors. For each sign, the type and 3D location is given. The dataset can be downloaded at https://btsd.ethz.ch/shareddata/.

Tsinghua Tencent 100k (TTK100) [490] provides $2048 \times 2048$ images for traffic signs detection and classification, with various illumination and weather conditions. It's the largest dataset for traffic signs detection, with 100,000 images out of which 16,787 contain traffic signs instances, for a total of 30,000 traffic instances. There are a total of 128 classes. Each instance is annotated with class label, bounding box and pixel mask. It has small objects in abundance and huge scale variations. Some signs which are naturally rare, e.g. signs to warn the driver to be cautious on mountain roads appear, have quite low number of instances. There are 45 classes with at least 100 instances present. The dataset can be obtained at https://btsd.ethz.ch/shareddata/.

Bosch Small Traffic Lights [13] is made for benchmarking traffic light detectors. It contains 13,427 images of size $1280 \times 720$ pixels with around 24,000 annotated traffic lights, annotated with bounding boxes and states (active light). Best performing algorithm is [285] which obtained a mAP of 53 on this dataset. Bosch Small Traffic Lights can be downloaded at https://hci.iwr.uni-heidelberg.de/node/6132.

A.2.7 Other Datasets

Some datasets do not fit in any of the previously mentioned category but deserve to be mentioned because of the interest the community has for them. iNaturalist Species Classification and Detection Dataset [394] contains 859,000 images from over 5,000 different species of plants and animals. The goal of this dataset is to encourage the development of algorithms for 'in the wild' data featuring large numbers of imbalanced, one-grained, categories. The dataset can be downloaded at https://github.com/visipedia/inat_comp/tree/master/2017.

Below we give all known datasets that can be used to tackle object detection with the different modalities that we presented in the Sec. 4.1.

A.3 3D Datasets

KITTI object detection benchmark [98] is the most widely used dataset for evaluating detection in 3D point clouds. It contains 3 main categories (namely 2D, 3D and birds-eye-view objects), 3 object categories (cars, pedestrians and cyclists), and 3 difficulty levels (easy, moderate and hard considering the object size, distance, occlusion and truncation). The dataset is public and contains 7,481 images for training and 7,518 for testing, comprising a total of 80,256 labeled objects. The 3D point clouds are acquired with a Velodyne laser scanner. 3D object detection performance is evaluated using the PASCAL criteria also used for 2D object detection. For cars a 3D bounding box overlap of 70% is required, while for pedestrians and cyclists a 3D bounding box overlap of 50% is required. For evaluation, precision-recall curves are computed and the methods are ranked according to average precision. The algorithms can use the following sources of information: i) Stereo: Method uses left and right (stereo) images ii) Flow: Method uses optical flow (2 temporally adjacent images) iii) Multiview: Method uses more than 2 temporally adjacent images iv) Laser Points: Method uses point clouds from Velodyne laser scanner v) Additional training data: Use of additional data sources for train-
The datasets and performance of SOTA detectors can be downloaded at http://www.cvlibs.net/datasets/kitti/, and the leaderboard is at http://www.cvlibs.net/datasets/kitti/eval_object.php?obj_benchmark=3d. One of the leading methods is [342] which is at an mAP of 67.72/64.00/63.01 (Easy/Mod./Hard) for the car category, at 50 fps. Slower (10 fps) but more accurate, [182] has a performance of 81.94/71.88/66.38 on cars. Chen et al. [47], Zhou and Tuzel [478] and Qi et al. [289] also gave very good results.

Active Vision Dataset (AVD) [5] contains 30,000+ RGBD images, 30+ frequently occurring instances, 15 scenes, and 70,000+ 2D bounding boxes. This dataset focused on simulating robotic vision tasks in everyday indoor environments using real imagery. The dataset can be downloaded at http://cs.unc.edu/~ammirato/active_vision_dataset_website/.

SceneNet RGB-D [239] is a synthetic dataset designed for scene understanding problems such as semantic segmentation, instance segmentation, and object detection. It provides camera poses and depth data and permits to create any scene configuration. 5M rendered RGB-D images from 16K randomly generated 3D trajectories in synthetic layouts are also provided. The dataset can be downloaded at http://robotvault.bitbucket.io/scenenet-rgbd.html.

Falling Things [384] introduced a novel synthetic dataset for 3D object detection and pose estimation, the Falling Things dataset. The dataset contains 60k annotated photos of 21 household objects taken from the YCB dataset. For each image, the 3D poses, per-pixel class segmentation, and 2D/3D bounding box coordinates for all objects are given. To facilitate testing different input modalities, mono and stereo RGB images are provided, along with registered dense depth images. The dataset can be downloaded at http://research.nvidia.com/publication/2018-06_Falling-Things.

A.4 Video Datasets

The two most popular datasets for video object detection are the YouTube-BoundingBoxes [303] and the ImageNet VID challenge [319]. Both are reviewed in this section.

YouTube-BoundingBoxes [303] is a data set of video URLs with the single object bounding box annotations. All video sequences are annotated with classifications and bounding boxes, at 1 frame per second. There is a total of about 380,000 video segments of 15-20 seconds, from 240,000 publicly available YouTube videos, featuring objects in natural settings, without editing or post-processing. Real et al. [303] reported a mAP of 59 on this dataset. This dataset can be downloaded at https://research.google.com/youtube-bb/.

ImageNet VID challenge [319] was a part of the ILSVRC 2015 challenge. It has a training set of 3,862 fully annotated video sequences having a length from 6 frames to 5,492 frames per video. The validation set contains 555 fully annotated videos, ranging from 11 frames to 2898 frames per video. Finally, the test set contains 937 video sequences and the ground-truth annotation are not publicly available. One of the best performing methods on ImageNet VID is [89] with a mAP of 79.8, by combining detection and tracking. Zhu et al. [484] reached 76.3 points with a flow best approach. This dataset can be downloaded at http://image-net.org/challenges/LSVRC.

VisDrone [481] contains video clips acquired by drones. This dataset is presented in Section 5.2.1.

A.5 Concluding Remarks

This appendix gave a large overview of the datasets introduced by the community for developing and evaluating object detectors in images, videos or 3D point clouds. Each object detection dataset presents a very biased view of the world, as shown in [169, 380, 381], representative of the user’s needs when they built it. The bias is not only in the images they chose (specific views of objects, objects imbalance [264], objects categories) but also in the metric they created and the evaluation protocol they devised. The community is trying its best to build more and more datasets with less and less bias and as a result it has become quite hard to find its way in this jungle of datasets, especially
when one needs: older datasets that have fallen out of fashion or even exhaustive lists of state-of-the-art algorithms performances on modern ones. Through this survey we have partially addressed this need of a common source for information on datasets.