Object Detection Techniques in Computer Vision Using Machine Learning

Prof. Sunil H. Chavda, Prof. Dhaval R. Chandarana
[1]*shchavda@gmiu.edu.in,* [2]*drchandarana@gmiu.edu.in*

I. INTRODUCTION

Object detection is a fundamental computer vision task that involves locating and classifying instances of objects (such as people, cars, animals, etc.) within an image [1]. In essence, an object detector answers the questions: *"What objects are present, and where are they?"* This capability is crucial as it provides foundational information for broader vision applications like image segmentation, tracking, and scene understanding [2]. Two key performance metrics of object detection frameworks are accuracy (both in correct classification and precise localization of objects) and speed (often measured in frames-per-second for real-time processing) [3].

Over the past two decades, object detection techniques have evolved dramatically. Earlier approaches relied on manually-crafted image features and simple classifiers, while modern methods leverage powerful machine learning models, especially deep convolutional neural networks (CNNs), to learn rich representations of objects [4][5]. The advent of deep learning has led to remarkable breakthroughs in detection accuracy and enabled real-time performance, making object detection a hot research area and enabling widespread deployment in industry [6].

Object detection today is employed in a wide range of real-world applications. For example, it is a key technology in autonomous driving, where detectors identify vehicles, pedestrians, and traffic signs to help navigate safely. It is also central to video surveillance and security systems for detecting intruders or hazardous objects, and to medical imaging in healthcare for locating anomalies like tumors in scans [7][8]. These diverse applications have different requirements (e.g. accuracy vs. speed, robustness to occlusions, scale invariance), which drive ongoing research. In this paper, we provide an intermediate-level review of object detection techniques, from the traditional methods predating deep learning to the modern machine learning approaches. We compare popular algorithms (R-CNN and its successors, YOLO, SSD, etc.), discuss their performance trade-offs, and highlight real-world use cases. We conclude with current challenges and potential future directions in this rapidly evolving field.

II. TRADITIONAL OBJECT DETECTION METHODS

Before the deep learning revolution, object detection was dominated by manual feature engineering and the sliding window paradigm. In a typical pipeline, an image was exhaustively scanned at various scales and positions by a classifier to decide whether each sub-window contained an object [9]. This approach was computationally intensive, but several innovations made it feasible on early-2000s hardware.

One milestone was the Viola-Jones detector (2001) for face detection. Viola and Jones introduced a cascaded classifier using simple Haar-like features, combined with techniques such as the integral image for fast feature computation and an AdaBoost-based feature selection mechanism [10][11]. This design achieved the first real-time face detection on a standard CPU, running 10–100× faster than contemporaries while maintaining comparable accuracy [12]. The Viola-Jones method demonstrated that with efficient features and cascaded rejection stages, sliding-window detection could run at interactive speeds, albeit primarily for rigid object classes like faces.

Another influential approach was the use of robust hand-crafted feature descriptors like SIFT and HOG. Scale-Invariant Feature Transform (SIFT), introduced by D. Lowe in 2004, extracted distinctive local keypoints that were invariant to scale and rotation. SIFT features enabled detecting specific objects by matching keypoints between images, and formed the basis for early object recognition systems. However, SIFT was more often used for object instance recognition than general object *detection* in cluttered scenes, since it excels at finding identical objects rather than arbitrary objects of a class.

The Histogram of Oriented Gradients (HOG) descriptor, proposed by Dalal and Triggs in 2005, became a cornerstone of pre-deep-learning object detection [13]. HOG features capture local shape (gradient) information on a dense grid, with normalization to improve invariance to illumination and contrast [14]. In their seminal work, Dalal and Triggs demonstrated HOG for pedestrian detection, showing significantly improved detection rates. A typical HOG-based detector would compute HOG features over fixed-size windows and use a linear SVM classifier to distinguish object vs. background. To detect objects at different sizes, the image is processed at multiple scales (an image pyramid) while the detection window size remains constant [15]. HOG, along with other descriptors of that era, substantially improved detection accuracy for rigid objects and remained the foundation of many systems for years [15].

Building on HOG, more sophisticated models like the Deformable Part-based Model (DPM) further advanced the state of the art in late 2000s. DPM (Felzenszwalb et al., 2008) combined the sliding-window approach with a part-based representation of objects [16]. In DPM, an object is modeled as a collection of parts (for example, a car is composed of sub-models for wheels, windows, etc.), arranged in a deformable configuration. The model uses HOG filters for each part and a root filter for the whole object, and it learns the optimal spatial arrangement of parts (allowing some deformation) to maximize detection score [17]. This "divide-and-conquer" strategy (detecting parts and assembling them) made DPM robust to pose and intra-class variation. Indeed, DPM won multiple years of the PASCAL VOC object detection challenge (2007–2009)

and was considered the pinnacle of the pre-deep-learning detectors [18][19]. DPM also introduced important concepts like mixture models (to handle different aspect ratios or sub-categories) and hard negative mining (iteratively improving the classifier by focusing on difficult negative examples), which would later be adopted in deep learning methods [19].

Despite these advances, traditional methods had limitations. They relied on the designer's ability to hand-craft features that capture all relevant variability of objects. Complex ensembles like DPM were algorithmically heavy, and the exhaustive sliding-window search was slow, even with speed-ups. By around 2010, detection performance had plateaued [20] – it became clear that a new approach was needed to significantly improve accuracy without sacrificing speed.

III. DEEP LEARNING–BASED OBJECT DETECTION

The breakthrough for object detection came in the early 2010s with the advent of deep learning. After the success of CNNs on image classification (AlexNet in 2012), researchers sought to apply CNN feature learning to detection. This led to a family of region-based CNN (R-CNN) detectors that dramatically improved accuracy, followed by "single-shot" detectors that achieved real-time speeds. Below, we discuss key techniques including R-CNN and its variants (Fast R-CNN, Faster R-CNN), as well as one-stage models like YOLO and SSD.

IV. TWO-STAGE DETECTORS: R-CNN, FAST R-CNN, AND FASTER R-CNN

R. Girshick *et al.*'s R-CNN (Regions with CNN features, 2014) was a seminal work that ushered in deep learning for detection. R-CNN bridged the gap between traditional detectors and CNNs by using an external region proposal method to generate possible object bounding boxes, and then classifying each proposed region with a CNN [21]. In practice, R-CNN used selective search to propose ~2000 candidate regions per image, warped each region to a fixed size, and passed it through a pretrained CNN (fine-tuned for detection) to extract features. Finally, class-specific linear SVMs classified the features, and bounding-box regressors refined the box coordinates. This multi-stage pipeline was complex but effective – R-CNN achieved a huge jump in accuracy, for example raising mAP on PASCAL VOC from the ~33% of DPM to ~58.5% [22]. This ~25 point leap was unprecedented, clearly demonstrating the power of learned CNN features over hand-crafted features. However, R-CNN's biggest drawback was its slowness: the CNN had to run ~2000 forward passes per image (one per region), resulting in only about 0.5~1 FPS even with GPU acceleration. The authors noted the redundant computation on overlapping proposals as a major inefficiency [22].

Researchers addressed R-CNN's shortcomings in subsequent upgrades. Fast R-CNN (Girshick, 2015) streamlined the pipeline by integrating proposal processing and classification into a single CNN graph [23]. Instead of cropping each region and running a separate CNN, Fast R-CNN feeds the *whole image* through a CNN once to produce a convolutional feature map. Then for each region proposal (from selective search), it uses an ROI pooling layer to extract a fixed-length feature vector from the shared feature map, which is fed into fully-connected layers for classification and

bounding-box regression. This design eliminated the redundant computations and allowed end-to-end training of the detector. The results were dramatic: Fast R-CNN achieved ~70% mAP on VOC and was 200× faster in detection time compared to R-CNN [23]. The overall detection pipeline was still two-stage (proposals + classification), but most heavy computation was shared, making it much more practical. One remaining bottleneck was that it still relied on the external region proposal algorithm, which was not learned and took significant time per image.

The next milestone, Faster R-CNN (Ren *et al.*, 2015), solved the proposal generation bottleneck by introducing a learnable Region Proposal Network (RPN). The RPN is a small CNN that operates on the convolutional feature map and yields object proposals (essentially predicting bounding boxes and objectness scores at each location) [24]. These proposals are then refined by the second stage, which is essentially Fast R-CNN (ROI pooling + classification/regression). By training the RPN and detector jointly, Faster R-CNN became an *end-to-end two-stage detector* that is highly efficient. It achieved near-real-time performance on GPUs (e.g. 5–7 FPS for VGA-resolution images) while maintaining state-of-the-art accuracy (on MS COCO it achieved ~59% AP@0.5, a big improvement) [24][25]. Faster R-CNN's unified approach (sometimes called a "region-based deep detector") set a new standard: many subsequent detectors are built on this paradigm of combining learnable proposals with deep feature extraction. Extensions like Feature Pyramid Network (FPN) further improved it by adding multi-scale feature layers [25]. Two-stage CNN detectors remained the champions of accuracy through the mid-2010s, though their inference speed, while greatly improved, was still lower than some real-time requirements (e.g. 5-7 FPS is insufficient for high-frame-rate video). This led to the pursuit of alternative, faster architectures described next.

V. ONE-STAGE DETECTORS: YOLO AND SSD

In contrast to two-stage approaches, one-stage (single-shot) detectors skip the explicit proposal generation step and directly predict object class probabilities and locations in a single evaluation of a network. This paradigm sacrifices some accuracy (especially on small objects) but vastly improves speed, making real-time object detection on live video feasible [26]. Two prominent one-stage detectors are the YOLO family and the SSD.

You Only Look Once (YOLO), introduced by Redmon *et al.* in 2016, framed detection as a regression problem done in one shot [27]. YOLO's CNN takes an input image and divides it into an $S \\times S$ grid; each grid cell outputs predictions for a fixed number of bounding boxes (with confidences) and class probabilities [28][29]. These predictions are then combined to produce final detections (with a threshold on confidence)[30]. Because the entire image is processed by the network *once*, YOLO is extremely fast. The original YOLO v1 model (with a 448×448 input) achieved about 45 FPS on a GPU, and a smaller-fast version even reached 155 FPS (albeit at lower accuracy) [31]. This was a massive speedup over two-stage detectors. YOLO's design also allows it to consider global context in the image when making predictions (since it sees the whole image), which led to fewer false detections on background regions compared to earlier methods [32].

However, the first YOLO model wasn't as accurate as state-of-the-art two-stage models; it attained roughly 63–66% mAP on PASCAL VOC, lower than Faster R-CNN's ~70+% [33][34]. It particularly struggled with small objects and precise localization, due in part to its coarse grid-based prediction (small objects might fall into the same grid cell and be missed) [35]. Subsequent versions YOLOv2, YOLOv3, and beyond improved accuracy significantly (e.g. YOLOv3 in 2018 closed much of the gap to two-stage detectors) while still running in real-time, making YOLO one of the most popular detection frameworks in practice.

Around the same time, Liu *et al.* (2016) proposed the Single Shot MultiBox Detector (SSD), another one-stage method that introduced useful ideas to boost one-stage accuracy. SSD uses a CNN (e.g. VGG-16 backbone) to generate feature maps at multiple scales and predicts detections on each of those feature maps [36][37]. By having predictions at multiple resolutions, SSD is able to detect both small and large objects better: early layers (higher resolution feature maps) handle smaller objects, while later layers (coarser resolution, more semantic) handle large objects [38]. SSD also employs a set of default anchor boxes of various shapes and scales tiled across the image, similar to the anchor concept in Faster R-CNN's RPN, but SSD directly predicts adjustments to these anchors in one pass [39][40]. The result was a one-stage detector that could match or exceed the accuracy of some two-stage methods while still being very fast. For example, with a 300×300 input, SSD300 achieved ~74% mAP on VOC2007 at 46 FPS, significantly outpacing YOLOv1's accuracy and speed [41][34]. A larger SSD512 model pushed accuracy further to ~77% mAP (surpassing Faster R-CNN), though at a cost of lower speed (~19 FPS) [42][43]. SSD's strength was its balanced design: by leveraging multi-scale features and eliminating the proposal step, it offered a sweet spot on the accuracy-speed trade-off for real-time applications.

To summarize, deep learning transformed object detection by replacing hand-crafted features with learned CNN features and by enabling end-to-end training of detection pipelines. The R-CNN family (two-stage detectors) achieved top accuracy, while models like YOLO and SSD demonstrated that real-time detection is possible with careful network design. Over time, further innovations like RetinaNet (2017) bridged the gap between one-stage and two-stage detectors by using a specialized focal loss to address class imbalance, achieving one-stage accuracy competitive with two-stage methods [44]. The landscape of object detection algorithms can thus be seen as a spectrum between speed and accuracy, as discussed next.

VI. PERFORMANCE COMPARISON AND TRADE-OFFS

Object detection techniques often face a trade-off between accuracy and speed. Two-stage detectors, which devote two passes to handling candidate regions, generally achieve higher accuracy (especially for small objects and under challenging conditions) but are slower. One-stage detectors forego the proposal step, gaining speed at the expense of some accuracy. Table 1 compares several landmark detectors on a common benchmark (PASCAL VOC 2007), and Figure 1 illustrates the speed-vs-accuracy trade-off.

Table 1: Performance of Representative Object Detectors (VOC 2007 test set) [45][34]

Detector	Year	Type	mAP (VOC'07)	Speed (FPS)
HOG + SVM (Dalal & Triggs)	2005	Traditional (sliding window)	~30% (pedestrians) [22]	<1 FPS (CPU)
DPM (Felzenszwalb *et al.*)	2008	Traditional (part-based)	33.7% [22]	~0.1 FPS (CPU)
R-CNN (Girshick)	2014	Two-stage (CNN + proposals)	58.5% [22]	0.02 FPS [45]
Fast R-CNN	2015	Two-stage (shared CNN)	70.0% [23]	0.4 FPS [46]
Faster R-CNN	2015	Two-stage (CNN+RPN)	73.2% [47][48]	~7 FPS [47][48]
YOLO v1 (Redmon)	2016	One-stage (CNN regression)	66.4% [41]	21 FPS [41][49]
SSD300 (Liu *et al.*)	2016	One-stage (multi-scale CNN)	74.3% [41]	46 FPS [50][34]
SSD512	2016	One-stage (multi-scale CNN)	77.2% [42]	19 FPS [42]

Figure 1: Speed (FPS) versus accuracy (mAP) for major object detection models on PASCAL VOC. Two-stage detectors (blue) like Faster R-CNN are highly accurate but slower, whereas one-stage detectors (orange) like SSD300 achieve real-time speeds with competitive accuracy [34][42]. (All models use a comparable CNN backbone, e.g., VGG-16, for fairness.)

From the above comparison, we observe that earlier traditional methods like HOG+SVM or DPM had quite low accuracy by today's standards and were far from real-time. The introduction of CNN-based detectors (R-CNN and Fast R-CNN) yielded substantial accuracy gains (nearly doubling mAP) [22][23], but initially these methods were extremely slow (R-CNN processed ~1 image in 50 seconds [45]). Engineering innovations (Fast/Faster R-CNN) improved the speed by orders of magnitude while also improving accuracy [23]. By 2015, Faster R-CNN could detect objects with ≈73% mAP at 5–7 FPS, which was a remarkable improvement, but still not sufficient for high-frame-rate video or time-critical systems.

On the other hand, one-stage models prioritized speed. YOLO v1, for instance, traded some accuracy to achieve real-time performance (~21–45 FPS) [31][49]. In Figure 1, YOLO sits on the high-FPS but lower-mAP end of the spectrum. SSD's results (74% mAP at 46 FPS) demonstrated that one-stage detectors can push accuracy into the 70+% mAP range while maintaining real-time speed [51]. In fact, as shown in Table 1 and Figure 1, SSD300 was the only model at the time to surpass 70% mAP *and* run at tens of FPS simultaneously [51]. This made SSD especially attractive for applications needing both good accuracy and speed. The larger SSD512 model achieved even higher accuracy, exceeding the two-stage Faster R-CNN, but its speed dropped closer to 20 FPS [42] – illustrating the constant balance between accuracy and speed.

In summary, two-stage detectors generally still outperform one-stage detectors in pure accuracy (especially on difficult benchmarks or when evaluating at strict localization criteria), but the gap has narrowed considerably with newer one-stage techniques. Techniques like focal loss (RetinaNet) and improved backbone networks have allowed one-stage methods to rival two-stage methods in accuracy, while two-stage methods have also become faster through better design (e.g. lighter R-CNN heads, optimization). Today, the choice between these approaches often depends on the application requirements: if maximal accuracy is needed and some latency is acceptable, a two-stage or advanced one-stage model may be used, whereas for real-time systems or embedded devices, efficient one-stage models (or compressed versions of two-stage models) are preferred [52][53]. The next section looks at how these detection techniques are applied in real-world scenarios.

VII. REAL-WORLD APPLICATIONS OF OBJECT DETECTION

Object detection technology has become a core component in numerous domains. We highlight a few notable real-world applications and how detection algorithms are employed:

- Autonomous Vehicles: Self-driving cars and advanced driver-assistance systems rely heavily on object detection to perceive the environment [54]. Detectors are used to identify other vehicles, pedestrians, cyclists, traffic signs, and road obstacles in real time, allowing the car's AI to make safe navigation decisions. For example, Tesla's Autopilot uses cameras with neural network detectors to recognize lane markings and vehicles [55]. In this domain, speed and accuracy are both critical: the detection system must react to hazards quickly (low latency) and with high precision to ensure safety.
- Video Surveillance and Security: In security cameras and surveillance systems, object detection is used to automatically spot people or objects of interest (e.g. detecting intruders in restricted areas, or finding unattended bags). Modern CCTV systems incorporate detectors for pedestrians, faces, or even weapons to alert security personnel of potential threats [56]. Object detection also enables people counting in smart cities and monitoring crowd density. In industrial settings, virtual "safety fences" use detectors to trigger alarms when a person approaches a dangerous zone [57]. These applications often require detecting objects in varied poses and lighting (day/night), demanding robust models (sometimes with infrared or thermal imagery for night vision).
- Healthcare and Medical Imaging: In medicine, object detection assists doctors by locating anatomical structures or anomalies in images. For instance, detectors can identify and localize tumors or polyps in radiology scans, count cells in microscopy images, or guide surgeons by detecting instruments/tissues in real time [8]. One example is using CNN-based detectors on X-rays or CT scans to highlight potential nodules (which could indicate early-stage cancer) for further examination. The challenge here is that accuracy is paramount – false negatives (missed detections) can be life-threatening – so detection models are often combined with other steps

and extensively validated. Speed is less critical than accuracy in many medical cases, but fast detection can enable real-time uses like surgical tool tracking or emergency diagnostics.

- Robotics and Manufacturing: Robots equipped with vision use object detection to recognize and localize objects they need to manipulate or avoid. For instance, a warehouse robot may detect packages of certain sizes for sorting, or an industrial robot may detect parts on an assembly line for pick-and-place. Drones with on-board cameras use object detection to track targets or avoid collisions. These settings often require lightweight models that can run on embedded hardware in real time, sometimes trading some accuracy to fit computational constraints.
- Retail and Consumer Applications: Object detection is also found in everyday consumer tech – for example, smartphone camera apps use face and human body detectors to focus and apply effects, or augmented reality apps detect physical objects (like furniture or landmarks) to overlay digital content. In retail stores, object detection can help in inventory management (e.g., detecting products on shelves) or in automated checkout systems (detecting items in a shopping cart).

These examples illustrate the versatility of object detection. Depending on the application, different detection algorithms may be favored. A surveillance system processing multiple camera feeds might opt for a very efficient one-stage model (like a Tiny-YOLO variant [58]) to handle real-time multi-stream input. In contrast, a medical imaging tool might use a heavier two-stage detector (or an ensemble) to maximize accuracy. The continual improvements in detection algorithms are expanding these applications further, but several challenges remain to be addressed for object detection to reach its full potential.

VIII. CURRENT CHALLENGES AND FUTURE DIRECTIONS

Despite tremendous progress, object detection still faces significant challenges, and ongoing research is exploring new directions to overcome them [59][60]. We conclude by outlining some key challenges and future directions:

- Small Objects and Scale Variability: Detecting very small or distant objects (e.g. a far-away pedestrian or a small lesion in an MRI) is difficult for current detectors. Small objects occupy few pixels, making it hard for CNNs to extract distinctive features. While methods like feature pyramids and super-resolution help, small object detection remains an open problem [61]. Future models may use enhanced multi-scale feature learning or higher-resolution inputs, though that can impact speed.
- Occlusion and Dense Scenes: In real-world scenes, objects may occlude each other (e.g. people in a crowded crowd or cars in traffic jams). Detecting partially visible objects or making sense of overlapping objects is challenging. Techniques like instance segmentation (detecting object masks) and depth sensing can aid here, but robust detection under heavy occlusion is still an area of active research [59].

- Open-World and Continual Learning: Most detectors are trained on a fixed set of classes and cannot recognize anything outside those classes. In an "open-world" scenario, an AI may encounter new object types and should learn to detect them incrementally without forgetting old classes. This is challenging because it requires on-the-fly learning and avoiding catastrophic forgetting. Progress is being made on incremental and few-shot detection, but enabling detectors to continuously learn new object categories (as a human would) remains largely unsolved [62]. This capability is important for long-lived robotic systems and evolving real-world applications.
- Multi-Modal Detection: Integrating information from multiple sensor modalities is a promising direction. For example, combining visual cameras with depth sensors (LiDAR or stereo) or thermal infrared cameras can improve detection in poor lighting or weather conditions [63]. Multi-modal object detection can be seen in autonomous vehicles that fuse camera and LiDAR data to detect 3D objects. However, challenges include synchronizing data, handling differing resolutions, and designing models that effectively fuse features. As new sensors become cheaper, future detectors will likely operate on richer inputs (RGB-D, thermal, radar, etc.) to improve robustness.
- Real-Time Efficiency and Edge Deployment: Even though current one-stage detectors are fast, achieving high accuracy at high FPS on low-power devices (like mobile or IoT devices) is an ongoing challenge. Models like YOLO have "tiny" versions for embedded use, but there is demand for detectors that can run at the edge with minimal latency. Techniques like model quantization, pruning, knowledge distillation, and efficient neural architectures (e.g. MobileNet, SqueezeNet backbones) are being actively explored to slim down detectors without too much loss in accuracy [64][65]. The future likely holds further optimized networks and perhaps dedicated hardware accelerators to push object detection to ubiquitous deployment.
- Beyond 2D Bounding Boxes: There is a trend toward richer output from detection systems. Instead of just 2D boxes, future detectors might output pixel-level segmentation (blur the line with segmentation tasks) or even understand 3D object properties (orientation, shape) for true scene understanding [66]. For instance, DETR (2020) is a recent transformer-based detector that predicts sets of objects in an end-to-end fashion, and others incorporate depth estimation or 3D bounding boxes. These approaches are still maturing, but they indicate a future where object detection is one part of a broader scene interpretation task, possibly incorporating transformer architectures and attention mechanisms for global reasoning.

In conclusion, object detection has evolved from simple algorithms using handcrafted features to highly sophisticated deep learning models that can detect dozens of object types in real time. Techniques like R-CNN, Fast/Faster R-CNN, YOLO, and SSD have defined the state of the art, each balancing the trade-offs between accuracy and speed in different ways. These advances have unlocked many real-world applications, from safer self-driving cars to smarter healthcare diagnostics. Yet, the journey continues: researchers are actively addressing current limitations such as improving detection of small/obscured objects, enabling continuous learning, and integrating multi-modal information. The future of object detection will likely involve more intelligent,

adaptable, and comprehensive vision systems that not only detect objects with high accuracy and speed, but also understand context and operate reliably in the open world[67]. Such progress will bring us closer to the goal of endowing machines with human-like visual perception capabilities.

REFERENCE

- [1] [2] [3] [4] [5] [6] [7] [9] [10] [11] [12] [13] [14] [15] [16] [17] [18] [19] [20] [22] [23] [24] [25] [59] [61] arxiv.org https://arxiv.org/pdf/1905.05055
- [8] [44] [52] [53] [54] [55] [56] [57] [58] [60] [62] [63] [64] [65] [66] [67] Object Detection 101: Applications, Challenges, and Future Directions · Neil Sahota https://www.neilsahota.com/object-detection-101-applications-challenges-and-future-directions/
- [21] [27] [28] [29] [30] [31] [32] [35] [1506.02640] You Only Look Once: Unified, Real-Time Object Detection https://ar5iv.labs.arxiv.org/html/1506.02640
- [26] [36] [37] [38] [39] [40] [41] [47] [50] SSD: Understanding single shot object detection | Manal El Aidouni https://manalelaidouni.github.io/Single shot object detection.html
- [33] [34] [42] [43] [45] [46] [48] [49] ssd_eccv_2016_final copy. Key http://www.cs.unc.edu/~wliu/papers/ssd_eccv2016_slide.pdf
- [51] Faster R-CNN vs YOLO vs SSD — Object Detection Algorithms | by Abonia Sojasingarayar | IBM Data Science in Practice | Medium https://medium.com/ibm-data-ai/faster-r-cnn-vs-yolo-vs-ssd-object-detection-algorithms-18badb0e02dc