

# Object detection

Viktoria Nikoleta Tsakalidou, Pavlina Mitsou

## Introduction

People have the ability from an early age to recognize objects through out the eyes when they have seen something similar. In general, the human visual system is fast and accurate, allowing us to perform complex tasks, store images and objects through memory. Always human beings were interested to have an automated way to detect objects in the environment like their ability. This strong desire is obviously shown from the prehistoric ages too. In the Ancient Greek mythology was the first sign for that desire with the bronze giant robot named Talos. (1) Talos was created by the Greek God Hephaestus and it was a gift for the King Minos to protect and detect invaders. Similar applications are all around the world in nowadays. In the world of technology, fast and accurate algorithms have been created for the detection of objects like surveillance cameras in combination with the alarms have the same ability that Talos had in that time, to detect invaders and ensure that the laws of the Crete were upheld by the inhabitants.

## Core concepts

Object recognition is a general term to describe a collection of related computer vision topics that involves identifying objects in digital images. It can be challenging at the begging to distinguish between different related computer vision tasks. (2)

- Image Classification is related to the prediction of the type or class of the dominant object in an image. The input will be an image with a single object, such as a photograph and we will expect as an output a class label (e.g. one or more integers that are mapped to class labels). (3)
- Object Localization is the localisation of the objects in an image and indicate their location with a bounding box. As an input will be an image with one or more objects and the output is one or more bounding boxes (e.g. defined by a point, width, and height). (3)
- Object Detection is the localisation of the presence of objects with a bounding box and the classification of them. Typically includes: proposing regions and then classify them. The input is an image with one or more objects and as output will be one or more bounding boxes (e.g. defined by a point, width, and height), and a class label for each bounding box.(3)

A core concept on the object recognition is the bounding box proposal that is the rectangular region that contains the object inside which can be represented as a 4-element vector either storing its 2 corner coordinates or storing its center location

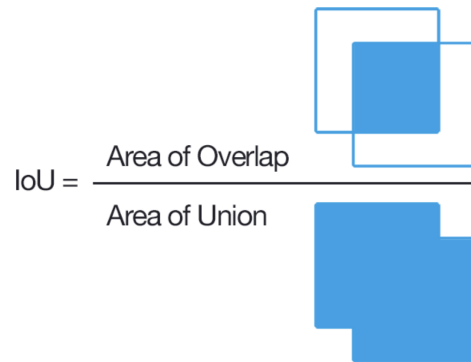

$$\text{IoU} = \frac{\text{Area of Overlap}}{\text{Area of Union}}$$

Fig. 1. Intersection over Union formula

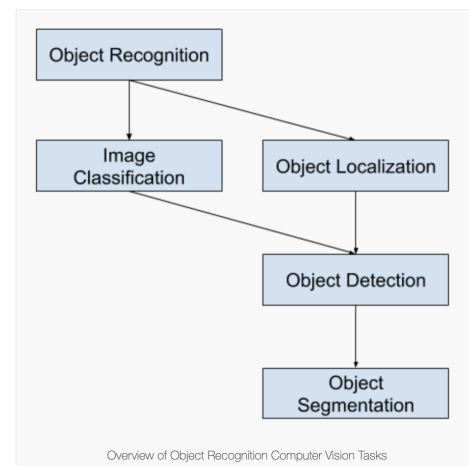


Fig. 2. Relationship between Computer Vision tasks

and its width and height(x, y, w, h). It is accompanied mostly by a confidence score that presents how likely an object is contained in the box. There is also an evaluation metric that checks the Intersection over Union (IoU) (also known as Jaccard index) by dividing the area of overlap with the area of union. The IoU evaluates how close the prediction bounding box is to the ground truth which is the manual annotated box from a human. It can be used by any algorithm that provides as output predicted bounding boxes. (4)

Non Maximum Suppression (NMS) is a common algorithm to merge these overlapping bounding boxes. If there is any bounding box that significantly overlaps with another one by checking the  $\text{IoU} > \text{IoU\_threshold}$  then the higher-confident box is suppressed. (5)

## Related work

**A. VJ Detector.** For the first time in 2001 P.Viola and M.Jones (6) achieved high detection rates of human faces without constraints like skin color segmentation etc and it

was super fast compared to the algorithms with the similar numbers in the accuracy with the "Viola-Jones" detector. This algorithm's name also was given by the creators as an honor to the significant contribution. The way that follows is straight forward and simple to understand. It slides windows in order to go through in all places and scales of the image and detect if it contains a face. The one of the most important techniques that dramatically improved the detection of VJ was the integral image. The integral image is a computational method to speed up box filtering or convolution process. Like other object detection algorithms in its time (7), the Haar wavelet is used in VJ detector as the feature representation of an image. The integral image makes the computational complexity of each window in VJ detector independent of its window size. P. Viola and M. Jones decided to use Adaboost (8) algorithm to select smaller portion of features that were more close related to face detection (about 180k dimensional). The last crucial technique that was introduced and had an impact on the performance was the detection cascades. A multi-stage detection paradigm that tried to reduce the computational overhead by spending more time on face targets rather than computation on background.

Figure 3 shows an example of VJ detector's result.

**B. HOG Detector.** The Histogram of Oriented Gradients is used in computer vision and image processing for purpose of object detection. The use of HOG was started in 2005 by N. Dalal and B. Triggs in their work to detect pedestrians in static images (9). The HOG works in local cells, it's unchanged in geometric and photometric transformations, except for the orientation of objects. HOG is also, designed to be computed in a dense grid of uniform cell distances and to use overlapping local contrast smoothing to enhance accuracy (10).

Figure 4 shows an Architecture of HOG. The image is inserted and then the pre-processing of the received image begins in order to improve the performance of the model. Then, the slope of the image is calculated. The size of the image slope is calculated as well as the direction of the image. The histogram is calculated where it separates the size of the slope according to its direction. After the histogram is created then the normalization is done for the best result of the image. We want to be independent numbers in order to have a satisfactory result. The final vector for training classifiers such as SVM is then calculated and then the object is detected. Finally, the desired result is displayed as output (11).

**C. Deformable Part-based Model (DPM).** DPM was originally an extension of the HOG detector and won the detection challenges of VOC-07, -08, and -09 by P. Felzenszwalb (10) in 2008 and after improvements was introduced and by R. Girshick. (12) (13) Related tasks that achieved leading performance, such as articulated human pose estimation (14), face detection (15) (16) and pedestrian detection (17) (18). This detection algorithm follows the "divide and conquer" philosophy. The training can be considered as the learning of a proper way of decomposition an object and the inference can be considered as an troupe of detections on differ-

ent parts. For instance the detection of a house can be decomposed on the detection of the windows, door and roof separately. Despite the fact that nowadays object detectors algorithms in terms of accuracy have far surpassed the DPM, there are a lot of them that still are deeply influenced by its insights, e.g. hard negative mining, mixture models, bounding box regression.

**D. RCNN Model Family.** The R-CNN family of algorithms refers the "Regions with CNN Features" or "Region-Based Convolutional Neural Networks". The first goal of RCNN was take an input image and produce a set of bounding boxes as output, where the each box contains an object and also the category of the object. The RCNN has been extended to perform other computer vision tasks. Below are some versions of RCNN that have been developed.

**D.1. RCNN.** The RCNN is not starts with introduction but with the extraction of a set of object proposal by selective search (19). After each proposal is rescaled to a fixed size image and fed into a CNN model trained on ImageNet (say, AlexNet ) to extract features. The end, linear SVM classifiers are used to predict the presence of an object within each region and to recognize object categories. Although RCNN has made great progress, it has some major drawbacks: It cannot be implemented real time as it takes around 47 seconds for each test image. The selective search algorithm is a fixed algorithm. Then, no learning is happening at that stage. This could lead to the generation of bad candidate region proposals (20). The redundant feature computations on a large number of overlapped proposals (over 2000 boxes form one image) leads to an extremely slow detection speed. But later in the same year , the SPPNet was proposed and get control of this problem.

As you can see in Figure 5, first you do Region proposal, where you use the selective search method to generate region proposal. This will create nearly 2000 different regions that will need to be considered. After taking each region proposal will create a feature vector representing this image in a much smaller dimension using a CNN. Along the way, feature vectors are sorted. Because, we want to detect which category of objects these attribute vectors represent, that's why we use an SVM classification. In conclusion, the Bounding Box Regressor is used to improve the average precision by 3

**D.2. Fast R-CNN.** The Fast R-CNN started from R. Girshick (21) in 2015 which is a further improvement of R-CNN and SPPNet (22), (23). The Fast RCNN give us to simultaneously train a detector and bounding box regressor under the same network configurations. On VOC07 dataset, the Fast R-CNN increased the mAP from 5.8% (R-CNN) to 70.0% while with a detection speed over 200 times faster than R-CNN. Observed that although the Fast R-CNN successfully integrates the advantages of R-CCN and SPPNet, but its detection speed is still limited by the proposal detection. To the CNN we input the image which in create the convolutional features maps. With usefulness these maps, it can extracted the regions of proposals. After, we use a Rol pooling layer to

reshape all the proposed regions into a fixed size. With this can be fed into a fully connected network.

**D.3. Faster R-CNN.** Shortly after the Fast R-CNN in the same year S. Ren et al. proposed the Faster RCNN detector (24), (25). The Faster RCNN is the first overall, and the first near-real time deep learning detector. The main contribution of Faster-RCNN is the introduction of Region Proposal Network (RPN) that enables nearly cost-free region proposals. From R-CNN to Faster-RCNN, most individual blocks of an object detection system e.g. proposal detection, feature extraction etc, have been gradually integrated into a unified, end-to-end learning framework. The architecture of Faster-RCNN is the pinnacle of the family of models and continues to achieve near state of the art results on object recognition tasks. The Faster R-CNN is composed of 3 neural networks. That's neural networks are(26):

- Feature Network, is pre-train image classification network. The function of this network is to generate very well features from the images. The output maintains the shape and structure of the original image.
- RPN, is network with 3 convolutional layers. The purpose of RPN is to generate a number of bounding boxes and called Region of Interest. The region of interest has high probability of containing any project. From this network, the output may be a number of bounding boxes identified by the pixel co-ordinates of two diagonal corners, and one value. The RCNN takes input from both the Feature Network and RPN, and generates the Final class and bounding box. The RPN and RCNN need to be trained both. Here is the complexities of Faster R-CNN lies.
- Training Detection Network: First of all, calculated the IOUs of all the 2000 or so OIs generated by the NMS following RPN against each ground truth bounding box. After the ROIs are labeled as background or foreground on the corresponding thresh values. Then, selected from the background and foreground a fixed number ROIS. Finally, some ROIs are duplicated at random, if there are not enough foreground or background ROIs to fill the fixed number.

**E. YOLO Model Family.** It was the first one-stage detector in deep learning era. It is an another popular family of object recognition models. The R-CNN models may be generally more accurate, yet the YOLO family of models are fast, much faster than R-CNN, achieving object detection in real-time.

**E.1. YOLO.** In 2015 R.Joseph was proposed the YOLO model(27). YOLO name comes from the abbreviation of "You Only Look Once". Firstly the model splits the input image into a grid of cells. Each cell is responsible for the prediction of a bounding box if the center of a bounding box falls within it. The class prediction is also based on each cell. Despite of its great improvement of detection speed, YOLO suffers from a drop of the localization accuracy compared with two-stage detectors, especially for some small objects.

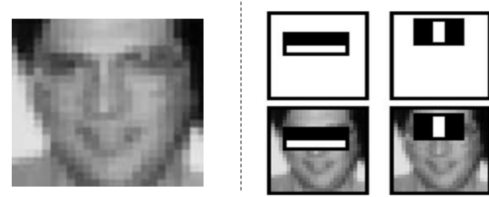


Fig. 3. Haar wavelets of VJ detector

**E.2. YOLO v2.** J. Redmon and A. Farhadi in 2016 were proposed the YOLO v2. YOLO v2 as the second version of YOLO improved significantly the accuracy while making it faster too(28). This is because YOLO v2 uses some techniques that YOLO didn't use. YOLO is one of the best models in object detection which can detect objects and process frames with the rate up to 150 FPS in small networks. YOLO v2 is trained on different architectures such as VGG-16 (29) and GoogleNet(30).

**E.3. YOLO v3.** The YOLO v3 is the third version of the YOLO. This version further refined the model architecture and training process. Although the accuracy of the models is close but not as good as Region-Based Convolutional Neural Networks (R-CNNs), they are popular for object detection speed, often demonstrated in real-time on video or with camera feed input. YOLO v3 has as the most salient feature the three different scales detections. This detection is done by applying  $1 \times 1$  detection kernels on feature maps of three different sizes and at three different places in the network (31).

**E.4. YOLO v4.** is the fourth version of the YOLO and it has been released in April 2020 from Alexey Bochkovskiy, Chien-Yao Wang and Hong-Yuan Mark Liao (32). It has the ability to recognize multiple objects in a single frame. (33). The YOLO v4 is a main improvement of YOLO v3. The modifications in the Neck and the implementation of a new architecture in the Backbone have improved the mAP by 10% and the number of FPS by 12%. Then with YOLO v4 has become easier to train this neural network on a single GPU. The CSPDarknet53(34) as a backbone spatial pyramid pooling additional module, PANet (35) path-aggregation neck and YOLO v3 head are composed the YOLO v4's architecture.

**E.5. YOLO v5.** After the discharge of YOLO v4, within two months of period, an another version of YOLO has been released and called YOLO v5. The YOLO v5 proposed from Glenn Jocher (32), who already know among the community for creating the popular PyTorch implementation of YOLO v3. YOLO v5 differentiates from all of the other prior releases, as this a PyTorch implementation rather than a fork from original Darknet. The YOLO v4 and YOLO v5 have a CSP (36) backbone and PA-NET (35) neck. The major improvements includes mosaic data augmentation and auto learning bounding box anchors.

## Software Libraries

Imagine a customer who wants to buy the same table as his friend's at an affordable price and wastes time searching on

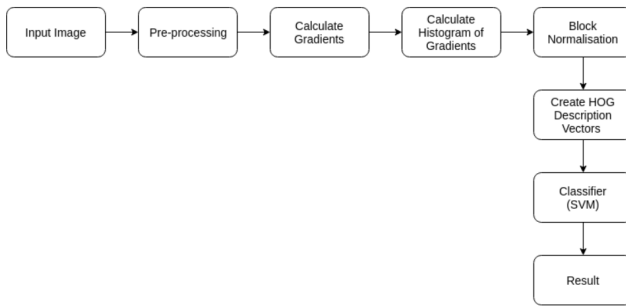


Fig. 4. Architecture of HOG

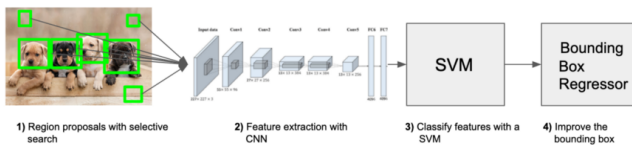


Fig. 5. The RCNN system

the internet where he could find what he is looking for by taking a photo of the table and with one click show him direct websites which have cheap prices for this table. The camera phone is a good optical input device, offering the ability to take photos and process data. Accurate real-time object detection is quite necessary in many applications. For example, Sam S. Tsai, David Chen, Vijay Chandrasekhar, Gabriel Takacs Ngai-Man Cheung, Ramakrishna Vedantham, Radek Grzeszczuk, Bernd Girod, have created a mobile product recognition application. The system they use is a local system based on features. Their application is built on a product database which contains an online product database using a Linux server. It was observed that there was a large reduction in the total product recognition response time. Also in the section of robotics a lot of object detection applications have been created. For example, Juan Fasola and Manuela Veloso (37) in their article describe an approach that renders visual detection of objects in real time by combining the power of processing the color segmented image along with that of the gray scale image of the same scene. This project was developed with in mind the annual RoboCup (38) competition and in particular the 4-Legged LEAGUE where football is played and competing Sony AIBO robot teams. To achieve all the above at the software part there are plenty of libraries that have been developed and extensively used in the industry as well as on the academia.

**F. Tensorflow.** Tensorflow is Google Brain open-source library that is used for numerical computation and large-scale machine learning in the process of acquiring data, training models, serving predictions, and refining future results. It has been developed in 2015 and at the beginning it was designed for internal usage by research team. Combines together Deep Learning and Machine learning algorithms and models that are efficiently running in C++ and there is a convenient front-end in Python. It provides the opportunity to create a graph of computations to perform. The nodes represents the mathematical operations and the edges are the data. As an Open-Source library for machine learning and deep

learning, TensorFlow is suitable for image recognition, text-based applications, voice search, and many more. Tensorflow is used by DeepFace, Facebook's image recognition system as well as by Apple's Siri for voice recognition. (39) On top of it, TensorFlow Object Detection API built which facilitated the construction, training and deployment of the object detection models. It already contains pre-trained models such as COCO (Common Objects in Context) dataset, the KITTI dataset, and the Open Images Dataset.

**G. PyTorch.** The PyTorch is, an open source machine learning library based on the Torch library(40), (41), (42). As say Stephanie Kim (43) the PyTorch was primarily developed by Facebook's artificial intelligence research group and was publicly introduced in January 2017. The Pytorch used for applications such as computer vision and natural language processing(44). Is's free and open-source software released under the Modified BSD. PyTorch provides two high-level features(45). The first is the Tensor computing with strong acceleration via graphics processing units(GPU) and the second is the Deep neural networks built on a tape-based automatic differentiation system. The PyTorch uses a method called automatic differentiation. A recorder records what operations have performed and then it replays it backward to compute the gradients. PyTorch has some strong advantages that makes so popular. One of these is the simplicity. It is very pythonic and it can integrate easily with the Python ecosystem. The second key advantage is the Great API that the PyTorch offers in term of usability due to the fact that is better designed with Object Oriented classes which encapsulate all of the major data choices along with the choice of model architecture. And third the Dynamic Graphs that the PyTorch implements dynamic computational graphs. Which means that the network can change behavior as it's being run, with little or no overhead. This is extremely helpful for debugging and also for constructing sophisticated models with minimal effort(46). Tensors are the workhorse of PyTorch. PyTorch has an extensive library of operations on them provided by the torch module. PyTorch Tensors are very close to the popular NumPy arrays. Also, one important thing is that the PyTorch allows tensors to keep track of the operations performed on them that helps to compute gradients or derivatives of output with respect to any of it's inputs(46).

**H. OpenCV.** Open CV is an open source library that contains functions for computer vision. It was created by Intel's research laboratories and continued to be developed by Willow Garage's research laboratories, and is now supported by its (47)researchers who developed the OpenCV. It's written in C++, C, Python and Java code and supports Windows, Linux, MacOS, iOS and Android. It has developed and continues to grow from research centers, containing code from basic structures of computational vision to complex and specialized algorithms that are optimized for better performance. In addition, the development of computational vision has included sections related to artificial intelligence and various mathematical structures that apply in the field of computational vision. The library is implemented in C/C++ and has



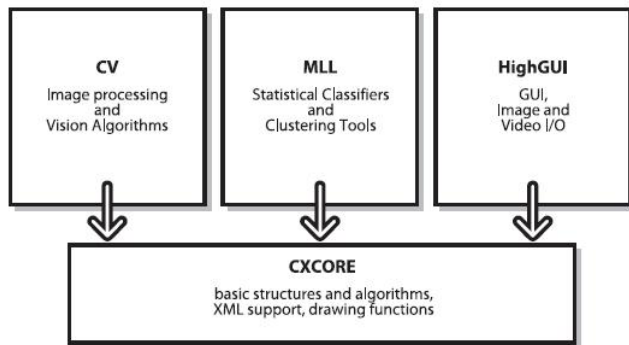


Fig. 6. Open CV building diagram

interfaces for using and customization of algorithms in different languages of different levels such as Python, Java, Matlab and others. Supported on most software platforms. Open CV aims in the maximum algorithm performance and contains methods that are used by specialized hardware, such as graphical processing units on processors and graphics cards using the CUDA or Open CL platform.

According to the Figure 6, the library is built in 5 parts(48).

- The CV component contains the basic image processing algorithms for motion analysis, feature detection, video, and generally higher-level it algorithms.
- The ML is the machine learning library and contains algorithms that OpenCV uses when running its algorithms as well as the OpenCV user. Includes many statistical sorters and grouping tools.
- The HighGUI contains processes for reading and storing files, images and videos as well as providing a simple graphical interface for setting up algorithms and displaying results.
- The CXCore contains basic data structures and content.
- End the CvAux, which contains both failed areas (integrated HMM facial recognition) and experimental algorithms (background/foreground segmentation) has been developed and merged with the CV component.

## Conclusions

In this paper have been considered some milestone detectors and some interesting software libraries that are used for object detection. After our extensive research and investigations, we conclude to the result that for a real time object detection system the most suitable family of algorithms is the YOLO family that has satisfactory results and it is super fast which is mandatory for this kind of application. From the software libraries, PyTorch that supports YOLOv5 looks to have promising results and it's worth to have a try for the application that we will build in the future.

## Bibliography

1. R.Graves. *The Greek Myths: Complete Edition*. Penguin, 1993.

2. A gentle introduction to object recognition with deep learning. <https://machinelearningmastery.com/object-recognition-with-deep-learning>.
3. Mobile product recognition. <https://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.208.3190&rep=rep1&type=pdf>.
4. Hamid Rezaatofghi, Nathan Tsoi, JunYoung Gwak, Amir Sadeghian, Ian Reid, and Silvio Savarese. Generalized intersection over union: A metric and a loss for bounding box regression. pages 658–666, 06 2019. doi: 10.1109/CVPR.2019.00075.
5. J. Hosang, R. Benenson, and B. Schiele. Learning non-maximum suppression. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 6469–6477, 2017. doi: 10.1109/CVPR.2017.685.
6. Paul Viola and Michael Jones. Rapid object detection using a boosted cascade of simple features. volume 1, pages I–511, 02 2001. ISBN 0-7695-1272-0. doi: 10.1109/CVPR.2001.990517.
7. C. P. Papageorgiou, M. Oren, and T. Poggio. A general framework for object detection. In *Sixth International Conference on Computer Vision (IEEE Cat. No.98CH36271)*, pages 555–562, 1998. doi: 10.1109/ICCV.1998.710772.
8. D. G. Lowe. Object recognition from local scale-invariant features. In *Proceedings of the Seventh IEEE International Conference on Computer Vision*, volume 2, pages 1150–1157 vol.2, 1999. doi: 10.1109/ICCV.1999.790410.
9. N. Dalal and B. Triggs. Histograms of oriented gradients for human detection. In *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, volume 1, pages 886–893 vol. 1, 2005. doi: 10.1109/CVPR.2005.177.
10. P. Felzenszwalb, D. McAllester, and D. Ramanan. A discriminatively trained, multiscale, deformable part model. In *2008 IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–8, 2008. doi: 10.1109/CVPR.2008.4587597.
11. Using histogram of oriented gradients (hog) for object detection. [https://iq.opengenus.org/object-detection-with-histogram-of-oriented-gradients-hog/?fbclid=IwAR3S7URbW1DJzV\\_AbJF8rvqPJ2d07GNY\\_E7Ts\\_XZrVLRMDHDXkLFY381A2A](https://iq.opengenus.org/object-detection-with-histogram-of-oriented-gradients-hog/?fbclid=IwAR3S7URbW1DJzV_AbJF8rvqPJ2d07GNY_E7Ts_XZrVLRMDHDXkLFY381A2A).
12. P. F. Felzenszwalb, R. B. Girshick, D. McAllester, and D. Ramanan. Object detection with discriminatively trained part-based models. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 32(9):1627–1645, 2010. doi: 10.1109/TPAMI.2009.167.
13. P. F. Felzenszwalb, R. B. Girshick, and D. McAllester. Cascade object detection with deformable part models. In *2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 2241–2248, 2010. doi: 10.1109/CVPR.2010.5539906.
14. Y. Yang and D. Ramanan. Articulated pose estimation with flexible mixtures-of-parts. In *CVPR 2011*, pages 1385–1392, 2011. doi: 10.1109/CVPR.2011.5995741.
15. Junjie Yan, Xucong Zhang, Zhen Lei, Dong Yi, and S. Z. Li. Structural models for face detection. In *2013 10th IEEE International Conference and Workshops on Automatic Face and Gesture Recognition (FG)*, pages 1–6, 2013. doi: 10.1109/FG.2013.6553703.
16. X. Zhu and D. Ramanan. Face detection, pose estimation, and landmark localization in the wild. In *2012 IEEE Conference on Computer Vision and Pattern Recognition*, pages 2879–2886, 2012. doi: 10.1109/CVPR.2012.6248014.
17. J. Yan, X. Zhang, Z. Lei, S. Liao, and S. Z. Li. Robust multi-resolution pedestrian detection in traffic scenes. In *2013 IEEE Conference on Computer Vision and Pattern Recognition*, pages 3033–3040, 2013. doi: 10.1109/CVPR.2013.390.
18. J. Yan, Z. Lei, D. Yi, and S. Z. Li. Multi-pedestrian detection in crowded scenes: A global view. In *2012 IEEE Conference on Computer Vision and Pattern Recognition*, pages 3124–3129, 2012. doi: 10.1109/CVPR.2012.6248045.
19. A. M. Smeulders, K. A. van de Sande, J. R. Uijlings, and T. Gevers. Segmentation as selective search for object recognition. In *2011 IEEE International Conference on Computer Vision (ICCV 2011)*, pages 1879–1886, Los Alamitos, CA, USA, nov 2011. IEEE Computer Society. doi: 10.1109/ICCV.2011.6126456.
20. R-cnn, fast r-cnn, faster r-cnn, yolo — object detection. <https://towardsdatascience.com/r-cnn-fast-r-cnn-faster-r-cnn-yolo-object-detection-algorithms>.
21. R. Girshick. Fast r-cnn. In *2015 IEEE International Conference on Computer Vision (ICCV)*, pages 1440–1448, 2015. doi: 10.1109/ICCV.2015.169.
22. Rich feature hierarchies for accurate object detection and semantic segmentation. [https://www.cv-foundation.org/openaccess/content\\_cvpr\\_2014/papers/Girshick\\_Rich\\_Feature\\_Hierarchies\\_2014\\_CVPR\\_paper.pdf](https://www.cv-foundation.org/openaccess/content_cvpr_2014/papers/Girshick_Rich_Feature_Hierarchies_2014_CVPR_paper.pdf).
23. Spatial pyramid pooling in deep convolutional networks for visual recognition. <https://arxiv.org/pdf/1406.4729>.
24. Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In C. Cortes, N. Lawrence, D. Lee, M. Sugiyama, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 28, pages 91–99. Curran Associates, Inc., 2015.
25. S. Ren, K. He, R. Girshick, and J. Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 39(6):1137–1149, 2017. doi: 10.1109/TPAMI.2016.2577031.
26. Faster r-cnn: Towards real-time object detection with region proposal networks. <https://arxiv.org/pdf/1506.01497.pdf>.
27. You only look once: Unified, real-time object detection. <https://arxiv.org/abs/1506.02640>.
28. <https://arxiv.org/abs/1612.08242>. <https://arxiv.org/abs/1612.08242>.
29. Vgg16 – convolutional network for classification and detection. <https://neurohive.io/en/popular-networks/vgg16/>.
30. Understanding googlenet model – cnn architecture. <https://www.geeksforgeeks.org/understanding-googlenet-model-cnn-architecture/>.
31. Yolo3: An incremental improvement. <https://pjreddie.com/media/files/papers/YOLOv3.pdf>.
32. Yolo5 is here: State-of-the-art object detection at 140 fps. <https://blog.roboflow.com/yolov5-is-here/>.
33. Yolo4: Optimal speed and accuracy of object detection. <https://arxiv.org/abs/2004.10934v1>.
34. Cspdarknet53. <https://paperswithcode.com/method/cspdarknet53>.

35. Panet: Path aggregation network in yolov4. <https://medium.com/cliq-ue-org/panet-path-aggregation-network-in-yolov4-b1a6dd09d158>.
36. Center and scale prediction: A box-free approach for pedestrian and face detection. <https://arxiv.org/abs/1904.02948>,.
37. J. Fasola and M. Veloso. Real-time object detection using segmented and grayscale images. In *Proceedings 2006 IEEE International Conference on Robotics and Automation, 2006. ICRA 2006.*, pages 4088–4093, 2006. doi: 10.1109/ROBOT.2006.1642330.
38. H. Kitano, M. Asada, Y. Kuniyoshi, I. Noda, Eiichi Osawa, and Hitoshi Matsubara. Robocup: A challenge problem for ai. *AI Mag.*, 18:73–85, 1997.
39. Real-time object detection using tensorflow. <https://www.mygreatlearning.com/blog/object-detection-using-tensorflow/>.
40. Facebook brings gpu-powered machine learning to python. <https://www.infoworld.com/article/3159120/facebook-brings-gpu-powered-machine-learning-to-python.html>,.
41. Why ai and machine learning researchers are beginning to embrace pytorch. <https://www.oreilly.com/radar/podcast/why-ai-and-machine-learning-researchers-are-beginning-to-embrace-pytorch>,.
42. Introduction to pytorch. [https://link.springer.com/chapter/10.1007%2F978-1-4842-2766-4\\_12](https://link.springer.com/chapter/10.1007%2F978-1-4842-2766-4_12),.
43. Exploring the deep learning framework pytorch. <https://algorithmia.com/blog/exploring-the-deep-learning-framework-pytorch>,.
44. Natural language processing (nlp) with pytorch. <https://dl4nlp.info/en/latest>.
45. Pytorch. <https://web.archive.org/web/20180615190804/https://pytorch.org/about>,.
46. The state of machine learning frameworks in 2019. <https://thegradient.pub/state-of-ml-frameworks-2019-pytorch-dominates-research-tensorflow-dominates-industry>,.
47. Intel acquires itseez. <https://opencv.org/intel-acquires-itseez>.
48. 1 a review on opencv. [https://www.researchgate.net/publication/280977983\\_A\\_Review\\_on\\_OpenCV](https://www.researchgate.net/publication/280977983_A_Review_on_OpenCV).