PEOPLE’S DEMOCRATIC REPUBLIC OF ALGERIA
Ministry of Higher Education And Scientific Research

University of Tlemcen – ABU BAKR BELKAID - Algeria
FACULTY OF SCIENCE
DEPARTMENT of COMPUTER SCIENCE

Graduation Project

To obtain a Master’s degree in Computer Science

Specialty: Intelligent Model and Decision (M.I.D)

On the subject:

Object detection in images using Faster R-CNN approach

Presented by:

TERBECHE Radjaa

ZEDDOUN Asma

Supervised by:

Mrs. CHAOUCHE RAMDANE Lamia

Examined on June 25 by:

Mr. HADJILA Fethallah	(President)
Mr. BERRABAH Sid Ahmed	(Examiner)

Academic year: 2022/2023

Abstract

Object detection in images is a significant research area in computer vision, enabling machines to locate and identify objects within images in real-time. This study aims to assess the effectiveness of the Faster R-CNN approach for object detection, implementing and testing several pre-trained models on the COCO dataset, and applying them to a test set from the challenging PASCAL VOC dataset. Despite some limitations, the study underscores Faster R-CNN's effectiveness and adaptability in handling diverse object detection tasks and its potential as a strong foundation for future research

Key words: Object detection, Deep learning, Faster R-CNN, Convolutional neural networks.

Résumé

La détection d’objets dans les images est un domaine de recherche important en vision par ordinateur, permettant aux machines de localiser et d’identifier les objets dans les images en temps réel. Cette étude vise à évaluer l’efficacité de l’approche Faster R-CNN pour la détection, la mise en œuvre et la mise à l’essai de plusieurs modèles pré-entrainés sur la base des données COCO, et leur application à un ensemble de tests à partir de PASCAL VOC. Malgré certaines limites, l’étude souligne l’efficacité et l’adaptabilité de Faster R-CNN dans la gestion de diverses tâches de détection d’objets et son potentiel comme base solide pour de futures recherches

Mots clés : Détection d’objets, Deep learning, Faster R-CNN, Réseaux neuronaux convolutionnels.

ملخّـص

يعد اكتشاف الكائنات في الصور مجالا بحثيا مهما في رؤية الكمبيوتر ، مما يمكن الآلات من تحديد و تصنيف الصور في الوقت الفعلي . تهدف هذه الدراسة إلى تقييم فعالية نهج Faster Rcnn لاكتشاف الأجسام ، و تنفيذ و اختبار العديد من النماذج المدربة مسبقا على مجموعة بيانات COCO ، و تطبيقها على مجموعة اختبار من مجموعة بيانات PASCAL VOC الصعبة . على الرغم من بعض القيود ، تؤكد الدراسة على فعالية Faster Rcnn و قابليتها للتكيف في التعامل مع مهام اكتشاف الكائنات المتنوعة و إمكاناتها كأساس قوي للبحث المستقبلي .

الكلمات الرئيسية : اكتشاف الكائن ، Faster R-CNN ، التعلم العميق ، الشبكات العصبية التلافيفية.

Dedicaces

We extend our appreciation to our families and friends, specially our parents for their unwavering support, patience, and understanding. Their love, encouragement, and belief in our abilities have been the driving force behind our perseverance and success.

That is why we are dedicating this work to them, wishing them health and wellness

Acknowledgments

First, we thank God the Almighty for giving us the strength and courage and patience to complete this modest work.

We extend our heartfelt gratitude to our supervisor. We are deeply thankful for your guidance, unwavering support and constructive critiques, which contributed immensely the successful completion of this work.

We also thank the jury members who agreed to read and evaluate this work.

We express our heartfelt gratitude to all the individuals who have directly or indirectly contributed to this work. Your contributions, whether big or small, have played a significant role in shaping our academic journey and the successful completion of this research. May God's blessings be upon you all.

Content table

General Introduction 12

Chapter I: Traditional methods of object detection 13

I.1. Introduction 14

I.2. Computer vision 14

I.2.1. Computer vision application 14

I.3. Object detection 15

I.3.1. History 16

I.3.1.1. Feature-based methods 16

I.3.1.2. Sliding window-based methods 16

I.3.2.3. Deep learning-based methods 16

I.3.1.4. One-stage and two-stage methods 16

I.3.2. Object detection challenges 17

I.4. Viola-Jones detection algorithm (VJ) 17

I.4.1. Histogram equalization 18

I.4.2. Haar-like features 18

I.4.3. Integral image 19

I.4.4. Ada boost 20

I.4.5. Cascading 20

I.5. Scale Invariant and Feature transform 20

I.5.1. Scale-space extrema detection 21

I.5.2. Keypoint localization 21

I.5.3. Orientation assignment 21

I.5.4. Descriptor computation 21

I.5.5. Keypoint matching 21

I.5.6. Transformation estimation 22

I.6. Histograms of Oriented Gradients (HOG) 22

I.6.1. Gamma / Color Normalization 22

I.6.2. Gradient Computation 23

I.6.3. Spatial / Orientation Binning 23

I.6.4. Normalization and Descriptor Blocks 23

I.6.5. Detector Window and Context 24

I.6.6. Classifier 24
I.7. Speeded Up Robust Features (SURF) 24
I.7.1. Scale-space extrema detection........ 25
I.7.2. Keypoint localization 25
I.7.3. Orientation Assignment 25
I.7.4. Feature Descriptor........ 26
I.8. Deformable Parts Model DPM........ 26
I.8.1. Feature Computation 27
I.8.2. Root filter 27
I.8.3. Part filter 27
I.8.4. Part models 27
I.8.5. Deformation models........ 28
I.9. Traditional methods limitations........ 28
I.10. Conclusion 28

Chapter II: Object detection using Faster R-CNN approach……………….29

II.1. Introduction 30
II.2. Convolutional neural network........ 30
II.2.1. CNN architecture 30
II.2.1.1. Convolutional layers 31
II.2.1.2. Activation function 32
II.2.1.3. Pooling layers........ 33
II.2.1.4. Fully connected layers 34
II.2.1.5. Loss function........ 35
II.2.1.6. Training 36
II.2.2. Popular architecture of CNN 37
II.2.3. OverFeat........ 38
II.3. Region based methods 38
II.3.1. Region proposal techniques 39
II.3.1.1. Selective search 39
II.3.1.2. EdgeBoxes........ 41
II.3.1.3. Region proposal networks (RPN) 42
II.3.2. Evolution of R-CNN to Faster R-CNN........ 44
II.3.2.1. Regions based Convolutional Neural Networks (R-CNN) 44
II.3.2.2. Fast R-CNN 47

II.3.2.3. The Need for Further Advancements 49
II.3.3. Faster R-CNN 50
II.3.3.1. Faster R-CNN architecture 50
II.3.3.2. Integration of RPN 51
II.3.3.4. The anchoring mechanism 51
II.3.3.3. Unified Training in Faster R-CNN 51
II.3.3.5. Strengths of Faster R-CNN over RCNN and Fast RCNN 52
I.4. Other approaches 54
I.5. Conclusion 55
Chapter III: Implementation and Faster R-CNN results 56
III.1. Introduction 57
III.2. Backbone Network Architecture 57
II.2.1. ResNet-50 57
II.2.2. Mobile Net 58
III.3. Dataset 59
III.3.1. Pascal Voc 2012 59
III.3.2. COCO 59
III.4. Software and Hardware setup 59
III.4.1. Hardware tools 59
III.4.2. Software 60
III.4.2.1. Libraries and Frameworks 60
III.4.2.2. LabelImg tool 60
III.5. Metrics 60
III.5.1. Intersection over Union (IoU) 61
III.5.2. Average Precision (AP) 61
III.5.3. Mean Average Precision (mAP) 62
III.6. Faster R-CNN pipeline 62
III.6.1. Preprocessing and Data preparation 62
III.6.2. Model configuration 63
III.6.3. Training Phase: 63
III.6.3. Testing phase 64
III.6.4. Model Tuning 64
III.7. Working method 64
III.8. Performance and Evaluation 67

III.9. Conclusion 73
General Conclusion 74

Figure table

Fig I - 1: Object detection result 16
Fig I - 2: Output of viola-jones detector 18
Fig I - 3: Three famous prototypes and their derivatives 19
Fig I - 4: The value of the integral image at pixel (x,y) is the sum of all the pixels above and to the left (grey area). (b): Matrix on viola jones algorithm 19
Fig I - 5: Example of SIFT results 22
Fig I - 6: Example of the output of the HOG detector 24
Fig I - 7: Left to right: the (discretised and cropped) Gaussian second order partial derivatives in y-direction and xy-direction, and our approximations thereof using box filters. The grey regions are equal to zero 25
Fig I - 8: Detection obtained using DPM 26

Fig II- 1: The architecture of CNN 31
Fig II- 2: Convolutional operation 32
Fig II- 3: Example of max pooling 33
Fig II- 4: Fully connected layers 34
Fig II- 5: Selective search method 40
Fig II- 6: EdgeBoxes method 41
Fig II- 7: Region Proposal Network 43
Fig II- 8: Anchor boxes for object detection 44
Fig II- 9: R-CNN architecture 44
Fig II- 10: the effect of non-max suppression 46
Fig II- 11: Fast R-CNN architecture 47
Fig II- 12: Example of RoI projection 48
Fig II- 13: Faster R-CNN architecture 50
Fig II- 14: R-CNNs series 53
Fig II- 15: Faster R-CNN results 54

Fig III- 1 : Resnet-50 architecture 57
Fig III- 2: MobileNet architecture 58
Fig III- 3 : Create annotations with LabelImg tool 66
Fig III- 4: Time executing and the detection results of simple and complex image by 2 models ... 69
Fig III- 5 : The Faster R-CNN results of an image with its ground truth 70
Fig III- 6 : Results of fasterrcnn_resnet50_fpn_v2 pre-trained 72

Table List

Table II- 1: R-CNN speed comparison 52

Table III- 1: mAP@0.6 results for different pre-trained models and different number of images. 67
Table III- 2 : AP@0.6 results for each class in PASCAL VOC dataset 68
Table III- 3: Executing time for 2 type of images and 2 different Faster R-CNN models 68
Table III- 4: The results of mAP measure with augmentation of IOU threshold 70

Abbreviations

Adam	Adaptive Moment Estimation
AP	Average Precision
CNN	Convolutional Neural Network
DoG	Difference of Gaussians
DPM	Deformable Parts Model
HOG	Histogram of Oriented Gradients
IoU	Intersection-over-Union
LoG	Laplacian of Gaussians
mAP	Mean Average Precision
NMS	Non Maximum Suppression
RANSAC	Random Sample Consensus
ReLU	Rectified Linear Unit
RoIs	Region of Interests
RPN	Region Proposal Networks
R-CNN	Region Convolution Neural Network
SIFT	Scale Invariant Features Transform
SGD	Stochastic Gradient Descent
SURF	Speeded Up Robust Features
SSD	Single-Shot Detector
SVM	Support Vector Machine
VGG	Visual Geometry Group
VJ	Viola-Jones
YOLO	You Only Look Once

General Introduction

Object detection is a significant and challenging task in the field of computer vision as it involves locating and identifying objects within images or videos, enabling machines to understand and interpret visual content. In recent years, the capability to detect and identify objects within images has become increasingly critical due to the exponential growth in digital image data and the expanding need for real-time processing. This capability has potential applications in numerous fields, including autonomous driving, surveillance, medical imaging, and robotics, among others. Traditional object detection methods have served us for many years but their limitations, particularly with regard to accuracy and processing speed, have constrained their applicability in an era demanding real-time results.

This leads us to the domain of deep learning, which has shown remarkable advancements in the field of object detection. Specifically, the Faster Region-based Convolutional Neural Network (Faster R-CNN), a state-of-the-art technique has been at the forefront of these developments. Faster R-CNN significantly mitigates the problem of computational inefficiency posed by its predecessors (R-CNN and Fast R-CNN), offering superior detection accuracy in real-time.

In this study, the main objective is to provide an in-depth understanding of the Faster R-CNN approach and implement it for object detection in images.

- Chapter 1 focuses on traditional object detection methods, setting the historical context for object detection, and illustrating the limitations that led to the development of modern deep learning-based techniques.
- Chapter 2 introduces Convolutional Neural Networks (CNNs), explaining how they work and why they are so effective at image classification tasks. It then transitions into the developments that led to the Region-based Convolutional Neural Networks (R-CNNs) and its subsequent improvements in the form of Fast R-CNN and finally the Faster R-CNN, which brought together the Fast R-CNN detector and region proposal network (RPN), demonstrating its advantages over its predecessors, introducing real-time capabilities while maintaining high accuracy.
- Chapter 03 presents the main portion of our work, where we detail the implementation of the Faster R-CNN approach for object detection. We also provide a comprehensive analysis of our results and draw a comparative study between the Faster R-CNN pre-trained models, highlighting the improvements brought forth by the Faster R-CNN.

And finally, we will conclude this dissertation with a general conclusion and some perspectives.

Chapter I
Traditional methods of object detection

I.1. Introduction

Object detection is the process of locating instances of objects of interest within an image. In computer vision, this problem has been extensively studied, and traditional methods of object detection have played a significant role in its development before the advent of deep learning-based approaches. These methods generally consisted of a series of hand-crafted features and algorithms for detecting objects based on these features. Traditional methods typically involved extracting low-level image features such as edges, corners, or texture descriptors, selecting a subset of the features, training a machine learning algorithm on the selected features, and using the trained model to detect instances of the object in new images.

In this chapter, we will provide a comprehensive overview of traditional methods of object detection, including their advantages, limitations, and applications. We will also discuss some of the most popular algorithms and techniques used in these methods.

I.2. Computer vision

Computer vision is a field of artificial intelligence focused on enabling computers to interpret, analyze, and understand visual data from the world around them. It involves the development of algorithms, models, and systems that can process digital Images or videos and extract meaningful information from them, such as object detection and recognition, facial recognition, image segmentation, motion analysis, and 3D reconstruction.

Computer vision algorithms are typically designed to mimic the human visual system, which is able to perceive and interpret complex visual scenes in real-time. However, while the human visual system relies on a network of interconnected neurons and biological processes, computer vision relies on the use of mathematical and computational models. These models use techniques such as machine learning, deep learning, and neural networks to analyze and understand visual data.

The development of computer vision has been made possible by advances in hardware, such as high-speed cameras and powerful processors, as well as the availability of large datasets for training and testing computer vision algorithms. As computer vision continues to evolve and improve, it has the potential to transform a wide range of industries, from healthcare to transportation and beyond.

I.2.1. Computer vision application

Computer vision (CV) has applications in a wide range of areas, including image and video analysis, object recognition, surveillance, robotics, medical imaging, and augmented and virtual reality [Szeliski, 2010].

Object Detection: Object detection involves identifying and localizing objects in images or videos. This technology is used in a variety of applications, including surveillance systems, autonomous vehicles, and robotics. Some popular object detection algorithms are Viola-Jones (VJ), Histogram of Oriented Gradients (HOG), Deformable Part Models(DPM), Faster R-CNN, You Only Look Once (YOLO), and Single-Shot Detector (SSD).

Face Recognition: Face recognition is the process of identifying or verifying the identity of a person from their digital image or video frame. It has various applications, such as in security systems, access control, and law enforcement. Some of the popular face recognition algorithms are FaceNet, VGGFace, and DeepFace.

Medical Image Analysis: CV techniques are used to analyze medical images for detecting diseases, analyzing magnetic resonance imaging scans, monitoring patient's vital signs, and more. It helps radiologists and doctors make more accurate diagnoses. Deep learning-based methods such as convolutional neural networks (CNNs) have shown promising results in medical image analysis [Litjens et al, 2017].

Autonomous Vehicles: CV is utilized in autonomous vehicles to detect objects, lanes, other cars, pedestrians, and road signs to navigate safely. It's a crucial component of the system that enables self-driving cars. Popular autonomous vehicle platforms such as Waymo and Tesla rely heavily on computer vision technology [Kyriakos et al, 2017].

Augmented Reality: CV is essential in AR for tracking user's movements and overlaying digital information in real world scenes. This technology is widely used in games, education, retail, and navigation [Klein et al, 2007].

I.3. Object detection

Object detection is a fundamental computer vision task that involves identifying and localizing specific object classes within an image or video. The goal of object detection is to automatically locate and classify objects of interest within an image or video frame (as shown in Fig I-1). It has numerous real-word applications, including autonomous driving, surveillance, robotics, and object recognition [Ren et al, 2016].

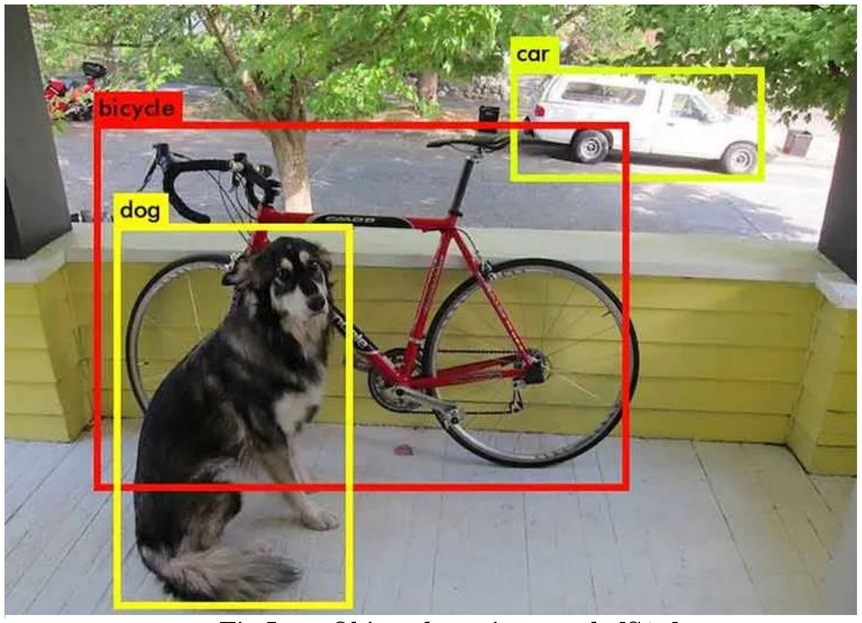


Fig I - 1: Object detection result [S01]

I.3.1. History

Object detection has been an active area of research in computer vision for several decades. Here is a brief historical overview of some of the important developments in the field:

I.3.1.1. Feature-based methods: In the early days of computer vision, feature-based methods such as the Viola-Jones algorithm were widely used for object detection. These methods involved manually designing features, such as Haar-like features, and using classifiers to detect objects based on these features [Viola & Jones, 2001].

I.3.1.2. Sliding window-based methods: Sliding window-based methods involve sliding a window of fixed size over an image and classifying each window as containing an object or not. This approach was popularized by the work of Dalal and Triggs, who used Histograms of Oriented Gradients (HOG) features and a support vector machine (SVM) classifier to detect pedestrians in images [Dallal & Triggs, 2005].

I.3.2.3. Deep learning-based methods: In recent years, deep learning-based methods have achieved state-of-the-art results in object detection. The breakthrough came with the development of the R-CNN algorithm, which uses a region proposal network (RPN) to generate candidate object regions and a convolutional neural network (CNN) to classify and refine these regions.

I.3.1.4. One-stage and two-stage methods: Recent advances in object detection have focused on improving the speed and accuracy of one-stage and two-stage methods. One-stage methods, such as YOLO [Redmon et al, 2016]. and SSD [Liu et al, 2016], detect objects directly from a single feedforward pass through a neural network. Two-stage methods, such as Faster R-CNN, use a separate region proposal network to generate candidate object regions, followed by a classification and regression network to refine these regions [Ren et al, 2016].

I.3.2. Object detection challenges

Object detection is a complex and challenging task in computer vision. Despite significant progress in recent years, many challenges still exist.

Scale variation: Objects can appear in different scales, which can make it challenging for object detection algorithms to detect them accurately [He et al, 2015].

Occlusion and truncation: parts of objects may be hidden behind other objects (occlusion), or parts of objects may be cut off at the edge of an image (truncation). These phenomena make detection difficult [Zhu et al, 2006].

Lighting conditions: Changes in lighting or poor lighting conditions, such as low light or harsh shadows, can affect the appearance of object in an image, making it difficult to recognize them [Lin et al, 2017b].

Object deformation: Objects can be deformed or distorted in shape [Felzenszwalb et al, 2010].

Background clutter: The presence of similar features in the background can confuse the detector [Viola & Jones, 2001].

Class imbalance: Some object classes may be rare or underrepresented in a dataset, object detection algorithms can struggle to detect the less common class [Lin et al, 2017a].

Real-Time processing: Ensuring object detection occurs in real-time is a significant challenge, especially when processing power is limited [Redmon et al, 2016].

Adversarial Attacks: Adversarial examples, which are intentionally designed to deceive the detection model, pose a significant challenge [Szegedy et al, 2013].

I.4. Viola-Jones detection algorithm (VJ)

Viola-Jones it is a detector developed by “Paul Viola” and “Michael Jones” in 2001. this detector care about the detection of human faces and for the first time without any constraints like skin color segmentation. Running on a 700 MHz Pentium III CPU, this detector was tens or even hundreds of times faster than other algorithms in its time under comparable detection accuracy.

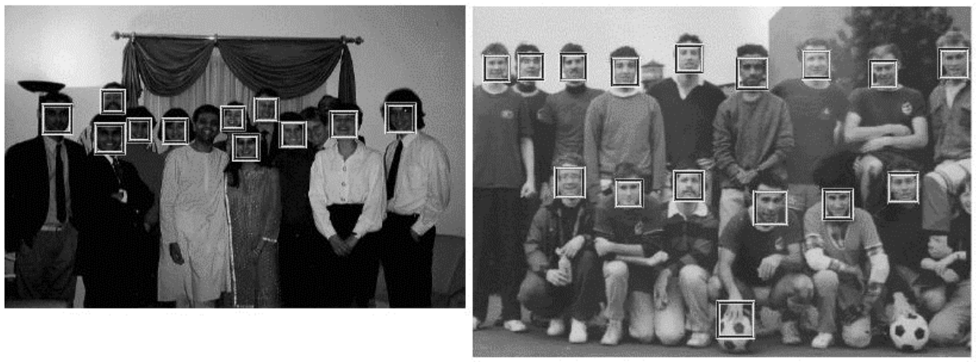


Fig I - 2: Output of viola-jones detector [Viola & Jones, 2001]

the VJ detector follows a most straightforward way of detection like sliding window. VJ detector has three important techniques: “integral image”,“feature selection” and “detection cascades” [Viola & Jones, 2001]

I.4.1. Histogram equalization

Histogram equalization is a technique for adjusting image intensities to enhance contrast. this technique depends on the effective distribution of intensity values. this method improves the general contrast of the image especially when using images in datasets for machine learning.

I.4.2. Haar-like features

“Haar-like” features are a different types of rectangular image features, that are used for object and face detection [Lienh & Maydt, 2002]. The first use was in “Viola Jones” algorithm, then they used in many of computer vision applications [S02].

"Haar-like" features are rectangular features that compute a value which represent the difference between the sum of pixel intensities in two adjacent rectangular regions : black region and white region , this value named "feature value"

Feature value = white average – black average [Takshi et al, 2005]

Haar-like” features have a lot of prototypes that vary in distribution between the white part and the black part where that they have the same dimension, and a lot of prototypes can be derived by changing the size and rotation [Lienh & Maydt, 2002].

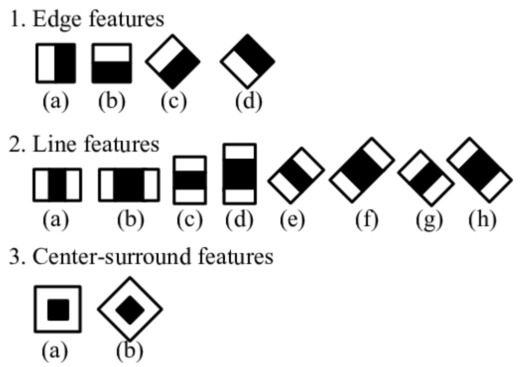


Fig I - 3: Three famous prototypes and their derivatives [Lienh & Maydt, 2002]

I.4.3. Integral image

To calculate "Haar-like" features very quickly can be used the integral image.

The integral image at the pixel whose location (x,y) is the sum of the pixels above and to the left of (x,y) , or in other words the sum of pixels from pixel (0,0) to pixel (x,y) , as shown in the equation[16]

$$ii(x, y) = \sum_{x' \le x, y' \le y} I(x', y')$$

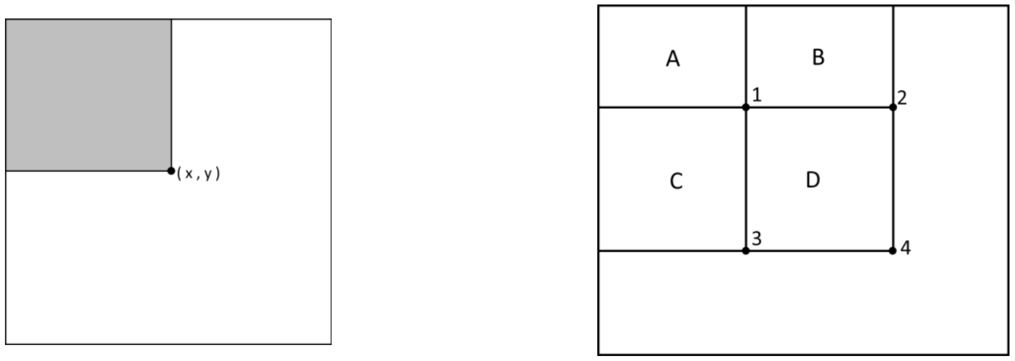


Fig I - 4: The value of the integral image at pixel (x,y) is the sum of all the pixels above and to the left (grey area). (b): Matrix on viola jones algorithm [Fachruddin et al, 2018]

To calculate the sum of pixels in rectangle region D can be used 4 array references. The integral image value at point whose location 1 is the sum of the pixels in rectangle A, and in location 2 is A+B . At point 3 the value is A+C , finally the value at location 4 is A+B+C+D . For this to calculate the sum of pixels in rectangle D can be use the integral image values calculated in location 1,2,3 and 4 as follows : 4 + 1 – (2 + 3) [Fachruddin et al, 2018].

I.4.4. Ada boost

The adaboost is “Adaptive boosting” which is a machine learning algorithm used in the viola-jones detector to select and combine a subset of haar-like features for a good results of face detection [Viola & Jones, 2001].

The adaboost calculation depends on choosing a proficient arrangement of feeble classifiers from countless components [Gudipati et al, 2016].

The fundamental ides is to select and combine the key features from an expansive number of haar-like features by iterative way [Gudipati et al, 2016].

Adaboost is an effective algorithm for features selection and classification in the viola-jones face detection method because it is able to select a small subset most discriminative features with high accuracy.

I.4.5. Cascading

Cascading is technique used in the viola-jones face detection algorithm to speed up the detection process by applying a series of classifiers in a cascade, where each classifier has a higher detection rate and a lower false positive rate than the previous one [Viola & Jones, 2001].

The first classifier in the cascade is designed to quickly reject non-object regions with a low false positive rate [Viola & Jones, 2001].

The cascading technique allows for efficient object detection by quickly rejecting non-object regions in the early stages of the detection accuracy. the viola-jones algorithm with its cascading technique has been widely used for object detection in real time application, such as face detection in video streams and pedestrian detection in autonomous driving [Viola & Jones, 2001].

I.5. Scale Invariant and Feature transform

The SIFT (Scale-Invariant Feature Transform) algorithm is a popular method for object detection in images. It was first introduced by David Lowe in 1999 and has since become one of the most widely used feature detection and matching techniques in computer vision. The SIFT algorithm is widely used in computer vision applications such as object recognition, image stitching, and 3D reconstruction. It has been shown to be robust to changes in scale, rotation, and illumination, and is capable of detecting distinctive and repeatable features in images [Lowe, 1999].

The SIFT algorithm consists of several key steps:

I.5.1. Scale-space extrema detection

a. Construct a scale space by convolving the image with Gaussian filters at different scales. The scale space is defined as a function of the spatial coordinates (x,y) and the scale parameter sigma, denoted as L(x,y,sigma).

b. Compute the difference of Gaussians (DoG) by subtracting two adjacent scales in the scale space. The DoG is defined as follows:

$$DoG(x,y,sigma) = L(x,y,k*sigma) - L(x,y,sigma)$$

where k is a constant factor, typically set to 1.6, and sigma represents the current.

c. Find local extrema in the DoG images by comparing each pixel to its eight neighbors in the current and adjacent scales. A point is considered an extremum if it has a higher or lower intensity than its neighbors in all directions.

I.5.2. Keypoint localization

The location of keypoints (extrema) is refined with sub-pixel accuracy using the Taylor series expansion of the DoG function. Keypoints with low contrast or poorly localized positions are eliminated.

I.5.3. Orientation assignment

a. Compute the gradient magnitude and orientation at each keypoint location. [Lowe, 1999]

b. Assign a dominant orientation to each keypoint based on the gradient orientations and magnitudes in the surrounding image patch. This is done by creating a histogram of gradient orientations and selecting the orientation with the highest count as the dominant orientation.

I.5.4. Descriptor computation

The image patch around each keypoint is divided into subregions (typically a 4x4 grid), and a histogram of gradient orientations is computed for each subregion, weighted by a Gaussian function centered on the keypoint. The histograms of all subregions are concatenated to form a 128-dimentional descriptor vector.

I.5.5. Keypoint matching

a. Match keypoints between different images by computing the Euclidean distance between their descriptors.

b. Use a nearest-neighbor ratio test to eliminate ambiguous matches. A match is considered valid if the distance between the two closest descriptors is significantly smaller than the distance to the next closest descriptor [Mikolajczyk & Schmid 2005].

I.5.6. Transformation estimation

The RANSAC (Random Sample Consensus) algorithm is used to estimate the transformation parameters between the matched keypoints in two images, and outliers are eliminated using a threshold on the reprojection error [Fischler & Bolles 1981].

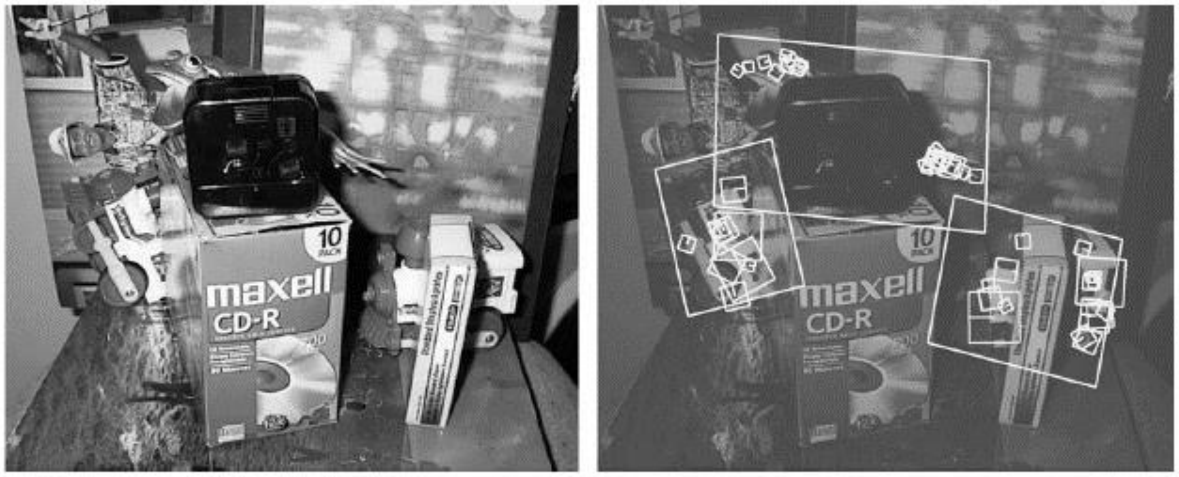


Fig I - 5: Example of SIFT results [Lowe, 2004]

I.6. Histograms of Oriented Gradients (HOG)

Histograms of Oriented Gradients was first introduced by Navneet Dalal and Bill Triggs in 2005. It is a features descriptor like Scale-Invariant-and -Transform (SIFT) that used in computer vision and image processing for the purpose of object detection. HOG is a feature-based method that counts the occurrences of gradient orientation in local image regions to create a descriptor for object detection. HOG has been used successfully for pedestrian detection [Dallal & Triggs, 2005].

The HOG algorithm requires that all images be of the same size. Therefore, we need a reprocessing step which is to resize the image to a fixed size 64 x 128 (the exact value used in the original paper).

The HOG method involves the following steps:

I.6.1. Gamma / Color Normalization

The input image is gamma-normalized to adjust its brightness and make it suitable for feature extraction. This is done using a power-law transformation with a gamma parameter

I_normalized = I^(1/gamma)

where I is the input image and gamma is a parameter that controls the strength of the normalization. The value of gamma is typically set between 0.2 and 0.5, depending on the specific application and the characteristics of the input image.

I.6.2. Gradient Computation

Horizontal and vertical gradients are calculated for each pixel using derivative filters, such as the Sobel filter, to detect edges. This results in matrices storing gradient components.

I.6.3. Spatial / Orientation Binning

The Spatial/Orientation Binning step can be further divided into the following sub-steps:

a. Divide the image into cells: The image is divided into small cells, typically 8x8 pixels in size. For each cell, the gradient magnitudes and orientations are computed using the following equations:

$$magnitude = \sqrt{G_x^2 + G_y^2}$$

$$orientation = arctan\left(\frac{G_x}{G_y}\right)$$

where Gx and Gy are the horizontal and vertical gradients, respectively.

b. Compute the histogram of gradient orientations: For each cell, a histogram of gradient orientations is computed by binning the orientations into a set of predefined bins (e.g., 9 bins spanning 0-180 degrees). The magnitudes of the gradients are used as weights for the bins. The histogram for each cell can be represented as:

H_i = [h_1, h_2, ..., h_n]

where h_i is the count of gradient orientations in the i-th bin, and n is the number of bins.

c. Concatenate the histograms: The histograms for all cells are concatenated to form a feature vector that represents the image. The resulting feature vector can be represented as:

V = [H_1, H_2, ..., H_N]

where N is the total number of cells in the image.

I.6.4. Normalization and Descriptor Blocks

The purpose of normalization is to ensure invariance to variations in illumination and contrast across different regions of the image. The image is divided into blocks, typically consisting of several adjacent cells. For each block, the histograms of gradient orientations are normalized by dividing them by the sum of squared magnitudes within that block. This normalization process ensures that the resulting feature representation is robust to changes in lighting conditions. The normalized histograms of all blocks are then concatenated into a single vector, which forms the final HOG feature descriptor representing the entire image.

I.6.5. Detector Window and Context

A sliding window is used to move across the image, and the HOG descriptor of each window is calculated. The HOG descriptor is then fed into a classifier to determine whether the window contains an object or not.

I.6.6. Classifier

The classifier can be any machine learning algorithm, such as Support Vector Machines (SVMs) or Neural Networks. The classifier is trained on a dataset of positive and negative examples to learn to distinguish between objects and non-objects.

After classification, non-maximum suppression is applied to suppress overlapping bounding boxes and keep only the most likely detection.

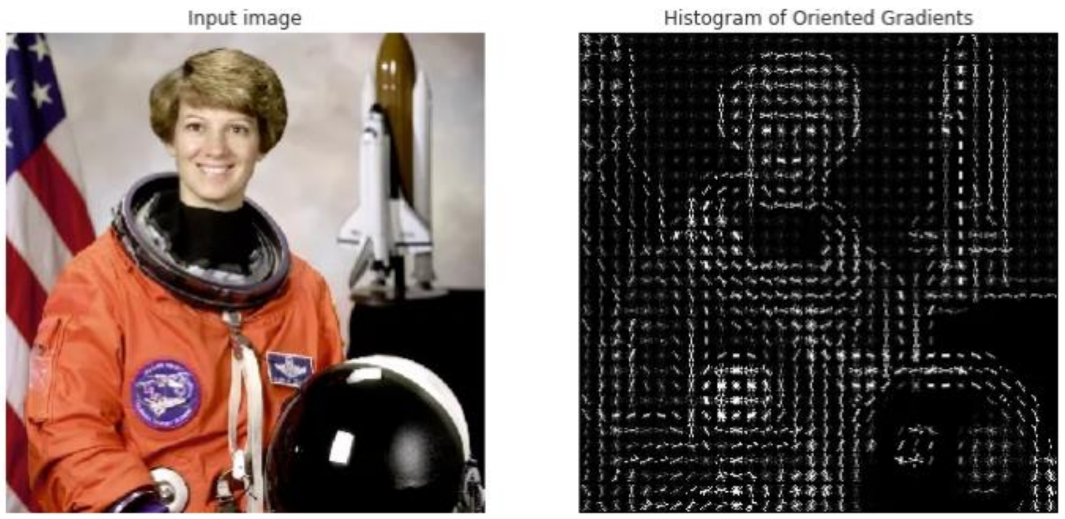


Fig I - 6: Example of the output of the HOG detector [S03]

I.7. Speeded Up Robust Features (SURF)

The Speeded Up Robust Features (SURF) algorithm is a popular method for feature extraction and matching in object detection and recognition applications. It was introduced by Herbert Bay, Tinne Tuytelaars, and Luc Van Gool.

SURF is designed to be faster than other feature detection algorithms, such as SIFT, while maintaining high accuracy and robustness to image transformations, such as rotation, scaling, and noise[Bay et al, 2006].

The SURF algorithm consists of several key steps:

I.7.1. Scale-space extrema detection

This initial step involves identifying scale-space extrema in the image via the Difference of Gaussians (DoG) approach, an approximation of the Laplacian of Gaussian (LoG) operator. Extrema in the resulting DoG pyramid are then found, noting their scale and location.

I.7.2. Keypoint localization

Once the extrema are identified, the next step is to refine their locations and eliminate low contrast and edge-like points using the Hessian matrix.

The Hessian matrix is used to detect blob-like structures at different scales. It is defined as:

$$H(x,\sigma) = \begin{vmatrix} L_{xx}(x,\sigma) & L_{xy}(x,\sigma) \\ L_{xy}(x,\sigma) & L_{yy}(x,\sigma) \end{vmatrix}$$

where Lxx, Lxy, and Lyy are the second-order Gaussian partial derivatives with respect to x and y, and σ is the scale factor.

For computational efficiency, SURF approximates the Gaussian second-order derivatives using box filters. The determinant of the Hessian matrix is used as a measure of the local response:

Det(H) = Lxx(x, σ) * Lyy(x, σ) - (Lxy(x, σ))^2

Local maxima in the response map are selected as keypoints. Non-maximum suppression is applied to select the best keypoints.

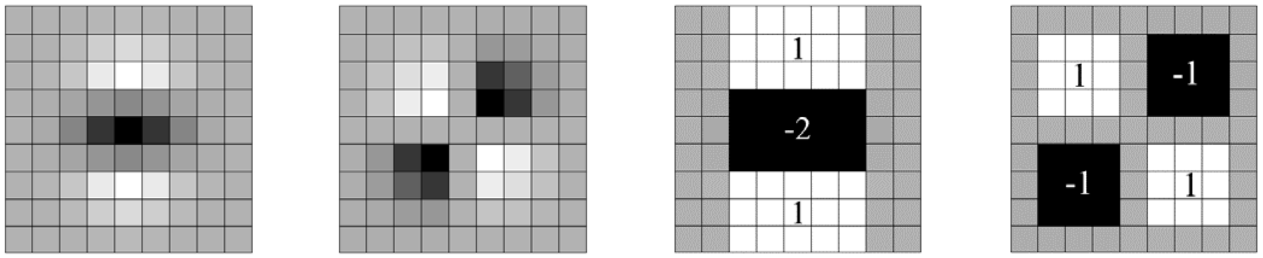


Fig I - 7: Left to right: the (discretised and cropped) Gaussian second order partial derivatives in y-direction and xy-direction, and our approximations thereof using box filters. The grey regions are equal to zero [Bay et al, 2006].

I.7.3. Orientation Assignment

SURF assigns an orientation to each keypoint based on the dominant gradient direction in its neighborhood to achieve rotation invariance. This process involves computing the Haar wavelet responses in the x and y directions around the detected keypoint at its scale. The Haar wavelet response is computed using the following equations:

$$dx = L(x+1,y) - L(x-1,y)$$

$$dy = L(x,y+1) - L(x,y-1)$$

magnitude = sqrt{dx^2 + dy^2}

orientation = atan2(dy,dx)

where L is the Gaussian smoothed image at the scale of the keypoint. The orientation is computed in the range of [0,360] degrees, and keypoints are duplicated for each dominant orientation.

I.7.4. Feature Descriptor

Finally, a 64-dimensional descriptor is constructed to represent the local neighborhood of each keypoint. The area around the keypoint is divided into 4x4 subregions, and Haar wavelet responses are calculated and summed for each, resulting in a 4D vector per subregion. The final descriptor is a concatenation of these vectors from all subregions.

The SURF algorithm is popular for its computational efficiency and robustness to image transformations, such as scaling, rotation, and illumination changes.

I.8. Deformable Parts Model DPM

Deformable Parts Model (DPM) is a widely used object detection framework that was proposed in 2010. DPM is a variant of the popular sliding window method that uses a deformable part-based model to represent an object as a collection of parts with flexible spatial relationships. Each part is represented by a feature vector and is associated with a deformation model that captures variations in part position, scale, and aspect ratio. The model uses a sliding window approach to evaluate the

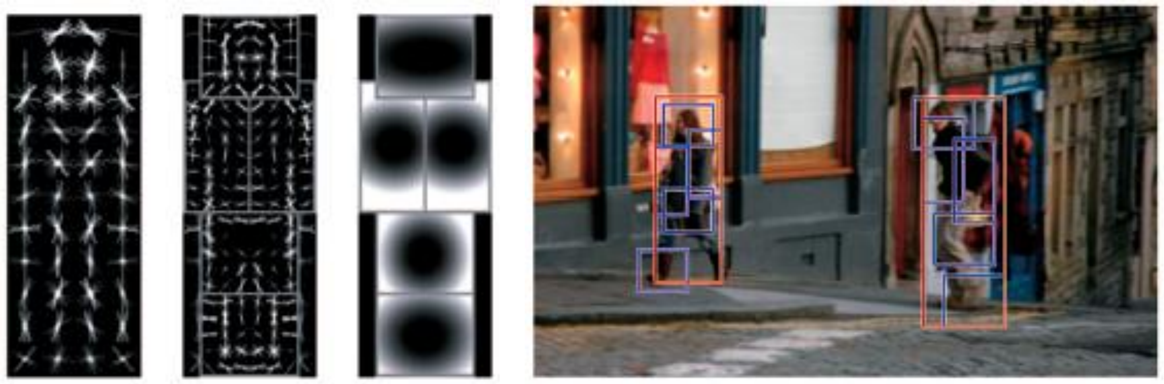


Fig I - 8: Detection obtained using DPM [Felzenszwalb et al, 2010]

likelihood of each part being present at each location and scale in the image, and combines these scores to produce a final detection result. The DPM has been shown to achieve high detection accuracy on a wide range of object categories, and has been widely adopted in practical applications [Felzenszwalb et al, 2010].

The DPM consists of a root filter that represents the global appearance of the object, and a part filter that represents the appearance of each part, along with part models that capture the spatial arrangement of the parts in the object. During detection, the DPM applies the root filter and part filters at multiple scales and locations in the image using a sliding window approach, and combines their scores to produce a final detection score. The root filter, part filter, and part models are important components of the DPM method that allow it to capture the global and local appearance of the object, as well as its flexible spatial arrangement.

I.8.1. Feature Computation

The method begins by computing Histograms of Oriented Gradients (HOG) features for the image [Felzenszwalb et al, 2010].

I.8.2. Root filter

A linear SVM classifier is trained for the whole object using the computed features. The response of the root filter at each location and scale is calculated by convolving the filter with the feature pyramid.

The root filter response can be computed as follows:

$$R(x,y,s) = w\_r^T HOG(x,y,s)$$

where w_r is the weight vector of the root filter and HOG(x,y,s) is the HOG feature vector at location (x,y) and scale s.

I.8.3. Part filter

Linear SVM classifiers are learned for each part of the object using the computed features. The response of each part filter is calculated similar to the root filter response.

The part filter response can be computed as follows:

$$R\_p(x,y,s) = w\_p^T HOG(x,y,s)$$

where w_p is the weight vector of the part filter and HOG(x,y,s) is the HOG feature vector at location (x,y) and scale s.

I.8.4. Part models

The root filter and part filters are combined to form a part-based model. The model's score is the sum of the root filter score and the maximum part filter score. This model is used to detect objects in the image.

The part-based model score can be computed as follows:

$$S(x,y,s) = R(x,y,s) + max\_p(\alpha\_p * R\_p(x,y,s))$$

where α_p is a weight for each part filter, R(x,y,s) is the root filter response, R_p(x,y,s) is the part filter response, and max_p() represents the maximum value over all part filters.

I.8.5. Deformation models

Deformation models are incorporated to account for variations in object shape and appearance. These models adjust the position and size of the parts relative to the root filter during detection.

I.9. Traditional methods limitations

Object detection algorithms such as Viola-Jones, HOG, SIFT, SURF, and DPM each have their own limitations

Viola-Jones, although efficient for simple object detection, struggles with complex objects, large scale and rotated object detection, and perform poorly in noisy images [Viola & Jones, 2001].

HOG, while effective for object detection, has limitations including issues with significant viewpoint changes, absence of color information, fixed resolution, and a heavy reliance on parameter tuning [Dalal and Triggs, 2005].

The SIFT algorithm, although robust to certain transformations, is computationally expensive, sensitive to some transformations, not fully invariant to affine transformations, and requires careful selection of parameters [Lowe, 2004] [Morel & Yu, 2009].

SURF, similar to SIFT, also has limitations, including not being fully invariant to affine transformations and occasionally failing to extract enough distinguishable keypoints [Bay et al, 2006].

Finally, the DPM algorithm, while powerful, is computationally expensive, lacks color information, and does not possess the semantic understanding provided by deep learning models [Felzenszwalb et al, 2010].

I.10. Conclusion

In conclusion this chapter has contained an overview of traditional methods to object detection. The field of computer vision for detecting objects in images is a various field of algorithms and techniques. so we try to define some of algorithms like Viola-Jones, HOG, DPM... despite the evolution of the object detection at the time , it had a lot of problems that we had mentioned earlier , as they tried to solve these problems by using modern methods based on deep learning .

In the next chapter, we will delve into the evolution of these modern techniques, focusing specifically on the pivotal model in object detection: Faster R-CNN.

Chapter II

Object detection using Faster R-CNN approach

II.1. Introduction

Deep learning has revolutionized the field of computer vision, propelling object detection in images to new heights, by providing more accurate and efficient methods for detecting objects. Convolutional Neural Networks (CNNs) are the backbone of most state-of-the-art object detection systems.

One notable advancement in this domain is the Faster R-CNN (Region-based Convolutional Neural Networks) algorithm. To fully appreciate the significance of Faster R-CNN, it is essential to understand the role of deep learning in object detection and the evolution of techniques leading up to it.

This chapter will delve into the details of the fundamentals of CNNs, R-CNN and Fast R-CNN, elucidating the core concepts, architectural advancements, and the impact of each technique on the development of Faster R-CNN. By comprehending the evolutionary path from R-CNN to Fast R-CNN and ultimately Faster R-CNN, we can appreciate the significance of each milestone and the contributions they made to the field of object detection.

II.2. Convolutional neural network

A convolutional neural network (CNN) is a type of deep learning neural network that is primarily used for processing and analyzing images and other multi-dimensional data. The main feature of a CNN is its ability to automatically learn and extract hierarchical features from the input data using convolutional layers, pooling layers, and other architectural components. CNNs have been shown to be highly effective for a variety of computer vision tasks, including object detection, image segmentation, and image classification [LenCun et al, 2015].

II.2.1. CNN architecture

The architecture of a CNN typically consists of several hidden layers, including convolutional layers, pooling layers, and fully connected layers (as shown in Fig II-2). Convolutional layers are designed to learn spatial features from the input image by performing convolution operations, while pooling layers reduce the dimensionality of the feature maps by down-sampling them. Fully connected layers are used to produce a final classification or regression output from the learned features [LenCun et al, 2015].

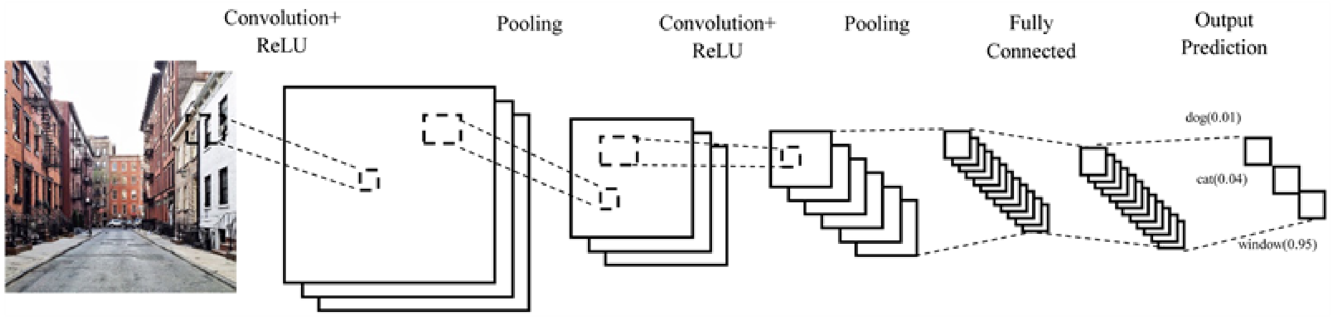


Fig II- 1: The architecture of CNN [S04]

II.2.1.1. Convolutional layers

Convolutional layers are a crucial component of CNNs and play an important role in image recognition and classification tasks.

The primary operation in a convolutional layer is the convolution operation, which applies a set of learned filters or kernels to the input data. Each filter is a small matrix of weights. The convolution operation involves sliding these filters across the input data, step by step, and computing the dot product (scalar product) between the filter values and the input values at each position (as shown in Fig II-2). This operation generates a feature map that represents the response of the filter to the input data, highlighting the presence of particular patterns or features in the input [Goodfellow et al, 2016].

The filters in convolutional layers are designed to detect specific features in the input data, such as edges, corners, or more complex patterns, depending on their size, shape, and learned weights. By applying multiple filters, a convolutional layer can detect a broad spectrum of features and patterns [Simonyan & Zisserman, 2014].

The output of each convolutional layer is then passed through a non-linear activation function, such as the Rectified Linear Unit (ReLU). This function introduces non-linearity into the model, allowing the CNN to learn and model more complex representations of the data. Specifically, the ReLU function sets all negative values in the feature map to zero while leaving positive values unchanged [Nair & Hinton, 2010].

Stacking multiple convolutional layers together forms a deep neural network that is capable of learning more abstract and complex features. As the output of one convolutional layer serves as the input to the next, the network can build a hierarchical representation of the input data, with lower layers learning basic features and higher layers learning more complex and abstract ones [Krizhevsky et al,2012].

In summary, the role of convolutional layers in the success of CNNs cannot be overstated. These layers allow the network to extract relevant features from the input data and learn complex representations, making them indispensable for tasks such as image classification, object detection, and segmentation.

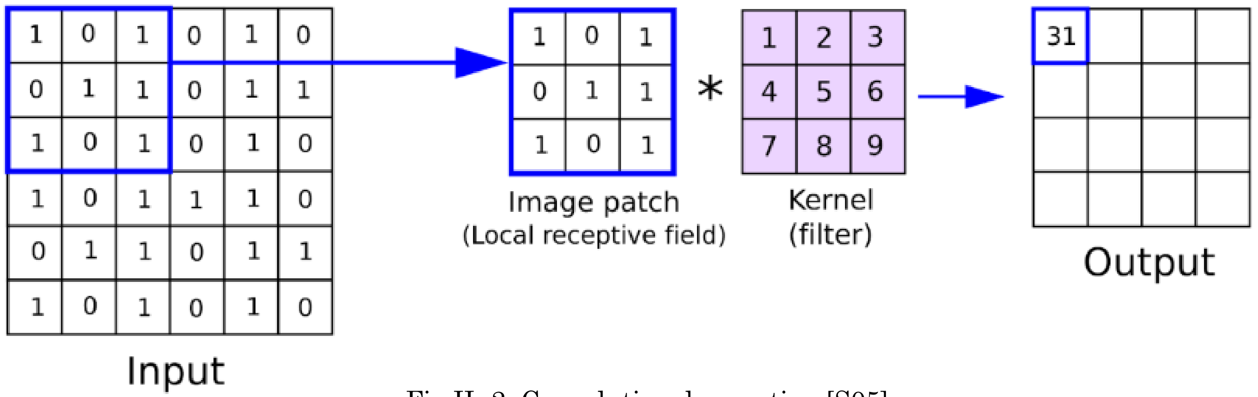


Fig II- 2: Convolutional operation [S05]

II.2.1.2. Activation function

After each convolution operation, an activation function is applied to the output of each neuron in a neural network to introduce non-linearity into the model, allowing it to learn complex patterns in the data and help to speed up the convergence of the model during training by preventing the signal from getting stuck in the same direction [Nwankpa et al,2018].

There are several types of activation functions used in CNNs, including:

a. **Sigmoid**: it is an S-shaped curve that maps any input value into a range between 0 and 1. It is especially useful for models where the output is a probability [Nwankpa et al,2018].

$$\sigma(x) = \frac{1}{1 + e^{-x}}$$

b. **ReLU (Rectified Linear Unit):** It is one of the most commonly used activation functions in deep learning models. It gives an output of x if x is positive and 0 otherwise. ReLU helps the model to solve the vanishing gradient problem [Nwankpa et al,2018].

$$ReLU(x) = max(0, x)$$

c. **Tanh (Hyperbolic Tangent):** It is similar to the sigmoid function but maps any input to a value between -1 and 1, and is typically used in multi-class classification problems [Goodfellow et al, 2016].

$$tanh(x) = 2\sigma(2x) - 1.$$

d. Softmax: is often used in the output layer of a classifier where the model needs to predict the probabilities for each class. The Softmax function takes as input a vector of K real numbers and normalizes it into a probability distribution consisting of K probabilities.

$$SoftMax(xi) = \frac{e^{xi}}{\Sigma_j e^{xj}}$$

In the above equation, xi is the ith element of the input vector x and the denominator $\Sigma j\ exp(xj)$ is the sum of the exponentials of all the elements in the input vector. This denominator ensures that the sum of the probabilities from the softmax function will be 1, which is a requirement for a probability distribution.

Choosing the Right Activation Function: The choice of activation function depends on the specific problem being addressed and the structure of the neural network being used. For example, ReLU is commonly used in CNNs due to its simplicity and efficiency, but it may not be suitable for all problems [Karpathy, 2016].

II.2.1.3. Pooling layers

Pooling layers in CNNs are responsible for reducing the spatial dimensionality of the feature maps generated by the convolutional layers. Pooling works by taking a small window (usually of size 2x2 or 3x3) and performing an operation on the values within the window, typically either taking the maximum or average value. The resulting output is a smaller feature map with reduced spatial resolution, but preserved important features [Zeiler & Fergus, 2014].

There are different types of pooling layers used in CNNs, including max pooling, average pooling, and global pooling [Sze et al, 2017].

a. Max pooling is the most commonly used pooling method, which takes the maximum value of each window. This is useful for detecting the most important feature in a given window.

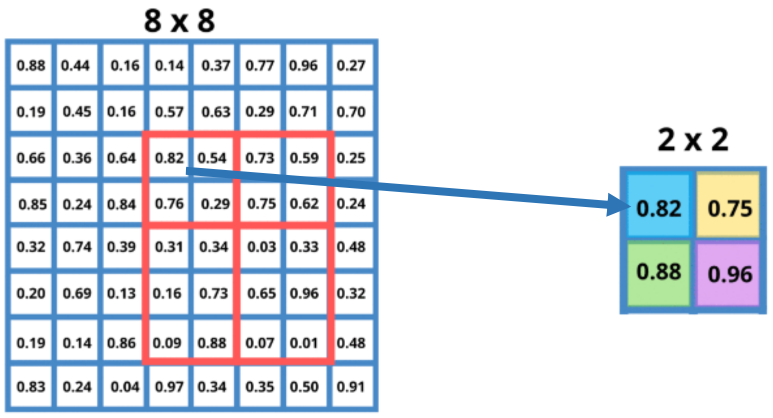


Fig II- 3: Example of max pooling [S06]

b. Average pooling, on the other hand, takes the average value of each window, which can be useful in reducing the amount of noise in the feature maps [Zeiler & Fergus, 2014].

c. Global pooling, also known as global average pooling, is another type of pooling that takes the average of all values in each feature map. This results in a fixed-size output that is independent of the input size, which can be useful for classification tasks [Zeiler & Fergus, 2014].

Pooling layers play an important role in reducing the spatial dimensionality of feature maps, which can help prevent overfitting and improve model efficiency. However, pooling can also lead to loss of information, so it is important to use it judiciously [LenCun et al, 2015].

II.2.1.4. Fully connected layers

Fully connected layers, also known as dense layers, are a type of layer commonly used in CNNs for classification and regression tasks. These layers are typically placed after the convolutional and pooling layers to combine the extracted features and produce a final output.

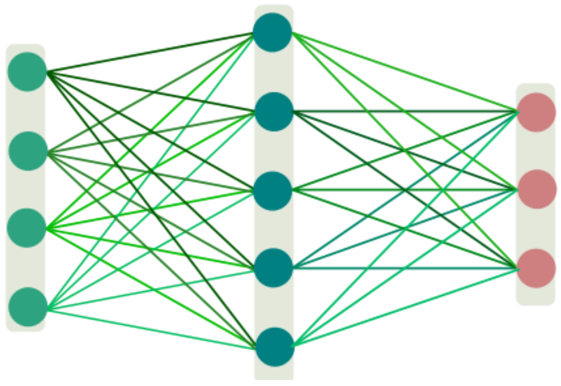


Fig II- 4: Fully connected layers [S07]

Fully connected layers are composed of multiple neurons that are connected to all the neurons in the previous layer. Each neuron in the layer receives input from all the neurons in the previous layer and produces an output, which is then passed on to the next layer [Goodfellow et al, 2016].

Input: The input to a fully connected layer is typically a vector of features that have been extracted from the previous layers of the CNN. These features are combined and transformed into a vector that represents the output of the layer.

Parameters: Each neuron in the fully connected layer has its own set of learnable parameters, including a weight matrix and a bias vector. The weight matrix determines the strength of the connections between the neurons, while the bias vector controls the output of each neuron. These parameters are learned through backpropagation during training to minimize the loss function [Goodfellow et al, 2016].

Activation function: Fully connected layers typically use an activation function, such as ReLU or sigmoid. The choice of activation function depends on the specific task and architecture of the CNN [Sze et al, 2017].

Output: The output of a fully connected layer is a vector that represents the predicted class probabilities or regression values for the input. This output is then compared to the true values during training using a loss function [LeCun et al, 2015].

Number of layers: The number of fully connected layers in a CNN varies depending on the task and complexity of the model. Typically, deeper networks with more layers have higher accuracy but require more training time and resources [Simonyan & Zisserman, 2014].

Implementation: fully connected layers are easy to implement in popular deep learning frameworks like TensorFlow and PyTorch. They are used in combination with convolutional and pooling layers to form a complete CNN architecture for tasks such as image classification, object detection, and segmentation.

II.2.1.5. Loss function

Loss function, also known as cost function or objective function, is a central component in training a CNN or any other machine learning model. It measures the inconsistency between predicted bounding boxes, class probabilities, and the ground truth.

the specific form of the loss function can vary depending on the algorithm used.

The loss function typically has two components: classification loss, which measures the error in predicting object classes, and localization loss, which measures the error in predicting bounding boxes. The total loss function is generally a weighted sum of these two components. It is used during the training of the network to update the model parameters. Here are some of the common loss functions used in CNNs:

a. **Mean Squared Error (MSE) Loss:** Also known as L2 smooth loss, this loss function calculates the average of squared differences between the predicted and actual values. It's commonly used in regression problems. While it's simple to understand and compute, it's sensitive to outliers due to the squaring of differences [Friedman et al, 2001].
b. **Cross-Entropy Loss:** Also known as Log Loss, it's commonly used in binary and multi-class classification problems. Cross-Entropy loss measures the dissimilarity between the predicted probability distribution and the actual distribution. It's preferred in classification problems because of its good mathematical properties [Goodfellow et al, 2016].

c. Hinge Loss: This is usually used for "maximum-margin" classification, most notably for support vector machines (SVMs). For an input-output pair, the hinge loss of a model is the loss incurred by that pair [Rosasco et al,2004].

d. Huber Loss/Smooth L1 Loss: Huber Loss is often used in regression problems and is less sensitive to outliers than the mean squared error loss. It combines the best properties of MSE loss and MAE loss by being quadratic for smaller errors and linear otherwise (and similarly for its gradient) [Girshick et al, 2014].

II.2.1.6. Training

a. Dataset Preparation: In supervised learning, the first step involves preparing the labeled dataset. Each sample in the dataset is an image with corresponding labels, indicating the class of the object and its location (bounding box coordinates). These annotations can be in the form of a bounding box, a pixel-wise mask, or other types of labels depending on the specific task [LeCun et al, 2015].

b. Model Architecture: The next step is to select or design a CNN architecture suitable for the task at hand. This involves choosing the number and types of layers (e.g., convolutional, pooling, fully connected), activation functions, and optimization algorithms [LeCun et al, 2015]. Examples of popular CNN architectures include VGG-16, ResNet-50, and Inception.

c. Initialization: Before training the model, the weights of the CNN layers need to be initialized. This can be done randomly or using pre-trained weights from a model that has been trained on a large dataset (e.g., ImageNet) [He et al, 2016].

d. Forward Propagation: During training, input data is fed through the CNN layers in a forward pass, producing a prediction for the output. Each layer applies a set of filters to the input, extracting features and reducing the dimensionality of the data.

e. Compute the loss: Calculate the loss function, which measures the difference between predictions and actual values. Common loss functions in CNNs are Mean Squared Error (MSE) for regression tasks and Cross-Entropy Loss for classification tasks.

f. Backward Propagation (Backpropagation): To minimize the loss function, backpropagation calculates the gradient of the loss with respect to each weight and bias, then updates their values. This process uses the chain rule from calculus, working from the output layer to the input layer [GoodFellow et al,2016].

g. Update the Weights and Biases (Optimization): There are various optimization algorithms that can be used to adjust the biases and weights during training, such as stochastic gradient descent (SGD), Adam. These algorithms control the learning rate and momentum of weight.

- SGD is a variant of gradient descent optimization, which uses a single or a batch of samples per iteration to update model parameters, providing computational efficiency and enabling large-scale dataset training [Bottou et al,2018].

- Adam is a method that computes adaptive learning rates for each parameter, combining the advantages of other extensions of SGD updates [Kingma & Ba,2014].

h. Training Loop: Repeat the forward propagation and optimizations steps for a predetermined number of iterations (epochs) or until the network's performance is satisfactory.

i. Validate: Validate the performance on unseen data to ensure the model generalizes well and doesn't overfit the training data, using a separate validation set.

j. Test: Evaluate the model on a testing set to measure its performance. The testing set is a different dataset not used during training.

Overall, training a CNN involves selecting and preparing a dataset, designing a model architecture, initializing weights, performing forward and backward propagation, optimizing the model, and iterating over the training loop until the model converges to a satisfactory performance.

II.2.2. Popular architecture of CNN

There are several types of CNN architectures that have been developed over the years. Here are some of the most common types of CNNs

LeNet: developed in 1998 by Yann LeCun and colleagues, LeNet was one of the first CNN architectures used for handwritten digit recognition. It consists of convolutional and subsampling layers for feature extraction and fully connected layers for classification [LeCun et al, 1998].

AlexNet: developed in 2012, is a deep CNN architecture that won the ImageNet Large Scale Visual Recognition Challenge (ILSVRC) by a significant margin. It employs convolutional and fully connected layers and incorporates techniques like data augmentation, dropout, and ReLU activation functions to improve accuracy [Krizhevsky et al, 2012].

VGG: The VGG (Visual Geometry Group) network is a deep CNN architecture developed by the Visual Geometry Group at the University of Oxford, which achieved state-of-the-art performance on the ILSVRC 2014 image classification task [Simonyan & Zisserman, 2014].

GoogLeNet/Inception: GoogLeNet, or Inception, is a CNN architecture developed by Google researchers. It won the ILSVRC 2014 challenge with significantly fewer parameters than VGG. GoogLeNet uses a combination of convolutional and pooling layers, along with the inception module that concatenates features from different filters [Szegedy et al, 2015].

ResNet: developed by Microsoft researchers, introduced residual connections to overcome the vanishing gradient problem. It uses skip connections to enable the network to learn an identity mapping, facilitating the training of deep neural networks [He et al., 2016].

DenseNet: it is a CNN architecture developed by researchers at Facebook AI Research. It connects each layer to every other layer, resulting in a densely connected network. This facilitates feature reuse and encourages the learning of discriminative features [Huang et al, 2017].

MobileNet: is a family of CNN architectures designed for mobile and embedded devices. Developed by Google researchers, MobileNet utilizes depthwise separable convolutions to reduce the number of parameters and computations required, making it lightweight and efficient [Howard et al, 2017].

II.2.3. OverFeat

OverFeat is a machine learning model that has been developed for the purpose of object detection, classification, and localization in images. OverFeat leverages the strengths of CNNs in a multi-scale and sliding window approach [Sermanet et al, 2013]. OverFeat was developed on the principles of convolutional networks, but with several key innovations:

Multi-Scale Processing: The model learns and applies convolutional networks at various scales of input resolution to improve performance. It uses an integrated framework to capture objects at different scales and aspects, unlike most CNNs that operate on a fixed size input.

Sliding Window Approach: In CNNs, a fully connected layer is often used for classification. In OverFeat, this layer is replaced with a convolutional layer. The output layer is treated as a grid of coarse detection windows, which are then used for object localization and detection.

Regression for Precise Localization: Along with class probabilities, OverFeat also predicts bounding box coordinates for objects, improving the model's localization accuracy [Sermanet et al, 2013].

II.3. Region based methods

Region-based methods are a family of object detection algorithms that focus on finding regions in an image that are likely to contain objects and then classifying these regions into object categories.

These methods typically have two stages: the first stage proposes potential regions of interest (RoIs), and the second stage classifies each RoI using a classifier, such as a CNN to classify the proposed regions into object classes or background [Girshick et al, 2014].

II.3.1. Region proposal techniques

Object detection involves both identifying the presence of an object in an image and localizing its position. However, localizing the object accurately is a challenging task. One of the conventional approaches for object localization is using a sliding window of varying sizes to scan the entire image, which is known as Exhaustive search. This approach is computationally expensive, as it requires searching through thousands of windows, even for a small image. Although some optimization techniques have been developed, such as using windows with different ratios instead of incrementing them by a few pixels, the number of windows still makes it inefficient.

Region proposal techniques provide a solution for The problems we have discussed so far by generating bounding boxes for image patches that are most likely to contain objects. Although these region proposals can be imperfect, overlapping, or noisy, among them, there will be at least one proposal that accurately corresponds to the object in the image. By classifying these region proposals with an object recognition model, we can identify the locations of the objects. The proposals with high probability scores are the most likely locations of the object in the image [Girshick et al, 2014].

There are several region proposal techniques that have been developed for object detection, each with its own strengths and weaknesses. However, the most popular techniques currently in use are: selective search, edge Boxes and region proposal networks.

II.3.1.1. Selective search

Selective Search is a technique for generating region proposals in object detection tasks. It is designed to be fast with a very high recall. It identifies candidate regions that potentially contain objects by combining segmentations at multiple scales and merging them based on similarity measures. It uses a hierarchical approach to merge regions based on color, texture, size, and shape compatibility [Uijlings et al, 2013].

We want to detect objects in an image (a) as show in Fig II-5, as shown in the ground truth (b). To achieve this, we utilize segmentation as a selective search strategy. Our objective is to achieve high recall by generating object locations of various scales and considering several scene conditions by using multiple invariant color spaces. The resulting object hypotheses are illustrated in (d).

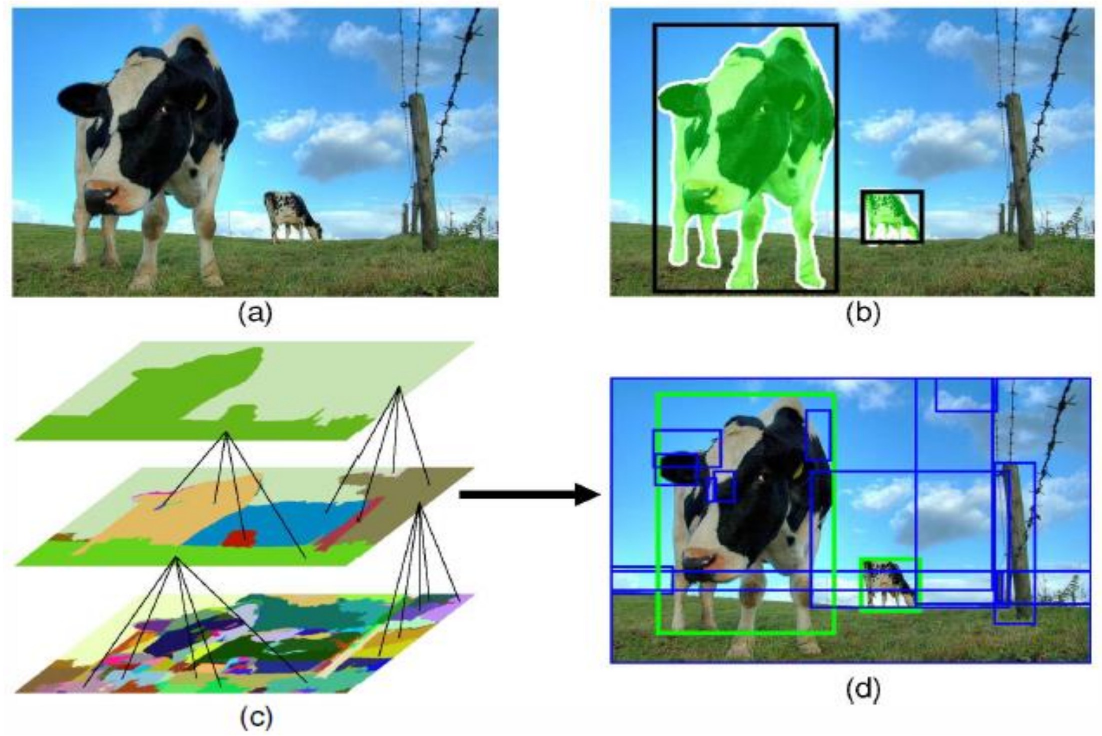


Fig II- 5: Selective search method [Uijlings et al, 2013]

The selective search algorithm consists of several steps, which are as follows;

a. **Initial segmentation:** The Selective Search algorithm begins by over-segmenting the input image into multiple small regions using a method called "Felzenszwalb and Hutten locher's efficient graph-based segmentation method." This technique groups together pixels that are similar based on intensity values to form multiple small segments or regions [Uijlings et al, 2013]

b. **Hierarchical grouping:** After initial segmentation, the algorithm starts merging the regions based on several similarity measures such as color, texture, size, and shape compatibility. This is done in a greedy hierarchical manner, with the most similar regions (according to the measures) being merged at each step. The process of merging continues until the entire image becomes a single region [Uijlings et al, 2013].

c. **Generate candidate regions:** During the merging process, the bounding boxes of the merged regions at each step are recorded and used as region proposals. These proposals represent a variety of scales and aspect ratios, increasing the likelihood of containing the objects in the image [Uijlings et al, 2013]

Selective Search has been widely used in object detection methods, such as R-CNN and Fast R-CNN, due to its computational efficiency and ability to generate a diverse set of high-quality region proposals. However, it can be relatively slow compared to more recent region proposal techniques like the Region Proposal Network used in Faster R-CNN.

II.3.1.2. EdgeBoxes

EdgeBoxes is a region proposal algorithm that generates a set of bounding boxes by evaluating edge information and geometric cues. It is based on the idea that object boundaries are often aligned with image edges, and it uses this information to generate proposals quickly and efficiently [Zitnick et al, 2014].

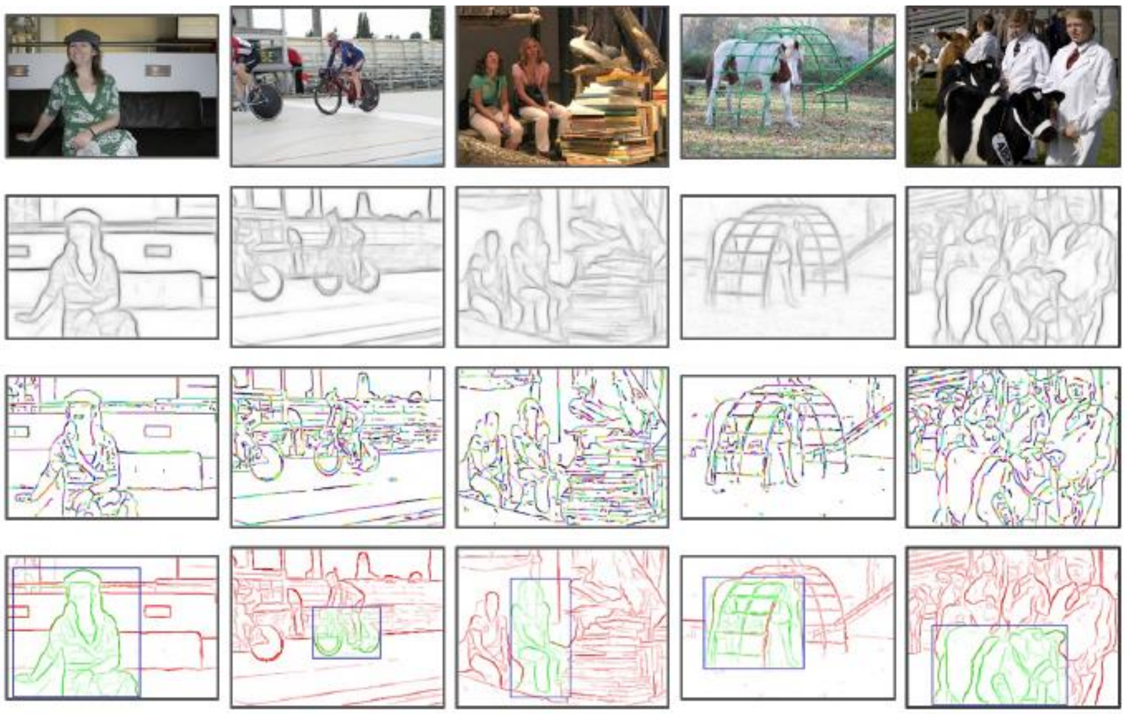


Fig II- 6: EdgeBoxes method [Zitnick et al, 2014]

The EdgeBoxes algorithm consists of several steps (as shown in Fig II-6), which are as follows:

a. Generate edge maps

The first step in the EdgeBoxes algorithm is to generate an edge map from the input image. The edge map is a binary image where the value of each pixel indicates the presence or absence of an edge in the corresponding location of the original image. The Structured Forests edge detector is typically used to generate these edge maps [Dollár et Zitnick, 2013].

b. Generate bounding box proposals

The algorithm generates a large set of bounding box proposals over the image. The bounding box proposals are generated using a sliding window approach with varying aspect ratios and scales [Zitnick et al, 2014].

c. Calculate the objectness score

For each bounding box proposal, the algorithm calculates an "objectness" score. The score is based on the number of edges or contours that are wholly contained within the box. The intuition here is that objects are typically surrounded by edges, so boxes that contain more complete edges are more likely to contain an object [Zitnick et al, 2014].

d. Perform the non-maximum suppression

To reduce redundancy among the proposals, the EdgeBoxes algorithm applies non-maximum suppression (NMS) to the objectness scores.

NMS works by comparing each proposal with its neighboring proposals and discarding those that have a high overlap with other proposals [Neubeck & Van Gool, 2006].

e. Group proposals

The remaining proposals can be optionally grouped based on their spatial proximity. This step ensures a diverse set of proposals that cover different parts of the image [Zitnick et al, 2014].

f. Refine proposals

Finally, the proposals can be refined by fitting a bounding box around each group of proposals [Zitnick et al, 2014].

The EdgeBoxes algorithm is frequently employed as part of a two-stage object detection pipeline, where the object proposals it generates are fed into a separate object detection network for final object classification.

II.3.1.3. Region proposal networks (RPN)

The RPN is an object detection method introduced by Ren et al. in 2016 as part of the Faster R-CNN framework. RPN is a fully convolutional neural network that generates region proposals directly from the feature maps of an image and outputs a set of object proposals, each represented by a bounding box and an objectness score. It is designed to improve the speed and efficiency of object detection compared to previous methods like Selective Search.

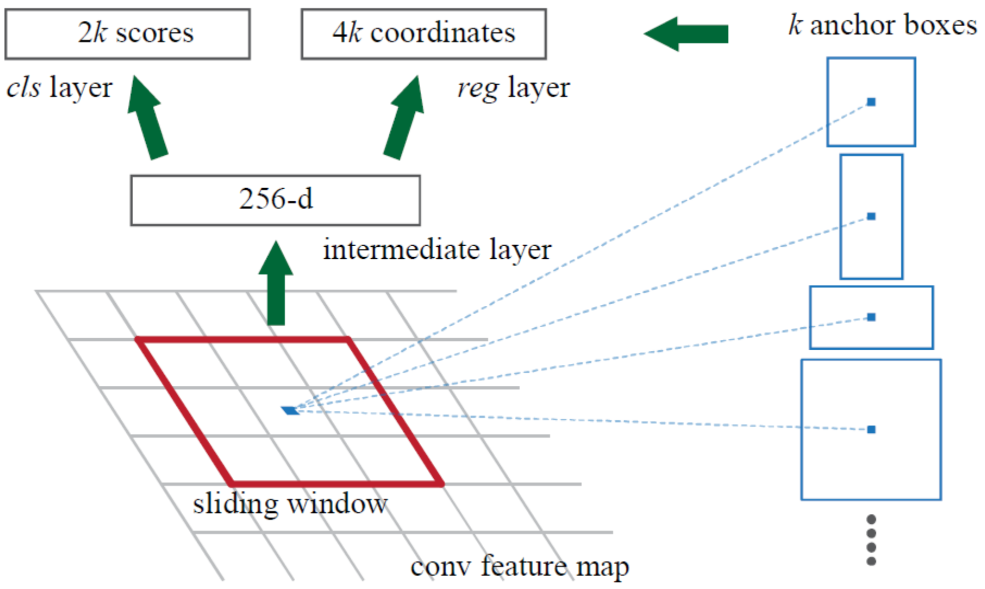


Fig II- 7: Region Proposal Network [Ren et al., 2016]

The RPN algorithm consists of several steps, which are as follows;

a. Shared Convolutional Layers: The RPN begins by passing the input image through a series of convolutional layers to produce a feature map. This feature map contains rich, high-level information about the image content. This is typically done using a pre-trained deep CNN, such as VGG-16 or ResNet, which have been trained on a large-scale image classification task .
The convolutional layers are shared with the detection network, which helps to reduce computational cost and allows the RPN to take advantage of the same high-level features used for object detection [He et al., 2016][Simonyan & Zisserman, 2014].

b. Anchors: Anchors are fixed bounding boxes of different scales and aspect ratios that are used as reference points for generating region proposals. At each location (i.e., each pixel) on the feature map, k anchors are defined, where k is typically 9 (3 scales and 3 aspect ratios) as show in Fig II-8. The anchors are designed to accommodate the various sizes and shapes of potential objects in the image[Ren et al., 2016].

c. Classification and Regression: For each anchor, the RPN simultaneously makes two predictions using two different convolutional layers (one for classification, one for regression) applied to the feature map
Classification (objectness score): The RPN predicts whether the anchor contains an object or not. This is done using a binary classifier that produces a score indicating the "objectness" of the anchor. A high score means the anchor is likely to contain an object, while a low score

means it is likely to contain background [Ren et al., 2016].

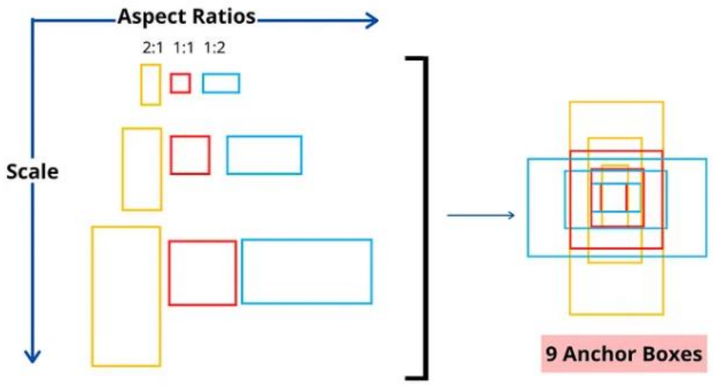


Fig II- 8: Anchor boxes for object detection [S06]

Regression (bounding box adjustments): The RPN also predicts four values that represent adjustments to the anchor box to better fit the object. These adjustments are in terms of the height, width, and x and y coordinates of the box. This is known as bounding box regression [Girshick, 2015].

d. **Non-Maximum Suppression:** The RPN then applies non-maximum suppression to the proposals to eliminate redundant overlapping proposals [Neubeck & Van Gool, 2006].

II.3.2. Evolution of R-CNN to Faster R-CNN

The evolution from R-CNN to Faster R-CNN represents a significant progression in the field of object detection

II.3.2.1. Regions based Convolutional Neural Networks (R-CNN)

R-CNN was one of the first applications of convolutional neural networks to the problem of object detection. It introduced the concept of using region proposals as a way to localize objects, combined with a CNN for feature extraction and classification [Girshick et al., 2015].

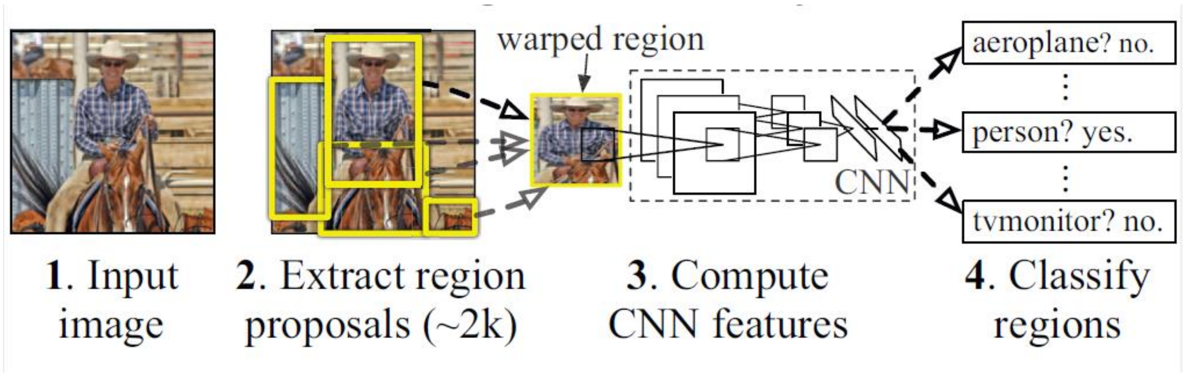


Fig II- 9: R-CNN architecture [Girshick et al., 2014].

The RCNN approach consists of four main steps that are crucial for object detection in images.

II.3.2.1.1. Object proposal generation

Object proposal generation is the first step in the RCNN approach for object detection. In this step, potential object regions are proposed from an input image using a selective search algorithm.

This step generates around 2000 region proposals that are likely to contain objects of interest.

The objective of object proposal generation is to reduce the computational cost of feature extraction and classification by selecting only the most relevant regions for further processing. The use of region proposals helps to focus the CNN on the most informative parts of the image and improves the accuracy of object detection [Girshick et al., 2014].

Once the candidate regions have been generated, they are resized and passed to the next step of the RCNN approach: feature extraction.

II.3.2.1.2. Feature extraction

The next step is to extract features from each proposed region. This is done by passing the warped and cropped region through a pre-trained CNN model (often pre-trained on ImageNet) and extracting the output feature map for that region. The CNN can be any pre-trained network such as VGG16 or ResNet50 [Girshick et al., 2014].

The input to this CNN is an RGB image, and the output is a 4096-dimensional feature vector that can be used for object classification and localization.

II.3.2.1.3. Object classification and localization

This step involves training a set of support vector machines (SVMs) for each object class to classify the proposed regions, and also training a bounding box regression model to refine the object localization within each region

To train the SVMs, the fixed-length feature vectors extracted from the proposed regions are first normalized by subtracting the mean and dividing by the standard deviation of the feature vectors across the training set. The normalized feature vectors are then used to train one SVM per object class using a one-vs-all strategy. The SVM outputs a probability score for each proposed region indicating the likelihood of it containing an object of that class [Cortes & Vapnik, 1995].

II.3.2.1.4. Bounding boxes regression

The final step in the R-CNN object detection pipeline is to refine the object proposals generated by the selective search algorithm using a technique called bounding box regression. This step involves training a regression model that learns to adjust the size and position of the proposed bounding boxes in order to better fit the object of interest within each region proposal

The regression model is trained using a set of ground truth bounding boxes and their corresponding region proposals. For each region proposal, the model learns to predict a set of four offsets that adjust the size and position of the proposed bounding box to better match the ground truth box. Specifically, the model predicts the offset values for the x and y coordinates of the top left corner of the bounding box, as well as the width and height of the box [Girshick et al, 2014].

During inference, the regression model is used to refine the bounding boxes of the proposed regions output by the object classification and localization step. For each proposed region, the model predicts the offset values and applies them to the original bounding box to obtain a refined box that better fits the object of interest within the region. This refined bounding box is then used to classify the object and determine its final position within the image [Girshick et al, 2014].

The use of bounding box regression in R-CNN has been shown to improve the accuracy of object detection by allowing for more precise localization of the objects of interest within each region proposal.

The resulting bounding box is then post-processed to remove overlaps and non-maximum suppression is used to select the final set of detected objects [Girshick et al, 2014]

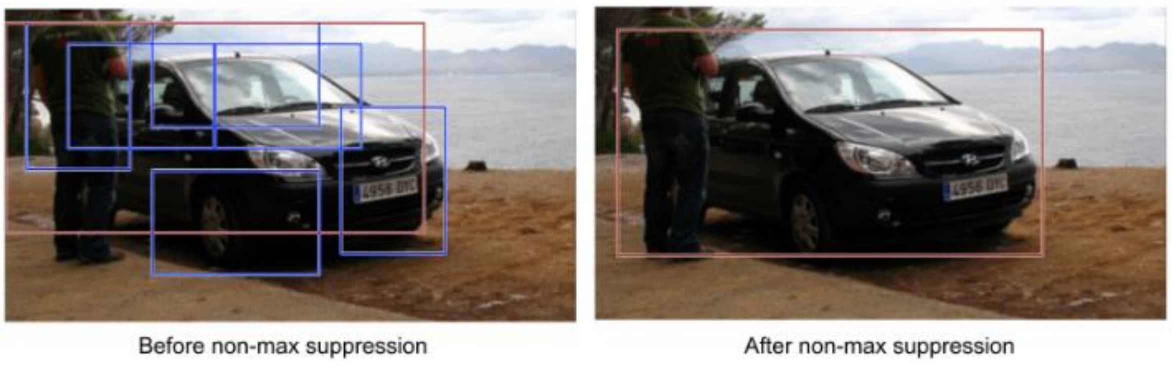


Fig II- 10: the effect of non-max suppression [S08]

Non-maximum suppression is a technique used to prevent such duplicate detections. Once the model has identified a set of bounding boxes that belong to the same object category, these boxes are sorted based on their confidence score, and those with low scores are discarded. The non-max suppression algorithm then iteratively selects the box with the highest confidence score, discards any boxes that have a high intersection over union (IoU) with the selected box, and repeats the process until all remaining boxes have low confidence scores or have been processed. This ensures that only the most confident and non-overlapping bounding boxes are retained for each object instance [Hosang et al, 2015].

II.3.2.2. Fast R-CNN

Fast R-CNN is a milestone model in the field of object detection that improved upon the original R-CNN model by Ross Girshick in 2015. It introduced several key improvements, most notably a mechanism called Region of Interest (RoI) Pooling and the use of a single network, and performs the tasks of classification and bounding box regression simultaneously [Girshick, 2015].

Fast R-CNN introduced a number of key advancements to address the shortcomings of R-CNN, including training speed, detection speed, and efficiency.

A significant change was the use of a single network for both object classification and bounding box regression, which was previously done in two separate stages in R-CNN. This made the training process more streamlined and effective [Girshick, 2015].

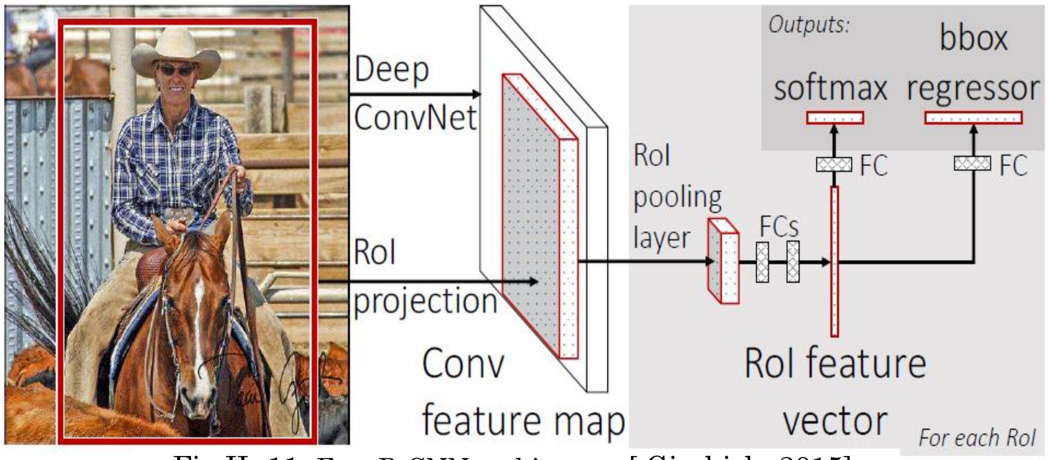


Fig II- 11: Fast R-CNN architecture[Girshick, 2015]

The architecture of Fast R-CNN is depicted in Fig II-11 It operates by taking an entire image and a collection of object proposals as input. The network initially processes the image through convolutional and max pooling layers to generate a convolutional feature map. Then, for each object proposal, a RoI pooling layer is applied to extract a fixed-length feature vector from the feature map. These feature vectors are subsequently fed into a sequence of fully connected layers, which split into two sibling output layers. The first layer produces softmax probability estimates for K object classes, including a "background" class. The second layer generates four real-valued numbers for each of the K object classes, encoding refined bounding-box positions [Girshick, 2015]

II.3.2.2.1. RoI Projection

It is the process of mapping or translating region proposals from the original image coordinates onto the feature map coordinates as show in Fig II-12. This is necessary because the region proposals are usually generated in the space of the original image, but they need to be transformed to the space of the feature map for further processing.

The exact process of projection typically involves scaling the region proposal coordinates by the subsampling ratio of the network (i.e., the ratio of the input image size to the feature map size), which accounts for the down sampling that occurs when the image is passed through the convolutional layers of the network [S06].

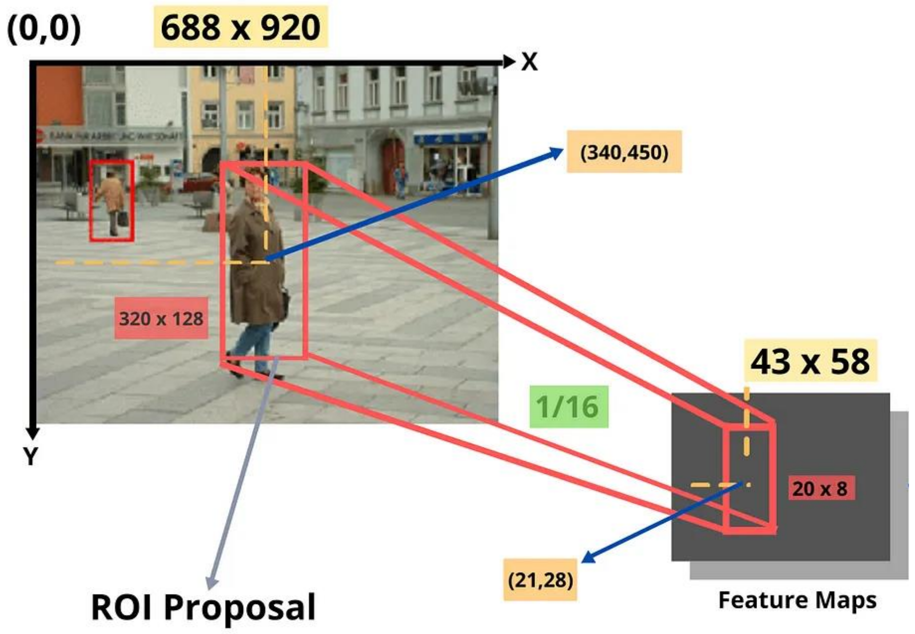


Fig II- 12: Example of RoI projection [S06]

II.3.2.2.2. Convolutional Feature Map

Fast R-CNN starts by taking an entire input image along with a set of object proposals, typically generated by an external method like Selective Search. This image is then passed through RoI projection and max pooling layers to produce a convolutional feature map. This approach is markedly different from R-CNN, which operated directly on each region proposal. The advantage of Fast R-CNN's approach is that the computationally expensive convolution operation is executed just once per image, rather than 2000 times per proposal, leading to significant improvements in efficiency [Girshick, 2015].

II.3.2.2.3. RoI Pooling

A critical part of the Fast R-CNN architecture is the RoI pooling layer. Unlike R-CNN, Fast R-CNN uses the RoI pooling layer extracts a fixed-length feature vector from the feature map for each region proposal. This process involves dividing the proposal region into a fixed number of subregions (e.g., a 7x7 grid), followed by max-pooling the values within each sub-region. This approach ensures that the extracted feature descriptors are consistently sized, irrespective of the size or aspect ratio of the region proposal. [Girshick, 2015]

II.3.2.2.4. Fully-Connected Layers and Outputs

The feature vectors are fed into a sequence of fully connected (fc) layers, finally leading to two output layers: one for object classification (with Softmax function) and another for bounding box regression. This design allows Fast R-CNN to train a single network that jointly learns to classify and localize objects [Girshick, 2015].

The feature vectors are then passed through a sequence of fully connected layers. The network branches into two output layers at the end: one produces Softmax probability estimates over the object classes (including a "background" class), bounding box regression loss, on the other hand, aims to refine the coordinates of the bounding box to better encapsulate the object by producing four real-valued numbers for each class. These four numbers parameterize the coordinates of a bounding box for the object in the proposal. This design allows Fast R-CNN to train a single network that can classify the object proposals and refine their bounding box coordinates [Girshick, 2015].

II.3.2.2.5. Training and Inference

The Fast R-CNN network is trained end-to-end using backpropagation and stochastic gradient descent. The training process optimizes the network parameters to minimize the classification and bounding box regression losses.

Once the model is trained, it can be applied to unseen images for object detection. During inference, an entire image and a set of object proposals are input into the network. The network processes the image to produce a convolutional feature map, extracts feature vectors for each proposal using the RoI pooling layer, and passes these vectors through fully connected layers.

The network then outputs class probabilities and bounding box coordinates for each proposal. The final detected objects are determined by selecting the class with the highest probability for each proposal, and applying a technique called non-maximum suppression to discard overlapping detections, leaving only the proposals with the highest confidence [Girshick, 2015].

II.3.2.3. The Need for Further Advancements

The development of Faster R-CNN was prompted by the need to overcome limitations in existing object detection algorithms such as R-CNN and Fast R-CNN. These methods faced challenges related to computational efficiency, disjointed processing, suboptimal integration, and real-time capabilities.

R-CNN relied on computationally expensive selective search for region proposal generation, while Fast R-CNN improved efficiency with shared convolutional features but still relied on separate region proposal methods. The desire for better accuracy and robustness also fueled the development of Faster R-CNN.

Additionally, a unified training approach was needed to optimize the entire object detection system, as opposed to training components separately.

Lastly, the growing demand for real-time object detection applications drove the optimization of the balance between speed and accuracy. Faster R-CNN addressed these challenges, resulting in significant advancements that have had a profound impact on the field of object detection.

II.3.3. Faster R-CNN

Faster R-CNN, as introduced by Ren et al. in 2016, is a deep learning algorithm for object detection that consists of two key modules [Ren et al, 2016]

- The first module is the region proposal network (RPN) for proposing candidate object bounding boxes
- The second module is the Fast R-CNN detector, utilizes these proposed regions for object detection.

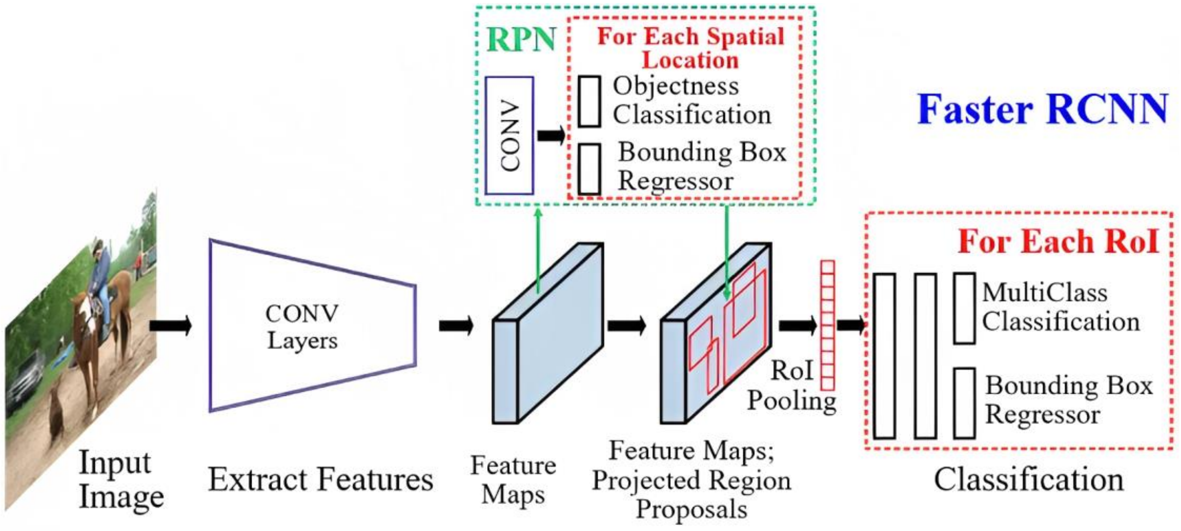


Fig II- 13: Faster R-CNN architecture [S05]

-

II.3.3.1. Faster R-CNN architecture

The architecture of Faster R-CNN (as shown in Fig II-13) includes the following steps:

a. **Feature Extraction:** The input image is processed through convolutional and pooling layers to create a feature map, capturing hierarchical and abstract representations.

b. **Region Proposal Network (RPN):** The RPN takes the feature map and generates rectangular object proposals with corresponding objectness scores. It uses a sliding window approach to predict potential bounding boxes at each location.
c. **RoI Pooling**: The generated proposals are reshaped using RoI pooling to have a fixed size, enabling them to be inputted into fully connected layers.
d. **Classification and Bounding Box Regression:** Each proposal undergoes two tasks in the fully connected layers. The first task is classification, where the model predicts the probability of the object belonging to a specific class. The second task is bounding box regression, where the model refines the object's location accuracy by predicting offsets for the bounding box coordinates.

II.3.3.2. Integration of RPN

By integrating the RPN into the Faster R-CNN architecture, the need for separate region proposal methods, such as selective search in R-CNN, is eliminated. The RPN generates region proposals by sliding a small network over the convolutional feature map, resulting in a more efficient and streamlined process. This integration enables end-to-end training of the entire object detection system, including the RPN and the subsequent stages of object classification and bounding box regression [Ren et al, 2016].

II.3.3.4. The anchoring mechanism

The anchoring mechanism in Faster R-CNN is pivotal for enhancing object detection accuracy. Anchors, which are predefined bounding boxes, serve as reference boxes during the regression process. They are positioned at various locations and scales across the image, generating multiple anchors at each spatial position on the feature map with diverse aspect ratios and scales.

This enables Faster R-CNN to effectively handle objects of different sizes and shapes. During training, the network learns to predict adjustments to transform the anchors into more accurate bounding boxes that tightly enclose the target objects.

By incorporating anchors, Faster R-CNN demonstrates flexibility and robustness in object detection, ensuring precise localization and classification of objects with varying scales and aspect ratios. The anchoring mechanism enhances Faster R-CNN's ability to handle object variations and ultimately leads to improved detection accuracy [Ren et al, 2016].

II.3.3.3. Unified Training in Faster R-CNN

In the architecture of Faster R-CNN, the RPN and Fast R-CNN share the same set of convolutional layers for feature extraction, which allows them to share computation, thereby improving efficiency and reducing training time. This design is also referred to as a "two-stage proposal + detection" method where the RPN first generates region proposals and then the Fast R-CNN uses these proposals to detect objects. This unified training approach allows the network to optimize all components simultaneously and globally, resulting in improved performance, enabling Faster R-CNN to achieve higher accuracy with fewer computational requirements [Ren et al, 2016].

II.3.3.5. Strengths of Faster R-CNN over RCNN and Fast RCNN

II.3.3.5.1. Speed and Efficiency

The "Faster" in Faster R-CNN refers to the model's significant improvement in computational efficiency and detection speed over its predecessors. In the R-CNN and Fast R-CNN models, the generation of region proposals was a time-consuming process, performed separately from the detection network and thus creating a computational bottleneck [Kim et al, 2020].

	R-CNN	**Fast R-CNN**	**Faster R-CNN**
Test Time Per image (with proposal)	50 sec	2 sec	0.2 sec
Speedup	1x	25x	250x
mAP (VOC2007)	66.0	66.9	66.9

Table II- 1: R-CNN speed comparison [Kim et al, 2020]

Faster R-CNN addresses this issue by incorporating a RPN within the network, which shares full-image convolutional features with the detection network [Ren et al, 2016]. This integration allows for nearly cost-free region proposals and dramatically accelerates the object detection process, making the system almost real-time detection (as shown in table II-1), such as autonomous driving and video surveillance.

Furthermore, the end-to-end training of Faster R-CNN, coupled with shared computation, significantly improves the computational efficiency. In contrast to Fast R-CNN, which also allowed end-to-end training but depended on selective search for proposing regions, Faster R-CNN completely embeds this process within the network. This streamlined architecture (as shown in Fig

II-14) not only eliminates the need for a separate module for region proposal but also reduces the number of model hyperparameters, simplifying the overall training process [Girshick et al, 2015].

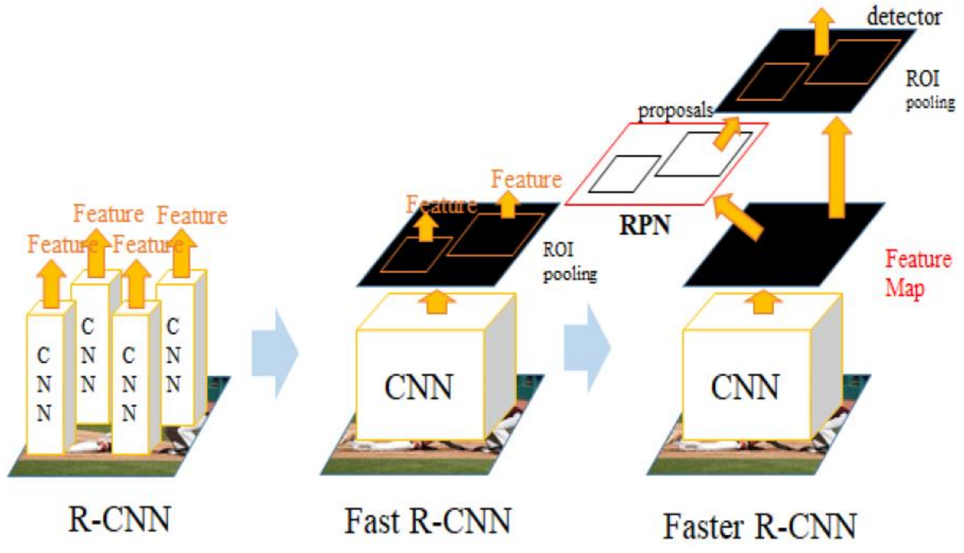


Fig II- 14: R-CNNs series [Kim et al, 2020]

II.3.3.5.2. Performance in Object Detection

Faster R-CNN also outperforms R-CNN and Fast R-CNN in terms of detection accuracy. The unified network in Faster R-CNN facilitates a unique "attention" mechanism, allowing the model to learn which regions of the image to focus on. By introducing the RPN, Faster R-CNN essentially performs a binary classification of object vs. not object for each proposed region, thereby refining the region proposals before the detection network processes them. This results in more accurate region proposals and subsequently more precise object detection [Lin et al, 2017b].

The introduction of RPNs also allows the model to propose regions with various scales and aspect ratios, further improving the model's ability to detect objects of different sizes and shapes. Hence, Faster R-CNN exhibits more robust performance across a wide variety of object detection tasks, including small object detection, compared to its predecessors [He et al, 2016].

And here is some result of the detecting object in images with Faster R-CNN that show the effectiveness of the algorithm

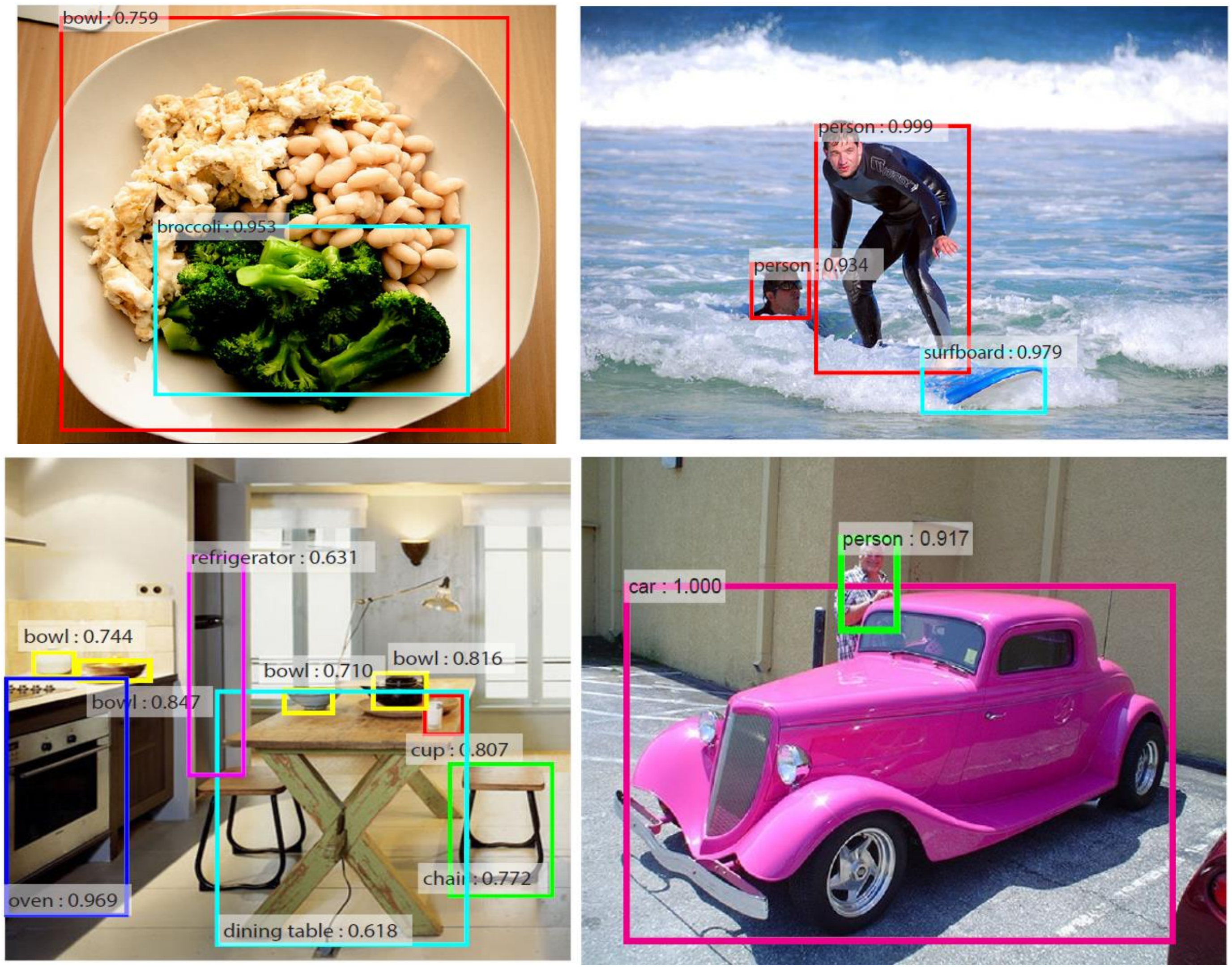


Fig II- 15: Faster R-CNN results [Ren et al, 2016]

I.4. Other approaches

Other common algorithms for detecting objects depend on deep learning and have been instrumental in the evolution of the field

YOLO (You Only Look Once) is a real-time object detection system that treats object detection as a regression problem. It divides the image into a grid and each grid cell predicts bounding boxes and class probabilities. YOLO is known for its speed but may have lower precision compared to two-stage methods like R-CNN [Redmon et al, 2016].

SSD (Single Shot MultiBox Detector) is another fast and efficient single-stage object detector. It improves upon YOLO by performing prediction on multiple feature maps of different scales, enabling better handling of objects of various sizes [Liu et al, 2016].

Mask R-CNN extends the Faster R-CNN model by adding a branch for predicting an object mask in parallel with the existing branch for bounding box recognition. This enables the model to generate precise segmentation masks for each detected object, which can be particularly useful in applications such as instance segmentation. [He et al, 2017].

Feature Pyramid Networks (FPN) address the challenge of detecting objects at different scales. They construct a feature pyramid within the network using a top-down architecture with lateral connections, allowing for multi-scale object detection. FPNs are commonly used as feature extractors in models like Faster R-CNN and Mask R-CNN [Lin et al, 2017b].

I.5. Conclusion

In this Chapter, we traced the evolution from the early days of convolutional neural networks to the state-of-the-art Faster R-CNN model for object detection. Faster R-CNN combines the region proposal network with the Fast R-CNN detector, resulting in improved accuracy and introducing real-time capabilities. Now, as we progress to the next chapter, we shift from theoretical comprehension to the practical application of Faster R-CNN. This transition allows us to explore the model's functionality in real-world scenarios, enhancing our understanding of this powerful object detection framework

Chapter III

Implementation and Faster R-CNN results

III.1. Introduction

In this chapter, we will delve into the implementation details of the Faster R-CNN algorithm, providing a comprehensive understanding of the step-by-step workflow involved.

The implementation of Faster R-CNN involves a series of critical steps that collectively contribute to its exceptional performance. We will explore each of these steps in detail, shedding light on the key concepts, techniques, and considerations involved in the process. We will discuss available libraries and frameworks that provide Faster R-CNN implementations, such as TensorFlow and PyTorch and explain the necessary configurations and hyper parameters for successful implementation. We will draw comparisons between Faster R-CNN pre-trained models, emphasizing the advancements made by Faster R-CNN and the reasons behind its superior performance.

III.2. Backbone Network Architecture

II.2.1. ResNet-50

ResNet50 is a variant of the ResNet (Residual Network) architecture that specifically refers to a ResNet model with 50 layers [He et al, 2016]. The model consists of a 7x7 convolutional layer, followed by multiple residual blocks (bottleneck layers), and finally a global average pooling, fully-connected layer, and Softmax function. Each residual block contains 3 layers (a bottleneck design), and the skip connection over these layers make it a ResNet-50 model.

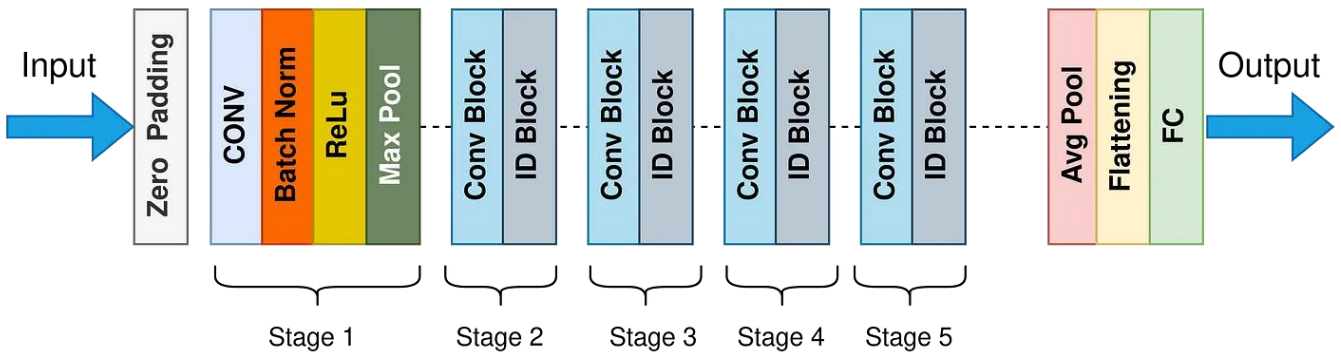


Fig III- 1 : Resnet-50 architecture [S09]

ResNet50 is a deep neural network with 50 layers. Its depth allows for the extraction of more complex and abstract features, enabling better representation of objects in images.

Residual Blocks: The core idea of ResNet is the residual block, which addresses the vanishing gradient problem encountered in training deep neural networks. A residual block contains a shortcut connection (skip connection) that adds the original input to the output of the block, allowing gradients to flow directly through the network and aiding in the training of deeper architectures.

When used as a backbone in Faster R-CNN, ResNet50 serves as a feature extractor. It processes the input image and produces feature maps with rich and hierarchical representations of the image content. These feature maps are then used for region proposal generation, object classification, and bounding box regression.

II.2.2. Mobile Net

MobileNet is a convolutional neural network architecture designed specifically for mobile devices with limited computational capacity. It introduces two new techniques - Depthwise Separable Convolutions and Width Multiplier - to reduce the computational cost of the network, making it efficient and fast [Howard et al, 2017].

Depthwise Separable Convolution: This technique separates the convolution process into depthwise convolution and pointwise convolution. In depthwise convolution, each channel of the input is convolved with its own set of filters, whereas in pointwise convolution, 1x1 convolutions are used to combine the outputs of the depthwise layer. This significantly reduces the computational cost.

Width Multiplier: It's a hyperparameter that allows the model to be thinner by reducing the number of channels, thus reducing computational cost.

MobileNet can be used for a variety of tasks, including object detection.

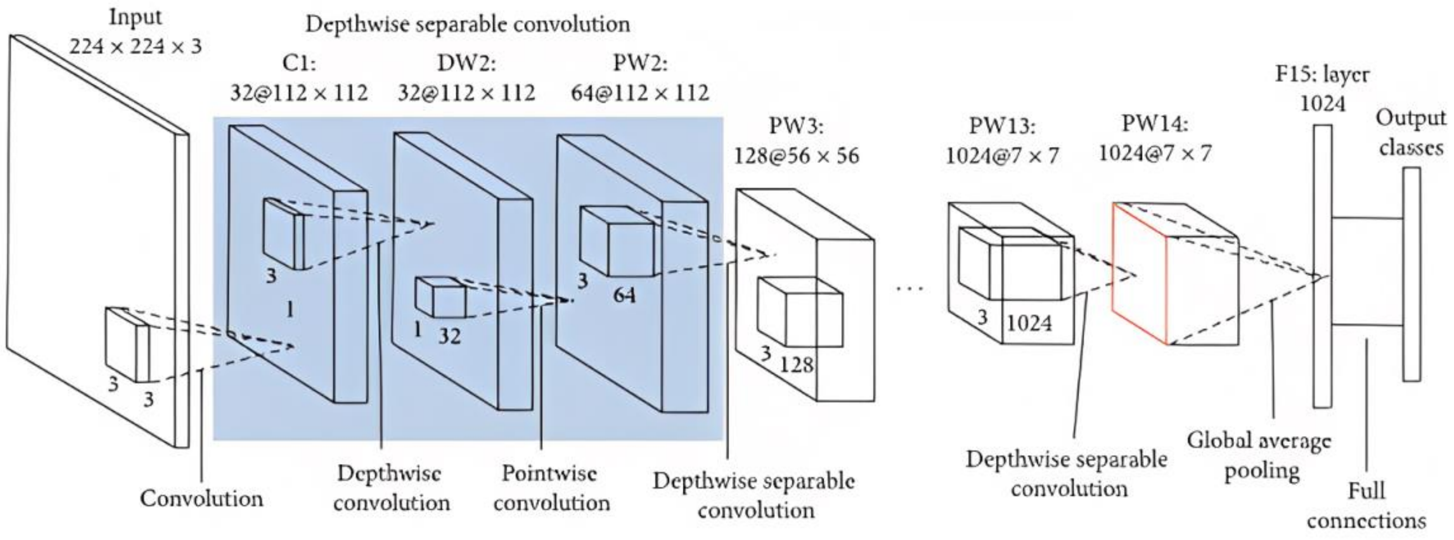


Fig III- 2: MobileNet architecture [S10]

III.3. Dataset

III.3.1. Pascal Voc 2012

The PASCAL Visual Object Classes (VOC) Challenge was a yearly image recognition competition that ran from 2005 to 2012, with the purpose of fostering research and development in object recognition by providing a standardized dataset and rigorous benchmarking of algorithms [S11].

The PASCAL VOC 2012 dataset is the last version of this competition. It contains a total of 11,530 images that include about 27,450 ROI-annotated objects and 6,929 segmentations for 20 object categories, which include:

- 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

III.3.2. COCO

The COCO (Common Objects in Context) dataset is a widely used benchmark dataset for object detection, segmentation, and captioning tasks. It contains a large collection of images with complex scenes and a diverse range of object categories. The COCO dataset is commonly used for evaluating the performance of object detection models, including Faster R-CNN.

The COCO dataset covers 91 different object categories, ranging from common objects like "person," "car," and "dog" to more specific categories like "hair drier" and "hot dog."

The training set contains approximately 118,000 images with over 860,000 annotated instances, providing bounding boxes, segmentations, and category labels. The validation set includes around 5,000 images with over 36,000 annotations. while the testing set contains approximately 41000 images [S12].

The evaluation metric for the detection task in COCO is the Average Precision (AP), and the mean Average Precision (mAP) is commonly used to assess the overall performance of the model across different object categories.

III.4. Software and Hardware setup

III.4.1. Hardware tools

All the results obtained were from our device, which has the following characteristics:

- CPU: Intel i5-1135G7 2.40 GHz
- RAM: 8G
- GPU: NVIDIA GeForce MX350 2G
- OS: Windows 11 Pro

III.4.2. Software

III.4.2.1. Libraries and Frameworks

PyTorch (Torch): It is another popular deep learning framework that supports Faster R-CNN implementation. It provides a flexible and intuitive interface for building and training object detection models. PyTorch allows users to leverage pre-trained models, such as Faster R-CNN, and provides tools for customizing and fine-tuning the models according to specific requirements [S13].

cv2 (OpenCV): Open Source Computer Vision Library is an open-source computer vision and machine learning software library. It is used for a variety of image and video processing operations like reading images and videos, performing transformations, applying filters, drawing bounding boxes, etc [S14].

PIL: The Python Imaging Library is often used in object detection tasks for image preprocessing and manipulation, such as loading and saving images, resizing, cropping, and applying transformations. It provides a convenient and efficient interface for handling image data in various formats [S15].

NumPy: it is a Python library used for working with arrays. It also has functions for working in the domain of linear algebra, Fourier transform, etc. In object detection, it's used for various mathematical operations on arrays and matrices [S16].

Sklearn (Scikit-learn): It is a free software machine learning library for Python. It is not used directly in object detection tasks but can be used for related tasks like splitting the dataset into train-test sets, evaluation metrics, etc. [S17].

Matplotlib: It is a plotting library for Python and its numerical mathematics extension NumPy. It provides a way to visualize images, plot graphs, display bounding boxes, etc.[S18]

III.4.2.2. LabelImg tool

LabelImg is a graphical image annotation tool written in Python. It's an excellent tool for manually annotating images - it allows you to draw bounding box on the objects in images and assign them class labels, creating the 'ground truth' labels required for object detection tasks. The tool supports both XML and JSON format, which are commonly used in object detection models [S19].

III.5. Metrics

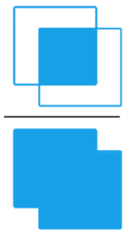
There are several metrics commonly used to evaluate the performance of object detection models

III.5.1. Intersection over Union (IoU)

Intersection over Union (IoU) is a commonly used evaluation metric in object detection tasks. It measures the overlap between two bounding boxes: the ground truth bounding box provided by the dataset and the predicted bounding box by the detection model. It is often used to evaluate the accuracy of object detection models.

The IoU is calculated as the area of the overlap between the two bounding boxes divided by the area of their union. Mathematically, it can be defined as:

$$IoU = \frac{Area\ of\ overlap}{Area\ of\ union}$$



Where:

- **Area of Overlap** is the area where the predicted bounding box and the ground truth bounding box intersect.
- **Area of Union** is the total area covered by the predicted bounding box and the ground truth bounding box.

IoU value ranges from 0 to 1, where:

- 0 means there is no overlap between the bounding boxes (worst case).
- 1 means there is a perfect overlap between the bounding boxes (best case).

In object detection tasks, a predicted bounding box is usually considered correct if its IoU with the ground truth box is above a certain threshold (for example, 0.5 in the PASCAL VOC challenge).

III.5.2. Average Precision (AP)

Average Precision (AP) is a critical metric employed to evaluate the performance of a detector across all threshold levels. AP computes the area under the Precision-Recall curve that is created by varying the decision threshold for assigning class labels.

The process starts by sorting all detections in descending order of their confidence scores. For each detection, we calculate Precision and Recall:

- **Precision** is the proportion of true positive predictions (TP) in the total predicted positives:

$$Precision = \frac{TP}{TP + FP}$$

- **Recall** is the proportion of true positive predictions in the total actual positives:

$$Recall = \frac{TP}{TP + FN}$$

Here, TP is the number of true positives, FP is the number of false positives, and FN is the number of false negatives.

The Precision-Recall curve is then plotted, and the Average Precision is calculated as the area under this curve.

Traditionally, in the PASCAL VOC challenge, the Average Precision is computed by taking the average of maximum precisions at a set of recall levels [0, 0.1, ..., 1,0]:

$$AP = \frac{1}{11} \sum_{r \in 0.0, 0.1, .., 1.0} P\_interp(r)$$

Where p_interp(r) is the maximum precision for recall greater than or equal to r.

III.5.3. Mean Average Precision (mAP)

Mean Average Precision (mAP) is a popular metric in the field of object detection that provides a single score that balances the precision and recall of a detector across all classes.

The mAP is then calculated as the mean of the APs over all classes:

$$mAP = \frac{1}{C} \sum_{i=1}^{c} AP_i$$

Where C is the number of classes, and AP_i is the Average Precision of class i.

In the more recent COCO challenge, mAP is computed across multiple IoU thresholds, making it a more robust measure of the performance of an object detector.

III.6. Faster R-CNN pipeline

Implementing the Faster R-CNN model for object detection involves a comprehensive pipeline with several intricate steps. Here, we present a concise overview of the general step-by-step process, encompassing both the training and testing phases

III.6.1. Preprocessing and Data preparation

The first step involves preparing your data for training and testing. This step typically involves the following sub-steps:

- **Data Collection:** We collect a dataset that contains the object(s) we're interested in detecting.
- **Annotation:** Using annotation tools, such as LabelImg, we draw bounding boxes around our object(s) of interest and label them. This creates the ground truth data that our model needs for training.
- **Cleaning:** Removing unwanted data or noises which might lead to inaccurate detection.
- **Augmentation:** Applying techniques such as scaling, cropping, flipping, rotation etc., to increase the variety of data. This helps the model generalize better and avoid overfitting.

- **Data Splitting:** Divide your data into training, validation, and testing sets. It is typical to use around 70-80% of data for training, 10-20% for validation (used during the model training to avoid overfitting), and 10-20% for testing (used to evaluate the final model).

III.6.2. Model configuration

Next, we'll configure the Faster R-CNN model for training. This involves setting up the architecture of the model (as shown in Fig II-11), including a backbone network (e.g., ResNet or VGG), a RPN and a classifier network.

III.6.3. Training Phase:

Stage 1 - Training the RPN:

In the first stage, the RPN is trained to generate accurate object proposals. The RPN takes the feature maps outputted by the backbone network and produces a set of object proposals, each with an objectness score. The RPN can accept inputs of any size and produce outputs correspondingly.

To train the RPN, each anchor box is assigned a binary class label (being an object or not). Anchor boxes that have the highest IoU with the ground truth boxes or those that have an IoU over a certain threshold (say 0.7) are assigned a positive label. The remaining boxes are assigned a negative label. The RPN is then trained on these labels using a binary cross-entropy loss.

Along with the binary labels, bounding box regression targets are also calculated for each anchor box. These targets are the offset required to make the anchor box match the ground truth box. This offset is then used to train the RPN using a smooth L1 loss.

Stage 2 - Training the Fast R-CNN detector:

Once the RPN has been trained, the proposals generated by it are used to train the Fast R-CNN detector. This step is similar to training a Fast R-CNN but instead of using selective search to generate proposals, the RPN-generated proposals are used.

To train the Fast R-CNN detector, each proposal is assigned a class label and a bounding box regression target. Proposals that have the highest IoU with the ground truth boxes or those that have an IoU over a certain threshold (say 0.5) are assigned the corresponding class label of the ground truth box. The remaining proposals are assigned a background label. The Fast R-CNN detector is then trained on these labels using a cross-entropy loss.

The bounding box regression targets are calculated in a similar way as the RPN, but now for each object class, these targets are then used to train the Fast R-CNN detector using a smooth L1 loss.

Stage 3 - Jointly fine-tuning: Once the RPN and the Fast R-CNN detector have been trained individually, they are combined and jointly fine-tuned. In this stage, the whole network (backbone + RPN + Fast R-CNN) is trained end-to-end. The fine-tuning process allows the RPN and Fast R-CNN to better adapt to each other.

Stage 4 – Loss calculation and Backpropagation:

Forward propagation: Pass the training images and the corresponding ground truth through the network. The network outputs class predictions and bounding box coordinates for each proposed region.

Loss calculation: The loss function quantifies the difference between the network's predictions and the ground truth. Faster R-CNN uses a combination of classification loss (Cross Entropy) and regression loss (Smooth L1 Loss).

Backpropagation and parameter update: Use backpropagation to compute the gradients of the loss with respect to the model's parameters, then use these gradients to update the parameters.

This procedure is repeated for several epochs until the model's performance plateaus on the validation data.

III.6.3. Testing phase

After training, the model is evaluated on a test dataset. The model generates bounding box proposals for each image in the test set, classifies the proposed regions, and refines the bounding box coordinates. These predictions are then compared with the ground-truth labels in the test dataset to compute performance metrics such as Precision, Recall, and mAP.

III.6.4. Model Tuning

Depending on the model's performance on the test data, adjustments might be required in the model's hyperparameters or data preprocessing steps to enhance the model's accuracy.

III.7. Working method

In this section, we provide details about our practical implementation of the Faster R-CNN approach for object detection. Specifically, we utilize two kind of pre-trained models (each one with 2 versions) to perform the detection tasks.

Type 01: we use the ResNet50 backbone with its two versions:

- **"fasterrcnn_resnet50_fpn"** it is a pre-trained model trained in 26 epochs, and uses 41 million parameters. [S20]

- **"fasterrcnn_resnet50_fpn_v2"** it is the same pre-trained model the difference is the number of parameters (43 million) [S21]. V2 refer to the second version.

Type 02: we use the MobileNet backbone with its two version:

- **“fasterrcnn_mobilenet_v3_large_fpn”** it is a pre-trained model trained in 26 epochs and uses 19 million parameters [S23].
- **“fasterrcnn_mobilenet_v3_large_320_fpn”** it is the same pre-trained model the difference is the size of input image which take 320 pixels [S24]

while the V3 refer to the third version of MobileNet.

These models, pre-trained on the COCO dataset, which covers a wide variety of object classes, including all 20 classes in the PASCAL VOC dataset.

The selection of these models was motivated by their diverse architecture offer diverse architectural attributes. The ResNet-50 based models bring the power of deep feature extraction, whereas the MobileNet v3 models provide efficiency, an important factor in real-time or resource-constrained applications. And Feature Pyramid Network (FPN), which is an architectural to capture multi-scale features [S22] improves the model's effectiveness in detecting objects at various scales.

By utilizing this pre-trained model, we save ourselves from the significant time and computational resources that would be required to train such a model from scratch.

For our detection task, we prepared a custom dataset, comprising **1000** diverse images. This collection was not random: the images incorporate various challenges of object detection such as variations in scale, viewpoint, illumination, and occlusion, offering a rigorous testing ground for our models.

To ensure accuracy, we meticulously annotated the images by manually applying bounding boxes and labels using the LabelImg tool (as shown in Fig III-3). This process enabled us to precisely identify and label the objects of interest within each image, ensuring that our dataset contains reliable and detailed information about the objects' locations and corresponding labels. These annotations, saved in XML files, serve as the 'ground truth' during the testing phase.

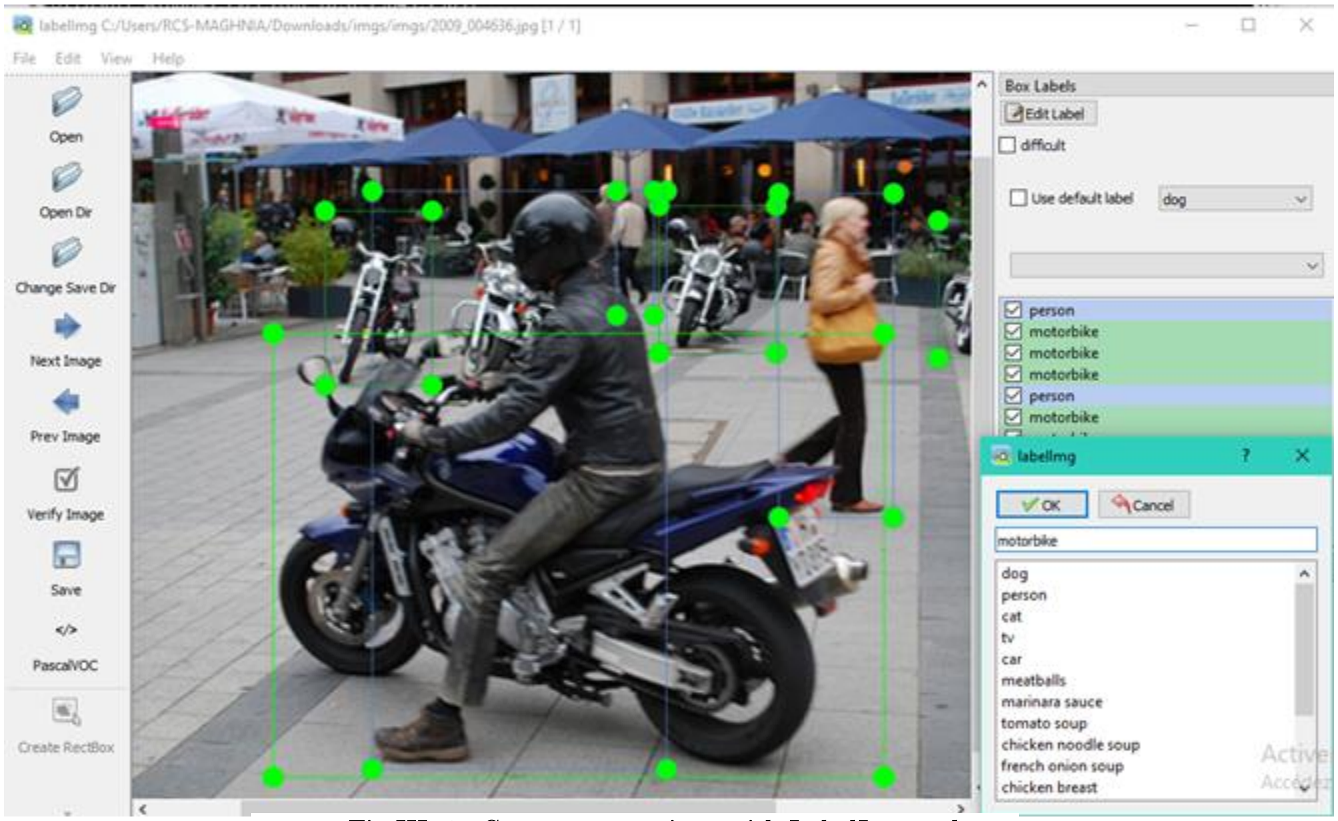


Fig III- 3 : Create annotations with LabelImg tool

During the testing phase, each pre-trained model generates generate predictions for object classes and bounding box coordinates for each image. To evaluate the accuracy of the object localization by the models, we implemented the IoU metric. IoU was calculated at different thresholds (0.5, 0.6, and 0.8), to assess the models' performance under varying levels of strictness in the localization precision

Finally, to measure the overall performance of the Faster R-CNN models for object detection, we calculated the mAP which aggregates the measure of the models' precision and recall performance across different object categories and IoU thresholds.

The detection of COCO objects will not affect our mAP results, as we implement the mAP function in a way that they only take into account Pascal Voc classes.

In summary, this methodology allows us to evaluate the effectiveness of different Faster R-CNN models for object detection tasks on the PASCAL VOC dataset, and analyze how they handle various object detection challenges. The results from the IoU and mAP measurements provide a detailed understanding of each model's performance, aiding in selecting the most suitable model for different object detection requirements.

III.8. Performance and Evaluation

In this section, we present the results of comparing pre-trained models with thoroughly discussed, supported by images

a. The mAP results

Pre-trained models \ The number of images in the dataset	100	411	1000
fasterrcnn_resnet50_fpn	0.651	0.642	0.540
fasterrcnn_resnet50_fpn_v2	0.715	0.654	**0.584**
fasterrcnn_mobilenet_v3_large_fpn	0.653	0.642	0.555
fasterrcnn_mobilenet_v3_large_320_fpn	0.643	0.599	0.516

Table III- 1: mAP@0.6 results for different pre-trained models and different number of images.

After using 4 pre-trained models and calculate mAP@0.6 metric, we got the results present in the table III-1 .We note that results vary according to the type of backbone used in the model, where the Resnet50 backbone offers better performance than MobileNet backbone in terms of object detection quality (mAP@0.6 measure).

Also we note the difference between the versions of each type of models, where in the model which use Resnet50 his second version is better than the first version, and this is due to the difference in the number of parameters. To contrary in the model which use MobileNet his first version is better than the second version, and this is due to the size determination of the input image (320).

Another important point, we observe that models quality depends also on the size of test set which is the more we develop it in size and add very difficult and complicated images, the more the challenge for models and therefore the mAP that may decrease slightly.

b. The AP results of classes:

In order to see the efficiency more precisely in terms of classes labels, we calculate the AP measure for each class and we choose the model which have the best result of mAP@0.6 measure from the table III-1 (fasterrcnn_resnet50_fpn_v2)

Classes	Aeroplane	Bicycle	Bird	Boat	Bottle	Bus	Car	Cat	Chair	Cow	Diningtable	Dog	Horse	Motorbike	Person	Pottedplant	Sheep	Sofa	Train	Tvmonitor
AP	0.618	0.607	0.620	0.464	0.537	0.680	0.503	**0.806**	0.430	0.442	**0.811**	0.714	0.681	0.572	0.515	0.426	0.425	0.580	0.695	0.566

Table III- 2 : AP@0.6 results for each class in PASCAL VOC dataset

In general, we observe that Faster R-CNN is very accurate especially in some classes as we indicated in the table III-2, like the class 'cat' with object detection quality 80.6%, and the class 'diningtable' with quality 81.1%. But we may note that some classes are of low quality like the class 'person' and 'sheep', and this is due to the type of object where can be a lot in the same image.

c. The time results

Based on the table III-1 we choose 2 model which have the best mAP measure to present the executing time of a simple image and a complex image, and we got the results shown in the following table:

Type of image / Models	Simple image	Complex image
fasterrcnn_resnet50_fpn_v2	4.09 sec	4.20 sec
fasterrcnn_mobilenet_v3_large_fpn	0.67 sec	0.87 sec

Table III- 3: Executing time for 2 type of images and 2 different Faster R-CNN models

Although the model use resnet50 backbone is better than a model use MobileNet backbone in terms of detection quality (mAP@0.6 metric) but MobileNet is higher than renet50 in terms of another measure , which is time executing as noted in table III-3.

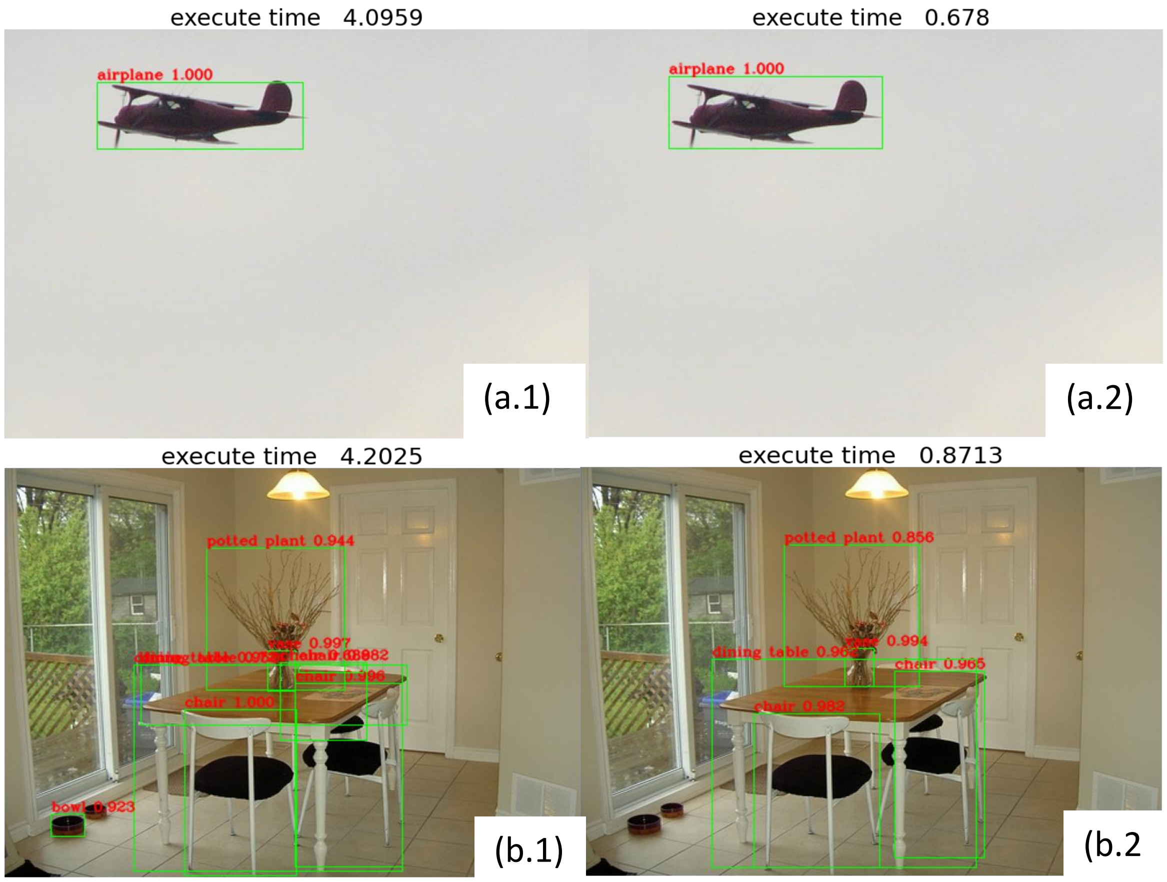


Fig III- 4: Time executing and the detection results of simple and complex image by 2 models

As we see in the previous figure (Fig III-4), in the simple image (a) where (a.1) is the output of 'fasterrcnn_resnet50_fpn_v2' and (a.2) is the output of 'fasterrcnn_mobilenet_v3_large_fpn' , we observe the same detection by the two pre-trained models . in the complex image (b) where (b.1) is the output of 'fasterrcnn_resnet50_fpn_v2' and (b.2) is the output of 'fasterrcnn_mobilenet_v3_large_fpn', Although MobileNet shows high efficiency in terms of time and exceeds Resnet50, the difference is also very clear in the specification as Resnet50 can detect two additional objects: the rear chair and also the bowl. The bowl is not within the PASCAL VOC classes, but Resnet50 detected it because it's pre-trained on COCO.

d. mAP results for different IoU threshold:

The calculation of IOU is essential for determining the precision and recall values used in the mAP metric, so we change the threshold of IOU and calculate mAP for 1000 images and 2 model (we choose this models in terms of the more accurate)

Models \ IOU values	0.5	0.6	0.8
fasterrcnn_resnet50_fpn_v2	0.590	0.584	0.502
fasterrcnn_mobilenet_v3_large_fpn	0.558	0.555	0.446

Table III- 4: The results of mAP measure with augmentation of IOU threshold

Thus as we note in the table III-4 if we augment the threshold of IOU the mAP measure lowed and this is due to how much the true boxes of objects and the prediction boxes match.

e. Object detection results

We present some of the results we obtained by applying the model that showed good performance on the images taken from our test set.

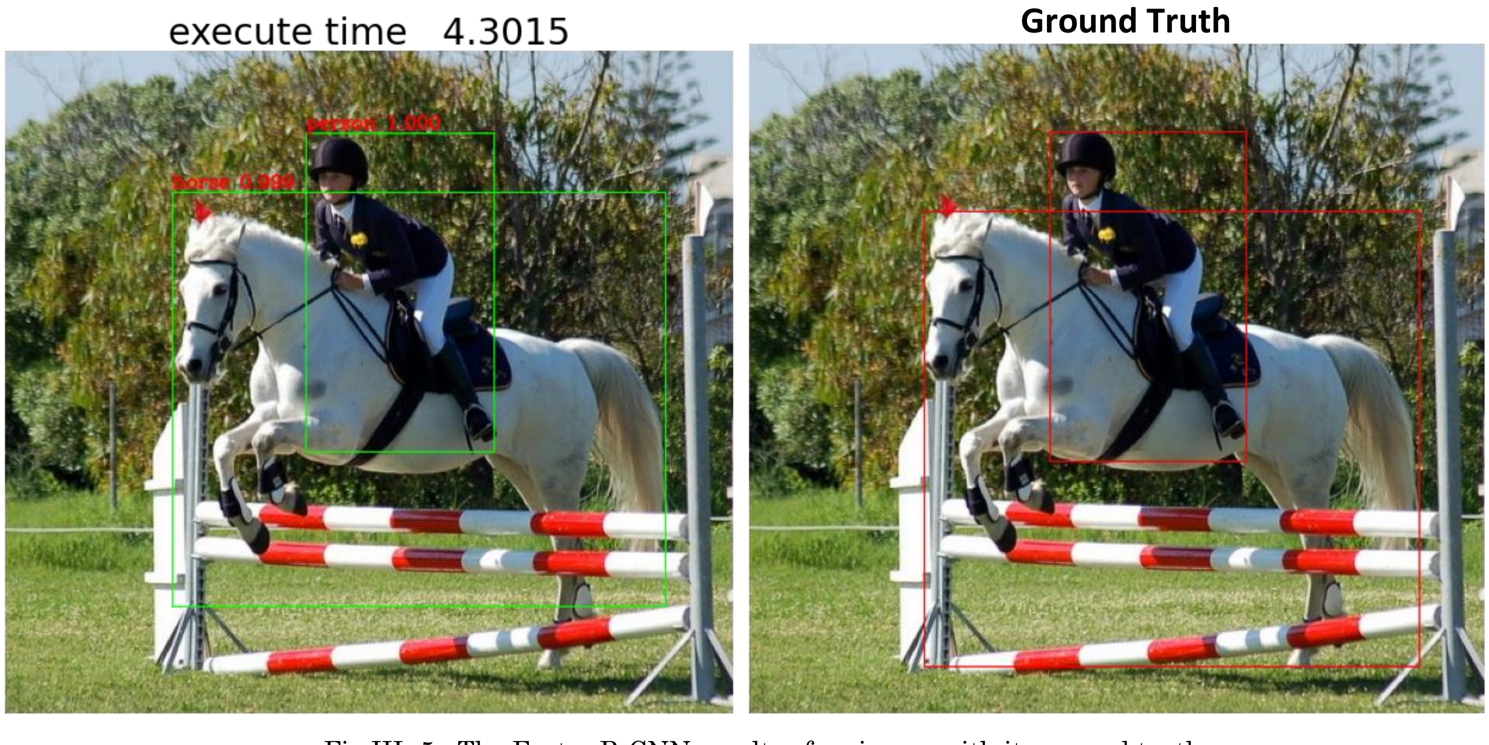
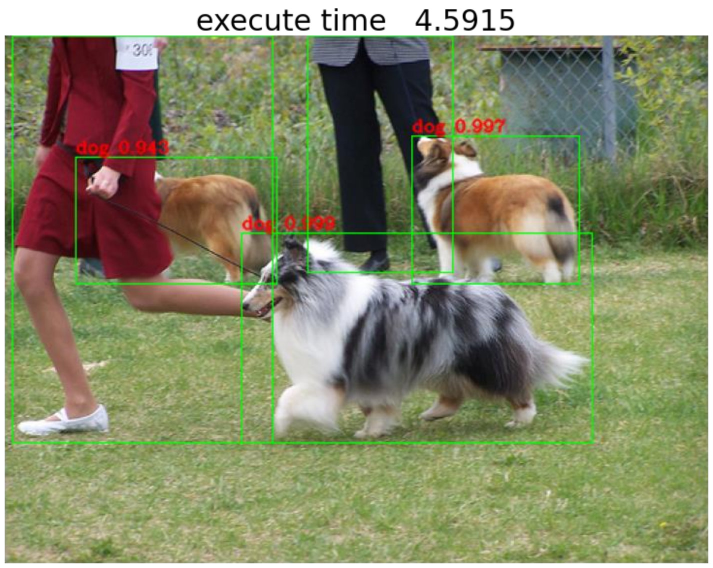
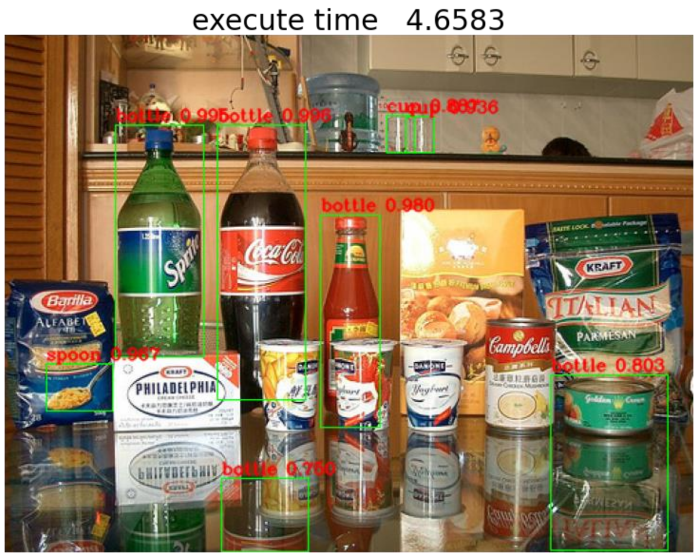


Fig III- 5 : The Faster R-CNN results of an image with its ground truth

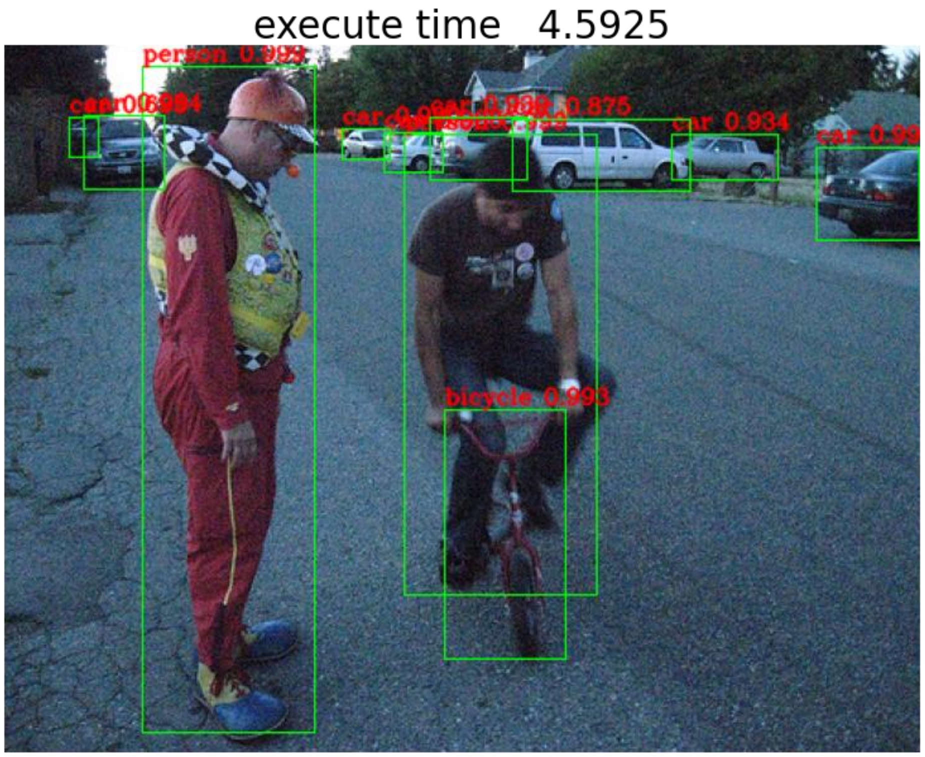
As shown in Fig III-5, the bounding boxes predicted by the Faster R-CNN model are ideal compared to the ground truth, such as what we see in the scores: around 99% for the horse label and exactly 100% for the person label.


execute time 4.5915
dog 0.943
dog 0.997
dog 0.999

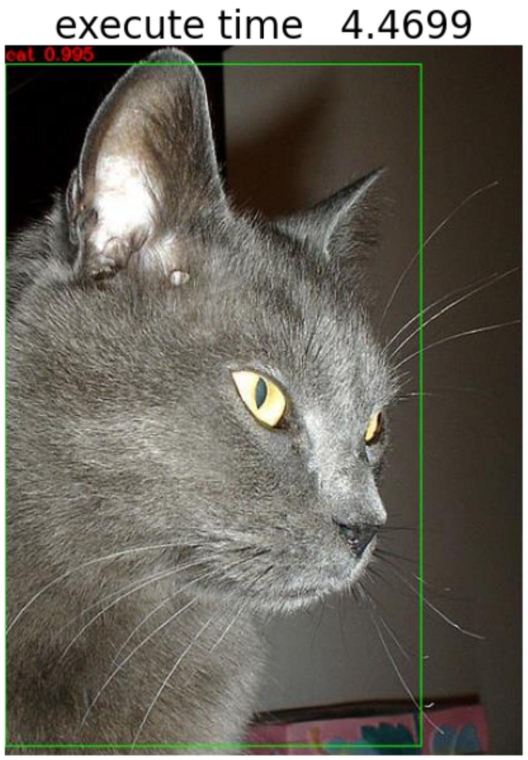
(a)


execute time 4.6583
bottle 0.995
bottle 0.996
bottle 0.980
spoon 0.967
bottle 0.803
bottle 0.750

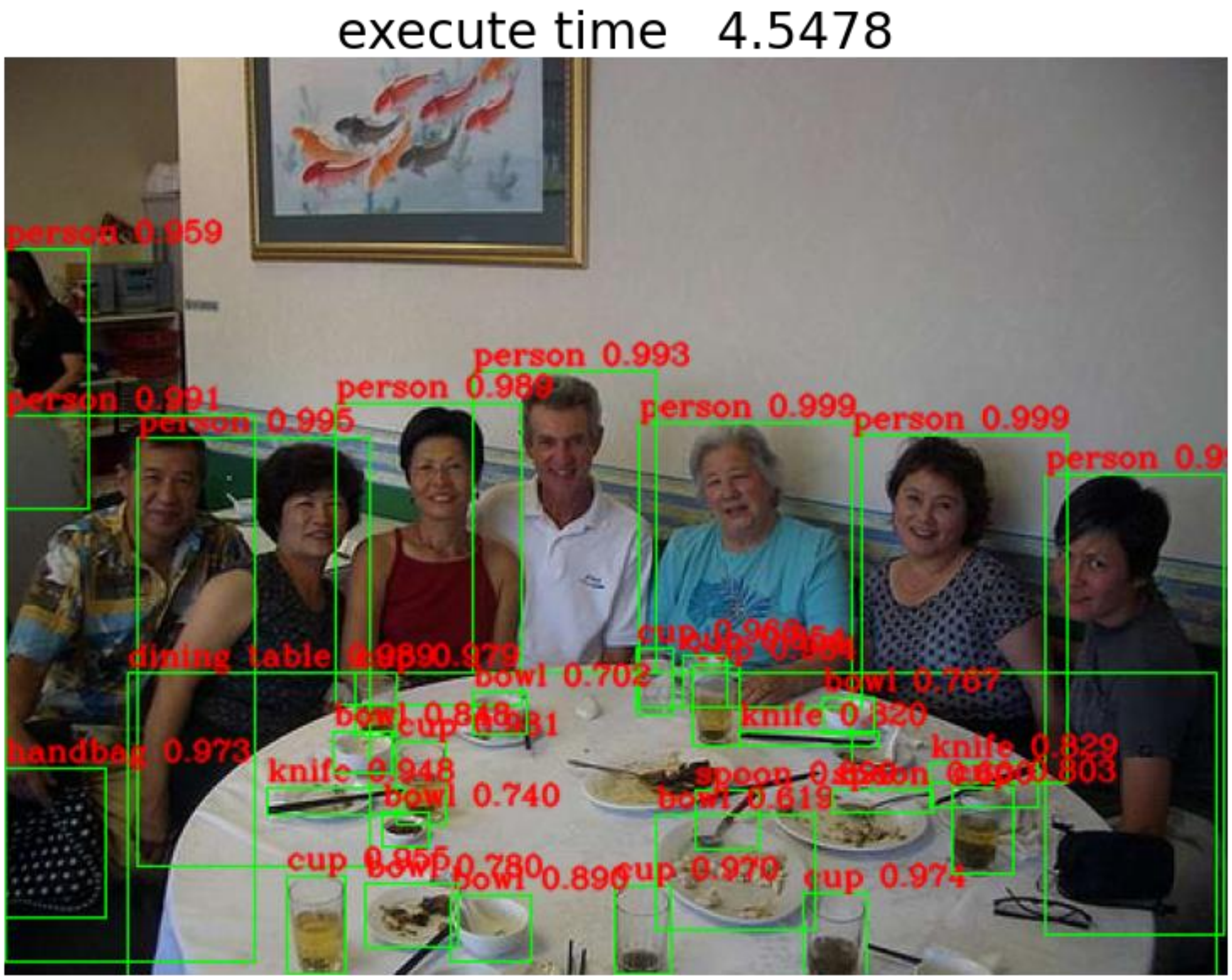
(b)


execute time 4.5925
person 0.998
car 0.934
car 0.99
bicycle 0.993

(c)


execute time 4.4699
cat 0.995

(d)



(e)

Fig III- 6 : Results of fasterrcnn_resnet50_fpn_v2 pre-trained

As shown Fig III-6, despite some problems such as lighting changes, occlusion of objects and different sizes (especially small object) that may cause problems for most detection algorithms, the Faster R-CNN showed a comprehensive adaptation and high performance in terms of precision in predicting bounding boxes and its labels and speed as well.

III.9. Conclusion

In this chapter, a comparative study was carried out on four pre-trained Faster R-CNN models, illustrating their efficacy in various object detection tasks. This in-depth evaluation not only highlighted each model's unique strengths and limitations but also their substantial contribution to the field of object detection. The results serve as a useful guide for selecting a suitable Faster R-CNN model for future research and applications, considering their differing abilities to handle complex object detection scenarios.

General Conclusion

In conclusion, this work presented a comprehensive exploration into the domain of object detection, underscoring its importance in various application in the real-world. We traced the evolution of object detection methodologies, from traditional methods with their inherent limitations to the rise of deep learning techniques, showcasing significant advancements in accuracy and processing speed.

Faster Region-based Convolutional Neural Network (Faster R-CNN) was central to this research, highlighted as a state-of-the-art technique that ameliorates computational inefficiencies of its predecessors while providing superior, real-time detection accuracy. The research demystified the working principles of Convolutional Neural Networks, Region-based Convolutional Neural Networks, and their advancements in the form of Fast R-CNN and Faster R-CNN.

At the core of this work was the implementation and evaluation of the Faster R-CNN approach for object detection, including an in-depth analysis of four pre-trained Faster R-CNN models on test set from the challenging Pascal Voc dataset. The annotation process for our images using LabelImg tool, the use of IoU as a metric to measure the accuracy of object detection, and the calculation of mAP for results analysis all illustrate the rigorous methodology adopted in this study. This comprehensive evaluation allows us to understand the effectiveness of each model in handling the complexity and variability of real-world object detection tasks, and demonstrated the resilience and advancements brought forth by Faster R-CNN, despite acknowledging certain limitations and challenges.

Looking ahead, we see a bright future for object detection, brimming with opportunities for refinement and the emergence of pioneering models. The insights gained from our experience with Faster R-CNN will undoubtedly steer these forthcoming advancements. It's incumbent upon us to persist in our research efforts to surmount the existing challenges, such as the dependence on high computational power and the enhancement of detection capabilities for smaller objects or those within congested scenes. A valuable area of future research lies in the exploration of single-stage detection models, such as YOLO or SSD, which offer potential improvements in efficiency. Moreover, it's crucial to explore strategies that harmonize processing velocity and accuracy, hence broadening the practical utility of these models in real-world scenarios.

Bibliography

[Bay et al, 2006] Bay, H., Tuytelaars, T., & Van Gool, L. (2006). SURF: Speeded up robust features. European Conference on Computer Vision, 404-417

[Bottou et al,2018] Bottou, L., Curtis, F. E., & Nocedal, J. (2018). Optimization Methods for Large-Scale Machine Learning. SIREV

[Cortes & Vapnik, 1995] C. Cortes and V. Vapnik, "Support-vector networks," in Machine Learning, vol. 20, pp. 273-297, 1995.

[Dallal & Triggs, 2005]: Dalal, N., & Triggs, B. (2005). Histograms of oriented gradients for human detection. In Proceedings of the IEEE conference on computer vision and pattern recognition (Vol. 1, pp. 886-893).

[Dollár et Zitnick, 2013] Dollár, P., & Zitnick, C. L. (2013). Structured forests for fast edge detection. In Proceedings of the IEEE international conference on computer vision (pp. 1841-1848).

[Fachruddin et al, 2018] Fachruddin , Errissya.R , Hendrawan , Yovi.P , Desi.K , Maria Rosario.B , (2018). " Real time detection on face side image with Ear Biometric imaging using integral image and Haar-like features" , ICECOS , IEEE .

[Felzenszwalb et al, 2010] Felzenszwalb, P. F., Girshick, R. B., McAllester, D., & Ramanan, D. (2010). Object detection with discriminatively trained part-based models. IEEE Transactions on Pattern Analysis and Machine Intelligence, 32(9), 1627-1645.

[Felzenszwalb et al, 2010] Felzenszwalb, P. F., Girshick, R. B., McAllester, D., & Ramanan, D. (2010). Object detection with discriminatively trained part-based models. IEEE Transactions on Pattern Analysis and Machine Intelligence, 32(9), 1627-1645.

[Fischler & Bolles 1981] Fischler, M. A., & Bolles, R. C. (1981). Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography. Communications of the ACM, 24(6), 381-395.

[Friedman et al, 2001] Friedman, J., Hastie, T., & Tibshirani, R. (2001). The elements of statistical learning (Vol. 1, No. 10). New York: Springer series in statistics.

[Girshick et al, 2014]: Girshick, R., Donahue, J., Darrell, T., & Malik, J. (2014). Rich feature hierarchies for accurate object detection and semantic segmentation. In Proceedings of the IEEE conference on computer vision and pattern recognition

[Girshick, 2015]**:** Girshick, R. (2015). Fast R-CNN. In Proceedings of the IEEE international conference on computer vision (pp. 1440-1448).

[Goodfellow et al, 2016] Goodfellow, I., Bengio, Y., & Courville, A. (2016). Deep learning. MIT Press

[Gudipati et al, 2016] Vamchi Krichna Gudipati , Oindrila Ray Barman , Mofica Gaffoor , Harshagandha and Abdelshakour Abuzneid , (2016). "Efficient facial expression recognition using adaboost and haar cascade classifiers" , university of Bridgeport USA , IEEE.

[He et al, 2015] He, K., Zhang, X., Ren, S., & Sun, J. (2015). Spatial Pyramid Pooling in Deep Convolutional Networks for Visual Recognition. IEEE Transactions on Pattern Analysis and Machine Intelligence, 37(9), 1904-1916.

[He et al, 2016] He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep residual learning for image recognition. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 770-778)

[He et al, 2017] He, K., Gkioxari, G., Dollár, P., & Girshick, R. (2017). Mask R-CNN. In Proceedings of the IEEE international conference on computer vision (pp. 2961-2969).

[Hosang et al, 2015] Hosang, J., Benenson, R., & Schiele, B. (2015). Learning non-maximum suppression. Proceedings of the IEEE conference on computer vision and pattern recognition, 5053-5062.

[Howard et al, 2017] Howard, A. G., Zhu, M., Chen, B., Kalenichenko, D., Wang, W., Weyand, T., ... & Adam, H. (2017). Mobilenets: Efficient convolutional neural networks for mobile vision applications. arXiv preprint arXiv:1704.04861.

[Huang et al, 2017] Huang, G., Liu, Z., van der Maaten, L., & Weinberger, K. Q. (2017). Densely connected convolutional networks. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 4700-4708)

[Karpathy, 2016] Karpathy, A. (2016). CS231n: Convolutional Neural Networks for Visual Recognition. Stanford University.

[Kim et al, 2020] Kim, J., Sung, J.-Y., & Park, S. (2020). Comparison of Faster-RCNN, YOLO, and SSD for Real-Time Vehicle Type Recognition. 2020 IEEE International Conference on Consumer Electronics - Asia (ICCE-Asia). doi:10.1109/icce-asia49877.2020.9277040

[Kingma & Ba,2014] Kingma, D. P., & Ba, J. (2014). Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980.

[Klein et al, 2007] : Klein, G., & Murray, D. (2007). Parallel tracking and mapping for small AR workspaces. In 2007 6th IEEE and ACM International Symposium on Mixed and Augmented Reality (pp. 225-234). IEEE.

[Krizhevsky et al, 2012] Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2012). ImageNet classification with deep convolutional neural networks. Advances in Neural Information Processing Systems 25, 1097-1105.

[Kyriakos et al, 2017] : Kyriakos, C., Tefas, A., & Pitas, I. (2017). A real-time single camera approach to mixed traffic vehicle tracking. IEEE Transactions on Intelligent Transportation Systems,

[LeCun et al, 1998] LeCun, Y., Bottou, L., Bengio, Y., & Haffner, P. (1998). Gradient-based learning applied to document recognition. Proceedings of the IEEE, 86(11), 2278-2324

[LenCun et al, 2015] LeCun, Y., Bengio, Y., & Hinton, G. (2015). Deep learning. Nature, 521(7553), 436-44

[Lienh & Maydt, 2002] R. Lienhart and J. Maydt. (2002). “An extended set of haar-like features for rapid object detection,” IEEE ICIP.

[Lin et al, 2017a] Lin, T. Y., Goyal, P., Girshick, R., He, K., & Dollár, P. (2017). Focal Loss for Dense Object Detection. Proceedings of the IEEE International Conference on Computer Vision (ICCV), 2980-2988.

[Lin et al, 2017b] Lin, T. Y., Dollár, P., Girshick, R., He, K., Hariharan, B., & Belongie, S. (2017). Feature pyramid networks for object detection. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 2117-2125).

[Litjens et al, 2017] : Litjens, G., Kooi, T., Bejnordi, B. E., Setio, A. A. A., Ciompi, F., Ghafoorian, M., ... & Sánchez, C. I. (2017). A survey on deep learning in medical image analysis. Medical image analysis

[Liu et al, 2016], Liu, W., Anguelov, D., Erhan, D., Szegedy, C., Reed, S., Fu, C. Y., & Berg, A. C. (2016). SSD: Single shot multibox detector. In European conference on computer vision (pp. 21-37). Springer.

[Lowe, 1999] Lowe, D. G. (1999). Object recognition from local scale-invariant features. Proceedings of the International Conference on Computer Vision, 1150-1157.

[Lowe, 2004] Lowe, D. G. (2004). Distinctive image features from scale-invariant keypoints. International Journal of Computer Vision, 60(2), 91-110

[Mikolajczyk & Schmid 2005] Mikolajczyk, K., & Schmid, C. (2005). A performance evaluation of local descriptors. IEEE Transactions on Pattern Analysis and Machine Intelligence, 27(10), 1615-1630

[Morel & Yu 2009] Morel, J. M., & Yu, G. (2009). ASIFT: A new framework for fully affine invariant image comparison. SIAM Journal on Imaging Sciences, 2(2), 438-469.

[Neubeck & Van Gool, 2006]: Neubeck, A., & Van Gool, L. (2006). Efficient non-maximum suppression. In International Conference on Pattern Recognition (pp. 850-855). IEEE.

[Nwankpa et al,2018] Nwankpa, C., Ijomah, W., Gachagan, A., & Marshall, S. (2018). Activation Functions: Comparison of trends in Practice and Research for Deep Learning. arXiv preprint arXiv:1811.03378

[Redmon et al, 2016]: Redmon, J., Divvala, S., Girshick, R., & Farhadi, A. (2016). You Only Look Once: Unified, Real-Time Object Detection. Proceedings of the IEEE conference on computer vision and pattern recognition, 779-788.

[Ren et al, 2016] Ren, S., He, K., Girshick, R., & Sun, J. (2016). Faster R-CNN: Towards real-time object detection with region proposal networks. In Advances in neural information processing systems (pp. 91-99).

[Rosasco et al,2004] Rosasco, L., De Vito, E., Caponnetto, A., Piana, M., & Verri, A. (2004). Are loss functions all the same?. Neural computation, 16(5), 1063-1076.

[Sermanet et al, 2013] Sermanet et al. Overfeat: Integrated recognition, localization and detection using convolutional networks. (2013). *arXiv preprint arXiv:1312.6229*.

[Simonyan & Zisserman, 2014]:Simonyan, K., & Zisserman, A. (2014). Very deep convolutional networks for large-scale image recognition. arXiv preprint arXiv:1409.1556.

[Sze et al, 2017] Sze, V., Chen, Y. H., Yang, T. J., & Emer, J. S. (2017). Efficient processing of deep neural networks: A tutorial and survey. Proceedings of the IEEE, 105(12), 2295-2329.

[Szegedy et al, 2013] Szegedy, C., Zaremba, W., Sutskever, I., Bruna, J., Erhan, D., Goodfellow, I., & Fergus, R. (2013). Intriguing properties of neural networks. arXiv preprint arXiv:1312.6199.

[Szegedy et al, 2015] Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., ... & Rabinovich, A. (2015). Going deeper with convolutions. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 1-9).

[Szeliski, 2010] Computer Vision: Algorithms and Applications. Springer

[Takshi et al, 2005] Takshi Mita , Toshimitsu Kaneko and Osamu Hori . (2005). "Joint Haar-like features for face detection". IEEE International conference on computer vision.

[Uijlings et al, 2013] Uijlings, J. R., van de Sande, K. E., Gevers, T., & Smeulders, A. W. (2013). Selective search for object recognition. International Journal of Computer Vision, 104(2), 154-171.

[Viola & Jones, 2001] Viola, P., & Jones, M. (2001). Rapid object detection using a boosted cascade of simple features. Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition. CVPR 2001, 1(I), 511-518.

[Zeiler & Fergus, 2014] Zeiler, M. D., & Fergus, R. (2014). Visualizing and understanding convolutional networks. In European conference on computer vision (pp. 818-833). Springer, Cham.

[Zhu et al, 2006] Zhu, Q., Yeh, M. C., Cheng, K. T., & Avidan, S. (2006). Fast Human Detection Using a Cascade of Histograms of Oriented Gradients. 2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06), 2, 1491-1498.

[Zitnick et al, 2014] Zitnick, C. L., & Dollár, P. (2014). Edge boxes: Locating object proposals from edges. In European conference on computer vision (pp. 391-405).

Web bibliography

[S01]:https://towardsdatascience.com/deep-learning-method-for-object-detection-r-cnn-explained-ecdadd751d22

[S02] https://en.wikipedia.org/wiki/Haar-like_feature

[S03] https://medium.com/analytics-vidhya/a-gentle-introduction-into-the-histogram-of-oriented-gradients-fdee9ed8f2aa

[S04] https://www.scirp.org/journal/paperinformation.aspx?paperid=115011

[S05] https://siddharthsankhe.medium.com/convolutional-neural-network-dc942931bff8

[S06]https://towardsdatascience.com/understanding-fast-r-cnn-and-faster-r-cnn-for-object-detection-adbb55653d97

[S07] https://www.modeldifferently.com/en/2021/10/image_classification/

[S08] https://lilianweng.github.io/posts/2017-12-31-object-recognition-part-3/

[S09] https://towardsdatascience.com/the-annotated-resnet-50-a6c536034758

[S10] https://wikidocs.net/165429

[S11] http://host.robots.ox.ac.uk/pascal/VOC/voc2012/

[S12] https://cocodataset.org/#home

[S13] https://pytorch.org/

[S14] https://docs.opencv.org/4.x/

[S15] https://pillow.readthedocs.io/

[S16] https://numpy.org/doc/stable/

[S17] https://scikit-learn.org/stable/documentation.html

[S18] https://matplotlib.org/stable/contents.html

[S19] https://github.com/heartexlabs/labelImg

[S20]https://pytorch.org/vision/main/models/generated/torchvision.models.detection.fasterrcnn_resnet50_fpn.html#torchvision.models.detection.fasterrcnn_resnet50_fpn

[S21]https://pytorch.org/vision/main/models/generated/torchvision.models.detection.fasterrcnn_resnet50_fpn_v2.html#torchvision.models.detection.fasterrcnn_resnet50_fpn_v2

[S22]https://jonathan-hui.medium.com/understanding-feature-pyramid-networks-for-object-detection-fpn-45b227b9106c

[S23]https://pytorch.org/vision/main/models/generated/torchvision.models.detection.fasterrcnn_mobilenet_v3_large_fpn.html#torchvision.models.detection.fasterrcnn_mobilenet_v3_large_fpn

[S24]:https://debuggercafe.com/using-any-torchvision-pretrained-model-as-backbone-for-pytorch-faster-rcnn/