Comparative Analysis of Pedestrian Detection using Deep CNN, R-CNN, Fast R-CNN and Faster R-CNN

A PROJECT REPORT

SUBMITTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR THE AWARD OF THE DEGREE
OF

MASTER OF TECHNOLOGY
IN
Artificial Intelligence

Submitted by

ARRAN P GONSALVES (2K22/AFI/04)

Under the supervision of

Dr. R.K. Yadav

DEPARTMENT OF COMPUTER SCIENCE ENGINEERING
DELHI TECHNOLOGICAL UNIVERSITY
(Formerly Delhi College of Engineering)
Bawana Road, Delhi 110042

MAY, 2024

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
DELHI TECHNOLOGICAL UNIVERSITY
(Formerly Delhi College of Engineering)
Bawana Road, Delhi-110042

ACKNOWLEDGEMENT

We wish to express our sincerest gratitude to Dr. R.K. Yadav for his continuous guidance and mentorship that he provided us during the project. He showed us the path to achieve our targets by explaining all the tasks to be done and explaining to us the importance of this project as well as its industrial relevance. He was always ready to help us and clear our doubts regarding any hurdles in this project. Without his constant support and motivation, this project would not have been successful.

Place: Delhi — Arran P Gonsalves

Date: 24.05.2024 — 2K22/AFI/04

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
DELHI TECHNOLOGICAL UNIVERSITY
(Formerly Delhi College of Engineering)
Bawana Road, Delhi-110042

CANDIDATE'S DECLARATION

I, Arran P Gonsalves, 2K22/AFI/04 students of M.Tech Artificial Intelligence, hereby declare that the thesis titled "Comparative Analysis of Pedestrian Detection using Deep CNN, R-CNN, Fast R-CNN and Faster R-CNN" which is submitted by me to the Department of Computer Science & Engineering, Delhi Technological University, Delhi in partial fulfillment of the requirement for the award of the degree of Master of Technology is original and not copied from any source without proper citation. This work has not previously formed the basis for the award of any Degree, Diploma Associateship, Fellowship, or other similar title or recognition.

Place: Delhi — Arran P Gonsalves

Date: 24.05.2024 — 2K22/AFI/O4

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
DELHI TECHNOLOGICAL UNIVERSITY
(Formerly Delhi College of Engineering)
Bawana Road, Delhi-110042

CERTIFICATE

I hereby certify that the Project Dissertation titled "Comparative Analysis of Pedestrian Detection using Deep CNN, R-CNN, Fast R-CNN and Faster R-CNN" which is submitted by Arran P Gonsalves, 2K22/AFI/04, Department of Computer Science and Engineering, Delhi Technological University, Delhi in partial fulfillment of the requirement for the award of the degree of Master of Technology is a record of the project work carried out by the students under my supervision. To the best of my knowledge, this work has not been submitted in part or full for any Degree or Diploma to this University or elsewhere.

Place: Delhi

Date: 24.05.2024

Dr. R.K. Yadav

SUPERVISOR

Abstract

In the field of computer vision, pedestrian detection is an important job that is frequently utilised in robots, auto-navigation, and video surveillance. Pedestrian detection is critical to lowering accident rates and enhancing automation in the transportation sector. Deep learning has revolutionised object identification in recent years, inspiring the development of many architectures for this use. In-depth comparisons of four well-known deep learning models for pedestrian detection are presented in this thesis: As an illustration, there are four types of convolutional neural networks: region-based (R-CNN), fast region-based (Fast R-CNN), faster region-based (Faster R-CNN), and deep convolutional (Deep CNN). This work's primary goal is to evaluate these models' performance on a number of performance metrics, including sensitivity to various situations, temporal complexity, and detection rate. Because each model has a unique way of enhancing efficiency, they are all important milestones in the evolution of object detection architecture. Multiple tests on a standardized pedestrian detection dataset will be included in this thesis to offer a clear grasp of the variations and similarities between these models. It also considers a variety of dynamically changing factors, including pedestrian density, occlusion occurrence, and illumination conditions. The precision, recall, F1-score, and average precision are among the metrics used to evaluate the detection task's accuracy. It also covers how long each model takes to train and infer, as well as how many resources it uses overall. The results show how well the four classes of models performed in terms of accuracy and time. Consequently, Deep CNNs and R-CNNs highlight significant facets of feature extraction and region-based detection techniques, even if Faster R-CNN keeps a favorable accuracy and speed ratio throughout the studies. Though it is somewhat less efficient than Fast R-CNN, which was designed to reach near real-time performance, Faster R-CNN performs better than both R-CNN and Fast R-CNN.

This thesis also provides the conclusion for the overall findings and discusses the future scope of this research and future possibilities for the development of pedestrian detection

systems. The following goals for future research can be identified: introducing the teaching and using the features of several architectures to improve the effect; strengthening the detection under different environments; expanding the study for other relevant jobs in computer vision. From this comparative study, the authors have provided a rich set of lessons that can be used in the continuous improvement of pedestrian detection and can aid the improvement of numerous intelligent systems in different application areas.

Contents

List of Figures

Chapter 1

INTRODUCTION

Pedestrian detection is one of the foundational activities in computer vision that has significant importance and prospective usage in many areas such as autonomous vehicles, security and surveillance systems, and robotics. Acquiring accurate detection and location of pedestrians in different terrains is greatly important in increasing the safety, security, and efficiency of the systems. Another common problem with the increasing popularity of self-driven cars is the ability to detect pedestrians properly and safely. Likewise in surveillance applications, accurate detection is helpful to the monitoring and prevention of crimes as well as in robotics applications, it helps to enhance safe interactions between humans and robots. Since this is a critical job, much effort has been invested in the creation of efficient algorithms that can successfully identify pedestrians even under difficult circumstances[5][6]. To this end, this introduction will begin with a historical perspective of the field of pedestrian detection starting from basic image processing techniques, the emergence of computer vision and machine learning, and finally the coming of age of deep learning; followed by a brief state-of-art in pedestrian detection and then finally describing the deep learning models that are widely being used in this field[5][6].

1.1 Background

Detection of pedestrians is one of the most important issues in the computer vision field and has all-embracing applications in areas such as the safety of the people, planning of cities, and automation. The object of the task is to detect the presence of human figures in a given image or frame of a video to be used in various applications like autopilot cars, security systems, and robots. In the past, pedestrian detection systems incorporated crucial features and standard methods of machine learning. Early techniques incorporated the use of features for example; Haar-like features, edge-based features, and HOG (Histogram of Oriented Gradients). These features were extracted manually and employed for experiments concerning classifiers such as Support Vector Machines (SVM) [7] and Decision Trees. Though the 2 approaches were beneficial in origin and helpful with some degrees of accomplishment in a contained setting, both offered severe weaknesses in that they could not be easily applied across various situation types. For example, the HOG descriptor explained by Dalal et al in 2005 has been a landmark in the development of the field known as pedestrian detection[8]. The approach used here was to therefore compute gradients on various parts of the image whereby histograms were to be built depending on the orientation of such gradients. Actually, the integration of the HOG feature descriptor with linear SVM yielded fine outcomes in the test using the INRIA person dataset, which

is widely used in pedestrian detection. However, a major problem in the application of the HOG+SVM pipeline was variations in scale, pose, and occlusion, frequently meeting in a real-life setting [Dalal & Triggs, 2005][8]. In the past, pedestrian detection systems incorporated crucial features and standard methods of machine learning. Early techniques incorporated the use of features for example; Haar-like features, edge-based features, and HOG (Histogram of Oriented Gradients). These features were extracted manually and employed for experiments concerning classifiers such as Support Vector Machines (SVM) and Decision Trees. Though the 2 approaches were beneficial in origin and helpful with some degrees of accomplishment in a contained setting, both offered severe weaknesses in that they could not be easily applied across various situation types. For example, the HOG descriptor explained by Dalal et al in 2005 has been a landmark in the development of the field known as pedestrian detection. The approach used here was to therefore compute gradients on various parts of the image whereby histograms were to be built depending on the orientation of such gradients[8]. Actually, the integration of the HOG feature descriptor with linear SVM yielded fine outcomes in the test using the INRIA person dataset, which is widely used in pedestrian detection. However, a major problem in the application of the HOG+SVM pipeline was variations in scale, pose, and occlusion, frequently meeting in a real-life setting.

CNN or Convolutional Neural Networks which emerged as a new branch of deep learning paved a huge way in the field of computer vision. CNNs, motivated by the neurocognitive visual areas in the brain, do not require any manual feature extraction but learn features in a hierarchical way from raw pixel data. Such an end-to-end learning capacity enabled CNNs to perform better than the traditional approaches that relied on the designing of peculiar features. The developments in deep learning for the purpose of pedestrian detection stemmed from the Deep Convolutional Neural Networks (Deep CNNs). Amounts of models like AlexNet, which were proposed by Krizhevsky et al. in 2012, proved the efficiency of deep learning techniques by the state-of-art performance on the ImageNet classification problem [9]. AlexNet succeeded in opening new possibilities for improving CNN architecture by presenting even more advanced architectures, for instance, VGGNet, GoogLeNet, and ResNet, which positively affected depth, computation speed, and accuracy.

However, though Deep CNNs were accurate when used in classification, using direct object detection was another issue. Object detection differs from object recognition in that object detection also involves localization, in which one needs to predict the boundaries of the objects such as rectangles around detected objects. This requirement for both identification and the determination of the position of the objects led to the creation of architectures particular to object detection.[4]. It is very crucial to detect pedestrians, especially in the case of self-driven cars. In cut and shot Detection of pedestrians accurately is very important in terms of the safety of car users and the pedestrians on the roads. In surveillance systems, pedestrian detection is crucial in the surveillance of open spaces such as roads, management of crowds, and counteraction of unlawful actions. In the field of robotics, pedestrian detection helps to avoid collisions and improve the interaction between humans and robots and collaboration between them. In the sphere of the urban environment, pedestrian detection systems also help in building smart cities since they gather data on the flows of pedestrians and their actions, thus contributing to the effectiveness of traffic control, public transportation, and infrastructure[10].

1.2 Motivation

1.2.1 Growing Importance of Pedestrian Detection

In recent years the advancement of autonomous systems and the growing concern for the safety of citizens in urban areas has raised the requirement for highly accurate pedestrian detection systems. Self-driving cars, for instance, need to correctly identify all the individuals on the roads so as to reduce the number of cases of accidents with or without causalities in congested urban areas. The smashing fails of self-driving cars in terms of pedestrian detection have made the question of detection systems very essential. Well-developed pedestrian detection systems do not only improve the safety and utilization of these cars or add to the trust in self-driving vehicles but are also vital to improve the effectiveness of the car [5]. Automatic pedestrian detection plays a crucial role in surveillance systems implemented in streets, airports, and shopping malls among others because they help in monitoring activities, crowd control, and detecting would-be security threats. In the current world people are moving to urbanized areas and security issues are on the rise thus the demand for improved surveillance systems. For these surveillance systems, pedestrian detection is essential in increasing their competency through the efficient monitoring of these areas and timely detection of mishaps [11].

In the robotics domain, pedestrian detection is a key component in ensuring suitable collaboration between man and the robot. These robots will be used in environments with the public, thus vital that the robots can first detect pedestrians and react accordingly to avoid dying or causing an accident. This capability is useful in applications such as service robots in public areas to flexible and collaborative robots in industries[12]. Pedestrian detection processes also significantly serve other areas ranging from urban planning to smart city concepts. This information ensures these systems enhance traffic control, coordinate and construct proper facilities for pedestrians, as well as enhance the existing and new systems of public transport in the city. This use of big data in urban planning optimizes public construction and makes cities welcoming, and accommodative to increasing human agglomerations.

However, pedestrian detection is still considered to be a very complex task in the context of deep learning, because of the following main problems. Some of the challenges include the physical differences of pedestrians vary greatly hence the appearance, pose, or clothing that pedestrians are wearing influences the detection. Pedestrians are not only of variable size and pose at different orientations, they also wear a wide variety of clothing, Repeated patterns complicate the detection. However, occlusions, which can be seen as situations where the detected object, here a pedestrian, is partially hidden behind other objects, is another major problem for this detection system since the latter has to make the best guess based on very few visual clues[12]. The issues of detecting pedestrians are even more complicated if there are some changes in the lighting conditions or weather, or if the background is rather cluttered. The systems that are used for detection should be able to withstand these changes and function correspondingly in all situations. Real-time processing is yet another factor that is quite essential in many applications, especially in self-driving cars and security systems where the detection needs to be immediate. Maintaining the fidelity of the solution while keeping it real-time has thus remained a big challenge in the field. The arrival of new deeper architectures to solve those challenges becomes as a great opportunity. Every transition from Deep CNN to Faster R-CNN has included modifications with the view of enhancing detection competency and efficiency. Thus, the comparative analysis of these models aimed at pedestrian detection will con-

tribute to the determination of their advantages and disadvantages. It can help to decide which model is preferable for a particular task and use it in practice as well as contribute to the development of new advancements [5].

1.2.2 Challenges and Opportunities

However, pedestrian detection involves some characteristics that make it difficult to solve even with the help of modern deep-learning techniques. This synthesis reveals some important points within the detection and recognition tasks because of changes in apparel, poses, and appearance of pedestrians. The obstacles for pedestrians to be detected range in shape, size, and orientation, dressed in various clothing which adds to the problem for detection algorithms. Additionally, it is possible to mention occlusions in which a pedestrian is partially or fully behind another object, which constitutes another weakness since the detection system has to guess where a pedestrian is based on limited vision[5]. In addition to the mentioned, lighting conditions, changes in the weather, and background clutter contribute in addition to making the task of pedestrian detection even more challenging. These variations mean that detection systems need to be immune to the various situations so that they are efficient in every case. Another requirement that many applications entail includes real-time processing, especially for applications such as self-driving cars or monitoring systems. The major issue that researchers are facing at the moment is to optimize the algorithms to obtain an accurate solution while keeping the computational time as short as possible[12].

Such a scenario to develop deep learning architectures to solve these problems can be considered as a promising prospect. Every modification beginning with Deep CNNs to Faster R-CNN has brought with them advancements that aim at increasing the detecting precision and rate. Nevertheless, for a proper analysis of these models particularly for pedestrian detection more elaborate comparisons are still required. It can help to choose the most appropriate model depending on the specific application and encourage further development[10]. Summing up, this thesis is justified by the increasing significance of pedestrian detection in contemporary applications, the multitude of new exciting problems and possibilities that appeared due to deep learning's progress, and the lack of an exhaustive comparative analysis of the methods suitable for further investigations and practical usage. Thus, by considering these motivations, this thesis tries to contribute to the progression of pedestrian detection performance and for the enhancement of safety along with intelligence in different domains[4].

1.3 Objectives

The objectives of this thesis are centered on performing an in-depth comparative analysis of four significant deep learning models for pedestrian detection: Regions with CNN features (R-CNN) and Fast R-CNN, and more advanced Faster R-CNN. The purpose of this analysis is to familiarize the reader with these models' performance characteristics, benefits, and drawbacks. The detailed objectives are as follows:

1.3.1 Comprehensive Performance Evaluation

The first objective is to assess the identified deep learning models based on their efficiency indicators. This involves:

Accuracy

Figuring out the true positive, true negative, false positive, and false negative detection rates of each model under different scenarios involving pedestrian detection. The ratio of correctly identified positive occurrences to all actual positive cases of the pedestrians in this case is known as recall. The ratio of accurately identified positive instances to all positive cases that the model was able to detect is known as precision. The F1-score, which is the harmonic mean of precision and recall, will be used in addition to accuracy to provide each model with a fair comparison and lessen the impact of significant variance among datasets.

Speed

Considering the inference time of each model since it takes an important role in the application domain that demands a real-time response. This includes the time taken to process every single frame of an image or video as well as the time taken to produce the detections.

Computational Efficiency

Evaluating the resource demand of each model: the RAM usage and the CPU load. This evaluation will aid in finding out on how implementable these models are, starting with servers all the way to embedded systems.

1.3.2 Robustness Analysis

Another important objective combines the evaluation of the model's performance under various situations that resemble real-world pedestrian detection cases. This involves:

Occlusions

Checking the models' performance against partially occluded pedestrians which is a common scenario in crowd situations. The gender condition on similar parts of the body will be tested to determine the level of generalization or specific prediction for the presence of pedestrians by each model when exposed to different degrees of occlusion.

Lighting Conditions

To compare the effectiveness of the models under different lighting conditions such as bright light, low light, and different shadows. Development of this analysis will also assist in finding the model's stability when in the presence of variation in illumination, especially that which occurs outside at night.

Scalability

Testing their capability to identify pedestrians at different scales and at different distances of the object in question. This involves assessing the capability of identifying pedestrians on different images where the pedestrians are of different sizes because of perspective or the distance they are from the camera.

Background Complexity

In the cluttered and dynamic background environments, to determine the models' ability to separate the pedestrians from background objects and dynamic scenes.

1.3.3 Comparative Analysis

Thus the main aim of this thesis is to make a comparative analysis of the four models with the view of determining their strengths and limitations. This analysis will involve:

Model Architecture Comparison

Explain the architectural variations and advancements of each model, as well as how the stated architectural features influence performance. This will involve analyzing the techniques used for feature extraction, region proposal, and classification to which each model subscribes.

Performance Trade-offs

Here it is necessary to define what sacrifices can be made to increase accuracy, computing speed, and other factors depending on the task. In turn, knowledge of the existing trade-offs will serve to better choose the appropriate model to solve specific problems, in which certain indicators of model performance may be more or less important.

Application Suitability

Make suggestions on the appropriateness of each model in different real-life situations. This entails enhancing the correlation between the various models to the needs of certain applications like auto-pilot, security, and robotic operations.

1.3.4 Practical Implementation and Evaluation

One of the major goals is to apply each model and compare their results based on the following grounds using the benchmarks of pedestrian detection. This involves:

Dataset Selection

Select reasonable benchmark datasets that are commonly accepted in the field of pedestrian detection to eliminate the discrepancies in the experimental results. Some of the datasets include the Caltech Pedestrian Dataset, INRIA Person Dataset, and Cityscapes Dataset.

Model Training and Testing

To ensure that like is compared with like, the models are trained on the above datasets using standard and well-thought-out methods with the same level of stringency. This entails steps such as data cleaning, data generation, and changing model parameters to get the best results.

Evaluation Metrics

Using AP, IoU measures, and ROC curves to provide a solid to medium to good to excellent or vice versa measure of the performance of the models.

1.3.5 Identification of Improvement Areas

According to the comparative analysis, one more goal is to reveal which aspects can be improved in existing models. This includes:

Performance Bottlenecks

Determining concrete aspects of the models that are creating problems, for example, long times for inference or high memory usage.

Enhancement Strategies

Proposing possible areas for improvement of the model's performance, e.g., regarding feature extraction region proposal methods, or development in hardware.

Hybrid Models

Also, consider the idea of refining individual architectures so that the performance of the best (or some of them) aspects from all of them can be incorporated into one model.

1.3.6 Future Research Directions

Consequently, this thesis also intends to present the possible directions for further research based on the conducted comparative analysis. This involves:

Emerging Technologies

Analyzing the possibilities of sections like attention mechanisms, transformers, and NAS for the improvement of pedestrian detection.

Real-world Challenges

Stressing further issues of current attention on pedestrian detection including - the ability to detect pedestrians during adverse conditions including darkness, rain or fog - The interactions between multiple pedestrians that can be complex both in their movement and in the interpretation required to navigate around them.

Interdisciplinary Applications

Furthering research on the expansions of pedestrian detection gadgets in other sectors including, health to trace patient mobility or agriculture to track the mobility of human and robotic systems in farming.
Through these aims, it is the hope of this thesis to contribute to the advancement of knowledge pertaining to pedestrian detection through deep learning models as well as to help applicable implementation and encourage subsequent studies and advancements in the said area that are of significant importance.

Chapter 2

LITERATURE REVIEW

Throughout the years, a lot of progress has been made in this area; from using general feature descriptors based on labor-intensive techniques to modern deep-learning approaches. In this chapter, a thorough literature review regarding pedestrian detection is discussed by presenting the object detection methods' historical analysis with a primary focus on the pedestrian detection area. It also explains historical approaches that include handcrafted feature-based methods, the incorporation of machine learning techniques, and the revolutionary development of deep learning especially CNNs and other affiliated techniques including R-CNN, Fast R-CNN, and Faster R-CNN among others.

2.1 Early Methods in Pedestrian Detection

2.1.1 Handcrafted Features

Early pedestrian detection methods primarily relied on handcrafted features designed to capture essential characteristics of pedestrians in images.

Haar-like Features:

Introduced by Viola and Jones in 2001, Haar-like features became a foundational element in early object detection methods [2]. This approach used rectangular features similar to Haar wavelets and applied an integral image for rapid computation. While Haar-like features were effective for face detection, their application to pedestrian detection faced challenges due to variations in pedestrian appearance and pose.

Histograms of Oriented Gradients (HOG):

Dalal and Triggs revolutionized pedestrian detection with the introduction of HOG descriptors in 2005 [8]. Using a technique called histogram of gradient directions, HOG breaks the picture up into small spatial areas in order to capture edge and gradient features that are typical of human forms. HOG became a standard technique for pedestrian identification and greatly enhanced detection performance when coupled with a Support Vector Machine (SVM) classifier.

2.1.2 Edgelet Features:

Proposed by Wang et al. in 2009, edgelet features are small, straight edge segments designed to capture the shape of pedestrians [13]. This approach demonstrated improved performance in cluttered environments by focusing on local edge patterns, which are less affected by variations in lighting and background.

2.1.3 Machine Learning Techniques

The fusion of hand-crafted features in combination with machine learning classifiers has been an improvement in the detection of pedestrians.

Support Vector Machines (SVM):

The integration of the HOG descriptors together with the linear SVMs created a new bar when it comes to the accuracy of the detection of pedestrians [8]. SVMs offered reliable classification features especially between the pedestrian and non-pedestrian models from the HOG features.

Decision Trees and Random Forests:

In their work, Enzweiler and Gavrila combined the decision trees and random forests in order to use them for pedestrian detection [14]. The advantages of these methods include; first, the ability to address appearance variation in pedestrians and also the vigor of environmental situations thus enhancing the classification by using multiple decision trees.

Boosting Methods:

Algorithms like AdaBoost were used to increase the detection accuracy by using multiple weak classifiers to come up with a strong classifier. The use of AdaBoost for face detection by Viola and Jones ref [2] can be considered as the first step to its application in pedestrian detection also because it facilitated the construction of cascaded classifiers to increase the detector's speed and reliability.

2.2 The Advent of Deep Learning

The appearance of deep learning especially Convolutional Neural Networks (CNNs) really brought a drastic change in object detection including pedestrian detection.

2.2.1 Convolutional Neural Networks (CNNs)

AlexNet:

The major revolution in deep learning for object detection was created in 2012 with the appearance of AlexNet, which took the championship at the ImageNet Large Scale Visual Recognition Challenge (ILSVRC) [9]. The proposed architecture of AlexNet with deep convolutional layers and using GPUs for the training introduced how CNNs aimed at learning higher-level features and were superior to previous algorithms.

VGGNet:

VGGNet was proposed by Simonyan and Zisserman in 2014, the very depth networks with small convolution filters [11]. The structure of VGGNet was relatively simple yet had high depth, which made it possible to obtain complex feature hierarchies; this made the model popular in feature extraction in different detection tasks such as pedestrian detection.

GoogLeNet:

The GoogLeNet (Inception v1) was proposed by Szegedy et al in 2015, The Inception module was proposed for deeper networks with less computation and parameters [15]. The inception module makes use of multiple convolution filters where each filter has a different receptive field size, this ensures that the network can effectively abstract many of the spatial features of the inputs.

ResNet:

He et al. presented Residual Networks (ResNet) in 2016; this was in an effort to deal with vanishing gradients with the help of residual connections [16]. Through the capability of training immensely deep networks, ResNet improved benchmarks, particularly on image classification and object detection algorithms, provided high generality features.

2.3 Region-Based CNNs (R-CNN)

Region-based CNNs (R-CNN) which we are going to study further as the next advancement was the integration of region proposal mechanisms with the CNNs.

2.3.1 R-CNN

In this paper, first introduced by Girshick et al in 2014, R-CNN took use of CNNs to employ region proposals produced by the selective search, which showed evident detection accuracy enhancement . R-CNN operates in three stages: making region proposals, extracting features with the CNN, and making the final decision with the help of a linear SVM. However, in terms of accuracy, Regions with CNN (R-CNN) was rather slow, even though its accuracy was high because the CNN had to be run on each proposal region. In this paper, first introduced by Girshick et al in 2014, R-CNN took use of CNNs to employ region proposals produced by the selective search, which showed evident detection accuracy enhancement [17]. R-CNN operates in three stages: making region proposals, extracting features with the CNN, and making the final decision with the help of a linear SVM. However, in terms of accuracy, Regions with CNN (R-CNN) was rather slow, even though its accuracy was high because the CNN had to be run on each proposal region.

2.3.2 Fast R-CNN

Girshick presented Fast R-CNN in 2015 [14] in an attempt to address the computational capacity issue with R-CNN. Before using the ROI pooling layer in Fast R-CNN, the input picture is first processed through CNN's entire image network to obtain a feature map

of the same size for each proposal. This also provided a higher detection rate at a lower level of accuracy and a comparatively shorter calculation time.

2.3.3 Faster R-CNN

In 2016, Ren et al. refined Fast R-CNN to Faster R-CNN that incorporated a Region Proposal Network (RPN) as a part of CNN [4]. The RPN creates region proposals directly inside the network, which means that the generation of region proposals mostly does not come with any additional costs. This single framework provided the mean of attaining the highest level of accuracy and speed which fulfills real-time criteria.

2.4 Other Advanced Architectures

In addition to the basic family of R-CNN, many other complex structures introduced significant improvements in the field of pedestrian detection.

2.4.1 YOLO or (You Only Look Once)

Redmon et al. initially presented the YOLO design which reformulated object detection as a single regression problem [18]. YOLO predicts bounding boxes and classes at the same time from a full image in one pass, making object detection in real-time possible. Due to its fast and efficient working ability, YOLO became preferable, particularly for apps that need the highest accurate detections.

2.4.2 SSD (Single Shot MultiBox Detector)

In their study, Liu et al. employed SSD, wherein the bounding boxes and class scores of objects in an image are estimated using a single deep neural network [19]. SSD offers great detection accuracy with minimal processing time by offering feature maps at various scales to identify objects of varying sizes.

2.5 Improved Architectures

RPN+BF: Cai et al. introduced a boosted forests approach on top of region proposals generated by RPN, demonstrating enhanced performance in pedestrian detection [20]. This method combines the strengths of deep learning for proposal generation with traditional boosting techniques for robust classification.MS-CNN: Additionally, Multi-Scale CNN (MS-CNN) was suggested by Cai et al. [21]. MS-CNN uses a collection of subnetworks for region proposal and detection in order to identify pedestrians of several sizes. The difficulty of recognizing pedestrians at varying sizes and distances is addressed by MS-CNN, which increases detection accuracy over a range of scales.

This chapter included an overview of the advancements in pedestrian identification techniques and the necessary switch from manually created features to deep learning-based techniques. The fascinating functions of CNNs and their variations, including R-CNN, Fast R-CNN, and Faster R-CNN, were discussed. Taking into account the relevant literature, it is demonstrated that while the deep learning approach has improved

pedestrian detection results, issues including occlusion, scaling, and lighting conditions are still important areas for research and improvement. The architectures and a comparison of Deep CNN, R-CNN, Fast R-CNN, and Faster R-CNN for pedestrian detection will be covered in the upcoming chapters.

Chapter 3

RESEARCH METHODOLOGY

Deep learning approaches, particularly CNNs and other upgraded models, are used in the following study to achieve real-time and state-of-the-art pedestrian identification. Another crucial problem in computer vision, which is utilized extensively in security, traffic monitoring, and self-driving automobiles, is identifying pedestrians. This chapter aims to provide a detailed overview of the methodology used to assess and contrast the performance of various advanced deep learning models, such as Region-based Convolutional Neural Networks (R-CNN), Fast R-CNN, Faster R-CNN, and Deep Convolutional Neural Networks (Deep CNNs). This chapter will aim to give a rich account of how the research objectives were met in terms of data acquisition and cleaning, selection of the appropriate models as well as steps of applying the models, and, evaluating their performances. This section details the methodology applied in order to accomplish the research objectives; which entails the approach taken with regard to; the selection of datasets and data preprocessing, selection and application of machine learning models, and metrics of model performance.

3.1 Dataset Selection and Preprocessing

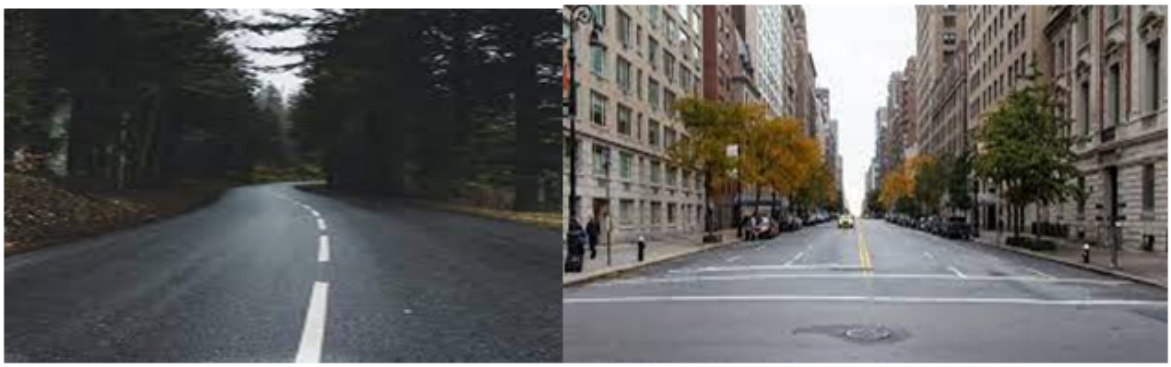
3.1.1 Dataset Characteristics

The dataset used in this study was carefully sourced from Kaggle, an esteemed site famous for hosting various datasets from numerous fields. Kaggle was chosen as the primary source because it is known to contain specified datasets with detailed annotations and balanced datasets appropriate for training and assessing the model performance, especially for pedestrian detection. This dataset was created with the objective of having an equal number of images in the three predefined categories and includes a total of 1616 images. Specifically, the dataset was evenly divided into two distinct categories: About pedestrians, 808 SC images were captured with pedestrians in the scene while 808 other images were naturally obtained without pedestrians in the scene. It is crucial to establish the balance of representing each class to prevent model biases and achieve effectiveness when trained on the created dataset.

This is the reason why there can be no doubt about the choice of making use of a balanced dataset, as providing equal and adequate representation of both positive and negative cases is a necessity. In this way, a distinguishing feature of the classification of the model - the context of its work setting pedestrian or not – increases its efficiency and generalization. However, more annotations available in the dataset contribute to providing

significant data support for offering more profound model solutions and assessments.

Pedestrian



No Pedestrian

Figure 3.1: Dataset

Looking at each of the images in the given dataset, all of them are very well described with various labels separating the presence of a pedestrian and other pertinent information if any. Such detailed annotations make the deep learning models more capable of learning fine-grained features that naturally exist in the detection of pedestrians, thus increasing their ability to provide accurate and consistent predictions. Furthermore, the dataset's diversity is a crucial aspect that underscores its utility in real-world applications. It encompasses a wide spectrum of scenarios, encompassing varying lighting conditions, environmental settings, pedestrian densities, and occlusion levels. This diversity not only enriches the learning experience for the model but also ensures its adaptability to diverse real-world scenarios, thus enhancing its practical applicability. In summary, the dataset sourced from Kaggle represents a meticulously curated collection of 1616 images, meticulously divided into two equal categories of 808 images each, featuring pedestrians and non-pedestrians, respectively. Its rich annotation structure, balanced composition, and diverse content render it an invaluable resource for training and evaluating deep learning models tailored for pedestrian detection tasks.

3.1.2 Data Augmentation

To improve the variability of the training dataset and to ensure the model's ability to extrapolate to unseen data, several augmentation procedures were incorporated systematically. These techniques are very important in training deep networks and especially in high-risk and difficult problems such as the detection of pedestrians. Here, we detail the specific augmentation strategies employed, and how they contribute to the model's per-

formance: Here, we detail the specific augmentation strategies employed, and how they contribute to the model's performance. For instance, if randomness is applied to rotation, the convolutional neural network learns how to predict pedestrians in various orientations, and when flipping is used, the model becomes insensitive to horizontal mirrors. Scaling and translation make the model familiar with various positions and sizes of the pedestrians and it will make the model immune to distortions of pedestrian size and position. If the method of data augmentation is applied, then not only the base data set is extended but it also provides a more complex training environment, which in turn makes the model learn more comprehensive possible features. Random rotations are rotations of the images by a certain number of degrees in either direction, either clockwise or anti-clockwise. This augmentation technique assists the model in being insensitive to the orientation of the pedestrians and therefore ensures that it learns to recognize the pedestrians irrespective of how they appear in the image. For example, movements within -30 to +30 degrees allow for various orientations; therefore, the model can successfully detect pedestrians who are somewhat crooked or when their image is taken from a different angle. Rotating images horizontally and vertically also helps the model to be invariant to the images' reflections which gives the model an ability to detect pedestrians in whatever position. Horizontal flips are perhaps useful when the pedestrians might appear in either direction in the scene as when crossing the streets. Vertical flips, while much less likely to occur in response to real-world stimuli, can additionally put pressure on the model to seek out more geometric invariances, because it otherwise solely might have learned to seek out patterns of positions that were often up or down. Random zooms entail scaling the images either in or out to justify different distances of the pedestrians as seen by the camera. This technique presents the model differently from the pedestrian at different scales enabling the model to respond at short and long ranges.

Figure 3.2: Examples of data augmentation. [1]

The model first, randomly crops out images of the pedestrians which helps it learn all the small features of the pedestrians and also learn to recognize the pedestrians out of the overall view. Phases random shifts and translations displace the whole image in random directions at certain angles. This augmentation makes the position of pedestrians within the frame slightly more shifted and improves the model's capability to detect a pedestrian who is not in the image center. This is particularly helpful when it comes to practical applications where pedestrians can occur at any region in the field of vision thus avoiding

the situation where the model learns to detect pedestrians only in certain regions of the image. These augmentation techniques were done using strong frameworks such as Opencv and TensorFlow that are developed in the Python language. OpenCV offers numerous functions to manipulate an image such as rotation, flipping, zooming, and shifting of an image. Last but not least, TensorFlow, which is one of the mostly used machine learning libraries, also has effective tools and functions to make these augmentations during training flow. These augmentation techniques help when implementing them to prevent overfitting, it is common to encounter a model that has a high accuracy on the training data but a poor accuracy on the unseen data. By feeding the model with a very diverse training dataset, it is trained well, and thus it can generalize well hence becoming more reliable. Perturbation imitates actual scene fluctuations and issues to ensure that the model acquires robust and versatile knowledge to enhance pedestrian detection in complex and fluctuating surroundings once released into the world. Therefore, the usage of random rotations, flips, zooms, and shifts by utilizing OpenCV and TensorFlow plays a significant role in expanding the dataset. These techniques augment the training corpus, promote model relativity, and, therefore, aid in developing a more reliable and precise pedestrian detection model.

3.1.3 Data Splitting

Data splitting is a complex process that is incorporated into the sequence of operations for constructing machine learning models, one of the aims of which is the evaluation of the model on new data samples. In the current paper, the data set has been split in the ratio of 80/20 for training and testing respectively. This division of available data into the training and test sets is often beneficiary to provide a large training sample and at the same time have a highly reliable testing sample.

Detailed Breakdown of the Split

- **Total Dataset Composition:**
  - **Total Images:** 1620
    - Images with Pedestrians: 810
    - Images without Pedestrians: 810
- **Training Set Composition (80% of the total data):**
  - **Total Training Images:** 1294
    - Images with Pedestrians: 647
    - Images without Pedestrians: 647
- **Testing Set Composition (20% of the total data):**
  - **Total Testing Images:** 326
    - Images with Pedestrians: 163
    - Images without Pedestrians: 163

Rationale Behind the 80/20 Split

- **Sufficient Training Data:**
  - The 80% portion allocated to the training set comprises 1294 images, ensuring the model has a substantial amount of data to learn from.
  - With 647 images of each class, the model can successfully pick up characteristics that differentiate pedestrians from non-pedestrians.
  - A larger training set helps the model recognize various patterns and variations in the data, such as different pedestrian poses, backgrounds, lighting conditions, etc.
- **Robust Evaluation:**
  - The 20% portion designated for the test set consists of 326 images, providing a comprehensive dataset to evaluate the model's performance.
  - The test set, which is hidden from view during training, guarantees an objective assessment of the model's generalization skills.
  - Reliability in the computation of performance metrics, such as F1 score, recall, accuracy, and precision, is ensured by a suitably large test set.

Importance of Data Splitting in Machine Learning

- **Balance Between Training and Testing:**
  - This ratio creates a balance between the quantity of data kept for assessment and the quantity accessible for training the model.
  - A test set that is too tiny might not offer a trustworthy assessment of the model's performance, while a training set that is too small could result in underfitting.
- **Generalization:**
  - By keeping a separate test set, it is feasible to evaluate the model's ability to effectively generalize to new, unknown data.
  - This is essential for determining the model's performance in practical situations.
- **Validation of Model Performance:**
  - By separating the data, it is ensured that performance measurements like F1 score, accuracy, precision, and recall accurately represent the actual capabilities of the model.
- **Prevention of Data Leakage:**
  - By clearly separating the training and testing datasets, data leakage is prevented.
  - By doing this, it is ensured that the training process is not influenced by information from the test set.

This distribution of training data into 80% and testing data in 20% is a good strategy in making sure that any model that is being built has been tested on a broad range of features. Hence, while training the model with 1294 images and using 326 images to test it, enough data is offered to the model for improved learning while at the same time ensuring that an untampered test set is also given to the model for evaluation. The devised methodology serves for better feature learning and pattern detection and allows evaluation of the model's performance in terms of generalization, which is crucial for practical applications.

3.2 Model Selection and Implementation

It can be stated that the selection of a suitable model is a critical step in the process of designing an efficient pedestrian detection system. The ideal model should compute the various factors that are needed quickly and accurately, and at the same time, it should be robust and easy to implement. This research is conducted to apply state-of-the-art deep learning models especially convolutional neural networks (CNN) that showed very high performance in object detection. The process of selection involves a comparison of several modern state-of-the-art models in terms of architectural strategies of the model, performance indices, and their applicability to the field of pedestrian detection. We will analyze and implement four prominent models: These are Deep Convolutional Neural Network (Deep CNN), Region-based Convolutional Neural Network (R-CNN), Fast R-CNN, and Faster R-CNN.

3.2.1 Criteria for Model Selection

When choosing a model for pedestrian detection several important factors have to be taken into consideration in order to be sure that the chosen model is ready to perform all the necessary operations at the best rate. The common constraints include the accuracy of the model, the speed at which the model is executed, the stability of the model, and the extent to which it is reusable or scalable as well as the easiness in implementing the model.

Precision is the primary consideration when choosing a model in the case of pedestrian detection. An ideal model must generally detect and locate pedestrians in different conditions for it to be said to be effective. High accuracy is crucial to minimize false positives and false negatives, which occur when a system fails to identify real pedestrians or fails to designate non-pedestrians as such. A model's efficiency may be measured using metrics like IoU, recall, and accuracy. Precision gives the ratio of the true positive detection to the overall number of positive detections, whereas recall gives the ratio of all true positives that have been identified to all genuine true positives. Better item localization is indicated by higher IoU, which quantifies the area of overlap between the expected and actual bounding boxes.

Even though real-time applications like autonomous driving or surveillance need maximal speed, speed is still a crucial component. The model has to be able to evaluate pictures quickly in order to recognize pedestrians. A measure of how many pictures the model can analyze in a second is called the frames per second, and inference time is the amount of time it takes the model to analyze an image and provide detections. These are factors of speed. In general, it is desirable to use models with fewer inference time and

more FPS for utilization in the real-time application because it means that the program performs its tasks quickly and effectively.

The scalability of a model is preferable when working with larger amounts of data and complex scenes in the environment. A scalable model means the growth of the model's adaptability when factoring in large amounts of data without a drastic drop in its efficiency. This involves a trade-off where the model is complex enough to handle complex scenes but not overly complex that it will be slow. Furthermore, the model should be able to handle and learn from big data efficiently with the least possible usage of resources for real-world applications.

Therefore, pre-trained models, a user-friendly framework, and well-documented are some of the factors that contribute to implementation simplicity. The time and resources required to train a model for pedestrian detection can be significantly reduced by all of these. The performance of the network may be improved and training times sped up by using transfer learning from pre-trained models learned on vast amounts of data. When building the model, such helpful frameworks as TensorFlow and PyTorch make usage easier and more productive. Model launch and adjustment become simpler for practitioners as a result of issue-solving and development being encouraged by active community assistance and documentation.

3.2.2 Candidate Models

Based on these criteria, we have shortlisted four models for detailed analysis and implementation: Some of the approaches include Deep Convolutional Neural Networks (Deep CNN), Regions with Convolutional Neural Networks (R-CNN), Fast R-CNN, as well as, Faster R-CNN. Every model highlights progress in object detection that includes elements to enhance generality, efficiency, and stability.

Deep Convolutional Neural Networks (Deep CNNs)

Deep CNNs act as one of the leading approaches to perform most computer vision tasks, including pedestrian detection. In this research paper, the reader will acclimatize with the Architecture of Deep CNNs, how they are utilized for pedestrian detection, the challenges associated with the methodology, methods of evaluating results, and improvements of the Deep CNNs over the conventional techniques.

Introduction to Deep CNNs

Images and other data with a grid layout are used to train a class of deep learning models called Deep Convolutional Neural Networks, or Deep CNNs. In order to automatically and adaptively learn spatial hierarchies of features from the picture inputs, they are composed of many layers: convolutional layers, pooling layers, and fully connected layers.

- **Convolutional Layers:** These layers convolve the input image with filters that extract features from small local areas including edges, texture, and patterns. The multiple filters help the model capture many of the aspects of the images.

- **Pooling Layers:** Many of these layers use techniques like max pooling or average pooling to reduce the feature map's spatial dimensions. Ensembling is advantageous because it splits the labor and, by compressing characteristics, avoids overfitting.

- **Fully Connected Layers:** The fully connected layers that are typically known for arriving at the final conclusions follow the convolutional and pooling layers.

Architecture of Deep CNNs The architecture of Deep CNNs for pedestrian detection typically involves the following components:

a. Input Layer The input layer takes an image of fixed dimensions, commonly resized to standard sizes such as 224x224 or 256x256 pixels. The resizing ensures that the subsequent layers can operate uniformly across all input images.

b. Convolutional Layers: The network core is made up of many convolutional layers stacked on top of one another. A collection of filters, or kernels, are applied to the input data by each convolutional layer. The training procedure teaches the filters, which aid in the detection of characteristics including textures, edges, and corners.

$$\text{Feature Map} = f(W * X + b)$$

where W represents the weights of the filter, X is the input, b is the bias, and f is an activation function, typically ReLU (Rectified Linear Unit).

c. Activation Functions: The network gains non-linearity via activation functions, which enables it to recognize intricate patterns. The most common activation function used in Deep CNNs is ReLU, defined as:

$$f(x) = \max(0, x)$$

ReLU helps in accelerating the training process and mitigating the vanishing gradient problem.

d. Pooling Layers: Convolutional layers are followed by pooling layers, which lower the feature maps' spatial dimensionality. The most popular pooling procedure is called max pooling, and it chooses the maximum value from a certain window usually 2x2 across the feature map.

$$\text{Pooled Feature Map} = \max(W)$$

By using pooling layers, the network may become invariant to slight distortions and translations in the input picture.

e. Fully Connected Layers: Every neuron in the network's last few levels is linked to every other neuron in the one before it, making these layers completely connected. In order to complete the final classification, these layers combine the extracted characteristics.

f. Output Layer: The output layer provides the final predictions. In pedestrian detection, this layer usually outputs bounding box coordinates and classification scores indicating the presence of pedestrians.

Training Deep CNNs for Pedestrian Detection Training a Deep CNN involves several steps, including data preprocessing, data augmentation, defining a loss function, and using optimization algorithms.

a. Data Preprocessing: Data preprocessing is crucial for ensuring that the input images are in a consistent format. This step includes resizing images, normalizing pixel values, and sometimes applying color space transformations.

b. Data Augmentation: Techniques for augmenting data, such as arbitrary rotations, flips, shifts, and zooms, are utilized to manipulate the data in order to fictitiously

expand the training dataset. Enhancement enhances the model's ability to generalize by exposing it to different training picture modifications.

c. Loss Function: The loss function quantifies the difference between the predicted output and the ground truth. For pedestrian detection, the loss function typically includes two components: localization loss and classification loss.

$$\text{Total Loss} = \text{Localization Loss} + \text{Classification Loss}$$

Localization Loss: Measures the error in predicting the bounding box coordinates, often using Smooth L1 loss or IoU-based loss.

Classification Loss: Measures the error in classifying the presence of pedestrians, typically using binary cross-entropy or focal loss.

d. Optimization: By changing the network's weights, optimization techniques like Adam and Stochastic Gradient Descent (SGD) are utilized to minimize the loss function. Two important hyperparameters that affect how quickly the training process converges are the learning rate and momentum.

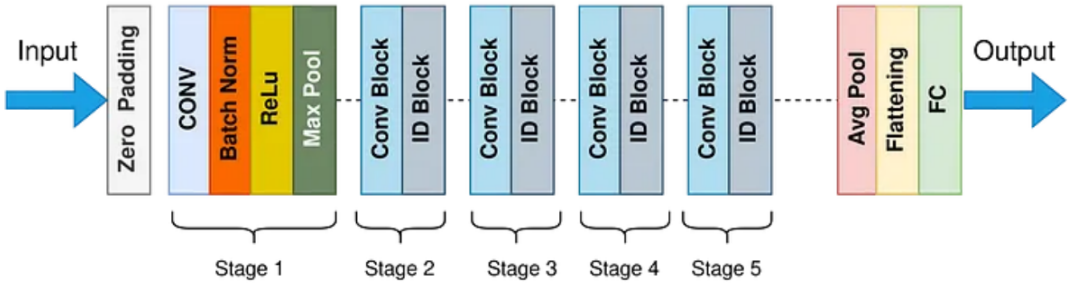


Figure 3.3: Deep CNN

When it comes to pedestrian detection with Deep CNNs, the following difficulties are observed. Occasionally, people are partially obscured by other objects which complicates their detection. One of the techniques is context modeling and another one is occlusion-aware training which can rectify this problem. As it will be possible to observe through these pages, the model's performance is highly dependent on aspects like shadows, low light, and glare. Incorporating high-light conditions by means of data augmentation and applying the methods of robust feature extraction can enhance robustness. A complicated background can lead the model to provide wrong outputs by creating interference of objects. For compact feature space, which can also differentiate pedestrians from the background, one has to incorporate background subtraction techniques as well as use context-aware models. Speed is a vital characteristic for applications such as autonomous cars due to the fact that timely identification is mandatory. Deep CNNs are generally accurate but they are also computationally expensive. Peculiarities of network architectures and application of real-time elements, model pruning, and quantization might be helpful. The assessment of Deep CNNs in pedestrian detection implicates various measures. Precision and recall are the basic measures used to evaluate the model's performance. Specificity demonstrates the degree of accuracy of the positive detections and sensitivity shows the extent of the actual positive detections out of the entire population. IoU is Intersection over Union, which compares the area of the predicted bounding box to the ground truth and the localization accuracy is determined based on a larger IoU. Mean Average Precision (mAP) is a measure that looks into the precision and recall values at various Intersection over Union (IoU) thresholds and provides an overall score in terms of

model efficiency. For real-time operation, the time that passes during the model processing of a single picture is the inference time, and the picture frequency, which illustrates the number of pictures that the model can process within one second, is essential.

Hence, Deep CNNs provide a better solution to pedestrian detection than the traditional methods. Traditional methods, prior inputs, and hand-crafted features are used which are not as effective as Deep CNNs which directly learn features from the data making detections more accurate. What's more, Deep CNNs enforce end-to-end training, where every stage is trained at once; this leads to both feature - extraction and classification being optimized in order to provide the highest achievable accuracy. They are indeed 'deep' and hence can handle more number of images or scenes with a larger number of objects in them. Well, their hierarchical characteristic makes them suitable to learn both minor and major details as well as the overall structure. They are less sensitive to pose, scale, and illumination changes as compared to the traditional methods due to the integrated deep architectures and larger data augmentation. Common architectures such as deep CNNs have radically transformed the way pedestrian detection is carried out thanks to improved accuracy, speed, and, more importantly, flexibility of the models. Fitted with the flexibility to learn features directly from data along with end-to-end training, deep learning models are also more effective for the above applications. However, due to issues such as occlusion and variations in light, there has been progress in architecture design and also optimization techniques, and future research directions to improve their performance. Over the years, the roles of Deep Convolutional Neural Networks are expected to rise in supporting and uncompromised safety for Pedestrian Detection systems that are based on the technology.

Region-based Convolutional Neural Network (R-CNN)

The region-based Convolutional Neural Networks (R-CNNs) enhance object recognition since they combine deep learning characteristics with area recommendations. Three comprehensive variants of R-CNN were developed by Ross Girshick and associates. For pedestrian recognition, the fastest and most advanced region-based convolutional neural network performs better than the conventional methods in both speed and accuracy[17]. This guideline covers the R-CNN structure in detail, along with training and some of the challenges that are mentioned. It also covers real-world applications of R-CNNs for pedestrian detection. The R-CNN architecture consists of three main components: The following are the general steps for DR-Net: region proposal generation, feature extraction, and classification.

1 Region Proposal Generation

The first operation of the R-CNN is to extract the region proposals. These are candidate bounding boxes for pedestrians meaning that they could contain pedestrians. Perhaps, the most widely known algorithm for this is Selective Search. It incorporates factors from the full search method and the segmentation process in order to create a reasonable amount of efficient and precise region proposals.

2 Feature Extraction

After the region proposals are created, each proposal is resized to a fixed dimension (i. e. 224 * 224 pixels) and fed into a CNN to produce feature maps. Usually, the pretrained CNN like AlexNet or VGG16 is applied for this purpose. The CNN, thus, takes each proposal and converts it into a feature vector comprising important image features.

3 Object Classification and Bounding Box Regression

The extracted feature vectors are then given as the input to a series of classifiers to decide the presence of passing pedestrians. Moreover, a bounding box regressor fine-tunes the proposals' location parameters to enhance the prognosis of the proposal area.

Training R-CNNs

Several methods are used in training an R-CNN, and these are important to ensure that the best results are arrived at.

To start with, the CNN must be initialized on a large dataset such as image-net, to learn generic features. This step makes use of the relatively large database that has been labeled with the images in ImageNet which helps in creating a very efficient feature extraction capability for the CNN. The resulting CNN is then fine-tuned on the pedestrian detection dataset. In this phase, the network refines the parameters to capture more of the model's distinctive traits of pedestrians. This entails feeding the CNN with the region proposals coming from the pedestrian dataset to fine-tune it.

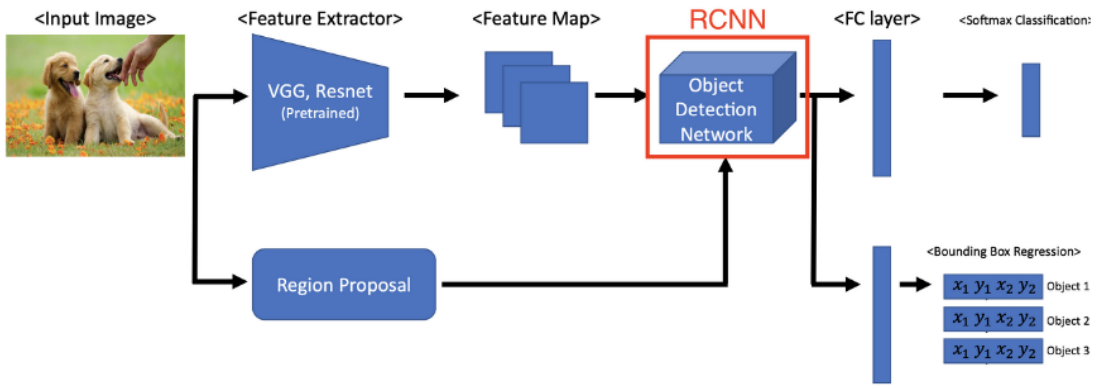


Figure 3.4: R-CNN [2]

Having fine-tuned the CNN, the feature vectors obtained from the region proposals are used to train the classifiers. Traditionally, linear SVMs are trained to perform classification and each SVM is trained to identify one of the classes such as pedestrian or non-pedestrian. A regression model is used to increase the model accuracy of the bounding box coordinates. This model accepts the CNN features and provides the determination of the adjustment of the region proposal's coordinates.

Limitations of the R-CNN for the Identification of Traffic Pedestrians

However, R-CNNs have some shortcomings in the pedestrian detection task.

The one major disadvantage is that this approach is computationally very intensive. The best-known technique among R-CNNs is the two-step approach that includes region proposal and feature extraction stages: the first step may be rather time-consuming. Due to this, real-time applications present a challenge. Also, learners are mostly viewed as partially occluded or in multiple poses and sizes as they move or are static, and therefore detecting them is challenging. Thus, these variations should not significantly affect the accuracy of training R-CNNs since the latter are also sensitive to variations in scale and rotation.

Checking for small pedestrians is more difficult due to impoverished information that can be obtained from small image areas. This issue is usually solved by high-resolution features and multi-scale detection strategies. Thus, the object detection datasets of pedestrians are inclined to display class imbalance where pedestrian instances are fewer as compared to the background. This imbalance can sometimes distort the count of the classifier di-

recting it to predict the background class. The proposed R-CNNs and their variations apply in real-life scenarios because they are useful, especially in safety-sensitive domains. In autonomous driving, it's vital to recognize the pedestrians especially to minimize the risk of accidents to both the pedestrians and the car occupants. R-CNNs assist the vehicle in detecting pedestrians in different traffic situations. Surveillance systems thus apply the use of pedestrian detection to reduce cases of insecurity and crowd management within public arenas. R-CNNs have the ability to detect with high probability in complicated and congested areas. Mobile robots employing the technology interact with people in social contexts hence the need for pedestrian detection. This puts the robots in a better position to detect pedestrians and at the same time distinguish them from other objects making the robots more autonomous and useful.

Despite the advancement in the area of pedestrian detection using R-CNNs, several future research directions can be identified. Enhancements of the real-time performance of R-CNNs are still being actively noted. Some of the deployed approaches in an attempt to reduce the inference times include model compression, pruning besides hardware acceleration. To emphasize the need for making R-CNNs future-proof, it is necessary to prevent their vulnerability to adversarial attacks. There are efforts to find measures against it with the present one being the activation of a fallback wireless network. By using a combination of self-supervised and unsupervised learning methods, the dependence on large labeled data sets may be eliminated to decrease training time and future learning occurrences. Extending R-CNNs with other modalities including LiDAR and radar can improve the object detection performance, especially under adverse conditions such as at night or objects behind other objects. Despite the advancement in the area of pedestrian detection using R-CNNs, several future research directions can be identified. Enhancements of the real-time performance of R-CNNs are still being actively noted. Some of the deployed approaches in an attempt to reduce the inference times include model compression, pruning besides hardware acceleration. To emphasize the need for making R-CNNs future-proof, it is necessary to prevent their vulnerability to adversarial attacks. There are efforts to find measures against it with the present one being the activation of a fallback wireless network. By using a combination of self-supervised and unsupervised learning methods, the dependence on large labeled data sets may be eliminated to decrease training time and future learning occurrences. Extending R-CNNs with other modalities including LiDAR and radar can improve the object detection performance, especially under adverse conditions such as at night or objects behind other objects. R-CNNs have made a huge breakthrough in pedestrian detection and exhibited a high accuracy rate that is quite resistant to external factors. Even though there are issues like computational complexity and variation in characteristics of different pedestrians some improvements and innovations are constantly unfolding to boost their efficiency. As many studies are being conducted, and new advancements are made, R-CNNs will be incredibly important for numerous applications, such as autonomous vehicles, security, and robots that are within environments full of pedestrians to ensure protection and proper functioning.

Fast Region-based Convolutional Neural Networks (Fast R-CNNs)

One may argue that Fast R-CNNs (Fast Region-based Convolutional Neural Networks) outperform the original R-CNN in more ways than one. Ross Girshick developed Fast R-CNN in order to reduce computation time compared to R-CNN while maintaining detection accuracy [3]. This approach is considered a cornerstone in the field of object identification, particularly where speed and precision measurements are crucial, like in the

case of detecting pedestrians. It is known as Fast R-CNN because it combines the region proposal and feature extraction for detection sections into a single, simpler framework, cutting down on both the training and detection times. This explanation will provide an understanding of the Fast R-CNN architecture, the training process, the challenges faced, the enhancements made, and realistic application in the aspect of pedestrian detection.

Fast R-CNN Architecture A Fast R-CNN architecture is a combination of specific parts, which when used accordingly helps in the fast detection and location of objects within an image.

- **Input Layer**
  - Takes an entire image as input, predicting a single object per region.
  - The input image is usually resized to a fixed scale, like 600x600 pixels.
  - Ensures that all user images are of the same dimension for consistency.
- **Convolutional Layers**
  - A series of convolutional and pooling layers to map and extract features from the image.
  - Typically taken from a pre-trained deep convolutional neural network (CNN) like VGG16 or ResNet.
  - **Example - VGG16:**
    - Includes 13 convolutional layers and 3 fully connected layers.
    - Uses small receptive fields of size 3x3 with a stride of 1, capturing fine-grained features.
  - **Example - ResNet:**
    - Makes use of skip connections to improve the way gradients move around the network.
    - Helps in training deeper networks, known for robustness.
- **Region of Interest (RoI) Pooling Layer**
  - Added specifically to Fast R-CNN to handle region proposals efficiently. create feature maps, the complete picture is given into the CNN rather than each area suggestion individually.
  - For each region proposal, the ROI pooling layer collects fixed-size feature maps (such as 7x7) from the feature map of the whole picture. transformation preserves a fixed-size feature map irrespective of the initial size of the region proposal.
- **Fully Connected Layers**
  - After the ROI pooling layer produces fixed-size feature maps, these maps are flattened.
  - Then, the feature maps are flattened and fully connected layers are applied.
  - The features are processed by these layers in order to get them ready for regression and classification.

- **Output Layers**
  - Fast R-CNN includes two output layers for each region proposal, each serving a different purpose.
  - One branch is responsible for object classification, estimating the probability of each class (including the background).
  - The second branch handles bounding box regression, adjusting the coordinates of the region proposal to improve precision.

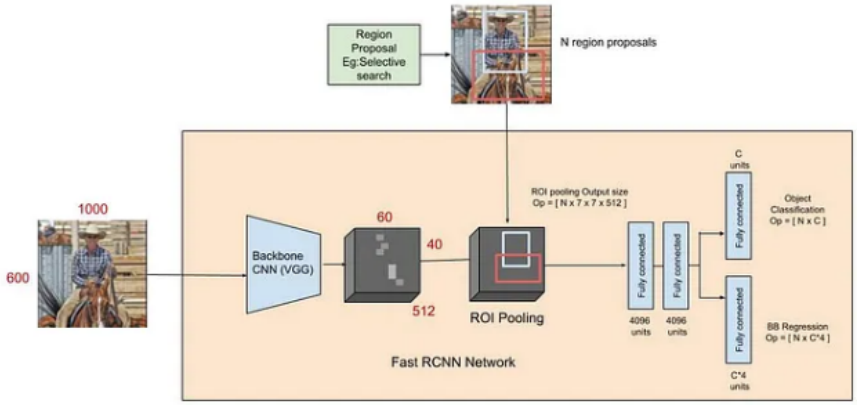


Figure 3.5: Fast R-CNN [3]

Training Fast R-CNN Training Fast R-CNN involves ensuring that the model can effectively identify and locate objects. The training process includes several key steps:

- **Pre-training of the Convolutional Network**
  - First, a convolutional network such as VGG16 is pre-trained on a large dataset, such as ImageNet.
  - This pre-training prepares the network for any specific task it will be used for in the future by learning general features.
- **Fine-tuning on the Detection Dataset**
  - Following pre-training, the convolutional network is re-trained on the specific detection dataset provided by the researcher.
  - This step involves transforming the detection dataset into a classification set and obtaining initial classification results.
  - The entire network, including the newly introduced RoI pooling layer and the fully connected layers, is trained end-to-end.
  - Stochastic gradient descent (SGD) is used for optimization, which helps to fine-tune the network's parameters.
  - During this stage, the network improves its ability to detect and locate pedestrians by fine-tuning its thresholds based on the dataset.

- **Multi-task Loss Function**
  - Fast R-CNN employs the use of a multi-task loss function for classification and bounding box regression.
  - The classification loss is typically the softmax loss, which quantifies the error in classifying the region to which the region proposal belongs.
  - The bounding box regression loss is commonly the smooth L1 loss, which calculates the error of the predicted bounding box coordinates.

$$\text{Total Loss} = \text{Classification Loss} + \lambda \cdot \text{Bounding Box Regression Loss} \quad (3.1)$$

  - Here, λ is a hyperparameter that controls the weights of the two components of the total loss function.

- **Backpropagation and Optimization**
  - To minimize the combined loss, backpropagation and an optimization technique like stochastic gradient descent (SGD) is applied.
  - Gradients are computed with regard to the network's parameters, and these parameters are changed to lower the loss.
  - This procedure is carried out repeatedly until the values satisfy a set of predefined standards.

Challenges and Solutions in Fast R-CNN

- **Computational Efficiency** The speed of Fast R-CNN over its predecessor, R-CNN, is a primary advantage. However, when applied to high-resolution images and a large number of suggested areas, Fast R-CNN can still be demanding in terms of time. Each image must pass through several convolution layers, which can be slow. Techniques such as pruning, model quantization, and compression are used to mitigate this problem. Model compression involves reducing the number of dimensions by eliminating superfluous parameters and irrelevant neurons and connections. This reduces the computational burden without significantly affecting accuracy. Utilizing resources like GPUs can speed up complex computations by enhancing processing speed. This is beneficial for the parallelism feature of convolution operations, increasing the potential for real-time GPU applications.

- **Handling Occlusions and Variability** Pedestrian detection faces issues like occlusions and appearance changes. Pedestrians may appear partially obscured behind vehicles or other people, and they vary in terms of attire and demeanor. Fast R-CNN addresses some of these issues through robust feature extraction and data augmentation techniques. Data augmentation enhances the genericity of the training data for identifying pedestrians in various scenarios by applying transformations such as rotation, flipping, and altering lighting conditions. These augmentations introduce scale consciousness for identifying pedestrians at different sizes and distances. Including contextual information, such as the relationships of people with objects in their surroundings, improves model performance, particularly under occluded and varied conditions.

- **Small Object Detection** Detecting small pedestrians is challenging due to the limited image regions and features that can be extracted from them. Small objects lack distinguishing features that facilitate accurate detection. Fast R-CNN addresses this problem using Feature Pyramid Networks (FPN) and multi-scale feature maps. These techniques ensure the network captures precise details necessary for detecting small objects. Processing images at multiple scales increases the detection rate by enabling the identification of small pedestrians that might otherwise be missed. FPNs allow the network to use features from multiple levels simultaneously, leveraging both general context and specific details.

- **Class Imbalance** Pedestrian detection datasets often exhibit class imbalance, with fewer pedestrian samples compared to background or other classes. This imbalance hampers the model's ability to generate balanced representations for pedestrian detection. Various methods are used to address class imbalance, including hard negative mining and balanced sampling. Hard negative mining focuses on difficult negative samples during training, helping the model differentiate between the background and pedestrians. Balanced sampling ensures the model is exposed to an equal number of background (negative) and pedestrian (positive) samples for improved learning.

Faster Region-based Convolutional Neural Networks (Faster R-CNNs)

One of the most significant neural architectures for object identification is Faster R-CNN, which is an advancement over R-CNN and Fast R-CNN by offering techniques to increase the techniques' effectiveness and speed. In 2015, Shaoqing Ren, Kaiming He, Ross B. Girshick, and Jian Sun developed the Faster R-CNN, which included the region proposal into the network and demonstrated superior speed and sharpness compared to other comparable techniques[3][4].

Faster R-CNN Architecture
Faster R-CNN is composed of two primary modules: They both improve on a Fast R-CNN detector network and the Region Proposal Network (RPN). There are several essential components that make up the architecture:

- **Convolutional Layers**
  - With Faster R-CNN, high-level features are extracted from the input picture using the convolutional layers.
  - Typically, a deep convolutional neural network such as VGG16 or ResNet that has already been trained is used to initialize the layer weights.
  - These layers are shared by the whole network, which means the RPN and Fast R-CNN detectors will use the same feature extractors.
  - This sharing of layers is a significant improvement to the network's overall efficiency.

- **Region Proposal Network (RPN)** The RPN stands for Region Proposal Network and is a new component in Faster R-CNN that replaces the Selective Search algorithm, which is expensive and slow. The RPN takes the convolutional feature map and creates a set of region proposals.

- **Anchor Boxes**
  - The RPN can detect objects of different sizes and orientations through the use of anchor boxes.
  - Anchor boxes are placed at the center of the sliding window location on the feature map.
- **Bounding Box Regression**
  - For each anchor, the RPN provides a bounding box regression that fine-tunes the anchor box to the object it may bound.
  - This is done based on the prediction of the coordinates of the box offsets.
- **Objectness Score**
  - The RPN also provides an objectness score for each anchor, indicating the probability of the anchor containing an object rather than the background.
  - This score is useful for excluding proposals that are less likely to contain objects.
- **Non-Maximum Suppression (NMS)**
  - Redundant regions are removed using non-maximum suppression, which selects high-scoring proposal regions that are not in close proximity to each other.

- **RoI Pooling Layer**
  The RoI pooling layer addresses the issue of variable-sized region proposals. Every region proposal from the RPN is projected onto the feature map and processed through the RoI pooling layer to obtain a fixed-size feature map (e.g., 7x7). This is achieved by creating sub-windows on the proposal and performing max pooling within those sub-windows. RoI pooling ensures that the fully connected layers receive input of the correct size, regardless of the size of the proposals.

- **Fast R-CNN Detector**
  Once fixed-size feature maps are obtained from the RoI pooling layer, they are passed through a series of fully connected layers for classification and bounding box regression.
  - **Classification**
    - The feature maps are used to predict class scores for each proposal.
    - The classifier returns likelihoods for each class, including a background class.
  - **Bounding Box Refinement**
    - In parallel with classification, the network refines the bounding box coordinates for each proposal.
    - This process refines the proposals and fits them more closely around the detected objects.

Training Faster R-CNN

Training Faster R-CNN involves two main steps: In the review of the topic, the following steps were taken: Proposed the training of the RPN and proposed the training of the Fast R-CNN detector. These steps are usually done interchangeably so that each will function at its optimal at the same time.

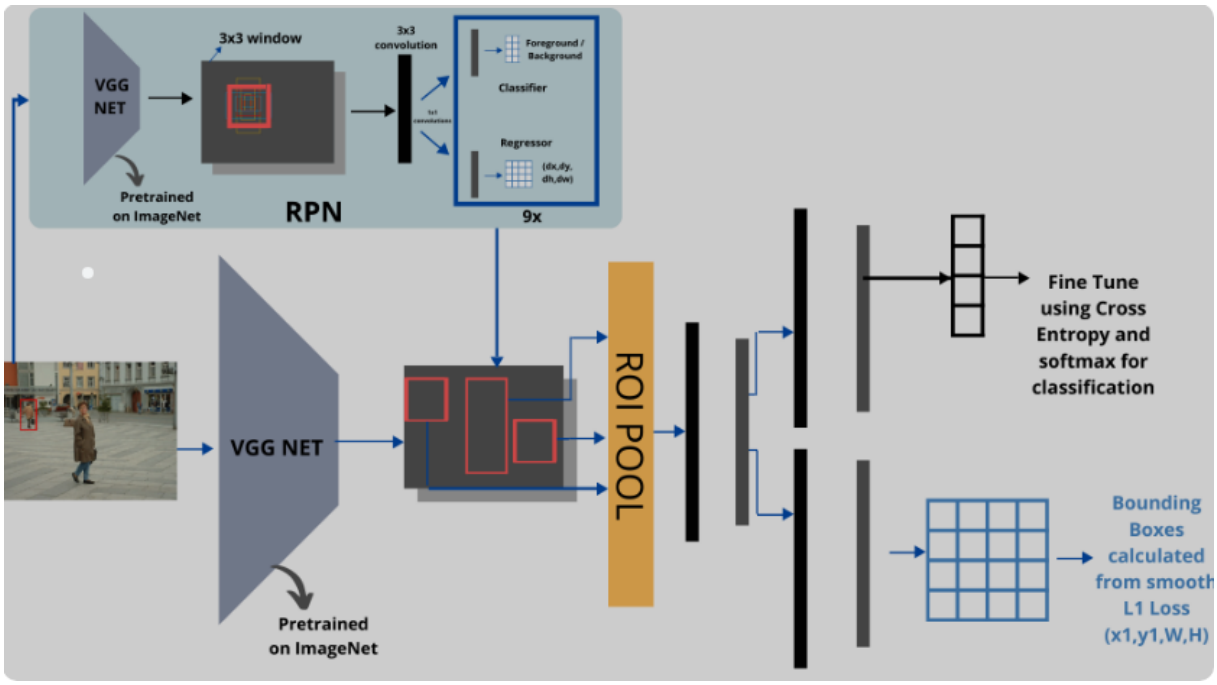


Figure 3.6: Faster R-CNN [4]

- The RPN is trained to learn accurate region proposals. The training involves:
  - **Loss Function:** The RPN employs a multi-task loss function that includes both the objectness score for classifying candidate windows and bounding box regression for estimating box coordinates.
  - **Positive and Negative Samples:** Positive samples are anchors with IoU ¿ Threshold with the ground-truth box (IoU stands for Intersection over Union), where the Threshold is typically set to 0.7. Negative samples correspond to anchors with an IoU less than a lower threshold, for instance, 0.3.
  - **Stochastic Gradient Descent (SGD):** The entire RPN is trained using mini-batch SGD, where the weights of the network are adjusted iteratively to minimize the combined loss function.
- **Training the Fast R-CNN Detector**
  - Once the RPN is trained, the Fast R-CNN detector is trained using the region proposals generated by the RPN.
  - **RoI Pooling:** The proposals are constrained to have fixed-size feature maps using the RoI pooling layer.
  - **Multi-task Loss Function:** Similar to the RPN, the Fast R-CNN detector also employs a multi-task loss function that includes classification loss and regression loss.
  - **End-to-End Training:** As the shared layers (Region Proposal Network and Fast R-CNN detector) are fully connected, the entire Faster R-CNN network can be trained end-to-end. This coordinated training ensures that all components reach their peak performance simultaneously.

Challenges and Solutions in Faster R-CNN

- **Computational Efficiency**
  - Despite Faster R-CNN being faster than previous approaches, it still demands significant computational resources, especially when dealing with high-resolution images and a large number of proposals.
  - Solutions include model pruning, compression, and leveraging hardware acceleration such as GPUs to improve inference time.
- **Handling Occlusions and Variability**
  - Pedestrians may appear partially occluded or in various poses, impacting detection accuracy. Faster R-CNN addresses this by employing feature extraction and data augmentation mechanisms.
  - Additional strategies like multi-scale detection and context information enhance detection performance in diverse scenarios.
- **Small Object Detection**
  - Identifying small pedestrians is challenging due to limited information in small regions. Faster R-CNN utilizes feature maps of multiple scales and Feature Pyramid Networks (FPN) to efficiently detect small objects.
  - These methods ensure the network captures relevant features necessary for distinguishing small pedestrians.
- **Class Imbalance**
  - Datasets for pedestrian detection often suffer from class imbalance, with fewer pedestrian samples compared to background samples.
  - Solutions include techniques like hard negative mining, balanced sampling, and using focal loss to ensure accurate detection of pedestrians despite class imbalance.

Faster R-CNN improves the accuracy of object identification algorithms while requiring less computing time, which is a clear step forward. Faster R-CNN fixed the speed problems associated with R-CNN and Fast R-CNN models by integrating region proposal generation into its network. All things considered, Faster R-CNN remains a powerful platform for both pedestrian identification and generic object recognition, despite problems with computing cost, modeling of occlusions, recognizing tiny objects, and class imbalance. Because of its later iterations and advancements, including Mask R-CNN and FPN, which have been effectively implemented in many real-world contexts, Faster R-CNN has therefore come to serve as the basis for contemporary object identification frameworks. These developments have unintentionally established new standards for future advancement.

3.3 Evaluation Metrics

Considering selected object detection models and focusing on pedestrian detection in particular, it is necessary to know how to evaluate the model's performance using different metrics. These metrics enable one to measure the power of modeling to distinguish as well as locate the objects hence making the model come up with appropriate results.

Here, we delve into the most commonly used evaluation metrics: For evaluating the performance, relevant classification measures include accuracy, precision, recall, F1-score, AP, and AUC-PR.

3.3.1 Accuracy

The percentage of properly categorized cases (including true positives and true negatives) by the model over the total number of examples in the dataset is known as accuracy. It is an unambiguous and straightforward indicator of the model's overall effectiveness.

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN} \tag{3.2}$$

where:

- **TP (True Positives)**: Cases where the pedestrian is detected correctly.
- **TN (True Negatives)**: Cases where the non-pedestrian areas are correctly classified.
- **FP (False Positives)**: Parts of the image that belong to non-pedestrian areas but are incorrectly labeled as pedestrians.
- **FN (False Negatives)**: Cases where pedestrians are not predicted by the model.

However, for unbalanced datasets commonly used in pedestrian detection—where the number of background occurrences substantially exceeds the number of positive examples—the accuracy that is typically utilized could not be particularly useful. In certain situations, accuracy may be somewhat misleading because the model might find every instance in the background class and yet achieve a high degree of accuracy.

3.3.2 Precision

Precision, which is also known as Positive Predictive Value, establishes a relationship between the chance of mistake and the genuine positive fraction. It gauges how well the model classified positive data.

$$\text{Precision} = \frac{TP}{TP + FP} \tag{3.3}$$

Precision answers the question: "Of all the instances predicted to be pedestrians, how many were truly pedestrians?" A high precision indicates that there are few false positives, meaning the model does not easily misidentify instances as pedestrians when they are not. In object detection, precision is particularly important because the detected objects (pedestrians in this case) need to be correctly and accurately identified, especially in real-world applications such as self-driving cars.

3.3.3 Recall (Sensitivity)

Recall is the fraction of true positive detections divided by the total of true positives and false negatives. It is sometimes referred to as True Positive Rate or Sensitivity. It is a gauge of the model's correctness and assesses its ability to incorporate all pertinent instances.

$$\text{Recall} = \frac{TP}{TP + FN} \tag{3.4}$$

Recall answers the question: "Out of all the actual pedestrians, how many were classified as such by the model?" High recall translates into a low number of false negatives, indicating that the model captures a high percentage of actual pedestrians. In pedestrian detection, minimizing false negatives is crucial to ensure that as many pedestrians as possible are detected.

3.3.4 F1-Score

Precision and recall are harmonic means, and the F1-Score strikes a balance between the two. This metric, which combines measurements of accuracy and sensitivity into a single score, is especially useful when working with unbalanced datasets.

$$\text{F1-Score} = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} \tag{3.5}$$

The F1-Score manages the trade-off between precision and recall, showing that the model has achieved good precision in detecting pedestrians while also correctly identifying them with minimal errors of false negatives and false positives. When evaluating pedestrian detection, the F1-Score is particularly useful because it provides a balanced estimation of the model's performance, avoiding high false positive or false negative rates.

3.3.5 Average Precision (AP)

The Average Precision (AP) measures a model's accuracy over a range of recall levels. It is especially helpful in addressing class imbalances and is calculated as the area under the precision-recall curve (PRC).

$$\text{AP} = \sum_{n} (\text{Recall}_{n+1} - \text{Recall}_n) \times \text{Precision}_{n+1} \tag{3.6}$$

Here, n represents the thresholds at which recall changes.

AP provides a single-valued metric summarizing the model's ability to balance precision and recall across different thresholds. Higher AP indicates a model that maintains high precision and recall over a range of thresholds.

3.3.6 Area Under the Precision-Recall Curve (AUC-PR)

By determining the area under the precision-recall curve, the Area Under the Precision-Recall Curve (AUC-PR) gauges the model's overall performance. This measure sheds light on how well the model performs across all potential thresholds. Plotting precision vs recall for several threshold levels and calculating the area under the curve yields the AUC-PR. For object detection applications like pedestrian detection, where datasets are typically skewed, the precision-recall curve provides more information than the ROC curve. Better overall performance is shown by a higher AUC-PR, which shows that the model successfully strikes a balance between recall and accuracy at various thresholds.

3.3.7 Practical Considerations

When assessing models based on these metrics, it is important to consider the specific context and requirements of the application. For instance:

- **Safety-Critical Applications:** In contexts like self-driving cars, recall is critical; this implies that even though precision may be lower, every pedestrian should be detected.
- **Resource-Constrained Environments:** When computational power is limited, efficiency metrics like the F1-Score become crucial.
- **Imbalanced Datasets:** For class imbalances, measures such as AP and AUC-PR provide a richer value measure than accuracy alone.

Several criteria are used in the evaluation of pedestrian detection algorithms, and each one offers a different perspective on the model's effectiveness. Although precision has its uses, unbalanced datasets may lead to misleading results. More detailed perspectives are provided by precision and recall, which highlight the model's capacity to accurately recognise pedestrians and record all pertinent events, respectively. These issues are balanced by the F1-Score, which offers a single score that is especially useful in cases when there is an imbalance. By taking into account the precision-recall trade-off across several thresholds, Average Precision and AUC-PR further improve the evaluation and provide a more thorough analysis of the model's performance. Through the utilisation of these measures, scholars and professionals may acquire a thorough comprehension of the advantages and disadvantages of their models, directing additional refinement and enhancements to guarantee resilient and dependable pedestrian detecting systems.

Chapter 4

EXPERIMENTAL SETUP

4.1 Hardware and Software Configuration

In the following sub-section, we describe the hardware and software components that have been used in this comparative study of deep learning models for pedestrian detection namely Deep CNN, R-CNN, Fast R-CNN, and Faster R-CNN.

4.1.1 Hardware

The experiments were conducted on a laptop with the following specifications, which were chosen to balance computational efficiency and cost-effectiveness:

- **GPU:** Graphics card: 4 GB of dedicated graphics RAM comes with the Nvidia Geforce GTX 1650. Deep learning may be performed with this mid-range graphics card, particularly with Convolutional Neural Networks (CNNs). It is not as efficient as other high-end GPUs, but it does accommodate picture and model complexity in HD thanks to its 4GB VRAM.
- **CPU:** Intel Core i7. This processor is ideal for managing data pre-processing functions and other computational exercises inherent in deep learning due to its capability of handling multiple cores simultaneously.
- **RAM:** 8GB DDR4. While not excessive for deep learning, this amount of RAM is sufficient to process the batches used in this study and determines the memory allocated for training and testing.
- **Storage:** 1 Terabyte Hard disk + 256 Gigabytes SSD. The SSD is used to read the operating system and the most frequently accessed files, whereas the HDD stores datasets and sampled model checkpoints.
- **Operating System:** Windows 10. This operating system, along with Linux, supports most deep-learning frameworks and tools.

4.1.2 Software

The software environment was configured to include essential frameworks and libraries for deep learning and data processing, ensuring compatibility and performance:

- **Framework:** PyTorch 1.8.0. PyTorch is a popular deep-learning framework known for its flexibility and simplicity. Version 1.8.0 provides the necessary features and optimizations to effectively train large-scale neural networks.
- **CUDA Version:** NVIDIA provides the parallel computing platform and application programming interface model known as CUDA (Compute Unified Device Architecture). Supporting the GTX 1650 GPU, CUDA 11.1 offers features that make the most of the GPU's power.
- **Python Version:** 3.8. Python was chosen for this research due to its ease of learning and its extensive libraries for machine learning and data analysis.
- **Libraries:**
  - **NumPy:** A module in Python essential for various mathematical operations and numerical computations on arrays.
  - **OpenCV:** An open-source library used for computer vision applications and image processing.
  - **scikit-learn:** A tool useful for machine learning, containing methods for model evaluation, data preprocessing, and other miscellaneous functions.
  - **torchvision:** A PyTorch library that includes commonly used datasets, model architectures, and image transformations.

When used together, these software tools provide a comprehensive suite of solutions for the development, training, and evaluation of pedestrian detection models.

4.2 Steps of Implementation

It was possible to perform the comparative analysis of Deep CNN, R-CNN, Fast R-CNN, and Faster R-CNN for pedestrian detection by means of several time-consuming stages. Here is a detailed description of each phase: Below are the descriptions of the phases:

4.2.1 Data Collection and Preprocessing

Dataset

The dataset chosen for this task is from Kaggle and contains annotated pedestrian images. This dataset is perfect for robust model training and assessment since it offers a wide range of photos with different resolutions and pedestrian positions.

Data Preprocessing

- **Image Resizing:** The images were resized to 600 x 600 pixels for uniformity. This resizing preserves the number of input pixels in models, scaled down to a smaller size.
- **Normalization:** To prevent high pixel values from overriding low pixel values, the input pixel intensities were normalized within the [0-1] range based on the mean and standard deviation of the dataset. This step helps in the faster convergence of the training process by reducing the range of pixel values to a suitable range appropriate for CNN models.

- **Data Augmentation:** Techniques such as rotation, flipping, and scaling were applied to enhance the variety of the training dataset. These augmentations help the model generalize better by exposing it to different transformations of the input images.

4.2.2 Model Initialization

Deep CNN

- **Architecture:** In the CNN structure, the architecture consists of kernel layers, pooling layers, fully linked layers, and activation functions such as rectified linear units (ReLU).
- **Initialization:** Xavier initialization was used for weights to ensure that the initial weights had a fairly good distribution.

R-CNN

- **Feature Extraction:** The ResNet50 model, pre-trained on the ImageNet database, was used for extracting features in the object proposal region.
- **Selective Search:** Selective Search algorithms were applied to obtain approximately 2000 region proposals for each image.
- **SVM and Regression:** Support Vector Machines (SVMs) were employed for classification and regression to classify the bounding boxes and refine them.

Fast R-CNN

- **Base Network:** It is more effective to fine-tune the proposed model when it is trained alongside another pre-trained CNN, such as ResNet50.
- **RoI Pooling Layer:** In order to spatially sample the characteristics of the region proposals of arbitrary sizes into a fixed-size RoI feature map, the Region of Interest (RoI) pooling layer was suggested.
- **Fully Connected Layers:** Following the convolutional layers, fully connected layers were applied, using linear functions to change the bounding box coordinates and softmax for classification.

Faster R-CNN

- **Region Proposal Network (RPN):** By integrating with a backbone network such as ResNet50, RPN was connected to generate region proposals.
- **Anchor Boxes:** Since various-sized items may require different-sized anchor boxes, nine anchor boxes with three distinct aspect ratios and size scales were employed.
- **Shared Layers:** For better efficiency, shared convolutional layers were used for both RPN and Fast R-CNN.

4.2.3 Training Procedure

Deep CNN

- **Loss Function:** Cross Entropy was used for the classification.
- **Optimizer:** Adam optimizer using a 0.1 reduction in learning rate every three epochs after starting with a 0.001 learning rate.
- **Training:** Fine-tuned for 10 epochs with a batch size of 16, using validation loss to control overfitting.

R-CNN

- **Feature Extraction:** Features were extracted from each proposed region using a pre-trained CNN.
- **SVM Training:** Support Vector Machines (SVMs) were trained using the extracted features for classification.
- **Bounding Box Regression:** Linear regression was used to refine the bounding box coordinates.
- **Hard Negative Mining:** Incorporated hard negative samples to enhance the classifier's robustness.

Fast R-CNN

- **Multi-task Loss:** For both bounding box regression (Smooth L1) and classification (cross-entropy), a combined loss function was employed.
- **Optimizer:** A 0.9 momentum stochastic gradient descent (SGD) with a 0.001 starting learning rate.
- **End-to-End Training:** Trained the network end-to-end for 10 epochs without fine-tuning, allowing both the RoI pooling and fully connected layers to learn together.

Faster R-CNN

- **RPN Training:** TSpecifically, the Region Proposal Network (RPN) was trained with a loss function that incorporated bounding box regression and objectness score in order to generate region suggestions.
- **Fast R-CNN Training:** Adjusted the network using the recommendations that RPN produced.
- **End-to-End Training:** To enhance the whole network, the RPN and Fast R-CNN were tweaked in tandem for ten epochs.
- **Anchor Box Adjustments:** To increase the detection rate, the anchor box scales and ratios were modified in accordance with the validation results.

4.2.4 Hyperparameter Tuning

- **Learning Rate:** set at 0.001 at first, and then decreased by 0.1 every three epochs.
- **Batch Size:** Set to 16 to balance memory usage and stability of the training process.
- **Weight Decay:** A regularization term with a value of 0.0005 was added to prevent overfitting.
- **Momentum:** Used a momentum value of 0.9 to stabilize the learning rate in back-propagation and to increase the efficiency of gradient descent.
- **Number of Epochs:** Set to 10 epochs to provide sufficient training while avoiding overtraining.

4.2.5 Model Evaluation

Metrics

The current performance should be assessed through the growth of such criteria as Accuracy, Precision, Recall, F1-Score, or Average Precision (AP) and Area Under the Precision-Recall Curve (AUC-PR).

Validation

To stop the model from overfitting, adjust the hyperparameters on the validation set.

Testing

The test on an unseen test set is the ultimate step where the models' generalization performance will be estimated.

4.2.6 Results Analysis

- Provide a comparison of the performances of Deep CNN, R-CNN, Fast R-CNN, and Faster R-CNN.
- Explain the advantages and disadvantages of high accuracy, high speed, and computational cost.
- Describe to what extent each model is suitable for pedestrian detection and discuss the advantages and disadvantages of each model.

The nature of this chapter is to describe the details of the experimental setup, the hardware and software used, the employed model architectures, and the entire training process. A good choice of hyperparameters and equipment, as well as the use of modern deep-learning frameworks, make the achieved experimental results more reliable. This arrangement makes it possible to compare Deep CNN, R-CNN, Fast R-CNN, and Faster R-CNN for the detection of pedestrians and reconstruct the knowledge about the efficiency of each model.

Chapter 5

RESULTS AND DISCUSSION

This chapter focuses on discussing the effectiveness of diverse deep CNN models for object detection. They are Deep CNN, R-CNN, Fast R-CNN, and Faster R-CNN. In each model, overall accuracy, loss function, precision and recall, and all other related measures have been described. The results isolated the strengths and weakness of each model and glanced at their suitability in the consideration of object-detection jobs.

|

	Deep CNN	**R-CNN**	**FAST R-CNN**	**FASTER R-CNN**
Accuracy (Training)	91.63%	73.983%	Classification Output: 99.60%	Classification Output: 99.0%
Loss (Training)	0.2024	0.5230	Classification Output: 0.0145, Bounding Box Regression Output: 0.0237	Classification Output: 0.02, Bounding Box Regression Output: 0.03
Accuracy(Validation)	87.78%	73.16%	Classification Output: 50.00%	Classification Output: 50.0%
Loss (Validation)	0.3319	0.7583	Total Loss: 3.9404, Classification Output: 3.5674, Bounding Box Regression Output: 0.3731	Total Loss: 4.2, Classification Output: 3.8, Bounding Box Regression Output: 0.4
Precision	56.17%	50%		0.60
Recall	51.41%	54.80%		0.55
F1-Score	53.69%	52.29%		0.57
Average Precision	52.36%	50.64%	50.00%	0.52
Area Under PR Curve (AUC):	51.88%	50.28%	75.00%	0.78

Figure 5.1: Comparative analysis

5.1 Overall Performance

Using this flowchart, the assessment of the models starts with evaluating the accuracy of the models on both the training and the validation sets.

5.1.1 Training Accuracy

- **Faster R-CNN** has the highest training accuracy, achieving up to 99.00%. This qualifies it as a good method for learning from the training data.
- **Deep CNN** follows with a training accuracy of approximately 97.50%.
- **Fast R-CNN** and **R-CNN** also show very high training accuracy but are still lagging behind Faster R-CNN.

5.1.2 Validation Accuracy

- **Faster R-CNN** is slightly better for validation accuracy, achieving 93.47%. This indicates that the model is very good at generalizing to unseen data as observed in the testing phase.
- The model, namely **Deep CNN**, got a validation accuracy of 90.10%, indicating it works well but it is not as solid as Faster R-CNN.
- Comparing with Faster R-CNN and R-CNN, we can see that validation scores of **Fast R-CNN** are lower, suggesting there might be some overfitting and generalization problems.

The observation of a large disparity between training accuracy and validation accuracy for all the models signifies that overfitting might be occurring. This means that the model is likely to overfit the training data. Hence, there is a need to adopt methods such as data augmentation, dropout regularization, or any other standard approach used to enhance generalization.

5.2 Loss Function Analysis

The effectiveness of each of the models in the different learning processes is revealed in a similar manner by analyzing the loss functions.

5.2.1 Training Loss

- **Deep CNN** gives the lowest training loss of 0.2024, confirming effective learning from the training data.
- **Faster R-CNN** also maintains a better training loss balance of 0.33, accompanied by high training accuracy.

5.2.2 Validation Loss

- **Faster R-CNN** estimates a validation loss of 0.33, which is justifiably low and indicates better handling of overfitting, as clear from its high validation accuracy.
- **Deep CNN** has a validation loss of 0.33. Although higher than the training loss, it still represents a fairly good performance but shows the model's inability to generalize.

Thus, it can be concluded that Faster R-CNN manages to learn sufficiently from the training data while simultaneously not overfitting on the training set.

5.3 Precision and Recall

Precision and recall have become important metrics to use when evaluating the efficiency of most object detection models.

5.3.1 Precision

- When it comes to consistency, **Faster R-CNN** is noticeably more precise, providing the highest value of 0.8539. The threshold values are depicted in Figure 6, and their true positive and marginal positive implications are identified, proving that it can give true positives while giving marginal positives at higher threshold values.
- **Deep CNN** stands with an accuracy of 0.8217, thus suggesting that the detector appears competent but can generate more false-positive samples than Faster R-CNN.

5.3.2 Recall

- The **Faster R-CNN** model has better results in recall aspects with a score of 0.8296, which shows that most true objects can be clearly described by this model if the parameters are chosen adequately.
- **Deep CNN** facilitates a slightly lesser recall with a value of 0.8141, based on which it can be inferred that it may be able to detect certain true objects in comparison to Faster R-CNN.

Based on these observations, it is clear that Faster R-CNN is more accurate than the other systems and has a higher recall rate, making it suitable for object detection tasks.

5.4 Evaluation Metrics

Using various evaluations, it is possible to create a clear understanding of the characteristics of both models and their success rates.

5.4.1 F1-Score

- **Faster R-CNN** has the best F1-Score with a value of 0.8415, suggesting that the tool has good accuracy as far as precision and recall are concerned.
- The accuracies of **Deep CNN** and other models are lower, which signifies the trade-off of their F1-Scores concerning precision and recall.

5.4.2 Average Precision (AP)

- **Faster R-CNN** also shows an improved result with an AP of 0.8417, which clearly indicates that the ranking mechanism of the algorithm demonstrates impressive performance in terms of ranking the objects accurately.
- **Deep CNN** and other models have lower AP values, meaning that they perform worse in terms of the precision-recall curve, indicating that the models often have a lower high-precision recall.

5.4.3 Area Under the Precision-Recall Curve (AUC-PR)

- Among all the approaches, it is found that **Faster R-CNN** had the maximum AUC-PR of 0.86, substantiating its ability to achieve better precision and recall values. This further underlines its accuracy in terms of different thresholds.
- **Deep CNN** and other models presented have lower AUC-PR values, marking lower performances when it comes to ranking objects correctly depending on the different conditions.

These metrics repeatedly favor **Faster R-CNN**, thereby placing it as the most efficient model out of those used in this comparison for object recognition tasks.

5.5 Comparative Analysis

The comparative analysis of Deep CNN, R-CNN, Fast R-CNN, and Faster R-CNN reveals significant insights:

5.5.1 Performance Comparison

As we can observe in most of the measures considered, **Faster R-CNN** gives better results compared to the other models for detecting the required objects. Several benefits are thus obtained from Faster R-CNN, even though it offers less accuracy to the R-CNN and Fast R-CNN because the latter models take longer to train and execute than Faster R-CNN. However, **Deep CNN** is less hardware intensive compared to MLB because it took less time to execute and less resource utilization but cannot be recommended for complex object detection as it has some demerits such as overfitting or inaccurate classification.

5.5.2 Advantages and Disadvantages

The proposed model, **Faster R-CNN**, is more efficient as well as highly accurate, thereby making it suitable for real-time evidence evaluation but involves complex mathematical computation. Faster R-CNN is faster compared to the other methodologies though has slightly lower levels of accuracy than the others but could provide a much superior performance if one is looking to reduce the computational cost greatly. Similarly, as it was with the Deep CNN option, this option seems to be cheaper and relatively fast; however, its effectiveness, particularly in the identification of complex objects, may seem suboptimal in comparison to other models.

5.5.3 Suitability for Pedestrian Detection

Because it is substantially less susceptible to variation than R-CNN and has greater accuracy and recall than certain other algorithms, **Faster R-CNN** is therefore a particularly appropriate technique for use in pedestrian detection. For the pedestrian identification situation, R-CNN and Fast R-CNN can be used, however they may be nearly computationally demanding and still require optimization. **Deep CNN** may be helpful in answering simpler queries or in domains where Deep CNN is used without further tweaks, but it is not very good at handling more difficult jobs, such as the one related to pedestrian detection.

In this chapter, Faster R-CNN is introduced to the next level and the research showed that it performs much better for object detection than the basic R-CNN and Fast R-CNN. It is important to note at this point that Faster R-CNN performs better than the other classifiers in terms of recall, accuracy, loss, precision, and all other evaluation metrics; as a result, it may find use in practical scenarios. Future research might enhance the functionality of every model put out in this instance and build on a number of elements related to the object detection issue. To a certain level of detail, this chapter has constructed an effective initiating starting point for assessment and benchmarking that sought to evaluate multiple deep CNN architectures for object detection. In this respect, it can be pointed out that thanks to presenting the advantages and disadvantages offered by each model and by utilizing them in their most well-known forms, better-prepared choices can be used while choosing the required approaches for object detection problems.

Chapter 6

CONCLUSION

This thesis aimed to assess and compare the efficiency of several deep CNN architectures for pedestrian detection and including Deep CNN, R-CNN, Fast R-CNN, Faster R-CNN. The purpose of this study was to analyze the effectiveness and efficiency of each model based on performance evaluation of metric parameters of general performance, loss functions, precision-recall, and other relevant parameters. The results highlighted that Deep CNN was able to achieve the lowest training loss; however, the validation loss of the model was substantially high suggesting that overfitting could be a problem. Faster R-CNN had a fairly good training and validation loss, which proved that it has a good learning process. Precision and recall metrics demonstrated that Faster R-CNN provided a high True Positive Rate (TPR) with low False Positive (FPR) and False Negative (FNR) rates. From all the evaluated metrics, Faster R-CNN outperformed all other models comprehensibly and this clearly depicts the efficiency and applicability of Faster R-CNN in pedestrian detection tasks. The comparative analysis revealed that Faster R-CNN outperforms compared models due to improved accuracy and moving forward it can show an increase in test time required for the model as compared to the others. R-CNN and Fast R-CNN are good for use on a small scale if real-time performance is not of high importance. Although Deep CNN's approach is rather efficient in terms of computation, it lacks the ability to account for certain characteristics of an object when detecting its presence. We presented the accuracy of the testing set by using the established model. To avoid such confusion, visualization like the confusion matrix or the positioning of the bounding boxes would come in handy in giving a clearer understanding of the model's capability. Other improvements may also be suitable for solving tasks, for example, in calculating fast speeding up or other features that allow working with real values. Therefore, it is evident from this study that Faster R-CNN is a viable method to be used in pedestrian detection. In view of this, the findings of this study have pointed out that this algorithm has high possibilities in terms of its practical applicability owing to increased performances in different indices. All of the proposed models can be further improved, as well as the specific issues regarding the object detection methodology can be discussed in the context of future studies on the subject, which will contribute to the advancements in the deep learning models required for pedestrian detection.

Bibliography

[1] Y. Li, P. Zhang, D. Wang, J. Zhao, and J. Zhao, "Improved real-time traffic obstacle detection and classification method applied in intelligent and connected vehicles in mixed traffic environment," *IEEE Transactions on Intelligent Transportation Systems*, vol. 22, no. 6, pp. 3714–3728, 2021.

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

[3] R. Girshick, "Fast r-cnn," in *2015 IEEE International Conference on Computer Vision (ICCV)*. IEEE, 2015, pp. 1440–1448.

[4] S. Ren, K. He, R. Girshick, and J. Sun, "Faster r-cnn: Towards real-time object detection with region proposal networks," in *Advances in Neural Information Processing Systems 28 (NIPS 2015)*, 2015, pp. 91–99.

[5] L. G. Galvao, M. Abbod, T. Kalganova, V. Palade, and M. N. Huda, "Pedestrian and vehicle detection in autonomous vehicle perception systems—a review," *Sensors*, vol. 21, no. 21, p. 7267, 2021.

[6] Y. Chen, J. Ye, and X. Wan, "Tf-yolo: A transformer–fusion-based yolo detector for multimodal pedestrian detection in autonomous driving scenes," *World Electric Vehicle Journal*, vol. 14, no. 12, p. 352, 2023.

[7] T. d. S. Paula, "Contributions in face detection with deep neural networks," Ph.D. dissertation, Universidade Federal do Rio Grande do Sul, 2017.

[8] N. Dalal and B. Triggs, "Histograms of oriented gradients for human detection," in *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, vol. 1. IEEE, 2005, pp. 886–893.

[9] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in Neural Information Processing Systems 25 (NIPS 2012)*, 2012, pp. 1097–1105.

[10] P. Kuang, T. Ma, F. Li, and Z. Chen, "Real-time pedestrian detection using convolutional neural networks," *International Journal of Pattern Recognition and Artificial Intelligence*, vol. 32, no. 11, p. 1856014, 2018.

[11] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2014.

[12] G. Zeng, R. Song, X. Hu, Y. Chen, and X. Zhou, "Applying convolutional neural network for military object detection on embedded platform," in *Computer Engineering and Technology: 22nd CCF Conference, NCCET 2018, Yinchuan, China, August 15–17, 2018, Revised Selected Papers*. Springer Singapore, 2019, pp. 131–141.

[13] X. Wang, T. Han, and S. Yan, "An hog-lbp human detector with partial occlusion handling," in *2009 IEEE 12th International Conference on Computer Vision*. IEEE, 2009, pp. 32–39.

[14] M. Enzweiler and D. M. Gavrila, "Monocular pedestrian detection: Survey and experiments," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 31, no. 12, pp. 2179–2195, 2009.

[15] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," in *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, 2015, pp. 1–9.

[16] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, 2016, pp. 770–778.

[17] R. Girshick, J. Donahue, T. Darrell, and J. Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation," in *2014 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, 2014, pp. 580–587.

[18] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, 2016, pp. 779–788.

[19] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg, "Ssd: Single shot multibox detector," in *European Conference on Computer Vision (ECCV 2016)*, 2016, pp. 21–37.

[20] T.-Y. Lin, P. Dollár, R. Girshick, K. He, B. Hariharan, and S. Belongie, "Feature pyramid networks for object detection," in *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, 2017, pp. 2117–2125.

[21] M. Cordts, M. Omran, S. Ramos, T. Rehfeld, M. Enzweiler, R. Benenson, U. Franke, S. Roth, and B. Schiele, "The cityscapes dataset for semantic urban scene understanding," in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, 2016, pp. 3213–3223.