IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received February 28, 2019, accepted April 3, 2019, date of publication April 9, 2019, date of current version April 19, 2019.

Digital Object Identifier 10.1109/ACCESS.2019.2909992

Pedestrian Detection in Automotive Safety: Understanding State-of-the-Art

N. K. RAGESH[1], (Member, IEEE), AND R. RAJESH[2], (Senior Member, IEEE)
[1]Tata Elxsi Ltd., Technopark, Trivandrum 695581, India
[2]Central University of Kerala, Kasaragod 671316, India

Corresponding author: N. K. Ragesh (ragesh.nk@gmail.com)

ABSTRACT Pedestrian detection is one of the important computer vision problems in automotive safety and driver assistance domain. It is a major component of the advanced driver assistance system (ADAS) which help the driver to drive safely. Recent literature shows a number of research activities addressing object detection/tracking in general and pedestrian detection in particular. The solutions proposed by different researchers vary in detection methods, detection scenario, feature descriptors, classification schemes, detection performance, as well as computational complexity. However, the average detection accuracy is not much promising even after many years of research. The fail-safe and real-time human detection from real life road scenes, even in standard resolution, is far from reality. Safety critical systems in the automotive industry have to follow well established stringent safety standards like ISO26262. Since the pedestrian detection system deals with human safety, it also has to follow these standards before integrating to the vehicle electronics. This paper is a study of different techniques used in pedestrian detection specific to the automotive application, along with a description of generic pedestrian detection solution architecture.

INDEX TERMS Road safety, advanced driver assistance systems (ADAS), smart image region of interest (ROI) selection, image feature descriptors, pedestrian detection.

I. INTRODUCTION

Advanced Driver Assistance System (ADAS) [1] is a set of intelligent solutions incorporated into new generation vehicles to support the driver to drive safely to avoid accidents, casualties and loss involved. Pedestrian safety [2] is one of the prime target of advanced driver assistance systems. The information provided by the system can save pedestrian life, but if the system is misbehaving or providing incorrect information, that may cause severe injuries or death of pedestrians and passengers of the vehicle. Because of this safety critical nature, the reliability of the system is at most importance and hence the system shall comply stringent safety standards like ISO26262 [3]–[5].

Pedestrian detection is one of the most important ADAS component, which is a typical object detection problem in digital image processing. The presence of pedestrians can be identified from the images captured through the cameras mounted on the vehicle by applying intelligent image processing techniques. Once the presence of pedestrian is identified, vehicle driver will be informed with appropriate warning mechanism including audio visual presentation as well as mechanical warnings like vibrating the steering wheel or seat.

The processing stages involved in pedestrian detection is same as any other typical object detection system. Simple objects with predefined shape may be detected through simple technique like template matching [6]. In template matching approach, sufficient number of templates of the object-of-interest are prepared in advance and the real image is compared with these templates to identify the presence of the object in the sample image. This approach is suitable for cases where the shape of the object is always predictable and can be uniquely represented with a limited number of templates. A typical case is to find a square or circular shape in an image. But this is not the case with pedestrian detection problem. Pedestrians may be standing or moving in any direction and thus the possibilities of the shape of the pedestrian is unlimited. Different dressing style and color, makes it extremely complex to represent pedestrians with a unique set of templates. We will need to look for more complex techniques to detect complex objects like a pedestrian.

The associate editor coordinating the review of this manuscript and approving it for publication was Michele Nappi.

Complex object detection in image processing can be implemented typically in two ways - a classifier based on predefined features (Feature Classifier [7] approach) or through a classifier with self learned features (Deep Learning approach). The feature classifier approach is the classical approach, which is still active, whereas recently more attention is given to deep learning based object detection and classification. Both these approaches are going in parallel, with some attempts to combine the good from these two.

Both these approaches are based on training. That is, the pedestrian detection system will first be trained with a large number of known pedestrian and non-pedestrian objects and representations/models of generic pedestrian and non-pedestrian objects are derived from these samples. A classifier then uses these generic models to identify weather an object-of-interest is a pedestrian on not by finding a match with these models.

In feature classifier (FC) approach, predefined features are used to represent pedestrian objects and a classifier classifies an object to a pedestrian or not based on the similarity of the features of the object-of-interest to that of the pre-trained model. The focus here will be how best a feature can be derived to distinguish a pedestrian object from other objects as well as how we arrive at a best pre-trained model to match the features of the object-of-interest. The first part deal with deriving and fine-tuning features where as the second part deals with developing a good training and classification algorithm. Along with use of the best features, a proper classifier trained with the best representative pedestrian dataset is also important in accurate detection of pedestrian objects.

In deep learning (DL) approach, there are no predefined features. The system learns best features by itself to classify the training samples. Here the training set is much more important than features. However, with a very good representative training set, a DL based system could out perform a FC based system.

In Automotive domain, Real time detection, sufficient accuracy, minimal memory footprint and computational requirements, are important to develop a practical pedestrian detection system as it is dealing with a safety critical problem.

This paper presents the state of the art pedestrian detection problem as a single point reference for researchers to make a better understanding of the system. This paper is organized as follows. Section II explains the automotive specific scenario and requirements in general and section III takes through the safety aspects of it. Section IV discusses on various data capture mechanism used for automotive scenario to capture live road scenes. Section V discusses on various datasets available for developing and evaluating automotive pedestrian detection solutions. Section VI explains classical feature-based pedestrian detection and section VII covers the modern deep learning based approach. Section VIII discusses special cases like occluded pedestrians and handling repeated pedestrian detection. Section IX discusses object localization and tracking in general. Section X discusses evaluation of the pedestrian detection algorithms. Section XI mention about the research opportunities, followed by conclusion and references.

II. AUTOMOTIVE SCENARIO AND SYSTEM REQUIREMENTS

Advanced Driver Assistance System (ADAS) will help driver to have a better understanding of the environment and particularly of the road ahead of the vehicle. Human errors in noticing vehicles, pedestrians, cyclists, and other objects (like Traffic signals, road-side entities etc.) in front of the vehicle may lead to accident and severe casualties. ADAS is meant to avoid such situation. System will monitor the road ahead and inform the driver on any possibility of a collision. The first step will be informing the driver, but as next step, the ADAS system may take control of the vehicle to avoid a collision. In case of an unavoidable collision, ADAS will work to minimize the casualties. ADAS typically have two parts - Active safety and Passive safety.

Active safety includes measures to avoid a collision whereas passive safety deals with measures to minimize casualties in case of an unavoidable collision. Figure 1 shows an overview of ADAS. Pedestrian detection is a part of active safety measures or collision avoidance system.

As automotive pedestrian detection deals with safety critical situations, the requirements are very stringent and should follow automotive safety standards. An object detection or person detection used in a consumer electronics system or digital signage system does not deal with such safety critical situations.

Automotive standards needs to be followed for any system through enhanced stability, predictability and reliability. Safety and security are of paramount importance. Since the misbehavior of systems in a vehicles may lead to hazardous situations to the passengers as well as other vehicles or pedestrians on the road, at most care to be taken on the system reliability. All software running on the vehicle systems need to be secured against vulnerabilities causing an external entity taking control of the vehicle.

Vision based ADAS components should be specifically trained to work with the images captured using a moving camera. The ever-changing scenes due to the moving vehicle will lead to increased computational complexity. The response time should be very minimum so as to have real-time feedback. Typical approach is to have a separate Engine Control Unit (ECU) for ADAS along with the Infotainment system. However, current trend is to combine In Vehicle Infotainment (IVI), Instrument Cluster (IC), and ADAS to be driven by a single module called eCockpit, to have the best integration and synchronization. The hardware platform should be carefully chosen to facilitate multi-core Central Preprocessing Unit (CPU) and Multi-core Graphics Processing Unit (GPU) support for computationally complex ADAS processing. The ADAS algorithms should be optimized for the hardware to provide real-time response to the driver. Also any automotive system should be designed with minimum processing requirement so as to

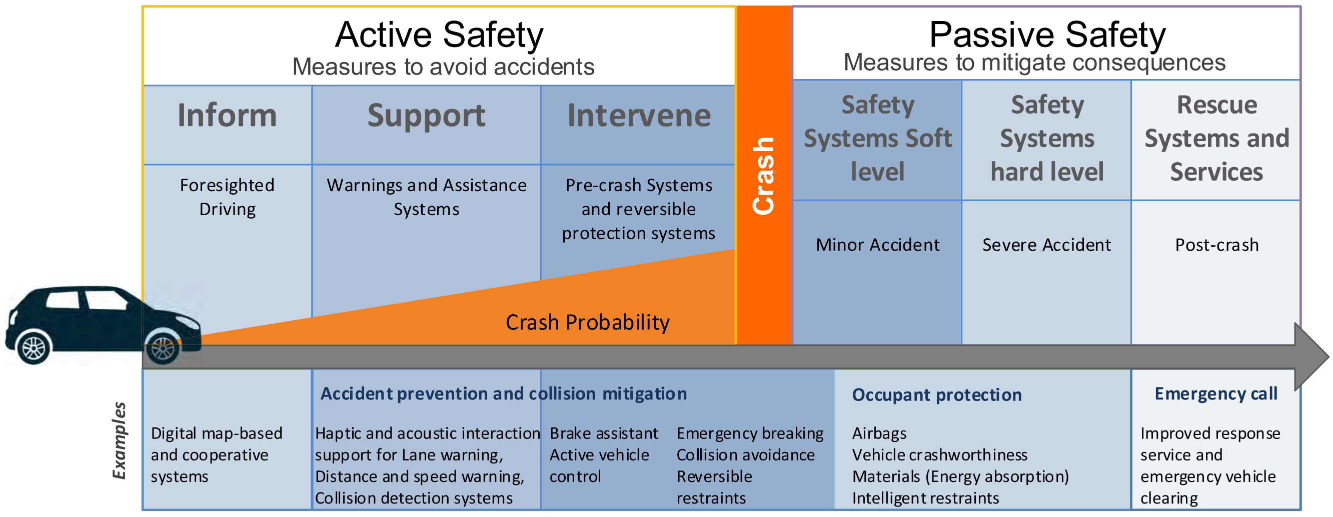
Active Safety
Measures to avoid accidents
Inform
Support
Intervene
Foresighted Driving
Warnings and Assistance Systems
Pre-crash Systems and reversible protection systems
Crash Probability
Crash
Passive Safety
Measures to mitigate consequences
Safety Systems Soft level
Safety Systems hard level
Rescue Systems and Services
Minor Accident
Severe Accident
Post-crash
Examples
Digital map-based and cooperative systems
Accident prevention and collision mitigation
Haptic and acoustic interaction support for Lane warning, Distance and speed warning, Collision detection systems
Brake assistant Active vehicle control
Emergency breaking Collision avoidance Reversible restraints
Occupant protection
Airbags Vehicle crashworthiness Materials (Energy absorption) Intelligent restraints
Emergency call
Improved response service and emergency vehicle clearing

FIGURE 1. Advanced driver assistance system (ADAS).

limit the load on vehicle battery and alternator for better fuel consumption.

Another important consideration in automotive domain is the ability to work on harsh weather conditions. Most of the vision-only-based systems fails here as they are designed to work on day-light or well-lit environments. However, combining vision-based algorithms with RADAR and LIDAR based approaches will help in addressing this requirement. In this paper, we primarily focus on vision-based systems.

III. SAFETY ASPECTS OF PEDESTRIAN DETECTOR

We have already discussed that pedestrian detection is one major component of ADAS. ADAS as a system also incorporates higher level features like collision detection and avoidance, highway assist (self driving in a controlled environment by following a lead vehicle on highways) etc., in addition to pedestrian protection. When we say safety aspect of pedestrian detection, it is same as the safety aspects of an ADAS implementation [5]. The brain of ADAS is a Machine Learning (ML) software, which takes decisions based on the critical observations of the objects in the surroundings. As machine learning is the core of ADAS, there where attempts [4] to assess the suitability of it for compliance with automotive safety standard ISO 26262 [3]. ISO 26262 standard includes guidance to minimize the risk of human hazards by ensuring a proper process in designing the hardware and software components of an automotive system. Figure 2 gives a high level picture of safety development process as defined by ISO 26262. It defines different Automotive Safety Integrity Levels (ASIL) to address the safety requirements depending on the criticality of a system. ASIL A represents the lowest and ASIL D the highest level of security requirements. Along with the system design guidelines, ISO 26262 also specifies the verification and validation requirements to ensure that the system always work at an acceptable risk level. The core of any process is to ensure that the system have a predictable

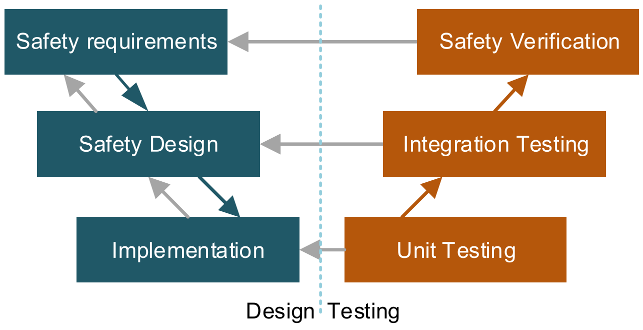

FIGURE 2. ISO 26262 defines development process in safety perspective.

level of working and any misbehavior can be easily traced to the root cause.

We can see that the process defined is suitable for both hardware and software. While saying a software, we mean a programmed component which is developed using a programming language either manually or automatically. However, an ML component is a software model that is learned through supervised on unsupervised training. There is no defined code and fully predictable output. Also there are some inherent properties of ML, which makes it difficult to comply to the software process. They are - *non-Transparency* (contains knowledge in an encoded form), *error rate* (the output is not fixed; there could be an error always.), *instability* (results will be different with different training process or dataset) and is *training based* (only a subset of possible inputs can be used to train the model). Salay *et al.* [4] analyze these in detail and suggests how the ML model can be modified to ISO 26262 standard compliance.

IV. DATA CAPTURE AND SENSORS

Pedestrian detection is an image processing problem. Image processing based ADAS systems work using 2D or

FIGURE 3. **Camera sensors used in automotive scenario.**

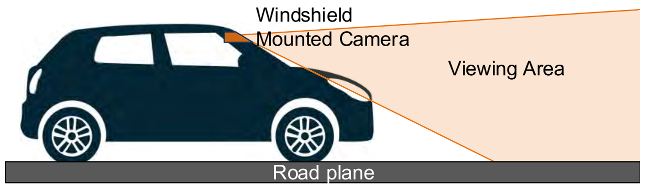

FIGURE 4. **Camera mounting on vehicle.**

stereoscopic 3D images. These images can be optical images captured using normal visible light cameras or thermal images captured using infrared cameras. These cameras will be fixed on windshield facing forward onto the road in front of the vehicle. Ranging information from RADAR or LIDAR sensors can also be used for additional verification of information in the process.

Optical cameras are the most used sensor type for capturing data required for pedestrian detection process. There could be mostly a single camera but there are also configurations with two synchronized cameras to capture stereo image pair. Solutions which focus on low-light/night-vision pedestrian detection use infrared cameras. There are systems which use a combination of optical and infrared cameras for better accuracy in all lighting conditions. For optical imaging, older systems were using monochrome (gray-scale) cameras whereas recent systems use full color cameras. These cameras are shared by other ADAS solutions running on the system. The algorithms may use gray-scale images or full color components as required.

When stereo image pairs are used, typically, image from one camera will be used for processing along with the depth map generated from the stereo processing unit. The depth map will help in identifying objects from the background, even though there is no significant intensity differentiation.

Most of the pedestrian detection algorithms proposed by various researchers are based on single camera images. The amount of research in stereo based pedestrian detection is considerably less compared to intensity only methods. This could be because of the difficulties in getting proper depth field information in different road conditions as well as the additional computational overhead and cost involved.

However, there are some approaches utilizing the information from sensors other than single camera. In [8] Keller *et al.* proposes a pedestrian detection method by computing histogram of oriented gradients (HOG) [9] from both depth field and intensity image. They apply the fusion of intensity and depth field at classifier level and claim that they achieved a performance improvement up to 7.5 times of that using only intensity image. Kira *et al.* used stereo disparity data along with edge response in [10] to detect long range pedestrians. In [11], Olmeda *et al.* uses far-infrared images from a vehicle mounted single infrared camera for pedestrian detection. Even if there are some exceptions as we discussed, day-light optical images are mostly used in pedestrian detection. In this survey, our main focus is intensity based pedestrian detection using single optical camera.

There are also some attempts using LIDAR along with camera input for pedestrian detection. In [12] Szarvas *et al.* propose a method using fusion of camera and LIDAR data with convolutional neural network classifier and claims they could reduce the false positive by a factor of 2 and improve the computational performance by 4 folds by achieving 10 fps computational performance. Because of the high cost involved in using LIDAR or RADAR sensors, these approaches are not widely accepted. But, in case the vehicle already have a LIDAR or RADAR sensor, then those can be shared with the pedestrian detection system and to improve the performance.

V. PEDESTRIAN DATASETS

Since most of the pedestrian detection systems are learning based, use of proper training data is very important. Also for better evaluation and comparison of the performance of different solutions, use of standardized test data with ground truth information is required. There are many different pedestrian detection datasets available. Broadly they can be classified as 'person' database and 'pedestrian' database. The Person database contains images of single or group of 'persons' or people in unconstrained pose and variety of domains. However, in our study, we focus on more relevant 'pedestrian' databases, where nearly upright and possibly moving people - referred as pedestrians - are imaged. This is matching with the pedestrian detection problem in automotive safety domain. The pedestrian detection system may use one or many of these datasets to train the classifier. Each dataset may contain a set of training vectors as well as a set of test vectors with ground truth information on the position and size of various pedestrians available in the scene covered by the test vectors. The pedestrians identified by the pedestrian detection system will be compared against the ground truth data to measure the detection rate and accuracy.

Table 1 shows a list of publicly available datasets suitable for automotive safety domain pedestrian detection applications. This is not an exclusive list. These datasets vary in size, the way the images collected as well as the size and resolution of pedestrians present. The last column in the table above shows the median pedestrian height in pixels. We can see that the most of the datasets except Daimler, TUD Brussels and Caltech are providing pedestrians of bigger size. However, in a typical portable setup with cameras fixed on moving vehicles, observing the road ahead, the size of pedestrians will be of medium size with average size of 96×48 pixels. This along with the availability of large number of training data makes Daimler and Caltech datasets very suitable for

TABLE 1. Datasets used for pedestrian detection.

Dataset	Imaging Setup	Training Data #Pos	Training Data #Neg	Median Ped.Height
MIT [7]	photo	924	-	128
USC-A [13]	photo	-	-	98
USC-B [13]	surveillance	-	-	90
INRIA [9]	photo	1208	1218	279
OSU Thermal [14]	photo	284	-	-
Daimler-CB [15]	mobile	2400	15000	36
USC-C [16]	photo	-	-	108
CVC [17]	mobile	1000	6175	83
ETH [18]	mobile	2388	-	90
TUD-Det [19]	mobile	400	-	218
NICTA [20]	mobile	18700	5200	72
CamVid [21]	mobile	640	-	-
TUD-Brussels [22]	mobile	1776	218	66
Daimler-DB [23]	mobile	15600	6700	47
Caltech [22]	mobile	192000	61000	48
KITTI [24]	mobile	25000	-	-
LSI FIR [11]	mobile	10208	43390	64
Cityscapes [25]	mobile	24400	-	-
CityPersons [26]	mobile	35016	-	-
VIPER [27]	virtual	134000	-	-

automotive applications. Pedestrians of larger size - or too close to vehicles are typically not common in automotive safety applications, where the system has to identify pedestrians from sufficient distance so as to avoid causalities.

Another aspect of the pedestrian dataset is how the images are captured. For automotive safety domain, a portable video capturing setup [28] is more preferred over a static imaging setup. This portable setup provides realistic imaging scenarios and helps to develop robust and optimum solutions for pedestrian detection in automotive safety.

Pedestrian detection in automotive safety is typically of two types: 1) pedestrian detection from infrared (IR) images to support night vision and 2) pedestrian detection from optical images to support day-time or lit-up scene detection. Both these cases are important to vehicle safety, however, most of the pedestrian detection datasets available are for day-time pedestrian detection. There are exceptions like OSU thermal image dataset [14] and LSI far infrared pedestrian dataset [11]. Some of the methods of day-light pedestrian detection are also applicable for night-time pedestrian detection; however, the feature descriptor should be selected considering this. We can also combine both optical images as well as IR images to improve the detection accuracy at an increased computational cost. In day-light images, singular or stereo images are available. Singular images are mostly used but there is an increasing trend of using stereo images for improving object detection. KITTI [24] is an extensive set of Computer Vision benchmark suite. The suite consists of real-life stereo and optical flow images particularly suitable for 3D object detection and tracking. The dataset includes objects of different types like pedestrians, cyclists and different types of vehicles. Stereo or color or monochrome images of objects as well as LIDAR data are available. There are around 25000 pedestrian objects with around 25% of them are semi-occluded and 5% are truncated.

Cityscapes [25] is a recent dataset specifically targeted for automotive applications, recorded on all normal weather conditions from 50 cities in Europe. The dataset is designed keeping deep learning based approaches also along with classical feature based approaches. The dataset contains more than 25000 annotated images with annotations for 30 different object types grouped to different classes including human, vehicle,nature, sky, construction etc. Both pixel-vise and instance-vise semantic annotations are available. The dataset includes hi-resolution and Hi-Dynamic Range (HDR) annotated stereo image pairs along with vehicle information. Annotations in Cityscapes dataset are compatible to other common datasets and the classifiers trained on the dataset perform better while validating with other datasets. Zhang *et al.* created a new set of pedestrian annotations on top of Cityscapes to form CityPersons [26] dataset. CityPersons include around 35,000 person annotations including training, testing and validation sets. Out of the 19,654 unique person annotations, 83% are of pedestrians, remaining 17% covers riders, passengers and other person objects. Using CityPersons dataset for training, authors could achieve better pedestrian detection performance on various datasets. CityPersons also include much more occluded pedestrian annotations than any other datasets. Out of the pedestrian annotations, only 30% are fully visible pedestrians. This makes CityPersons a more practical dataset for pedestrian detection.

The Visual Perception benchmark (VIPER) [27] dataset introduces the new concept of virtual world. The dataset is generated from 250,000 full hi-definition (HD 1080) realistic-virtual video frames, generated using advanced gaming concepts. All the images are annotated to be used with various types of object detection approaches. The dataset includes optical flow images as well as 3D scene layouts along with the annotated images. Pixel-wise and instance-wise semantic annotations available. Being a virtual dataset, the possibilities are unlimited. The model can also be upgraded to include more object categories as well as environment conditions. Virtual datasets have a better future in preparing a near-complete dataset to address any problem. This also saves the cost of data capturing from the real world but have limitations of computer graphics, which is getting narrowed as the graphics rendering techniques are improving.

There are many other datasets like Microsoft COCO [29], PASCAL Context [30], PASCAL VOC [31], ImageNet [32] etc. used in computer vision and deep learning, which are not designed for automotive use cases and we are excluding them as our focus is on automotive domain. In this paper, we focus more on methods using day-light pedestrian detection from singular images.

We have discussed about various dataset but there is no universal dataset which can produce a perfect classifier. In other

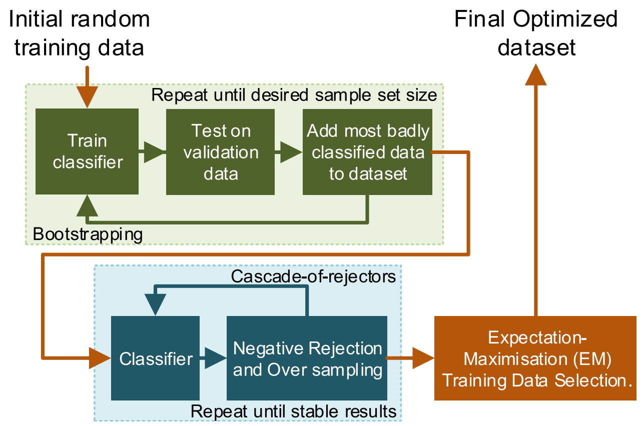
Initial random training data
Final Optimized dataset
Repeat until desired sample set size
Train classifier
Test on validation data
Add most badly classified data to dataset
Bootstrapping
Cascade-of-rejectors
Classifier
Negative Rejection and Over sampling
Repeat until stable results
Expectation-Maximisation (EM) Training Data Selection.

FIGURE 5. Dataset optimization.

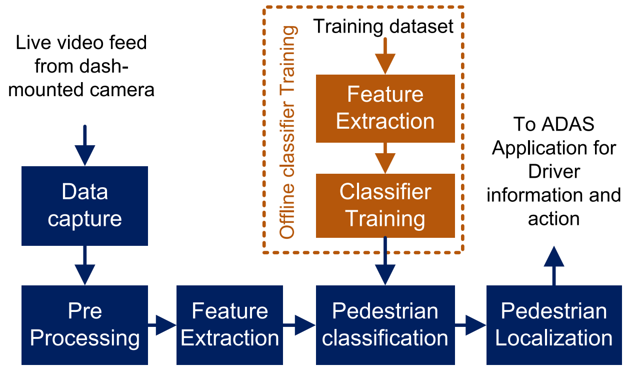
Live video feed from dash-mounted camera
Offline classifier Training
Training dataset
Feature Extraction
Classifier Training
To ADAS Application for Driver information and action
Data capture
Pre Processing
Feature Extraction
Pedestrian classification
Pedestrian Localization

FIGURE 6. Feature-based pedestrian detection system.

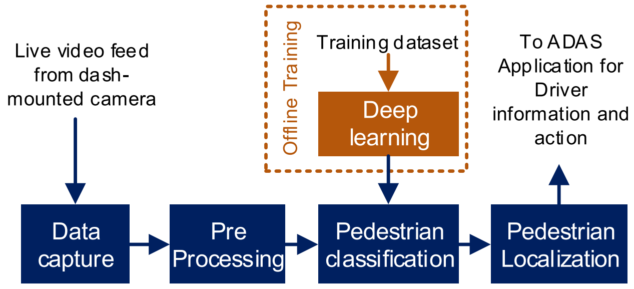
Live video feed from dash-mounted camera
Offline Training
Training dataset
Deep learning
To ADAS Application for Driver information and action
Data capture
Pre Processing
Pedestrian classification
Pedestrian Localization

FIGURE 7. Deep Learning model based pedestrian detection system.

words, none of these datasets represent a hundred percent perfect object class. As we already mentioned, the quality of any classifier depends on the dataset with which it is trained. A perfect dataset leads to the best of a classifier at the same time, an imbalanced dataset limits the classification efficiency. It is not possible to simply tell which data contributes to a good classification and which do not. There are different attempts to improve the dataset for pedestrian detection. FairTrain [33] is one of the recent attempt, where the authors propose an improved data selection and training process. FairTrain proposes an optimum initial dataset selection through bootstrapping followed by a cascade of rejectors to prepare an optimized dataset for the final classifier. Figure 5 shows the dataset optimization process as described in [33]. The process start with a small set of randomly selected data samples and iteratively refines the dataset by including the most mis-qualified validation samples. 1000 iterations are recommended to refine the dataset while a cascade of rejectors are employed and 5000 iterations recommended when classifier is standalone. Authors also shows that with optimized dataset, a simple Histogram of Oriented Gradients (HOG) based Support Vector Machine (SVM) [34] classifier outperforms many of the best-known classifiers.

VI. PEDESTRIAN DETECTION - FEATURE-BASED APPROACH

The classical approach of pedestrian detection is a feature-based classifier as shown in figure 6. There are typically two stages in the pedestrian detection process - an offline training stage and online detection stage. During the training stage, the pedestrian detector is trained with many positive (True pedestrian) samples as well as negative (non-pedestrian) samples. A classifier engine is constructed based on the features extracted from these samples. During the actual detection stage, live video feed from the dash-mounted camera is fed into the system, where the system will detect the pedestrians visible in the scene and appropriately inform the driver with details.

The classifier training is an offline activity through which we will create a classifier engine, which can discriminate a pedestrian image from a non-pedestrian image based on the intelligence learned through the iterative training process. The actual classification is an online activity where the pedestrians are identified from the live video/image sequence. Because of the computational complexity of the pedestrian detection process, there are very less solutions available with near real-time performance and the detection accuracy may be compromised in some solutions to meet the computational requirements. One of the best pedestrian detection solution in terms of speed and detection performance is CrossTalk [35] proposed by Dollar *et al.*, which runs at 14 fps with Caltech USA test video of VGA (640x480) resolution at around 50% detection rate using an Intel core i7 computer. Another solution by Dollar *et al.* [36] detects pedestrians at around 5 fps from real-life VGA video. The recent proposed solution from Dollar *et al.* [37] improved the detection time by using feature pyramids to avoid re-computation of feature descriptors at different scales. Other solutions are far behind this in detection time. Of the 16 top speed systems tested on Caltech dataset, the average pedestrian detection rate is less than 30%. However, in closed environments and non real-time systems, many solutions provide better detection rates. In this section, we will discuss about common features and classifier architectures used in feature-based pedestrian detection systems.

A. ROI SELECTION AND PREPROCESSING

The object detectors are trained with fixed size object samples and expect similar input while actual classification. For

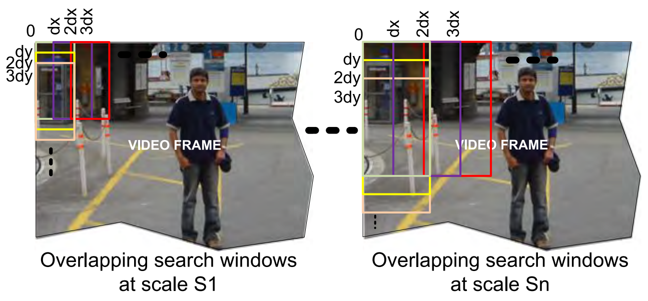
0
dx
2dx
3dx
dy
2dy
3dy
VIDEO FRAME
Overlapping search windows at scale S1
Overlapping search windows at scale Sn

FIGURE 8. Scanning window ROI selection.

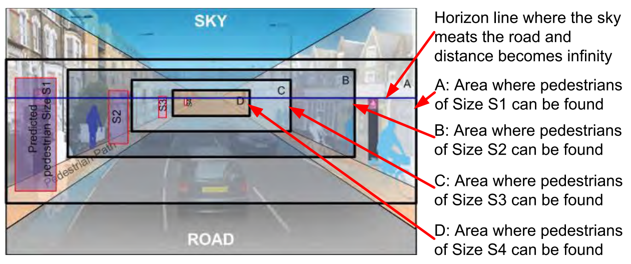
SKY
ROAD
Predicted pedestrian Size S1
S2
S3
Pedestrian Path
A
B
C
D
Horizon line where the sky meats the road and distance becomes infinity
A: Area where pedestrians of Size S1 can be found
B: Area where pedestrians of Size S2 can be found
C: Area where pedestrians of Size S3 can be found
D: Area where pedestrians of Size S4 can be found

FIGURE 9. Limiting search window and scale using a road model.

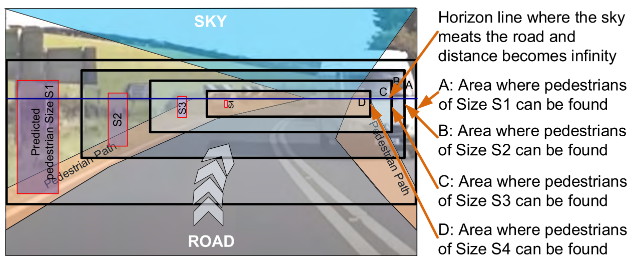
SKY
ROAD
Predicted pedestrian Size S1
S2
S3
Pedestrian Path
Pedestrian Path
A
B
C
D
Horizon line where the sky meats the road and distance becomes infinity
A: Area where pedestrians of Size S1 can be found
B: Area where pedestrians of Size S2 can be found
C: Area where pedestrians of Size S3 can be found
D: Area where pedestrians of Size S4 can be found

FIGURE 10. Context-based road model - turn right.

example, to train a pedestrian detector, cropped and scaled pedestrian images are given as positive samples as well as equal size non -pedestrian sample images are given as negative samples. The classifier will classify a similar size image as pedestrian image or non-pedestrian image based on the feature extracted from them. To detect pedestrians in a real life image, we actually extract smaller portions of the image and check for presence of pedestrian in them. Images are extracted from all possible positions and sizes, and then scaled to an acceptable size before giving to classifier. This process is called Region Of Interest (ROI) extraction.

Standard way of selecting ROI from test image is using a scanning window to search through the image using overlapped ROI windows. Figure 8 shows the scanning window process for ROI selection. These windows will be placed at all possible pixels and scales in the image. If multi-scale detection is used, the ROI windows are used in different scales and then these are scaled to the uniform size used while training the classifier before proceeding with extracting the feature. The ideal method is to use all possible ROI size and place the windows on all possible pixel locations in the image. However, this will greatly increase the computational complexity. To reduce the complexity, only limited scales and window locations are used.

Another way to reduce the computational complexity in scanning window approach is to reduce the search space instead of limiting the window scale and position. If unnecessary portions of the image including the image background and the areas of the scene where objects of interest are not expected can be excluded from the search space, there will be huge saving in computational cost. Complex techniques may be used for background removal and ROI selection considering this computational savings. One of our previous papers [38] analyzes the possibility of using the camera position and road model in predicting the potential area and size of the pedestrians expected in the so as to reduce the search space and hence the computational complexity. Figure 9 shows the road image model used in [38] to reduce the search space. We could see that by applying a road model on the image, we can reduce the search space by a factor of 8. However, the road model need not be same in all conditions. This may change when the vehicle is turning, ascending or descending. Figure 10 shows the context based road model, which can be applied while the vehicle is turning right. During actual detection process, the system can get the vehicle status and adaptively adjust the road model to minimize search space.

References [38] and [39] discusses the possibility of applying background removal using Active Contour Model (ACM) as described by Chan and Vese [40]. By applying ACM, the search space could be further reduced by a factor to 2 to 4 [38] depending on the image.

The images captured are usually preprocessed before providing to the classifier algorithm for analysis. Preprocessing used for training is repeated on the test Regions of Interest (ROI) also, to improve the detection performance. The most common preprocessing operations are intensity normalization and scaling of cropped ROIs. The images are prepared so as to be efficiently processed by the algorithm. The preprocessing techniques used may vary from simply cropping of images to transforming the images into a different domain. The common preprocessing operations include image background removal and enhancing image quality(brightness, contrast, gamma, sharpness/smoothness etc) of the ROI.

Different types of filtering may be applied on the image to enhance or highlight the features that will be used by the algorithm for object detection. This could be simple convolution filters or even complex transformations, which could considerably improve the detection accuracy or speed.

B. FEATURE EXTRACTION

The preprocessed ROI image is not directly fed to the classifier. An image can be considered as a 2-dimensional array of correlated intensity values. It will be easy for humans to identify the object present from this image data, but for an automated Artificial Intelligence (AI) system like pedestrian detection, limited information can be directly taken from this intensity map. To improve the classification efficiency, we need to extract some feature descriptors from the ROI, which will help the classifier to identify whether the ROI contain an object of interest. A feature descriptor should help the classifier to uniquely identify different classes of input types - in our case a pedestrian and a non-pedestrian. There are different feature descriptors used by different researchers. It is not easy to simply say one is better to another. The feature descriptor to be used will depend on what type of information we are searching from the image. The feature could be as simple as the average intensity of the image or a complex vector like co-occurrence histogram of oriented gradients (CoHOG). The algorithm may use a single feature descriptor or a combination of multiple feature descriptors extracted from the image. In some case, the size of combined feature vector may be more than the size of the image under processing. The purpose is to extract some vector from the image, which will uniquely represent its content to the algorithm.

It is not possible to specify a particular feature descriptor which will uniquely represent the image content in all scenarios. Some descriptors may perform better in some cases and may fail in other cases. Researchers are working on identifying new feature descriptors or improving the existing ones to be used in different cases like pedestrian detection. Years of research improved the feature descriptors a lot. Highly capable feature descriptors as well as combination of multiple feature descriptors were introduced. Here we are discussing some of the feature descriptors and combinations commonly used for pedestrian detection. We should use the optimum feature descriptor with sufficient uniqueness at minimum computational complexity to develop an efficient pedestrian detection solution.

Since we are focusing on automotive safety applications, where low or medium resolution pedestrians has to be detected, we focus on sliding window based approach, which is more suitable for the target application. There are basic features addressing image intensity variations, object shapes, histograms, and many other aspects of images like human motion. There are a large number of feature descriptors proposed by various researchers. We address the most common ones, their enhancements and combinations in the remaining part of this section. An extensive survey of all the feature descriptors used for object detections outside the scope of this document. The readers can refer detailed surveys by Dollar *et al.* [28], Enzweiler and Gavrila [23] and Geronimo *et al.* [41] for more details.

1) HAAR WAVELET

Haar wavelet is first proposed by Papageorgiou and Poggio [7] as a feature for object detection. This is further taken up by Viola *et al.* [42]. Papageorgiou used SVM [34] as a classifier whereas Viola and Jones used an Adaboost [43] decision tree classifier. We will discuss SVM and Adaboost while we discuss classifiers in detail. The wavelet function and its scaling function can be represented as in

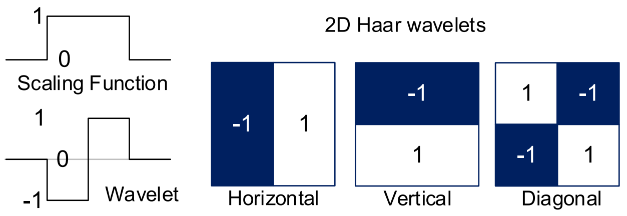

FIGURE 11. **Haar wavelets for pedestrian detection.**

equations 1 and 2.

Wavelet function:

$$\varphi(t) = \begin{cases} -1 & 0 < t < \frac{1}{2}, \\ 1 & \frac{1}{2} < t < 1, \\ 0 & otherwise. \end{cases} \tag{1}$$

Wavelet scaling function:

$$\phi(t) = \begin{cases} 1 & 0 < t < 1, \\ 0 & otherwise. \end{cases} \tag{2}$$

Figure 11 shows the basic construction of a Haar wavelet and some of the generated filter masks. Viola and Jones Haar-like feature was one of the oldest and successful feature description used for human face detection and human detection in general. The integral image representation reduced the computational complexity significantly which helped near-real-time human detection. Viola *et al.* [42] further proposed computing Haar-like features on difference images from motion sequence which further reduced the computational complexity. There is a reference implementation [44] which is available as part of Open CV [45] library, implementing Viola Johns Haar features (VJ) for pedestrian detection. Trompouki *et al.* [46] took this implementation and optimized it for multi-CPU-multi-GPU platforms with specific focus to automotive use-cases. They could achieve around 90 times improvement on computation time and could meet real-time detection at 21fps.

2) EDGELET AND SHAPELET

Shape of the object is a key point in identifying any object of interest. Shapes can be represented by the silhouette of the object, which can be a combination of line and arc segments, called edgelets. The edges can be extracted from image by applying edge filters like sobel filter. Wu and Nevatia [13] used specific combinations of edgelets of 4 to 8 pixels in length at specific locations for detecting a pedestrian silhouette. Line and arc edgelets and their symmetric pairs were used to identify Head and shoulder, Torso and legs along with whole body. Wu and Nevatia proved that the combined detection of these body parts together will be a good detector for pedestrian. Figure 12 shows the use of edgelets as described in [13]. Sabzmeydani and Mori combined weighted directional gradient of image ROI to learn a detector called shapelet [47] by using a variant

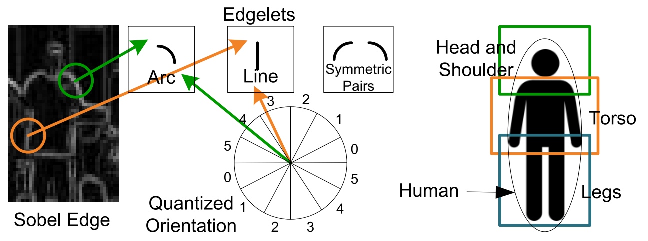

FIGURE 12. Edgelet feature.

of Adaboost decision. Adaboost is further used to build a classifier from these shapelets. However shapelet is a feature learned from rather weak directional gradients, whereas edgelets are defined features directly used with classifier.

A feature called edge affinity function can be computed from the edgelets which will represent the distribution of the edgelets at different positions in the image ROI. Edge affinity will in effect represent the shape of the object in the ROI. Edge affinity function can be computed as follows.

Let an image I be represented as edge intensity $M^I(p)$ and normal $n^I(p)$ at position p. The Edge intensity M^I and the edge direction Θ^I can be computed by applying an edge filter like Sobel operator as below. Normal n^I is computed from edge direction Θ^I.

$$G_x = \begin{bmatrix} +1 & 0 & -1 \\ +2 & 0 & -2 \\ +1 & 0 & -1 \end{bmatrix} * I, \quad G_y = \begin{bmatrix} +1 & +2 & +1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix} * I$$

$$\text{Edge Intensity } M^I = \sqrt{G_x^2 + G_y^2},$$

and

$$\text{Edge Direction } \Theta^I = atan\left(\frac{G_x}{G_y}\right)$$

Neighboring edge points can be combined to form edgelets, which are small edge segments, represented with the position and normal vectors of the edge points. Let an edgelet E contains k edge points. The edgelet E can be represented as $E = \left[\{u_i\}_{i=1}^k, \{n_i^E\}_{i=1}^k\right]$. Now the affinity between edgelet E and the image I at position p can be computed as:

$$f(E; I, p) = \frac{1}{k}\sum_{i=1}^{k} M^I(u_i + p)\left|\left\langle n^I(u_i + p), n_i^E\right\rangle\right| \tag{3}$$

For simplifying the computation, the dot product $\left|\left\langle n^I(u_i + p), n_i^E\right\rangle\right|$ between two normal vectors can be approximated as

$$l[\delta] = \begin{cases} 1 & \delta = 0, \\ 4/5 & \delta = \pm 1, \pm 5, \\ 1/2 & \delta = \pm 2, \pm 4, \\ 0 & otherwise. \end{cases}$$

where the input δ is the difference between two quantized orientations. Let the quantized edge orientations of image I and the edgelet E are represented by $V^I(p)$ and $\left\{V_i^E\right\}_{i=1}^k$ respectively. Now the affinity computation will be simplified as

$$f(E; I, w) = \frac{1}{k}\sum_{i=1}^{k} M^I(u_i + w).l(V^I(u_i + w) - V_i^E) \tag{4}$$

3) HISTOGRAM OF ORIENTED GRADIENTS (HOG)

Gradient based feature descriptors were widely accepted because of their invariance to intensity changes. HOG is one of the most successful gradient based feature descriptor for pedestrian detection which was proposed by Dalal and Triggs [9]. HOG was subject of many researches and there are many variants of HOG and combination of HOG with other methods, which led to better detection performance. Mostly all of the modern detectors use HOG in some form. Computation of HOG is explained below.

Let $I(x, y)$ be the gray-scale image of $(w \times h)$ resolution. The gradients of I, g_x and g_y in x and y directions can be computed by filtering with $[-1; 0; +1]$ and $[-1; 0; +1]^T$. Then the gradient orientation $\theta(x, y)$ and magnitude $M(x, y)$ is computed as

$$\theta(x, y) = atan\left[\frac{g_y}{g_x}\right]$$
$$M(x, y) = \sqrt{g_x^2 + g_y^2} \tag{5}$$

To simplify the computation, gradient directions can be quantized into b bins as below:

$$\theta_q(x, y) = round\left(\frac{b \times \theta(x, y)}{\pi}\right) \text{ mode } b$$

Finally, the gradient image $G(x, y, z)$, a sparse matrix of size $(w \times h \times b)$ can be computed as:

$$G(x, y, z) = \begin{cases} M(x, y) & z = \theta_q(x, y), \\ 0 & otherwise. \end{cases}$$

Histogram of the orientations are then computed over fixed size overlapped blocks. Let $B(s \times s)$ be the defined block at location (x, y) of G. The HOG of the block B is a vector of length b, where b is the number of orientation bins. The block-HOG of block B can be represented as:

$$HOG_B = \left\{\sum_{d_x=1}^{s}\sum_{d_y=1}^{s} G(x + d_x, y + d_y, z)\right\}_{z=1}^{b} \tag{6}$$

HOG of the ROI is then computed by concatenating block-HOGs of all overlapped blocks within the ROI. The block-HOGs are also optionally normalized over neighboring blocks to minimize variations on image brightness. Figure 13 gives a pictorial representation of the computation of HOG feature descriptor.

Felzenszwalb *et al.* [48], [49] improved the use of HOG by using coarse scale HOG as in [9] and additional fine scale HOG of body parts, which can be moved and deformed. Zhu *et al.* [50] used integral histograms for faster HOG computation. Watanabe *et al.* [51] proposed a high dimensional

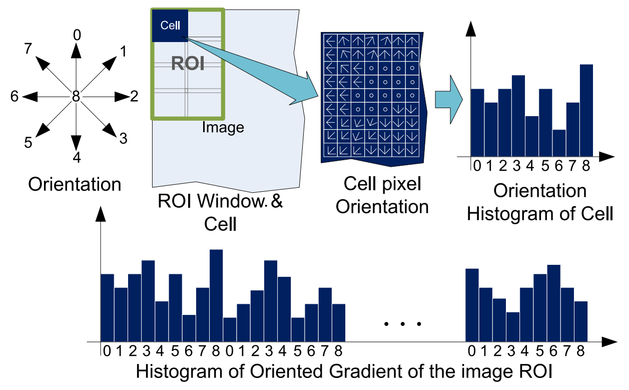
Orientation
Cell
ROI
Image
ROI Window & Cell
Cell pixel Orientation
Orientation Histogram of Cell
Histogram of Oriented Gradient of the image ROI

FIGURE 13. **HOG feature.**

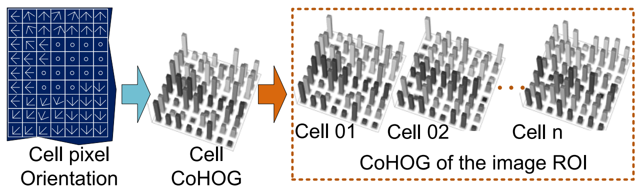

FIGURE 14. **CoHOG feature.**

feature called Co-occurrence Histogram of Oriented Gradients (CoHOG) based on HOG where instead of taking the histogram of the gradients, histogram of gradients of pixel pairs are taken. This significantly improves the detection performance of classifier at the cost of increased feature size and computational complexity. Figure 14 shows the computation of CoHOG. SVM is usually used as classifier when HOG or CoHOG are used as feature descriptor. In [52] Nan *et al.* proposed a different improvement to HOG. They used Histograms-Of-Salience (HOS), by aggregating saliency map learned using histogram based contrast with the oriented gradients.The authors showed large performance improvement compared to simple HOG on INRIA dataset using this approach.

4) LOCAL BINARY PATTERNS (LBP)

Local binary patterns are also used along with HOG for improving the pedestrian detection performance [53]. LBP captures the local texture properties in an image. As in HOG, the image will be divided into blocks for calculating the LBP. Value of each cell is compared with each of its 8-neighbor cells. If the cell value is greater than the neighbor, a value of 0 is put in the neighbor cell. Otherwise a value of 1 is placed. Once all 8 neighbors are checked, the decimal representation of the binary number obtained by arranging the generated 1s and 0s in a clock-vise order starting from the top neighbor is called the LBP of the cell. Let $\{P(i,j)\}_{i,j=1}^{k}$ be the cell at location (i,j) in the image block and $\{N(k,P)\}_{k=1}^{8}$ represent its 8 neighbors in clock-vise order starting from the top one, then the LBP can be computed as the decimal equivalent of

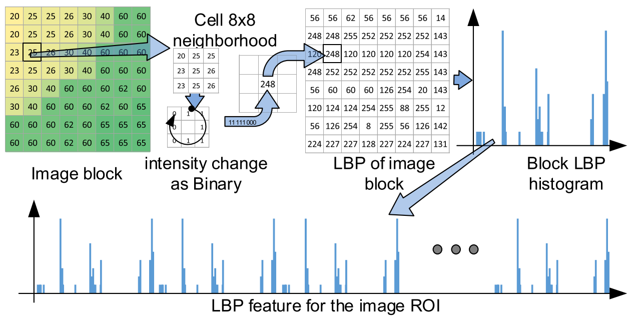

FIGURE 15. **LBP feature.**

the Binary number $B[k]$, which is defined by:

$$LBP = Integer(B) \tag{7}$$

where the binary pattern B is constructed as,

$$B = \{B[k]\}_{i=1}^{8} = \begin{cases} 0 & P > N(k,P), \\ 1 & otherwise. \end{cases}$$

LBPs of each cell is computed to generate the Block LBP. To generate a feature vector out of this, histograms of each LBP block is computed and then concatenated for an image ROI similar to HOG computation. Figure 15 shows a pictorial representation of LBP computation process.

5) CHANNEL FEATURES

Enhancements of the basic feature descriptors as well as combination of them with other methods were proposed to improve the detection performance as well as reducing computational complexity. Instead of computing a single complex feature, recent trend is to use a combination of simpler features called channels on a boosting decision tree. These feature channels will be of same size as of the image and may be processed in same way by the classifier. Another advantage of these channels is they are capable of parallel implementation as well as further optimization. Originally Dollar *et al.* proposed the integral channel features [54] for pedestrian detection, where multiple registered image channels generated through linear and non-linear transformations on the image, features like local sums, histograms, and Haar features are efficiently computed using integral images. They prove that these combinations can outperform HOG or other individual features when computed efficiently. Dollar further improved the detection speed by utilizing the correlation between neighboring feature channel responses combined with a soft classifier model in [35].

Mathias *et al.* used the integral channels along with a modified occlusion specific detectors called Franken classifiers [55] to improve detection of partially occluded pedestrians. Lim *et al.* used 3 gradient magnitude channels by applying Gaussian filtering at three levels along with other integral channels and computed self-similarity features [56] on them. The featured dimension is then reduced before applying the boosting classifier. Zhang *et al.* modified

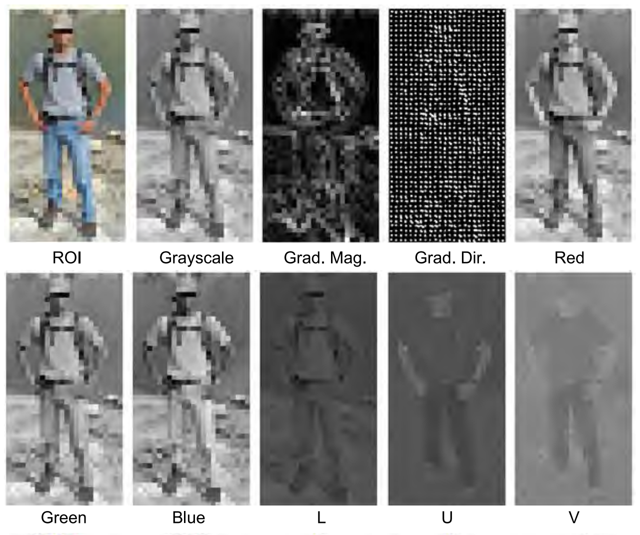

FIGURE 16. **Channel features.**

the channels by applying adaptive location specific Haar features [57] on the computed channels to reduce the feature size. Benenson *et al.* [58] also improved the detection by applying filtering on channels as described in [59] and strengthening the boosting classifier.

In [37], Dollar *et al.* use Aggregate Channel Features (ACF), which use a set of 10 features including normalized gradient magnitude, 6 channel HOG, and LUV color channels. They use channel pyramids to reduce channel computation at different scales. Nam *et al.* [60] and Dollar further improved the ACF features by using Locally De correlated Channel Features (LDCF) generated by decorrelating ACF. They could achieve a 10x reduction in false positives against ACF using LDCF at same detection rate. The features like Integral channel, ACF and LDCF are designed to be used in a feature pyramid so as to reduce the computational complexity in re-computing the features for different scales. Figure 16 shown some features used in generating feature pyramids.

Lin *et al.* proposed feature pyramid networks (FPN) [61] to take the feature pyramid concepts to deep learning algorithms. Figure 17 shows an optimized way of computing multi-scale channel pyramids as proposed in [37]. There are many other variations of channel features. Rajaram *et al.* proposed an improvement on ACF detector by training with multi-scale vectors [62]. They could achieve 6% improvement on the detection accuracy compared to single-resolution training.

Yang *et al.* used Convolutional Channel Features (CCF) [63] learned from raw image data using Convolutional Neural Network (CNN) or ConvNet [64], [65] instead of the computed channels discussed above. They also tried by combining CCF and ACF channels and resulted in better detection performance than both of these two. Zhang *et al.* extended the channel features to Checkerboard features by dynamically applying a filter bank [66] on HOG-LUV channels. Cai *et al.* developed a complexity

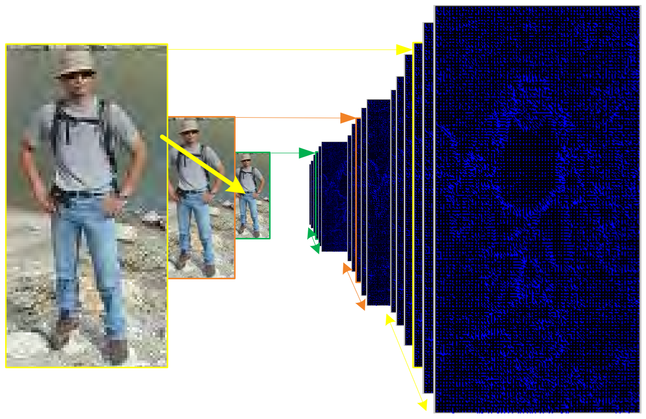

FIGURE 17. **Optimized multi-scale channel pyramid computation.**

aware cascade training (CompACT) method [67] which will dynamically select features from a pool of features with varying complexity. The feature pool contains features like ACF, self-similarity features, Checkerboard features, Locally decorrelated HOG features and some ConvNet features to further improve the detection performance.

6) OTHER FEATURE DESCRIPTORS AND COMBINATIONS

Channel features are of typically same size and can be easily aggregated. There are approaches like [67] where features of different type and length and complexity are also combined together to form a bigger feature descriptor. Wojek and Schiele [68] proposed a multi-feature, which is combination of Haar-like features, shapelets, shape context and HOG, and showed it outperforms any of these individual features. Walk *et al.* [69] further added local color self similarity and motion features to this multi-feature. Wu and Nevatia [70] combined HOG with edgelet and covariance features. Wang *et al.* [53] combined HOG with local binary patterns (LBP) texture descriptor.

Many researchers used combination of shape information and multi-part detection approaches to improve detection performance. Mohan *et al.* [71] used a two stage approach where the results of head, arm and legs detectors where matched with a rough human model. Lin and Davis [72] used a part-template tree to model human body parts and used HOG along with shape outline information. Enzweiler and Gavrila [73] performed human classification combined with labeling into one of four canonical orientations. Liu et al. proposed Context-Aware saliency detection on ROIs along with HLS (HOG, LBP and Scale-invariant feature transform (SIFT)) feature model to reduce the computational complexity in [74]. SIFT [75] is a scale invariant image feature represented by computing key-points and their direction in the image. The key-points are computed based on local neighborhood features. 16x16 neighborhood of each Key-point is divided into 4x4 blocks and 8-bin oriented histogram is computed for each of these blocks which then combine to generate 128 bin vector called key-point descriptor.

Key-point descriptors are shown as efficient scale invariant image feature.

7) FEATURE OPTIMIZATION AND CHOOSING THE BEST

As we combine different features together, the overall feature size increases and the computational time required to compute as well as classify will also be drastically increased. Most of the approaches use some sort of sampling of the feature descriptors to reduce the overall length. As the number of individual feature descriptors are increased, sampling will have less effect on overall detection performance. The sampling can vary from simply picking from a set, to applying a complex filtering as used in [66]. Another approach to reduce overall computational complexity is to chose the features which can be independently computed using a parallel processing hardware like GPU or Digital Signal Processors (DSP). Also the computational algorithms will be modified to utilize the parallel processing architecture.

We have discussed different standard feature descriptors as well as their combinations. It is difficult to tell which individual feature is the best; however, HOG is considered as one of the most useful feature for pedestrian detection. This is evident from the amount of literature describing the HOG and their combination. Another important point is the use of channel features. Channel features and their multi-scale pyramids are easy to compute with the help of a parallel processing hardware. Also instead of directly using the individual feature descriptors, filtering them with intelligent filters will help both in decorrelating the data as well as reducing the feature length. We can see that some features and methods perform best on some pedestrian test datasets where they fails on other datasets, which shows that, no feature can be treated as the ideal or best for all possible scenarios. The best one is yet to come!.

8) SELF-LEARNED FEATURES

As we have already discussed that no feature can be treated as *'The best'*, the alternate way is to use some technique to learn features from the data. This concept is used in Convolutional Neural Networks (CNN), where a set of convolutional filter kernels are learned from the training data. Each of these convolutional kernels are 2D or 3D convolutional filters. The filter coefficients are learned from the data during the training process. In a typical CNN, there could be multiple layers of convolutional filters, arranged sequentially. The filters in each layer is applied on the image progressively. These set of convolution filters will generate a set of feature vectors, which can best discriminate the input data provided during the training. These features are typically given to a fully connected Neural Network (NN) classifier. However, the convolutional features learned through the CNN training can also be used with other classifiers. We will further discuss NN while we discuss classifiers in section VI-C.1 and CNN while we discuss Deep learning based pedestrian detection in section VII

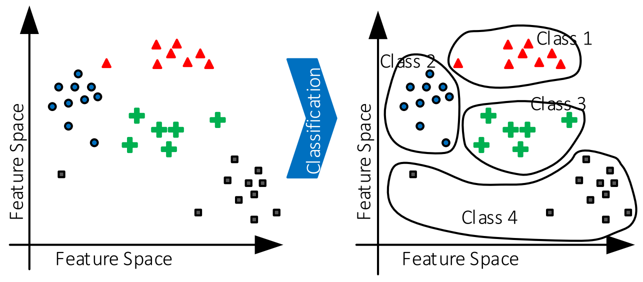

FIGURE 18. Classification of object feature vectors plotted on feature space.

C. MACHINE LEARNING IN PEDESTRIAN DETECTION

Machine Learning (ML) is the process of making a machine to perform intelligent tasks like classification. The system is learned to execute tasks which are typically handled by humans using their intelligence. Humans learn by examples. Same concept is applied on a machine, where the machine is provided with a vast set of examples and provided with special algorithms to learn the underlying similarities and differences, which will help them to group into different classes or clusters. In this section, we discuss some of the commonly used machine learning architectures and how they can be used for pedestrian detection.

Classification is the process of grouping data-points in an n-dimensional feature-space into a given set of m buckets or clusters or classes. By applying classification on a data point, we will assign a class label to it depending on its position in the feature-space. Figure 18 shows the classification process. Classifier is being addressed from the very start of clustering studies and there are plenty of research work happened on this area. A very simple classifier is a nearest-neighbor (NN) classifier, where each data point is attached to the class, which is nearest to it. There are different ways to measure the distance between a data point and a class - a simple way is to find the euclidean distance between the data point and each of the class centers. Typically, in object classifiers, every object is mapped to an n-dimensional feature space for a classification. For this, a feature is extracted from the raw object data. Various feature descriptors are explained in the previous section. NN classifier is generalized to k-NN, where a data point is assigned to k-neighboring classes with an attached probability value. Simple methods like this will be useful in cases where the data points can be easily clustered. However, for complex problems, like pedestrian detection, more advanced classification techniques are needed. In this section, we will discuss the different classifier approaches used in computer vision. There are many approaches to classification and thus a variety of classifiers. Table 2 shows a typical classification of the classification methods. An extensive study of various classifiers and their implementations are available in [76].

From the vast pool of classification approaches, some are relatively common to object classification in digital image processing. A detailed discussion on their architectures and implementation details are out of the scope of this survey,

TABLE 2. Classifier approaches.

Criteria	Classifier Type	Example
Training samples	Supervised	Maximum likelihood, Minimum distance, Artificial neural network, Decision tree classifier
	Unsupervised	ISODATA, K-means clustering algorithm
Parameters	Parametric	Maximum likelihood, Linear discriminant analysis
	Non-parametric	Artificial neural network, Decision tree classifier, Evidential reasoning, Support vector machine, Expert system
Pixel information	Subpixel	Fuzzy-set classifiers, Subpixel classifier, Spectral mixture analysis
	Per-pixel	Most of the classifiers like - Maximum likelihood, Minimum distance, Artificial neural network, Decision tree, and Support vector machine
	Per-field	GIS-based approaches
	Object-oriented	eCognition
Output	Hard	Most of the classifiers like - Maximum likelihood, Minimum distance, Artificial neural network, Decision tree, and Support vector machine
	Soft (fuzzy)	Fuzzy-set classifiers, Subpixel classifier, Spectral mixture analysis
Spatial information	Spectral	Maximum likelihood, Minimum distance, Artificial neural network
	Contextual	Iterated conditional modes, Point-to-point contextual correction, Frequency-based contextual classifier
	Spectral-contextual	ECHO, Combination of parametric or non-parametric and contextual algorithms

but we will give an overview of some of the commonly used techniques.

1) ARTIFICIAL NEURAL NETWORK (ANN)

ANNs [77] try to mimic how human brain works. Human brain contains a vast network of neurons and the information or memory is stored in their interconnections. When some trigger is received from the sensory organs, brain will try to pickup all the interconnected information from the neural networks based on the context. ANNs also work in a similar fashion. The basic element of ANN is an artificial Neuron (or Perceptron), which taken n inputs and produce

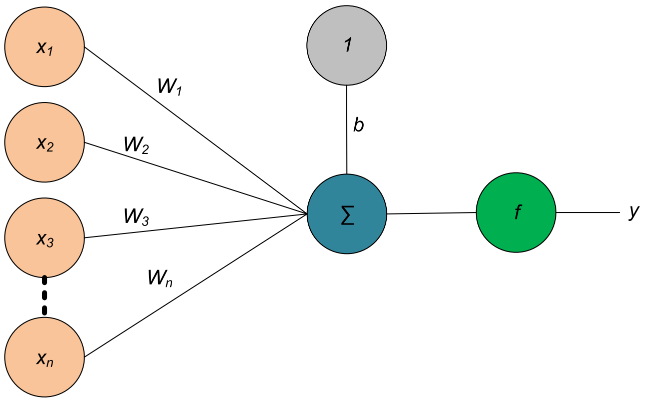

FIGURE 19. Perceptron - building block of ANN.

one output. The Neuron implements a transfer function, which is typically a weighted sum of its inputs $\sum_{i=1}^{n} w_i x_i$, where w_i is the weight assigned to the i^{th} input x_i. A bias b will be added to each perceptron and a nonlinear activation function is applied to the final value to control the output range. Figure 19 shows a pictorial representation of a perceptron. Typical activation functions used are - Sigmoid, Hyperbolic Tangent, and Rectified Linear Unit (ReLU). ReLU is the most common and simple activation function used. ReLU allows non-negative values pass through. The computation can be represented as $f(x) = max(0, x)$.

Considering the activation function as ReLU, the final output of a perceptron with n inputs can be computed as:

$$y = max(0, \sum_{i=1}^{n} w_i x_i + b)$$

where, b is the bias, x_i is the ith input and w_i is the corresponding weight. The output can be controlled by adjusting the weight w_i.

An interconnection of multiple such perceptrons, organized in different layers is called a Multilayer Perceptron(MLP) or Artificial Neural Network (ANN). Such networks can learn information by adjusting the interconnection weights during the training process. The first layer is called input layer, which will have the number of neurons corresponding to the feature vector size and the last layer will have one output for each of the classes to be identified. During the training process, the interconnection weights are adjusted so as to associate each of the training samples to their correct classes. The technique used to adjust the network weights to match the training sample output is called back propagation. The difference between predicted output and actual expected output is considered as error and the weight of the previous layer are adjusted to minimize this error. This then continues to each neuron in the previous layer until we reach the input layer. This is why the technique is called back propagation. The network can then be used to classify any test objects provided we have used enough number of representative training samples. Many variants of ANNs are in use. Figure 20 shows a typical ANN for a two-class classification with 1 input layer, 1 output layer and two hidden layers.

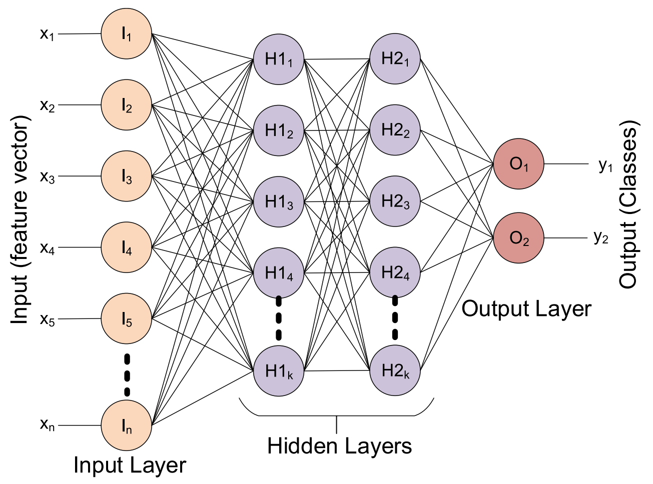

FIGURE 20. **Artificial neural network as object classifier.**

2) SUPPORT VECTOR MACHINE (SVM)

SVM [34] is one of the most common classifier used for pedestrian detection. SVM tries to map each of the object onto an n-dimensional feature space and tries to draw $(n-1)$-dimensional hyperplanes on it so as to separate the objects into different classes. To illustrate the idea, let us consider two heaps of mangoes and apples kept near to each other on a table. This can be mapped to a two dimensional feature-space considering their x and y positions and a line can be drawn between the two heaps so as to separate them into two classes. Let us say, whatever comes left to the line are apples and those on the right of the line are mangoes. The example is too simplistic and requiring a 2-dimensional feature space to correctly separate the classes where a practical object detection will require a high dimensional space. But the concept is same. In a basic SVM classifier, there will be n inputs $\{x_i\}_{i=1}^{n}$ and an n-dimensional weight vector $\{w_i\}_{i=1}^{n}$. The output of the classifier is computed as $\sum_{i=1}^{n} w_i x_i$. If the classifier needs to identify only two classes, the sign of the output will decide the same. SVM is also a learning classifier, where the weights w_i are adjusted to correctly classify the training samples. The hyperplane dimension may not be required as the same of feature size; in fact, that may be much higher than the feature size for correct classification. For a multi-class classification, there could be multiple outputs on SVM which will provide a probability of the input sample to be in each of the classes.

3) DECISION TREES (DT) AND BOOSTING

Decision trees are another approach towards classifier implementation. Here a set of week decision rules are defined along with a hierarchical sequence of application of these decision rules. The input is repeatedly refined at each stage and finally grouped into one of the output classes. Each decision rule is a simple weak rule considering the final classification problem; but helps in gradually refining the final decision by sequentially applying these simple rules. The rules are defined in a tree form and the leaves are final classes. Boosting is a type of classifier implementation applying this refinement concept. Here, a set of week classifiers are defined and are sequentially applied on input data by refining the classification at each stage. A very low complex classifier can be implemented at the first stage where the input count is more. Better and computationally intensive classifiers can be used in later stages, where the amount of data is getting reduced at each stage. Adaboost [43] is one of the most used boosting classifier in pedestrian detection. In the pedestrian classification section, we will discuss different variants of the boosting classifiers in pedestrian detection.

D. PEDESTRIAN DETECTOR TRAINING AND CLASSIFICATION

In this section, we discuss on different aspects of training a feature-based pedestrian detector and the process of classification.

1) TRAINING OF FEATURE-BASED PEDESTRIAN CLASSIFIER

As discussed in the Introduction, All pedestrian detection systems are learning based. The system will be trained using a set of training data, where images of pedestrians in different size, shape and orientation are fed to the system.

In classical feature-based approach, appropriate features are extracted from the training images and used them to train a classifier. Sufficient number of non-pedestrian images also will be used for training so that the classifier will be able to discriminate pedestrians and non-pedestrians once trained. A pedestrian classifier is typically a pattern classifier. The classifier architecture is independent of the type of object/pattern to be classified, but a trained classifier engine is specific to a particular problem and object type. The classifier architecture is selected based on the performance requirements as well as the feature descriptors used to uniquely identify the objects.

The three key elements in building a good classifier are 1) the choice of correct feature descriptor, 2) the use of correct dataset, and 3) the selection of a good classifier engine. Good feature selection helps to uniquely identify the object of interest (pedestrian in our case) from other objects in the scene. A good dataset provides the classifier with enough number of positive and negative samples which will help the classifier to formulate an engine which can correctly classify object of interest from other samples based on the feature vector values. The classifier is actually learning the classification by example during the training process. A good engine will optimally formulate the classifier function so as to provide a correct classification with minimum usage of computational power and storage.

From the generic framework of a pedestrian detection system, we can find that, in the core of a pedestrian detection system is a pedestrian classifier, which takes the feature descriptors extracted from the preprocessed image ROIs and identifies whether the image contains a pedestrian or not. Preprocessing is used to select the proper ROI for training as well as to enhance the ROI so as to improve the uniqueness of

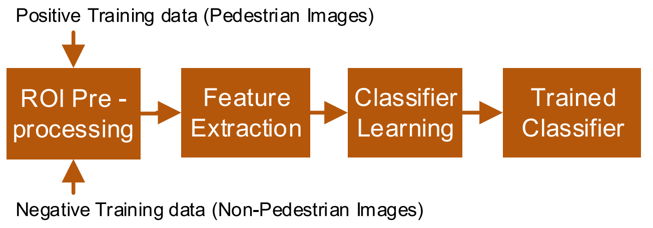
Positive Training data (Pedestrian Images)
ROI Pre - processing
Feature Extraction
Classifier Learning
Trained Classifier
Negative Training data (Non-Pedestrian Images)

FIGURE 21. Training of feature-based classifier for pedestrian detection.

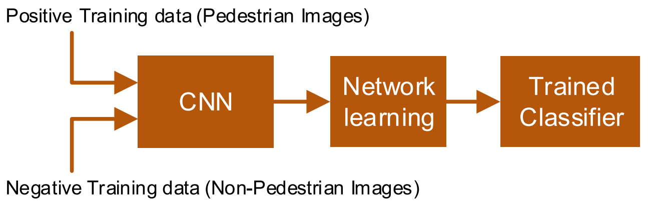
Positive Training data (Pedestrian Images)
CNN
Network learning
Trained Classifier
Negative Training data (Non-Pedestrian Images)

FIGURE 22. Training in deep learning with self-learned features for pedestrian detection.

the extracted features to represent the object of interest, This classifier is actually comparing the subject ROI against a set of known classes of objects in feature space. The set of these known classes are generated by training the classifier with sufficiently large set of images of all the known classes. The classifier training is an offline and iterative process, where, the classifier engine or the detector will be improved in each iteration. The sample data used for training should properly represent each of the classes to be identified. For a pedestrian detection system, we can view the classifier as a two-class system, where the classes are named as Pedestrian and Non-pedestrian. We can label a pedestrian sample as positive sample and a non-pedestrian sample as negative sample. There should be sufficient number of positive and negative samples to be used for training the classifier based on the uniqueness of the feature vector used.

Through training, the classifier learns to discriminate pedestrian and non-pedestrian image segments. Figure 21 shows the stages involved in training a classifier. Sufficient numbers of positive training vectors containing pedestrians in different orientation are given to the classifier, along with enough number of non-pedestrian images. These images will be preprocessed so as to normalize the intensity variations and to improve the detectability. If the feature descriptors used are not scale invariant, these images are scaled into uniform size before the features are extracted. Classifier will be typically trained with images of a representative size (after scaling the samples if required) depending on the detection requirements. Training may also go in multiple iterations by including false positives and false negatives obtained from testing as additional negatives and positives in next iteration. This may be repeated multiple times till the classifier gets saturated in detection performance.

The classifier architecture used in different solutions may be different. The selection of an appropriate classifier will depend on the type of feature used as well as the specific requirements of the solution. The availability of training dataset, the training convergence time, the classification time and classification accuracy are also counted in selecting the correct feature-classifier combination.

2) TRAINING A CNN WITH SELF-LEARNED FEATURES

In classifiers with self-learned features, feature extraction is not a separate stage. Optimum features were learned from the training dataset for best classification. The features here will mostly be simple convolution filter kernels but there will be many such kernels spread across multiple convolution layers in the CNN. The networks with many layers are typically called as deep networks and the machine learning approach involving deep networks are referred as Deep Learning (DL). The heart of a DL framework is a feature learning network like CNN. The DL framework learns best suitable features as convolution kernels to properly distinguish the training data set. Once the convolution-pooling layers are completed, the intermediate output - say features - are fed to a dense neural network with multiple fully-connected hidden layers.

Learning is done in CNN in a similar way to ANN. However, the computational requirement is much higher here. Back-propagation is used for network learning by adjusting the coefficients of convolution filters and the network weights. The difference between actual output and expected output is computed and the network parameters are adjusted progressively from the output layer to input layer (hence back-propagation). The underlying algorithm is called Gradient- Descent (GD). GD is computationally complex and required to pass through all training samples in a stretch to arrive at an optimized network parameter set. Generally we use a approximation of GD, the Stochastic Gradient Descent (SGD) in back propagation for deep networks, where sample dataset is huge. SGD allows to learn incrementally on each training sample. For each input (training sample), the actual output is computed in a forward pass from input to output layer. Output of each layer is formulated as a an n-dimensional vector with the layer parameters (like inter-connection weights, convolution filter coefficients etc.) as variables. This vector is adjusted by repeatedly adding an n-dimensional error vector, which is the difference between actual layer output and expected output. The best error vector is computed in the direction with minimum difference with the expected output. This is repeated until a stable minimum (local minimum) error is achieved. The value of parameters at this local minimum are taken as the learned parameters. This process is repeated for each training sample.

One problem with gradient descent algorithm is that the function may settle at a local minimum without reaching the global minimum, which will lead to less accurate network. To avoid this issue, a learning rate parameter is introduced. Higher the learning rate, faster the convergence but lesser accuracy. Similarly, the accuracy will be more at cost of slower convergence when the learning rate is lower. An optimum case will be dynamic learning rate where the learning rate will be high in the beginning but will reduce as the function converges to the global minimum.

Deep learning classifiers require more memory and computational power to get trained. Also as the features are self-learned, the dependency on proper and sufficient training data is more compared to feature classifier approach. However, the DL architecture is highly parallelizable and many standard DL architectures optimized to highly parallel GP-GPU platforms are now available free of cost for researchers to use. Users don't need to know the internals of these complex DL framework; they are encapsulated with easy to use interface. Users can experiment with different network architectures with a wide variety of tools and chose the best to match their requirement.

3) FEATURE-BASED PEDESTRIAN CLASSIFICATION

A trained classifier will learn its internal parameters to distinguish an input - image or a feature vector - belong to a particular class or not. In our case, to distinguish whether the object is a pedestrian or not. A classifier may be classifying only two classes - a class of interest and others - or a set of classes. If the number of classes a classifier detects is more than two, the classifier is called a multi-class classifier. In general, for every input, a classifier will provide a class membership probability for each output classes. The output class with maximum class-membership probability value will be selected and the object will be mapped accordingly.

Object detection is done by using a classifier engine to separate the image into different known classes based on the extracted feature descriptors. A classifier will typically find the best match for a given feature descriptor from a set of known feature descriptor classes. The classifier will compute a distance metric between the subject feature and the class-centers of known classes in the feature space. If there are multiple classes, a membership probability value will be assigned to the subject for each of those classes based on the distance metric. The subject will be assigned to the class with minimum distance in feature space, provided the distance is within an allowable threshold. If the class membership function is equal for different classes, the subject is either marked as not a member of any of these classes or specific decision will be taken depending on the target application.

Selection of proper classifier is also very important in implementing a pedestrian detection system. There are standard feature classifiers like Artificial Neural Network (ANN) [77] classifier, Support Vector Machine (SVM) [34], and boosting based decision tree classifiers like Adaboost [43]. Many researchers are working on improving the classification techniques or combining these techniques for faster convergence in classification. Weak feature cascades like Haar features are typically combined with boosting whereas HOG and its combinations are usually combined with SVM or its variants. The channel features also use boosting based classifiers. Neural network based classifier were used less where computational time needs to be minimum. Conventionally in pedestrian detection, SVM [34] and its variants are used for classification. There are variants of SVM to match different classification requirements. Linear SVM is low complex version with longer convergence time compared to a more complex non-linear SVM. Two-class SVM can be used for applications involving only one type of object and multi-class SVM is used where objects of more than two types need to be handled. There are many variations of SVM like histogram intersection kernel SVM (HikSVM) proposed by Maji *et al.* [78] and Latent SVM [48], which use latent information about the object like hierarchical data, pose information etc. Most of the original literature in pedestrian detection are based on HOG or its combinations combined with variants SVM detector.

Recent papers in pedestrian detection are using variants of boosting [79] based classifier cascades [80] more than SVM. Dollar *et al.* use feature pyramids [36], [37], [54], [60] combined with Adaboost classifier which is one of the best in the state of the art solutions for pedestrian detection in both detection accuracy as well as computational complexity. Saberian and Vasconcelos [81] proposed a new cascade boosting algorithm called fast cascade boosting (FCBoost), which generalizes the Adaboost as well as minimizes the Lagrangian risk so as to improve classification accuracy and speed. The authors showed that FCBoost outperforms current state-of-the-art methods in both detection accuracy and speed.

In general, in feature-classifier approach, boosting based adaptive decision trees provide better pedestrian classification. There are plenty of recent papers discussing boosting based decision trees as the primary classification approach in pedestrian detection. We have already discussed many improvements suggested in the boosting based classifiers while we discussed channel features. However, DL approaches outperform feature-classifier approaches in recent benchmarks.

VII. PEDESTRIAN DETECTION - DEEP LEARNING APPROACH

We have already discussed ANN and back propagation learning process. We also discussed self-learning of features with Convolutional Neural Network (CNN). The number of hidden layers in ANN defines its depth. As the number of hidden layers are more, the network will become more deep. Deep learning uses deep network, however, it is not just deep neural network; the structure of the network itself is different.

A. DEEP LEARNING NETWORKS

One of the most common method used in deep learning for visual images/videos is a Convolutional Neural Network (CNN) [64]. Interested readers can find an overview of DL techniques in [82]. CNN uses multiple convolution and pooling layers along with a dense fully connected network for classification. The input is not feature vector; but the image itself. A set of convolution kernels are learned by the system, by applying which we can extract the features which best represent properties of the training image set. We can also say

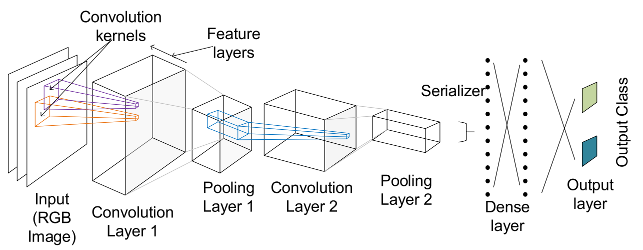

FIGURE 23. Convolution neural network as classifier.

that, a CNN is a variant of MLP with input layer constructed with convolution kernels.

Figure 23 shows a typical CNN construction for an object classification. The input is the image in RGB format. The features are different convolution kernels used at multiple convolution layers. Only the size of these kernels are defined initially. The actual kernel will be learned during training. In the CNN represented in figure 23, there are two convolution layers and two pooling layers each. In each convolution layer, a set of convolution kernels are used to do 2D/3D convolution on the input image and output of each of these convolution operations are combined. The convolution output may also be passed through some activation function like rectified linear unit (ReLU) to control the output value range. The next step is reduce the size/volume of the convolution layer output (called pooling). Size of the image volume is reduced by simple operations like taking only the maximum of non-overlapped 2D/3D blocks (called max-pooling). Other operations can also be used here to reduce the size of the image volume. Such convolution-pooling layer combinations may be repeated multiple times in a CNN for processing the input image.

Once the input processing is complete, the output image data is serialized and passed to a dense neural network called the dense layer. A dense layer may contain multiple hidden layers. The dense layer is then connected to the output layer where the output class is presented.

During the learning process, all the convolution kernels as well as weights in the dense layer are learned. The computational and memory requirement for CNN is much higher than ANN. However, due to the availability of highly parallelizable GP-GPU platforms, and availability of DL architectures optimized for them, recently there is an increased focus on DL for pedestrian detection. This is true to automotive domain also due to the increasing computational power. Recently, DL based methods outperform conventional feature-classifier based approaches in pedestrian detection benchmarks. Here in the remaining part of this section, we discuss on adaptation of CNNs and DL to automotive pedestrian detection.

CNN is not the only architecture used in DL. There are other commonly used complex architectures available for users to implement their learning problem. Each of these architectures differ in their network organization and complexity. Some of the common architectures are:

- AlexNet [83]
- VGG Net [84]
- GoogleNet [85]
- ResNet [86]
- ResNeXt [87]
- RCNN (Region Based CNN) [88]
- YOLO (You Only Look Once) [89]
- SqueezeNet [90]
- SegNet [91]
- GAN (Generative Adversarial Network) [92]

Many of these architectures can be freely downloaded and used with DL programming frameworks like TensorFlow, Caffe, CNTK, PyTorch, Keras, Deeplearning4j, Matlab Deep learning toolkit etc. to implement different DL solutions based on our requirement. We can either freshly train these networks for a new problem or can use the existing knowledge from the pre-trained models and additionally train for the new problem using transfer learning [93]. These pre-trained networks are not specific to pedestrian; but they are typically multi-object detectors where pedestrian could be one of the detectable object. These networks were pre-trained with a huge collection of training vectors, in such high volumes, which will be practically impossible for a fresh network as the training time required will be too long. By transfer learning, we can make use of their object detection capabilities and convert these generic frameworks to match the automotive pedestrian detection problem.

B. DEEP LEARNING IN PEDESTRIAN DETECTION

Now let us discuss some of the works in DL in pedestrian detection. Hosang *et al.* explains the inherent issues and improvement possibilities of CNN for pedestrian detection [94] by proposing Vanilla ConvNet as an example. Luo *et al.* improves CNN based pedestrian detection by adding new layers built with a new switchable restricted Boltzmann machine to form a Switchable Deep Network (SDN) [95] which automatically learns hierarchical features, salience maps, and mixture representations of different body parts.

Xiang *et al.* [96] and Zhang *et al.* [97] use region-based CNN (R-CNNs) for pedestrian detection. They use a Region Proposal Network (RPN) followed by boosted forests on shared, high-resolution convolutional feature maps. The authors show that R-CNNs produce detection performance comparable to state-of-the art. However, faster implementations [97] degrades the performance. Cai *et al.* proposed [98] a unified deep neural network called as multi-scale CNN (MS-CNN) for fast multi-scale object detection using CNN. They could show state of the art performance at lower computational complexity with 15fps detection speed on Caltech and KITTI dataset. Li *et al.* proposed a similar approach called Scale aware Fast R-CNN (SAF R-CNN) [99], which uses multiple networks to detect pedestrians at different scales and then combine them adaptively to generate the final score.

Du *et al.* proposed a Fused-Deep Neural Network (F-DNN) [100], improving both robustness and computational performance of DNN based pedestrian detection.

F-DNN uses a single-shot Deep CNN followed by multiple DNNs used in parallel for refinement of the detection. The outputs of these DNNs are fused using a soft-rejection based fusion method to compute the final detection confidence. Du *et al.* also proposed F-DNN-SS [100] by integrating pixel-wise semantic segmentation to F-DNN. Both these approaches outperform the state-of-the art pedestrian detection while tested on Caltech and INRIA datasets.

C. DEEPENING THE NETWORK TO IMPROVE PERFORMANCE

There is a trend in increasing the depth of network for improved performance. However, deep networks will become difficult to train as the number of hidden layers increase. He et al. propose, residual learning framework ResNet [86], where, Residual functions are used for efficient and fast learning of deeper networks with layer numbers in order of hundreds. Wu *et al.* [101] further improved the ResNet by proposing an ensemble of shallow networks instead of a single ultra-deep network for improved performance and computational efficiency. Also to reduce memory footprint required to train deeper networks, Bulò *et al.* [102] proposed a method by dropping some intermediate results, which could be recovered during the backward pass with the inversion of stored-forward results. They could show around 50% reduction in memory footprint with only less than 2% increase in computation time. In [103], Zhao *et al.* discuss modification on input layers of a deeper network by implementing pyramid pooling layer with pyramid scene parsing network (PSPNet). They use a global context fused with information from subregions for efficient segmentation. Zhuang *et al.* [104] also suggested improvements on the global context representation. They propose aggregating dense relation network (DRN) and context-restricted loss (CRL) to combine global context and local information.

D. PIXEL-LEVEL AND INSTANCE-LEVEL SEGMENTATION AND CITYSCAPES DATASET

Cityscape dataset provides both pixel-level and instance level semantic labeling and this lead to different methods on these two approaches. In pixel-level semantic segmentation, each pixel is mapped to a class and the segmented class will be represented as a set of pixels. Instance-level segmentation gives a fine segmentation by also separating each instances of an object of the same class.

1) PIXEL-LEVEL SEMANTIC LABELING

Semantic labeling deals with pixel mapping and normally discards the spatial relation between neighborhood pixels. In [105] Ke *et al.* proposes Adaptive Affinity Field (AAF) concept to capture the relation between neighborhood pixels during training to improve pixel-level semantic segmentation using PSPNet.

To improve inter-class distinction efficiency of deep networks, Yu *et al.* [106] propose Discriminative Feature network (DFN), where two subnets, a smooth one and a Border one are used to separately deal with the class and its boundary discrimination. This improves the detection and segmentation performance of fully convolutional networks.

Chen *et al.* proposed DeepLab [107] where, they improve the Deep CNNs by incorporating convolution with up-sampled filters (atrous convolution), atrous spatial pyramid pooling (ASPP), and improving object boundaries at output layer by using fully connected conditional random fields (CRF). Atrous convolution gives control at the resolution and field of view of the convolution process. ASPP allows robust segmentation at multiple scales. Authors also suggested improvements on convolution by employing atrous convolution in cascades and in parallel [108]. In [109] they further improved DeepLab by adding a decoder module to refine segmentation results across object boundaries. They also improved the framework by incorporating meta-learning techniques and recursive search space for dense image prediction [110].

2) INSTANCE-LEVEL SEGMENTATION

Instance level segmentation is aimed at segmentation and classification of every individual object in the scene. There are many diverse approaches for instance level segmentation with deep networks. The instance boundary represented with a bounding box may not be accurate which leads to partial instance segmentation. Hayder *et al.* [111] propose boundary-aware instance segmentation (BAIS) network, which incorporates an object mask network (OMN) to identify correct object boundary from a not-perfect bounding box. OMN refines the output of a deep network to generate perfect object representation. Arnab and Torr [112] propose a way to individually identify each instances of objects in a scene by combining the detector bounding box, semantic segmentation and the object shape information. They use an end-to end deep network with semantic and instance segmentation modules.

Liu *et al.* [113] proposes a sequential grouping of network (SGN) to address instance-level segmentation instead on standard CNN flow. SGN includes 3 subnets - breakpoint predicting CNN, a 2-layer fully connected LineNet, and class specific 3-layer MergerNet. Breakpoint prediction predicts object boundaries both in horizontal and vertical direction. These are then used to form line segments and connected components by LineNet. MergerNet extracts instances from the connected components. Liu *et al.* [114] proposes two parallel similar deep networks to predict pixel level semantic score and pixel affinities (the relation between two pixels of the same instance). The pixel affinity and semantic score are then combined using a graph merge algorithm to group pixels into instances.

He *et al.* [115] proposes a method called Mask R-CNN, by extending Faster R-CNN [88] by adding a mask prediction parallel to the available bounding box prediction. Achuna *et al.* [116] uses their automatic instance annotation method, PolygonRNN++ on results of a Faster R-CNN detector output for instance level segmentation. For this, they designed a new CNN encoder, reinforced learning technique and a graph Neural Network (GNN) to increase the output resolution. Liu *et al.* [117] integrated bottom up path augmentation to FPN [61] backbone of Faster R-CNN and combined with adaptive feature pooling for their path aggregation network (PANet). PANet shows the best instance level segmentation performance on COCO 2017 challenge and Cityscapes benchmarking.

Bai and Urtasun [118] integrates the watershed transform [119] within the deep network for improved instance level segmentation. In [120], Kendall *et al.* proposes an efficient way of simultaneous multi-class instance segmentation using deep network by using a multi-task loss function. They show that their approach can outperform a pool of individual classifiers trained on each class separately. Transfer learning is also getting more interest among researchers where learnings from one problem is transferred to another related problem for faster convergence. Li *et al.* [121] discusses several suggestions including an explicit inductive bias for improving transfer learning when a pre-trained coarse classifier is used and fine-tuned for a fine classification. There are many other approaches in semantic segmentation which we might have missed out, but we feel that readers will get an idea of the current trends from this paper.

VIII. SPECIAL CASES - OCCLUSIONS AND REPEATED DETECTIONS

We were discussing different methods to detect a pedestrian from a scene. Most of the features and process we discussed are for mostly visible pedestrians. But, in practical case, many of the pedestrians visible may be partially occluded or hidden behind other objects or pedestrians. Many of the approaches we have already discussed can be modified to include partially occluded pedestrians by adaptively training with partially occluded pedestrians as well as improving the classifier. However, there are some works [55], [122]–[125] particularly targeting this problem of occluded pedestrian detection. We also discussed CityPersons [26] dataset, which is having special focus to pedestrians occluded at different levels.

In a search-window based algorithm, there will be multiple overlapped ROI windows processed at different scales. There are chances that many of these ROI windows represent same object from the scene and are detected as different objects by the classifier. There should be a post processing stage after object classification to identify these multiple-detections of same object and selecting only the best to represent the detected object. This process is known as detection window clustering or non maximum suppression (NMS). Efficient detection window clustering will avoid multiple detection of same object without affecting the detection of different adjacent objects. In [126], we discussed various methods to find the best representative detection window without affecting the detection of nearby pedestrians. However, there are approaches utilizing the presence of multiple pedestrians and their mutual relationship to improve pedestrian detection as in [125] and [127]. In [49] Felzenszwalb *et al.* used a contextual feature based on normalized scores of multiple overlapped detections in an SVM classifier trained on Pascal dataset to re-score the detections. The best among the re-scored detection is selected. This method introduced further computational complexity but shows significant detection improvement.

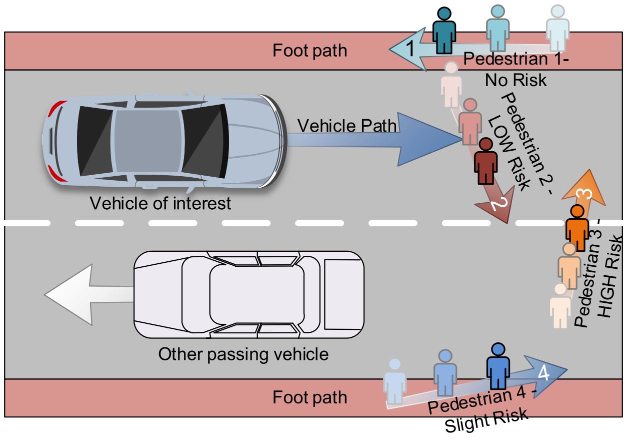

FIGURE 24. Tracking pedestrians and detection of collision risk.

IX. OBJECT LOCALIZATION AND TRACKING

The information generated by the core modules like pedestrian detection by processing the camera images are fed to the ADAS system, where different applications will analyze this data and take appropriate actions. The action may be simply providing an indication to the driver or even actively controlling the vehicle so as to avoid any critical situation.

Object detection based ADAS applications may need to identify the type of object and its current position with respect to the vehicle. Appropriate actions will be taken to analyze this and avoid any possible safety threat. Object tracking based applications will track these objects in successive video frames and compute the movement path of the object of interest. The motion paths of each objects detected by the object detection module will be closely analyzed and the potential future movement will be predicted from it. This information will be compared against the predicted motion path of the vehicle so as to identify the possibility of a hit. This analysis report will be provided to the ADAS application and the applications will then communicate the driver or automatically control the vehicle so as to avoid the casualty. Figure 24 illustrates tracking pedestrians and the associated risk of collision. Here 4 pedestrians are being tracked and two cases are found more risky. Pedestrian 2 is moving out of the vehicle path in a fast pace, but is still risky because if he stopped moving, there will be a collision. Pedestrian 3 is surely a high risk case because he is moving towards the vehicle path in a slow pace. Pedestrian 4 follows a slanted path and in case he decides to cross the road, it will lead to a collision and thus treated as a slightly risky case.

Object tracking is different from object detection. Object need not be recognized (like as a pedestrian) before tracking; any moving object can be tracked. If only one object is present, tracking will be straightforward. However, in a practical case where there could be multiple objects identified in a frame, they needs to be uniquely numbered and each of them needs to be tracked in successive frames. For this the objects in a frame should be matched to objects in next frame before tracking. When the objects are similar, like in the case of multiple pedestrians, the matching and tracking becomes more complex. ROIs corresponding to each of the pedestrian objects are extracted as templates from a frame and will be used to map the detections in the next frame. These ROI templates or some features generated from them can be used to find the corresponding object in the next frame. The techniques to match the objects vary from simple template matching through Mean Square Error (MSE), or fairly complex shape or contour tracking to application of a classifier like SVM.

Because of the importance of tracking in pedestrian detection, there is an increasing trend in combining them together. Methods of tracking by detection or detection by tracking are also attempted. K Okumo et al. use a mixture model that incorporates information from the dynamic models of each object and the detection hypotheses generated by Adaboost to detect and track hokey players in [128]. Wu and Nevatia [13] use human body part detection for detecting and tracking humans. In [129], Avidan used an ensemble of weak classifiers which then combined into a strong classifier using Adaboost where the peak detection of the strong classifier is used to track pedestrians in successive images. Leibe *et al.* [130] considered object detection and tracking estimation as a coupled optimization problem. They used an implicit object model and combined local features to detect and track humans. Andriluka *et al.* [19] improved this approach by combining advantages of detection and tracking in a single framework.

Object tracking itself is a wide area, with a huge collection of literature. The focus of this paper is pedestrian detection and a detailed study of tracking methods is out of the scope of this document. However, a rough outline of object tracking is given to highlight that, in ADAS, it is also equally important like object detection. Ragland and Tharcis provides details of different tracking approaches in their survey [131]. Interested readers can also refer recent paper by Rangesh and Trivedi [132] for more details on multi-object tracking in automotive scenario.

X. ALGORITHM EVALUATION

This section provides a general discussion and comparison of different pedestrian detection solutions proposed by the research community. We are mainly comparing the algorithm performance on two different datasets. Firstly, the Caltech dataset and secondly the Cityscapes dataset. Caltech

TABLE 3. Classifier-dataset combinations used in different solutions evaluated on Caltech dataset.

Algorithm	Features	Classifier	Training
ACF [37]	Channels	Adaboost	INRIA
AdaptFasterRCNN [26]	Pixels	DeepNet	Caltech+
AFS [135]	Multiple	LinearSVM	INRIA
AFS+Geo [135]	Multiple	LinearSVM	INRIA
CCF [63]	Deep	Adaboost	Caltech
CCF+CF [63]	Deep+Channels	Adaboost	Caltech
ChnFtrs [54]	Channels	Adaboost	INRIA
CompACT-Deep [67]	Multiple	Boosting	Caltech
ConvNet [65]	Pixels	DeepNet	INRIA
CrossTalk [35]	Channels	Adaboost	INRIA
DBN-Mut [125]	HOG	DeepNet	INRIA/Caltech
DeepParts [122]	Pixels	DeepNet	Caltech
F-DNN [100]	Pixels	DeepNet	Caltech+
F-DNN+SS [100]	Pixels	DeepNet	Caltech+
F-DNN2+SS [136]	Pixels	DeepNet	Caltech+
FasterRCNN+ATT [137]	Pixels	DeepNet	Caltech+
FeatSynth [138]	multiple	linear SVM	INRIA
FisherBoost [80]	HOG+COV	FisherBoost	INRIA
FPDW [36]	Channels	Adaboost	INRIA
FtrMine [139]	Channels	Adaboost	INRIA
GDFL [140]	Pixels	DeepNet	Caltech+
HikSvm [78]	HOG	HikSVM	INRIA
HOG [9]	HOG	LinearSVM	INRIA
HogLbp [53]	HOG+LBP	LinearSVM	INRIA
InformedHaar [57]	Channels	Adaboost	INRIA/Caltech
JointDeep [124]	Color+Gradient	DeepNet	INRIA/Caltech
Katamari [58]	Channels	Adaboost	INRIA/Caltech
LatSvm-V1 [48]	HOG	LatentSVM	PASCAL
LatSvm-V2 [49]	HOG	LatentSVM	INRIA
LDCF++ [141]	Channels	Adaboost	Caltech
MLS [142]	HOG	Adaboost	INRIA
MS-CNN [98]	Pixels	DeepNet	Caltech+ImageNet
MultiFtr (MF) [68]	Multiple	Adaboost	INRIA
MF+CSS [69]	Multiple	LinearSVM	TUD-Motion
MF+Motion [69]	Multiple	LinearSVM	TUD-Motion
MF+Motion+2Ped [127]	Multiple	LinearSVM	TUD-Motion
NAMC [143]	Channels	Adaboost	INRIA/Caltech
pAUCBoost [144]	HOG+COV	pAUCBoost	INRIA
PCN [145]	Pixels	DeepNet	Caltech+ImageNet
Pls [146]	Multiple	PLS+QDA	INRIA
PosInv [72]	HOG	Adaboost	INRIA+
RandForest [147]	HOG+LBP	RandomForest	INRIA/Caltech
Roerei [59]	Channels	Adaboost	INRIA
RPN+BF [97]	Pixels	DeepNet+Adaboost	Caltech+ImageNet
SA-FastRCNN [99]	Pixels	DeepNet	Caltech+ImageNet
SCCPriors [148]	Channels	Adaboost	INRIA/Caltech
SDN [95]	Pixels	DeepNet	INRIA/Caltech
SDS-RCNN [149]	Pixels	DeepNet	Caltech+ImageNet
Shapelet [47]	Gradients	Adaboost	INRIA
SketchTokens [56]	Channels	Adaboost	INRIA+
SpatialPooling [150]	Multiple	pAUCBoost	INRIA/Caltech
SpatialPooling+ [151]	Multiple	pAUCBoost	Caltech
TA-CNN [123]	Pixels	DeepNet	Caltech++
TLL-TFA [152]	Pixels	DeepNet	Caltech++
UDN+ [153]	Pixels	DeepNet	Caltech+ImageNet
VJ [42]	Haar	Adaboost	INRIA
WordChannels [154]	WordChannels	Adaboost	INRIA/Caltech

dataset is generally used for all types of algorithms, whereas, Cityscapes are primarily used by DL based algorithms. We have used the reference code provided by Dollar in his website [133] for performance comparison of different pedestrian detection solutions on Caltech dataset whereas data available on Cityscapes website is used for performance evaluation of algorithms on it. Interested readers can visit website of department of Computational Vision, California Institute of Technology (Caltech) [134] to get more details on the Caltech benchmarking suite. The comparison code was run on an Intel core-i5 PC with NVIDIA GeForce 940 MX GPU card to measure the execution time. The execution time needs to be treated as relative to compare the algorithm performance. Absolute values may slightly vary system to system. We are providing a snapshot of the benchmarking results to make this document as a single point reference.

A. DATASETS AND ALGORITHMS FOR PERFORMANCE EVALUATION

Table 3 lists different algorithms used on Caltech dataset for performance comparison and the combination of feature descriptor, classifier and training dataset used to generate the results. We can see that Caltech dataset is preferred by many algorithm. Cityscapes and CityPersons datasets are relative new and thus used by less number of researchers. Channel filters and their variants are most preferred by researchers mainly because of the ease of parallelizing the execution using GPUs. For DL based methods, image pixels is directly fed to the system as the features are learned internally. We have already discussed the importance of these in previous sections. We can also see that boosting based decision trees are the most preferred and most successful classifier of choice in recent studies for feature classifier approaches. SVM and its variants are also still getting sufficient attention.

We have also captured the best performing DL based algorithms on Cityscapes [25] dataset. Both pixel-based as well as instance based approaches are considered. Top 10 methods with proper reference available as of the time of preparation of this paper are selected for illustration. Table 4 gives a listing of the algorithms.

B. EVALUATION METRICS

Some of the common metrics used for benchmarking the performance of pedestrian detection solutions (also applicable for any object detection problem) are listed in Table 5, which is self explanatory. The typical metrics used on Caltech dataset are *Precision*, *Recall*, *FPPI*, and *MissRate*, whereas the metrics used in Cityscapes datasets are *IoU* and *AP*.

None of these metrics provide a perfect representation of performance of a pedestrian detection algorithm. However, they help us in relatively ranking the algorithms.

C. EVALUATION RESULTS

On Caltech dataset, we have used both $PR-curve$ and $Miss-rate$ vs $FPPI$ curve to represent top 15 algorithms.

TABLE 4. Deep learning algorithms used for performance evaluation on Cityscapes dataset.

Pixel-level	Instance-level
DPC [110]	PANet [117]
DRN-CRL-Coarse [104]	Mask R-CNN [115]
DeepLabv3+ [109]	Mask R-CNN [fine] [115]
DeepLabv3 [108]	PolygonRNN++ [116]
PSPNet [103]	GMIS [117]
ResNet-38 [101]	SGN [113]
BatchNorm [102]	PIS-DIN [112]
L2-SP [121]	Multi-task Learning [120]
DFN [106]	Dwatershed [118]
AAF on PSPNet [105]	BaIS [111]

TABLE 5. Common acronyms and metrics used for evaluating pedestrian detectors.

Metric	Notation	Meaning
Bounding Box	BB	Rectangular Object ROI
Overlap Ratio	OR	$OR = \frac{Area(BB_{dt} \cap BB_{gt})}{Area(BB_{dt} \cup BB_{gt})}$ $(gt = GroundTruth, dt = Detection)$
Ground truth data	GT	Set of BBs and details of all available objects in dataset
True Positives	T_P	Number of objects correctly detected
True Negatives	T_N	Number of non-objects correctly rejected
False Positives	F_P	Number of non-objects wrongly detected as objects of interest
False Negatives	F_N	Number of objects wrongly rejected
False Positives Per Window	$FPPW$	$FPPW = \frac{F_p}{T_p+T_n+F_p+F_n}$
False Positives Per Image	$FPPI$	$FPPI = \frac{F_p}{N_{images}}$
Precision	P	$P = \frac{T_p}{(T_p+F_p)}$
Recall	R	$R = \frac{T_p}{(T_p+F_n)}$
Miss Rate	MR	$MR = 1 - \frac{T_p}{(T_p+F_n)}$
Log Average Miss rate	AMR	Average miss rate of 9 FPPI values evenly spaced in Log space
Intersection over Union	IoU	$IoU = \frac{T_P}{(T_P+F_P+F_N)}$

Results of Viola Johns Haar features (VJ) and Histogram of Oriented Gradients (HOG) are also given for an idea on how far we have improved from the beginning. Figure 25 shows the *Missrate* vs *FPPI* performance of top 15 algorithms whereas figure 26 shows the top 15 when we compare the *Precision* vs *Recall*. We can see that the algorithms comes in top on these two metrics are different. This also affirms that no metric can be universally recommended as the best one. However, *Missrate* vs *FPPI* plot is more preferred against $PR-curve$.

We have also compared the algorithms on Caltech for their execution speed. Figure 27 shows the 15 fastest algorithms on

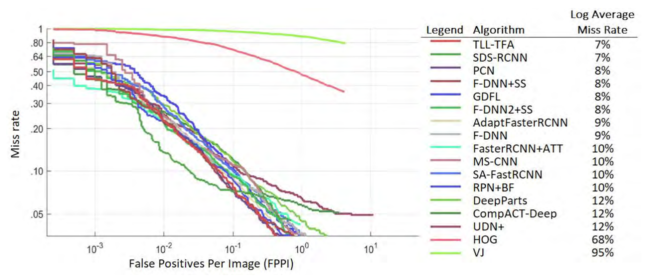

FIGURE 25. Miss Rate vs FPPI Plot of best 15 algorithms with VJ and HOG on Caltech USA dataset.

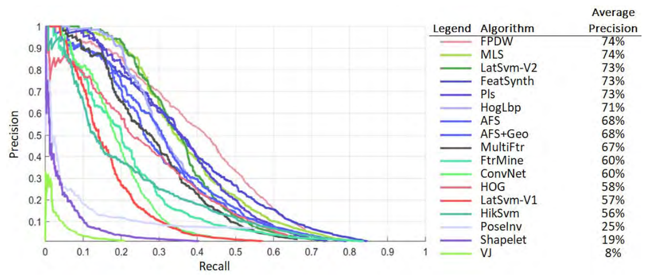

FIGURE 26. PR Curve for best 15 algorithms with VJ and HOG on Caltech USA dataset.

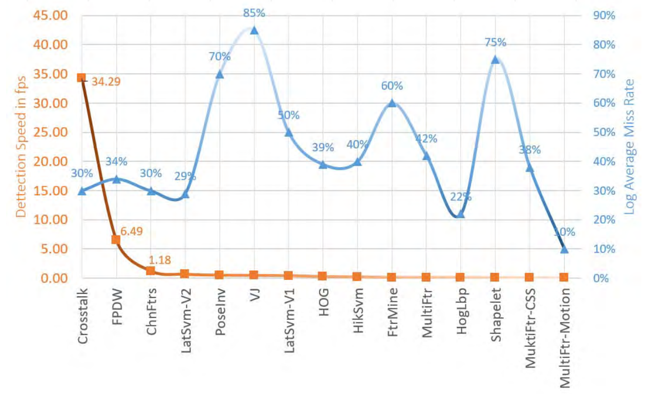

FIGURE 27. Execution speed and miss rate of 15 fastest algorithms on Caltech USA dataset.

Caltech evaluation. Log-average miss rate of these algorithms are also plotted on the same graph for a better evaluation. We can see that even though the detection performance is increasing, the computational requirements and execution speed is not yet to the expected levels. Only two algorithms are showing reasonably good execution speed. Crosstalk is clearly dominating the evaluated algorithms in execution speed (with 34 frame per second execution) and 30% log-average miss rate (AMR).

Cityscapes dataset is specially designed for deep learning based object detection. Cityscapes provides benchmark performance on Instance level as well as pixel level semantic labeling. For pixel level semantic labeling performance comparison, they use PASCAL VOC intersection-over-union (IoU) [31] as performance metric and for instance level semantic labeling, they use Average Precision (AP) [155] as performance metric.

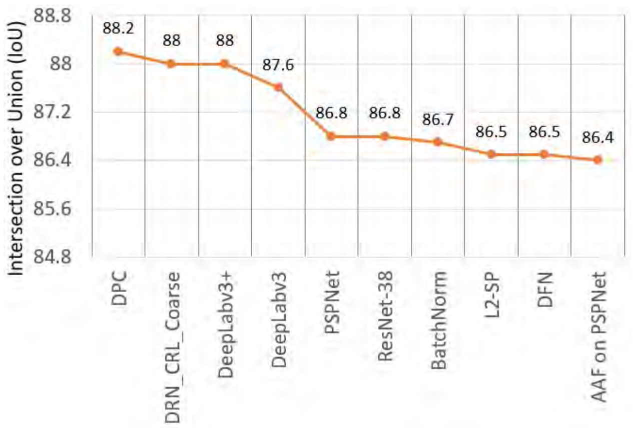

FIGURE 28. Pixel-level semantic labeling performance (IoU) of top 10 algorithms on Cityscapes dataset.

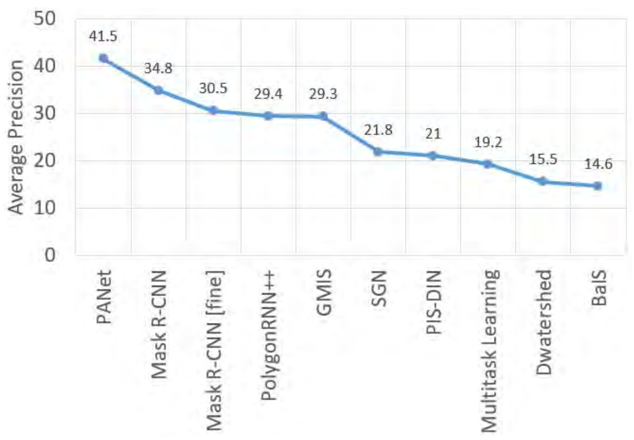

FIGURE 29. Instance-level semantic labeling performance (AP) of top 10 algorithms on Cityscapes dataset.

We include the pedestrian detection performance of top 10 algorithms (with accessible reference) performing best on both pixel-level and instance-level semantic labeling on Cityscapes dataset in figure 28 and figure 29 respectively. Interested users can visit Cityscapes website [25] for a detailed performance benchmarking of various algorithms on their dataset.

From the analysis, we can see that the algorithm performing best on one dataset may not perform well on another dataset. This shows that the solutions proposed are far from a generic ideal solution. The performance of algorithms depend heavily on the dataset used to train the detector as well as the dataset on which the detector is tested.

XI. RESEARCH TRENDS AND OPPORTUNITIES

We have discussed some of the recent advances in pedestrian detection problem by addressing different stages of the process separately. Deep learning based pedestrian detection is getting more focus recently against the classical-feature classifier approaches. Many techniques in feature-classifier approach are now getting applied into DL networks for improved performance. Availability of highly parallelizable platforms like multi-core GP-GPU hardware and NVIDIA CuDNN like software libraries help in efficient implementation of DL based object detection solutions efficiently. However, the feature-classifier approach is still in focus and improvements are being suggested by researchers. The general trends we can find in feature-classifier approach can be summarized as 1) use preprocessing techniques so as to reduce the search space and improve the feature extraction, 2) combine best known features to form stronger features, 3) make use of feature pyramids as possible to reduce computational load, 4) chose multi-stage boosting with a bundle of weak classifier instead of single strong classifier, and 5) chose features and classifiers which can efficiently run on parallel hardware platforms. In DL, the focus is on optimizing the convergence time and improving the performance, computational complexity and memory requirements of the network. Even with all the advances in object detection in recent years, fail-safe pedestrian detection that will work on all weather conditions and road conditions is still far from reality.

XII. CONCLUSION

Pedestrian detection is one of the hot topics in ADAS. Researchers are working to develop a robust real-time solution with high detection rate and minimum false detections particularly in harsh environment. However, the detection performance as well as the computational complexity of available solutions is far behind the expectation from automotive industry, especially when the automotive industry is moving towards the driver-less cars. This paper is an attempt to understand the techniques used in pedestrian detection particularly in automotive domain. We conclude this discussion by highlighting that there is high demand for low-cost and robust solutions. There is high potential of research in this area. The solutions that can solve the pedestrian detection issues will also find application in other object detection and tracking problems across the digital image processing domain.

ACKNOWLEDGMENT

The authors would like to thank the authorities at Bharathiar University, Central University of Kerala, and Tata Elxsi for the support extended during the preparation of this paper.

REFERENCES

[1] J. P. Thalen. (Aug. 2006). *ADAS for the Car of the Future*. [Online]. Available: http://essay.utwente.nl/58373/

[2] K. Bartolomeos *et al.*, "Pedestrian safety: A road safety manual for decision-makers and practitioners," World Health Org., Geneva, Switzerland, Tech. Rep., 2013.

[3] B. J. Czerny, J. D'Ambrosio, R. Debouk, and K. Stashko, "ISO 26262 functional safety draft international standard for road vehicles: Background, status, and overview," in *Proc. 28th Int. Syst. Saf. Conf.*, Minneapolis, MN, USA, Aug./Sep. 2010. [Online]. Available: http://www.sesamo-project.eu/content/iso-26262-functional-safety-draft-international-standard-road-vehicles-background-status-and

[4] R. Salay, R. Queiroz, and K. Czarnecki. (2017). "An analysis of ISO 26262: Using machine learning safely in automotive software." [Online]. Available: https://arxiv.org/abs/1709.02435

[5] B. Spanfelner, D. Richter, S. Ebel, U. Wilhelm, and C. Patz, "Challenges in applying the iso 26262 for driver assistance systems," in *Proc. 5th Conf. Driver Assistance Focus Netw.*, Munich, Germany, no. 16, 2012, pp. 28-1–28-3.

[6] R. M. Dufour, E. L. Miller, and N. P. Galatsanos, "Template matching based object recognition with unknown geometric parameters," *IEEE Trans. Image Process.*, vol. 11, no. 12, pp. 1385–1396, Dec. 2002.

[7] C. Papageorgiou and T. Poggio, "A trainable system for object detection," *Int. J. Comput. Vis.*, vol. 38, no. 1, pp. 15–33, 2000.

[8] C. G. Keller, M. Enzweiler, M. Rohrbach, D. F. Llorca, C. Schnorr, and D. M. Gavrila, "The benefits of dense stereo for pedestrian detection," *IEEE Trans. Intell. Transp. Syst.*, vol. 12, no. 2, pp. 1096–1106, Dec. 2011.

[9] N. Dalal and B. Triggs, "Histograms of oriented gradients for human detection," in *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit. (CVPR)*, vol. 1, Jun. 2005, pp. 886–893.

[10] Z. Kira, R. Hadsell, G. Salgian, and S. Samarasekera, "Long-range pedestrian detection using stereo and a cascade of convolutional network classifiers," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, Oct. 2012, pp. 2396–2403.

[11] D. Olmeda, C. Premebida, U. Nunes, J. Armingol, and A. de la Escalera, "Pedestrian detection in far infrared images," *Integr. Comput.-Aided Eng.*, vol. 20, pp. 347–360, Aug. 2013.

[12] M. Szarvas, U. U. Sakai, and J. Ogata, "Real-time pedestrian detection using LIDAR and convolutional neural networks," in *Proc. IEEE Symp. Intell. Vehicles*, Jun. 2006, pp. 213–218.

[13] B. Wu and R. Nevatia, "Detection of multiple, partially occluded humans in a single image by Bayesian combination of edgelet part detectors," in *Proc. 10th IEEE Int. Conf. Comput. Vis. (ICCV)*, vol. 1, Oct. 2005, pp. 90–97.

[14] J. W. Davis and M. A. Keck, "A two-stage template approach to person detection in thermal imagery," in *Proc. IEEE Workshops Appl. Comput. Vis. (WACV)*, vol. 1, Jan. 2005, pp. 364–369.

[15] S. Munder and D. M. Gavrila, "An experimental study on pedestrian classification," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 28, no. 11, pp. 1863–1868, Nov. 2006.

[16] B. Wu and R. Nevatia, "Cluster boosted tree classifier for multi-view, multi-pose object detection," in *Proc. 11th IEEE Int. Conf. Comput. Vis. (ICCV)*, Oct. 2007, pp. 1–8.

[17] D. Geronimo, A. Sappa, A. López, and D. Ponsa, "Adaptive image sampling and windows classification for on-board pedestrian detection," in *Proc. 5th Int. Conf. Comput. Vis. Syst. (ICVS)*, vol. 39, 2007, pp. 1–10.

[18] A. Ess, B. Leibe, and L. Van Gool, "Depth and appearance for mobile scene analysis," in *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, Oct. 2007, pp. 1–8.

[19] M. Andriluka, S. Roth, and B. Schiele, "People-tracking-by-detection and people-detection-by-tracking," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2008, pp. 1–8.

[20] D. Overett, L. Petersson, N. Brewer, L. Andersson, and N. Pettersson, "A new pedestrian dataset for supervised learning," in *Proc. IEEE Symp. Intell. Vehicles*, Jun. 2008, pp. 373–378.

[21] G. J. Brostow, J. Fauqueur, and R. Cipolla, "Semantic object classes in video: A high-definition ground truth database," *Pattern Recognit. Lett.*, vol. 30, no. 2, pp. 88–97, 2009.

[22] P. Dollár, C. Wojek, B. Schiele, and P. Perona, "Pedestrian detection: A benchmark," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2009, pp. 304–311.

[23] M. Enzweiler and D. M. Gavrila, "Monocular pedestrian detection: Survey and experiments," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 31, no. 12, pp. 2179–2195, Dec. 2009.

[24] A. Geiger, P. Lenz, and R. Urtasun, "Are we ready for autonomous driving? The KITTI vision benchmark suite," in *Proc. Int. Conf. Pattern Recognit. (CVPR)*, Jun. 2012, pp. 3354–3361.

[25] M. Cordts *et al.*, "The cityscapes dataset for semantic urban scene understanding," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, vol. 20, Jun. 2016, pp. 3213–3223.

[26] S. Zhang, R. Benenson, and B. Schiele. (2017). "CityPersons: A diverse dataset for pedestrian detection." [Online]. Available: https://arxiv.org/abs/1702.05693

[27] S. R. Richter, Z. Hayder, and V. Koltun, "Playing for benchmarks," in *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, vol. 20, Oct. 2017, pp. 2232–2241.

[28] P. Dollár, C. Wojek, B. Schiele, and P. Perona, "Pedestrian detection: An evaluation of the state of the art," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 34, no. 4, pp. 743–761, Apr. 2012.
[29] T.-Y. Lin *et al.*, "Microsoft COCO: Common objects in context," in *Proc. Eur. Conf. Comput. Vis. (ECCV)*, 2014, pp. 740–755.
[30] R. Mottaghi *et al.*, "The role of context for object detection and semantic segmentation in the wild," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2014, pp. 891–898.
[31] M. Everingham, S. M. A. Eslami, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman, "The Pascal visual object classes challenge: A retrospective," *Int. J. Comput. Vis.*, vol. 111, no. 1, pp. 98–136, Jan. 2015.
[32] O. Russakovsky *et al.*, "ImageNet large scale visual recognition challenge," *Int. J. Comput. Vis.*, vol. 115, no. 3, pp. 211–252, Dec. 2015.
[33] R. Trichet and F. Bremond, "Dataset optimization for real-time pedestrian detection," *IEEE Access*, vol. 6, pp. 7719–7727, 2018.
[34] C.-W. Hsu, C.-C. Chang, and C.-J. Lin, "A practical guide to support vector classification," Dept. Comput. Sci. Inf. Eng., Univ. Nat. Taiwan, Taipei, Taiwan, Tech. Rep., 2010, pp. 1–16.
[35] P. Dollár, R. Appel, and W. Kienzle, "Crosstalk cascades for frame-rate pedestrian detection," in *Proc. Eur. Conf. Comput. Vis. (ECCV)*, in Lecture Notes in Computer Science, vol. 7573, A. Fitzgibbon, S. Lazebnik, P. Perona, Y. Sato, and C. Schmid, Eds. Berlin, Germany: Springer, 2012, pp. 645–659.
[36] P. Dollar, S. Belongie, and P. Perona, "The fastest pedestrian detector in the west," in *Proc. Brit. Mach. Vis. Conf. (BMVC)*, 2010, pp. 1–11.
[37] P. Dollár, R. Appel, S. Belongie, and P. Perona, "Fast feature pyramids for object detection," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 36, no. 8, pp. 1532–1545, Aug. 2014.
[38] N. K. Ragesh and R. Rajesh, "Speeding up of pedestrian detection by smart ROI," in *Proc. Int. Symp. Res. Innov. Qual. Improvement Higher Edu.*, 2014, pp. 1–9.
[39] P. V. Viswajith, N. K. Ragesh, and S. N. Madhu, "ACM based ROI extraction for pedestrian detection with partial occlusion handling," *Procedia Comput. Sci.*, vol. 46, pp. 45–52, Jan. 2015.
[40] T. F. Chan and L. A. Vese, "Active contours without edges," *IEEE Trans. Image Process.*, vol. 10, no. 2, pp. 266–277, Feb. 2001.
[41] D. Geronimo, A. M. Lopez, A. D. Sappa, and T. Graf, "Survey of pedestrian detection for advanced driver assistance systems," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 32, no. 7, pp. 1239–1258, Jul. 2010.
[42] P. Viola, M. J. Jones, and D. Snow, "Detecting pedestrians using patterns of motion and appearance," in *Proc. 9th IEEE Int. Conf. Comput. Vis. (CVPR)*, vol. 2, Oct. 2003, pp. 734–741.
[43] J. D. Novakovic and A. Veljovic, "AdaBoost as classifier ensemble in classification problems," in *Proc. Infoteh-Jahorina*, Mar. 2014, pp. 616–620.
[44] R. M. Badia, Y. Etsion, S. Girbal, A. Portero, and M. Lujan, "D2.1 report on the reference set of applications chosen, and initial characterization of the applications," TERAFLUX, Univ. Siena, Siena, Italy, Tech. Rep. #D2.1, Project#249013, 2010. [Online]. Available: http://www.teraflux.eu/
[45] OpenCV. (1999). *OpenCV Library: Open Source Computer Vision Library*. [Online]. Available: http://www.willowgarage.com/pages/software/opencv
[46] M. M. Trompouki, L. Kosmidis, and N. Navarro, "An open benchmark implementation for multi-CPU multi-GPU pedestrian detection in automotive systems," in *Proc. 36th Int. Conf. Comput.-Aided Design (ICCAD)*, Nov. 2017, pp. 305–312.
[47] P. Sabzmeydani and G. Mori, "Detecting pedestrians by learning shapelet features," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2007, pp. 1–8.
[48] P. Felzenszwalb, D. McAllester, and D. Ramanan, "A discriminatively trained, multiscale, deformable part model," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2008, pp. 1–8.
[49] P. F. Felzenszwalb, R. B. Girshick, D. McAllester, and D. Ramanan, "Object detection with discriminatively trained part-based models," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 32, no. 9, pp. 1627–1645, Sep. 2010.
[50] Q. Zhu, M.-C. Yeh, K.-T. Cheng, and S. Avidan, "Fast human detection using a cascade of histograms of oriented gradients," in *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit. (CVPR)*, vol. 2, Jun. 2006, pp. 1491–1498.
[51] T. Watanabe, S. Ito, and K. Yokoi, "Co-occurrence histograms of oriented gradients for pedestrian detection," in *Advances in Image and Video Technology* (Lecture Notes in Computer Science), vol. 5414, T. Wada, F. Huang, and S. Lin, Eds. Heidelberg, Germany: Springer, 2009, pp. 37–47.
[52] Y. Nan, C. Liang, W. Fang, X. Wang, Y. Yang, and J. Chen, "Histograms of salience for pedestrian detection," in *Proc. Int. Conf. Internet Multimedia Comput. Service (ICIMCS)*, 2014, pp. 275–278.
[53] X. Wang, T. X. Han, and S. Yan, "An HOG-LBP human detector with partial occlusion handling," in *Proc. IEEE 12th Int. Conf. Comput. Vis. (ICCV)*, Sep. 2009, pp. 32–39.
[54] P. Dollar, Z. Tu, P. Perona, and S. Belongie, "Integral channel features," in *Proc. Brit. Mach. Vis. Conf. (BMVC)*, 2009, pp. 91.1–91.11.
[55] M. Mathias, R. Benenson, R. Timofte, and L. V. Gool, "Handling occlusions with franken-classifiers," in *Proc. Int. Conf. Comput. Vis. (ICCV)*, Dec. 2013, pp. 1505–1512.
[56] J. J. Lim, C. L. Zitnick, and P. Dollár, "Sketch tokens: A learned mid-level representation for contour and object detection," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2013, pp. 3158–3165.
[57] S. Zhang, C. Bauckhage, and A. B. Cremers, "Informed Haar-like features improve pedestrian detection," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2014, pp. 947–954.
[58] R. Benenson, M. Omran, J. Hosang, and B. Schiele, "Ten years of pedestrian detection, what have we learned?" in *Proc. Eur. Conf. Comput. Vis. (ECCV) Workshops*. Cham, Switzerland: Springer, 2014, pp. 613–627.
[59] R. Benenson, M. Mathias, T. Tuytelaars, and L. Van Gool, "Seeking the strongest rigid detector," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2013, pp. 3666–3673.
[60] W. Nam, P. Dollár, and J. H. Han, "Local decorrelation for improved pedestrian detection," in *Advances in Neural Information Processing Systems*, Z. Ghahramani, M. Welling, C. Cortes, N. Lawrence, and K. Weinberger, Eds. Red Hook, NY, USA: Curran Associates, 2014, pp. 424–432.
[61] T.-Y. Lin, P. Dollár, R. Girshick, K. He, B. Hariharan, and S. Belongie, "Feature pyramid networks for object detection," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jul. 2017, pp. 936–944.
[62] R. N. Rajaram, E. Ohn-Bar, and M. M. Trivedi, "Looking at pedestrians at different scales: A multiresolution approach and evaluations," *IEEE Trans. Intell. Transp. Syst.*, vol. 17, no. 12, pp. 3565–3576, Dec. 2016.
[63] B. Yang, J. Yan, Z. Lei, and S. Z. Li, "Convolutional channel features," in *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, Dec. 2015, pp. 82–90.
[64] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition,"*Proc. IEEE*, vol. 86, no. 11, pp. 2278–2324, Nov. 1998.
[65] P. Sermanet, K. Kavukcuoglu, S. Chintala, and Y. LeCun, "Pedestrian detection with unsupervised multi-stage feature learning," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2012, pp. 3626–3633.
[66] S. Zhang, R. Benenson, and B. Schiele, "Filtered channel features for pedestrian detection," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2015, pp. 1751–1760.
[67] Z. Cai, M. J. Saberian, and N. Vasconcelos, "Learning complexity-aware cascades for deep pedestrian detection," in *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, Dec. 2015, pp. 3361–3369.
[68] C. Wojek and B. Schiele, "A performance evaluation of single and multi-feature people detection," in *Pattern Recognition* (Lecture Notes in Computer Science), vol. 5096, G. Rigoll, Ed. Heidelberg, Germany: Springer, 2008, pp. 82–91.
[69] S. Walk, N. Majer, K. Schindler, and B. Schiele, "New features and insights for pedestrian detection," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2010, pp. 1030–1037.
[70] B. Wu and R. Nevatia, "Optimizing discrimination-efficiency tradeoff in integrating heterogeneous local features for object detection," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2008, pp. 1–8.
[71] A. Mohan, C. Papageorgiou, and T. Poggio, "Example-based object detection in images by components," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 23, no. 4, pp. 349–361, Apr. 2001.
[72] Z. Lin and L. S. Davis, "A pose-invariant descriptor for human detection and segmentation," in *Computer Vision—ECCV* (Lecture Notes in Computer Science), vol. 5305, D. Forsyth, P. Torr, and A. Zisserman, Eds. Berlin, Germany: Springer, 2008, pp. 423–436.

[73] M. Enzweiler and D. M. Gavrila, "Integrated pedestrian classification and orientation estimation," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2010, pp. 982–989.
[74] C. Liu, X. Fang, X. Wang, and S. Gong, "Pedestrian recognition in aerial video using saliency and multi-features," *Int. J. Comput. Elect. Eng.*, vol. 6, no. 4, pp. 307–315, 2014.
[75] D. G. Lowe, "Distinctive image features from scale-invariant keypoints," *Int. J. Comput. Vis.*, vol. 60, no. 2, pp. 91–110, 2004.
[76] D. Lu and Q. Weng, "A survey of image classification methods and techniques for improving classification performance," *Int. J. Remote Sens.*, vol. 28, no. 5, pp. 823–870, 2007.
[77] S. B. Maind and P. Wankar, "Research paper on basic of artificial neural network," *Int. J. Recent Innov. Trends Comput. Commun.*, vol. 2, no. 1, pp. 96–100, 2014.
[78] S. Maji, A. C. Berg, and J. Malik, "Classification using intersection kernel support vector machines is efficient," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2008, pp. 1–8.
[79] Y. Ding and J. Xiao, "Contextual boost for pedestrian detection," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2012, pp. 2895–2902.
[80] C. Shen, P. Wang, S. Paisitkriangkrai, and A. van den Hengel, "Training effective node classifiers for cascade classification," *Int. J. Comput. Vis.*, vol. 103, no. 3, pp. 326–347, 2013.
[81] M. Saberian and N. Vasconcelos, "Boosting algorithms for detector cascade learning," *J. Mach. Learn. Res.*, vol. 15, no. 1, pp. 2569–2605, 2014.
[82] J. Schmidhuber. (2014). "Deep learning in neural networks: An overview." [Online]. Available: https://arxiv.org/abs/1404.7828
[83] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet classification with deep convolutional neural networks," in *Advances in Neural Information Processing Systems*, F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, Eds. Red Hook, NY, USA: Curran Associates, 2012, pp. 1097–1105.
[84] K. Simonyan and A. Zisserman. (2015). "Very deep convolutional networks for large-scale image recognition." [Online]. Available: https://arxiv.org/abs/1409.1556
[85] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna. (2015). "Rethinking the inception architecture for computer vision." [Online]. Available: https://arxiv.org/abs/1512.00567
[86] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2016, pp. 770–778.
[87] S. Xie, R. B. Girshick, P. Dollár, Z. Tu, and K. He. (2016). "Aggregated residual transformations for deep neural networks." [Online]. Available: https://arxiv.org/abs/1611.05431
[88] S. Ren, K. He, R. Girshick, and J. Sun, "Faster R-CNN: Towards real-time object detection with region proposal networks," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 39, no. 6, pp. 1137–1149, Jun. 2017.
[89] J. Redmon, S. K. Divvala, R. B. Girshick, and A. Farhadi. (2015). "You only look once: Unified, real-time object detection." [Online]. Available: https://arxiv.org/abs/1506.02640
[90] F. N. Iandola, M. W. Moskewicz, K. Ashraf, S. Han, W. J. Dally, and K. Keutzer. (2016). "SqueezeNet: Alexnet-level accuracy with 50x fewer parameters and 0.5 MB model size." [Online]. Available: https://arxiv.org/abs/1602.07360
[91] V. Badrinarayanan, A. Kendall, and R. Cipolla. (2015). "SegNet: A deep convolutional encoder-decoder architecture for image segmentation." [Online]. Available: https://arxiv.org/abs/1511.00561
[92] I. J. Goodfellow *et al.*, "Generative adversarial nets," in *Proc. 27th Int. Conf. Neural Inf. Process. Syst. (NIPS)*, vol. 2, 2014, pp. 2672–2680.
[93] J. Yosinski, J. Clune, Y. Bengio, and H. Lipson, "How transferable are features in deep neural networks?" [Online]. Available: https://arxiv.org/abs/1411.1792
[94] J. Hosang, M. Omran, R. Benenson, and B. Schiele, "Taking a deeper look at pedestrians," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2015, pp. 4073–4082.
[95] P. Luo, Y. Tian, X. Wang, and X. Tang, "Switchable deep network for pedestrian detection," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2014, pp. 899–906.
[96] Y. Xiang, W. Choi, Y. Lin, and S. Savarese. (2016). "Subcategory-aware convolutional neural networks for object proposals and detection." [Online]. Available: https://arxiv.org/abs/1604.04693
[97] L. Zhang, L. Lin, X. Liang, and K. He, "Is faster R-CNN doing well for pedestrian detection?," in *Proc. Eur. Conf. Comput. Vis. (ECCV)*, Oct. 2016, pp. 443–457.
[98] Z. Cai, Q. Fan, R. S. Feris, and N. Vasconcelos, "A unified multi-scale deep convolutional neural network for fast object detection," in *Proc. Eur. Conf. Comput. Vis. (ECCV)*, B. Leibe, J. Matas, N. Sebe, and M. Welling, Eds. Cham, Switzerland: Springer, 2016, pp. 354–370.
[99] J. Li, X. Liang, S. Shen, T. Xu, and S. Yan. (Jun. 2015). "Scale-aware fast R-CNN for pedestrian detection." [Online]. Available: https://arxiv.org/abs/1510.08160
[100] X. Du, M. El-Khamy, J. Lee, and L. S. Davis. (2016). "Fused DNN: A deep neural network fusion approach to fast and robust pedestrian detection." [Online]. Available: https://arxiv.org/abs/1610.03466
[101] Z. Wu, C. Shen, and A. van den Hengel. (2016). "Wider or deeper: Revisiting the resnet model for visual recognition." [Online]. Available: https://arxiv.org/abs/1611.10080
[102] S. R. Bulò, L. Porzi, and P. Kontschieder. (2017). "In-place activated batchnorm for memory-optimized training of DNNs." [Online]. Available: https://arxiv.org/abs/1712.02616
[103] H. Zhao, J. Shi, X. Qi, X. Wang, and J. Jia. (2016). "Pyramid scene parsing network." [Online]. Available: http://arxiv.org/abs/1612.01105
[104] Y. Zhuang *et al.*, "Dense relation network: Learning consistent and context-aware representation for semantic image segmentation," in *Proc. IEEE Int. Conf. Image Process. (ICIP)*, vol. 2, Oct. 2018, pp. 3698–3702.
[105] T.-W. Ke, J.-J. Hwang, Z. Liu, and S. X. Yu, "Adaptive affinity fields for semantic segmentation," in *Proc. Eur. Conf. Comput. Vis. (ECCV)*, Sep. 2018, pp. 587–602.
[106] C. Yu, J. Wang, C. Peng, C. Gao, G. Yu, and N. Sang. "Learning a discriminative feature network for semantic segmentation." [Online]. Available: https://arxiv.org/abs/1804.09337
[107] L.-C. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. L. Yuille, "DeepLab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected CRFs," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 40, no. 4, pp. 834–848, Apr. 2017.
[108] L. Chen, G. Papandreou, F. Schroff, and H. Adam. (2017). "Rethinking atrous convolution for semantic image segmentation." [Online]. Available: https://arxiv.org/abs/1706.05587
[109] L. Chen, Y. Zhu, G. Papandreou, F. Schroff, and H. Adam, "Encoder-decoder with atrous separable convolution for semantic image segmentation," in *Proc. 15th Eur. Conf. Comput. Vis. (ECCV)*, Munich, Germany, Sep. 2018, pp. 833–851.
[110] L. Chen *et al.* (2018). "Searching for efficient multi-scale architectures for dense image prediction." [Online]. Available: https://arxiv.org/abs/1809.04184
[111] Z. Hayder, X. He, and M. Salzmann. (2016). "Shape-aware instance segmentation." [Online]. Available: https://arxiv.org/abs/1612.03129
[112] A. Arnab and P. H. S. Torr, "Pixelwise instance segmentation with a dynamically instantiated network," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Honolulu, HI, USA, Jul. 2017, pp. 879–888.
[113] S. Liu, J. Jia, S. Fidler, and R. Urtasun, "SGN: Sequential grouping networks for instance segmentation," in *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, Venice, Italy, Oct. 2017, pp. 3516–3524.
[114] Y. Liu *et al.*, "Affinity derivation and graph merge for instance segmentation," in *Proc. 15th Eur. Conf. Comput. Vis. (ECCV)*, Munich, Germany, Sep. 2018, pp. 708–724.
[115] K. He, G. Gkioxari, P. Dollár, and R. B. Girshick, "Mask R-CNN," in *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, Venice, Italy, Oct. 2017, pp. 2980–2988.
[116] D. Acuna, H. Ling, A. Kar, and S. Fidler, "Efficient interactive annotation of segmentation datasets with polygon-RNN++," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Salt Lake City, UT, USA, Jun. 2018, pp. 859–868.
[117] S. Liu, L. Qi, H. Qin, J. Shi, and J. Jia, "Path aggregation network for instance segmentation," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Salt Lake City, UT, USA, Jun. 2018, pp. 8759–8768.
[118] M. Bai and R. Urtasun, "Deep watershed transform for instance segmentation," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Honolulu, HI, USA, Jul. 2017, pp. 2858–2866.
[119] S. Beucher and C. D. M. Mathmatique, "The watershed transformation applied to image segmentation," in *Proc. Scanning Microscopy Int.*, 1991, pp. 299–314.

[120] A. Kendall, Y. Gal, and R. Cipolla, "Multi-task learning using uncertainty to weigh losses for scene geometry and semantics," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Salt Lake City, UT, USA, Jun. 2018, pp. 7482–7491.
[121] X. Li, Y. Grandvalet, and F. Davoine, "Explicit inductive bias for transfer learning with convolutional networks," in *Proc. 35th Int. Conf. Mach. Learn. (ICML)*, Stockholm, Sweden, Jul. 2018, pp. 2830–2839.
[122] Y. Tian, P. Luo, X. Wang, and X. Tang, "Deep learning strong parts for pedestrian detection," in *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, Dec. 2015, pp. 1904–1912.
[123] Y. Tian, P. Luo, X. Wang, and X. Tang, "Pedestrian detection aided by deep learning semantic tasks," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2015, pp. 5079–5087.
[124] W. Ouyang and X. Wang, "Joint deep learning for pedestrian detection," in *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, Dec. 2013, pp. 2056–2063.
[125] W. Ouyang, X. Zeng, and X. Wang, "Modeling mutual visibility relationship in pedestrian detection," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2013, pp. 3222–3229.
[126] N. K. Ragesh and R. Rajesh, "Avoiding multiple overlapped detection of single pedestrian," *Int. J. Adv. Comput. Commun. Control*, vol. 2, no. 4, pp. 151–155, 2014.
[127] W. Ouyang and X. Wang, "Single-pedestrian detection aided by multi-pedestrian detection," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2013, pp. 3198–3205.
[128] K. Okuma, A. Taleghani, N. de Freitas, J. J. Little, and D. G. Lowe, "A boosted particle filter: Multitarget detection and tracking," in *Computer Vision—ECCV* (Lecture Notes in Computer Science), vol. 3021, T. Pajdla and J. Matas, Eds. Berlin, Germany: Springer, 2004, pp. 28–39.
[129] S. Avidan, "Ensemble tracking," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 29, no. 2, pp. 261–271, Feb. 2007.
[130] B. Leibe, K. Schindler, and L. Van Gool, "Coupled detection and trajectory estimation for multi-object tracking," in *Proc. IEEE 11th Int. Conf. Comput. Vis. (ICCV)*, Oct. 2007, pp. 1–8.
[131] K. Ragland and P. Tharcis, "A survey on object detection, classification and tracking methods," *Int. J. Eng. Res. Technol.*, vol. 3, no. 11, pp. 622–628, Nov. 2014.
[132] A. Rangesh and M. M. Trivedi. (2018). "No blind spots: Full-surround multi-object tracking for autonomous vehicles using cameras & LiDARs." [Online]. Available: https://arxiv.org/abs/1802.08755
[133] P. Dollar. *Piotr's Computer Vision MATLAB Toolbox*. Accessed: Jul. 1, 2018. [Online]. Available: http://vision.ucsd.edu/~pdollar/toolbox/doc/index.html
[134] P. Dollar, C. Wojek, B. Schiele, and P. Perona. (2018). *Caltech Pedestrian Database*. [Online]. Available: http://www.vision.caltech.edu/archive.html
[135] D. Levi, S. Silberstein, and A. Bar-Hillel, "Fast multiple-part based object detection using KD-ferns," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2013, pp. 947–954.
[136] X. Du, M. El-Khamy, V. I. Morariu, J. Lee, and L. S. Davis. (2018). "Fused deep neural networks for efficient pedestrian detection." [Online]. Available: https://arxiv.org/abs/1805.08688
[137] S. Zhang, J. Yang, and B. Schiele, "Occluded pedestrian detection through guided attention in CNNs," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2018, pp. 6995–7003.
[138] A. Bar-Hillel, D. Levi, E. Krupka, and C. Goldberg, "Part-based feature synthesis for human detection," in *Computer Vision—ECCV*. Berlin, Germany: Springer, 2010, pp. 127–142.
[139] P. Dollar, Z. Tu, H. Tao, and S. Belongie, "Feature mining for image classification," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jul. 2007, pp. 947–954.
[140] C. Lin, J. Lu, G. Wang, and J. Zhou, "Graininess-aware deep feature learning for pedestrian detection," in *Proc. Eur. Conf. Comput. Vis. (ECCV)*, Sep. 2018, pp. 732–747.
[141] E. Ohn-Bar and M. M. Trivedi. (2017). "To boost or not to boost? On the limits of boosted trees for object detection." [Online]. Available: https://arxiv.org/abs/1701.01692
[142] W. Nam, B. Han, and J. H. Han, "Improving object localization using macrofeature layout selection," in *Proc. IEEE Int. Conf. Comput. Vis. Workshops (ICCV)*, Nov. 2011, pp. 1801–1808.
[143] C. Toca, C. Pătraşcu, and M. Ciu, "Performance testing and functional limitations of normalized autobinomial Markov channels," in *Proc. IEEE Int. Conf. Intell. Comput. Commun. Process. (ICCP)*, Sep. 2015, pp. 401–405.
[144] S. Paisitkriangkrai, C. Shen, and A. van den Hengel, "Efficient pedestrian detection by directly optimize the partial area under the ROC curve," in *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, Dec. 2013, pp. 1057–1064.
[145] S. Wang, J. Cheng, H. Liu, and M. Tang. (2018). "PCN: Part and context information for pedestrian detection with CNNs." [Online]. Available: http://arxiv.org/abs/1804.04483
[146] W. R. Schwartz, A. Kembhavi, D. Harwood, and L. S. Davis, "Human detection using partial least squares analysis," in *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, Sep. 2009, pp. 24–31.
[147] J. Marín, D. Vázquez, A. M. López, J. Amores, and B. Leibe, "Random forests of local experts for pedestrian detection," in *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, Dec. 2013, pp. 2592–2599.
[148] Y. Yang, Z. Wang, and F. Wu, "Exploring prior knowledge for pedestrian detection," in *Proc. Brit. Mach. Vis. Conf. (BMVC)*, Sep. 2015, pp. 176.1–176.12.
[149] G. Brazil, X. Yin, and X. Liu, "Illuminating pedestrians via simultaneous detection & segmentation," in *Proc. IEEE Int. Conf. Comput. Vis.*, Venice, Italy, Oct. 2017, pp. 4950–4959.
[150] S. Paisitkriangkrai, C. Shen, and A. van den Hengel, "Strengthening the effectiveness of pedestrian detection with spatially pooled features," in *Proc. Eur. Conf. Comput. Vis. (ECCV)*, 2014, pp. 546–561.
[151] S. Paisitkriangkrai, C. Shen, and A. van den Hengel, "Pedestrian detection with spatially pooled features and structured ensemble learning," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 38, no. 6, pp. 1243–1257, Jun. 2014.
[152] T. Song, L. Sun, D. Xie, H. Sun, and S. Pu, "Small-scale pedestrian detection based on somatic topology localization and temporal feature aggregation," in *Proc. ECCV*, Munich, Germany, Jul. 2018, pp. 1–16.
[153] W. Ouyang, H. Zhou, H. Li, Q. Li, J. Yan, and X. Wang, "Jointly learning deep features, deformable parts, occlusion and classification for pedestrian detection," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 40, no. 8, pp. 1874–1887, Aug. 2018.
[154] A. D. Costea and S. Nedevschi, "Word channel based multiscale pedestrian detection without image resizing and using only one classifier," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2014, pp. 2393–2400.
[155] B. Hariharan, P. A. Arbeláez, R. B. Girshick, and J. Malik, "Simultaneous detection and segmentation," in *Proc. 13th Eur. Conf. Comput. Vis. (ECCV)*, Zurich, Switzerland, Sep. 2014, pp. 297–312.

N. K. RAGESH received the Diploma degree in computer engineering from the State Board of Technical Education, Kerala, in 1996, the B.E. degree in computer engineering from Madras University, in 2001, and the M.Tech. degree in computer science with specialization in digital image computing from the University of Kerala, in 2003. He is currently pursuing the Ph.D. degree in computer science with Bharathiar University, Coimbatore, under the supervision of Dr. R. Rajesh (coauthor).

He has more than 16 years of experience in industry out of which 15 years in multimedia and signal processing including image, video, and audio processing. He has also worked with the Centre for Development of Advanced Computing (CDAC) and Network Systems and Technologies (NeST) during this tenure. For past 10 years, he is involved in research and development activities focusing on image processing, automotive infotainment, and advanced driver assistance systems (ADAS). He is currently a Senior Specialist in digital signal processing with the Automotive Business Unit, Tata Elxsi Ltd., Technopark, Trivandrum. His research interests include digital image processing, advanced driver assistance systems (ADAS), and computer graphics.

R. RAJESH received the B.Sc. degree in mathematics and the M.Sc. degree in computer science in 1996 and 1998, respectively, and the Ph.D. degree in computer science from the University of Kerala, in 2005, under the guidance of Prof. Dr. M. R. Kaimal on the topic titled *Fuzzy Models and Model Based Control for Nonlinear Systems*.

He has five years fulltime research experience including four years of experience in RESPOND project, funded by Indian Space Research Organisation (ISRO), titled *Development of Methodologies Based on Neuro Fuzzy and Genetic Algorithms for Modeling and Control of Systems*. He has one year industrial Postdoctoral Training as a Senior Research Associate/Research Consultant with Network Systems Technologies (NeST) Pvt. Ltd., Thiruvananthapuram. He has also done his Postdoctoral training in medical imaging on the topic titled *Influence of Noise in Human Brain* under the guidance of Dr. C. Kesavadas, Department of Radiology and Interventional, Sree Chitra Tirunal Institute of Medical Science and Technology, in 2007. He was an Associate Professor with the Department of Computer Science, Central University of Bihar, from 2013 to 2016, and an Assistant Professor with the Department of Computer Applications, Bharathiar University, from 2005 to 2013. He has also work experience at the Indian Institute of Information Technology and Management-Kerala (IIITM-K), University of Kerala, and Hindustan Latex Ltd. He is currently an Associate Professor with the Department of Computer Science, Central University of Kerala, Kerala, India. His research interests include computational intelligence (fuzzy systems and genetic algorithms) and image processing.

Dr. Rajesh was a recipient of the IEEE Award for Outstanding Leadership & Service and the IEEE Award for the Outstanding Design Project at the University of Kerala–Kariavattom student branch, in 2004. He was also a recipient of UGC-NET & JRF (1998), the ISRO RESPOND Junior Research Fellowship, in 1999, and the ISRO RESPOND Senior Research Fellowship, in 2001. He is a member of the IEEE Computational Intelligence Society and Computer Society of India.