International Journal of Multimedia Information Retrieval (2021) 10:171–184
https://doi.org/10.1007/s13735-021-00212-7

REGULAR PAPER

Optimized MobileNet + SSD: a real-time pedestrian detection on a low-end edge device

Chintakindi Balaram Murthy[1] · Mohammad Farukh Hashmi[1] · Avinash G. Keskar[2]

Received: 27 November 2020 / Revised: 10 May 2021 / Accepted: 30 June 2021 / Published online: 16 July 2021

Abstract
One of the most fundamental challenges in computer vision is pedestrian detection since it involves both the classification and localization of pedestrians at a location. To achieve real-time pedestrian detection without having any loss in detection accuracy, an Optimized MobileNet + SSD network is proposed. There are four important components in pedestrian detection: feature extraction, deformation, occlusion handling and classification. The existing methods design these components either independently or in a sequential format, and the interaction among these components has not been explored yet. The proposed network lets the components work in coordination in such a manner that their strengths are improved and the number of parameters is decreased compared to recent detection architectures. We propose a concatenation feature fusion module for adding contextual information in the Optimized MobileNet + SSD network to improve the detection accuracy of pedestrians. The proposed model achieved 80.4% average precision with a detection speed of 34.01 frames per second (fps) when tested on the Jetson Nano board, which is much faster compared to standard video speed (30 fps). Experimental results have shown that the proposed network has a better detection effect during low light conditions and for darker pictures. Therefore, the proposed network is well suited for low-end edge devices.

Keywords Pedestrian detection · Computer vision (CV) · Optimized MobileNet + SSD Network · Caltech pedestrian dataset · Jetson Nano board

1 Introduction

Computer vision has had major growth in numerous fields such as robotics, medical imaging, microscopy, image retrieval, face recognition and modern industrial applications. In recent years, pedestrian detection is also a conspicuous problem in CV applications like semi-autonomous vehicles and self-driving cars. It is also an indispensable and remarkable task in an intelligent video surveillance system. It has a clear-cut extension to automotive applications because of its use in safety systems. This was offered by many car manufacturers (e.g. Ford, Nissan, GM, Volvo) as an ADAS option in 2017.

✉ Mohammad Farukh Hashmi
mdfarukh@nitw.ac.in

[1] Department of Electronics and Communication Engineering, National Institute of Technology, Warangal, India

[2] Department of Electronics and Communication Engineering, Visvesvaraya National Institute of Technology, Nagpur, India

In daily life, we often drive through busy environments, traffic and challenging weather conditions. Road accidents are one of the major causes of death. So, we may be able to design safer automobiles by providing tools to inform and warn the driver about pedestrians and other relevant information. Therefore, it can save many lives in road accidents, i.e. the concept of self-driving cars. For pedestrian detection, we should be able to differentiate between multi-scale pedestrians and other complicated objects that are also present in the image backgrounds. For that, we need to withdraw the features of pedestrians such as shape, colour and behaviour. The intra-class variations of pedestrians such as clothing, backgrounds, lightning, articulation and occlusion are the main challenges. The detection is more precise if the expected and plausible features are extracted perfectly from the images.

A feature is an interesting part of an image, and the main focus of feature extraction is to extract information from the image and make some decisions at every image point so that the given features of any object are present in that image. So the extraction of image information can be done by convolutional neural networks (CNNs) as at present they have very good capability of high-level feature extraction of an image and are also useful in low-level feature detections. Deep convolutional neural network (DCNN) is classified basically into two categories, base and detection network. AlexNet [1], VGGNet [2], Xception [3], MobileNet [4] and DenseNet [5] are widely used baseline networks. High-level features are provided by classification or detection by the base network. MobileNet uses convolution to produce high-level features just like another base network as well to decrease the number of network parameters. For image classification, a fully connected (FC) layer is the final layer of CNN. Classification layers can be removed and replaced by detection networks. Examples of detection layers include feature-fused [6], feature pyramid network (FPN) [7], RCNN ("Region-based Convolutional Neural Network") series [8–10], YOLO ("You Only Look Once") series [11] [12], SSD ("Single Shot Multi-Box Detector") [13], M-RCNN ("Mask-Region-based Convolutional Neural Network") [14], etc. The use of SSD on the last convolutional layer results in a detection task.

Some studies on pedestrian detection [15] [16] use different types of targets as well as detection networks that are implemented using either traditional methods or deep learning-based methods. Traditional methods widely used in machine learning are "Histogram of Oriented Gradients" (HOG) [17], Haar-like features using patterns of motion and appearance [18], deformable models [19], etc. Deep learning-based methods include RCNN [8], Fast RCNN [9], Faster RCNN [10], YOLO [11], YOLOv2 [12], SSD300/512 [13] and Mask-RCNN [14]. However, there is still a need for further optimization in these networks for real-time multi-scale pedestrian detection on low-end edge devices.

To improve the performance of real-time pedestrian detection, the proposed model makes full use of convolutional layers. The proposed model used Keras [20] as the main deep learning framework with Tensor Flow [21] in the backend, and various data augmentation techniques are applied to the dataset while training. Keras provides fast experimentation with deep neural networks, and it is easy to use and extensible. Keras allows distributed fast training of deep neural network models on clusters of GPUs and Tensor processing units (TPU's). The proposed network is trained on Pascal Voc-2007 [22] dataset with Adam as optimizer by replacing SGD optimizer used in original SSD + VGG network. Finally, the proposed method is tested in real-time on a low-end edge device.

The key contributions of the proposed work are summarized as follows:

1. The SDD + VGG [13] model fails to detect dense pedestrians and offset this shallow convolution layers conv4_3 and conv5_3 are concatenated in the proposed Optimized MobileNet + SSD network which improves the feature map information and this in-turn improves detection performance.
2. This paper is the first to propose a concatenation feature fusion module for adding contextual information in the Optimized MobileNet + SSD network to improve the detection accuracy of pedestrians.
3. The proposed method seeks to extract the best hyper-parameters because fine-tuning hyper-parameters such as depth, stride, filter, shape, optimizer play a vital role in the optimization of the network and also help in reducing computational power while execution.
4. Experimental results show that the proposed model outperforms on Pascal Voc-2007 test dataset and showed a better detection effect while detecting denser and multi-scale pedestrians during low light and darker images and runs at 34.01 fps on Jetson Nano board.

The rest of this paper is organized as follows: Section II covers related work on pedestrian detection. In sections III and IV, methods and materials and the proposed methodology to solve the problem related to real-time pedestrian detection were discussed. In section V, we illuminated the findings of our model and also conducted experiments on a low-end device Jetson Nano; finally, the results, limitations and scope for future research in the area are discussed.

2 Related work

Recently, pedestrian detection methods based on deep learning techniques have exhibited state-of-the-art (SOTA) performance. Most existing pedestrian detectors employ either single-stage or two-stage strategy as their backbone architecture. Liu W et al. [13] proposed SSD using a single deep CNN to detect objects of various scales. This method separates the output space of bounding boxes into a set of anchor boxes over different aspect scales and ratios for every extracted feature map location. Szarvas M et al. [24] implemented pedestrian detection using CNN. The demonstrated method used a difficult pedestrian database collected from a city with no restrictions on background, pose, action, lighting and environmental conditions. Fukui H et al. [26] proposed a pedestrian detection based on deep CNN with an ensemble inference network to achieve high accuracy. To achieve such generalization, they introduced "ensemble inference network" (EIN) and "random dropout" for performing classification and training processes separately.

Joint deep learning for pedestrian detection was proposed by Ouyang [27]. A new deep CNN architecture by joint deep learning, i.e. through collaboration feature extraction, deformation and occlusion handling and classification learn jointly to maximize the strengths by formulating it into a joint deep learning framework. To detect occluded pedestrians, Zhang [28] proposed guided attention in CNNs. This method employs an attention network with self or external guidances on the baseline Faster RCNN detector to detect heavily occluded pedestrians.

Kuang et al. [29] implemented real-time pedestrian detection in a low illumination environment and distance estimation from the camera using a smartphone-based thermal camera. A "high-level semantic feature detection" [30] new perspective for pedestrian detection took pedestrian detection as a reference and a new perspective was developed for detecting objects which influence object detection to a modified level or a task, i.e. high-level semantic feature detection task. This method scans for feature points in the entire image such as corners, edges, blobs and where the convolution is naturally suited. Cheng et al. [31] proposed enhanced SSD for pedestrian detection. The proposed pedestrian detection with enhanced SSD depicts small-scale objects and most of the missed pedestrian targets (dense targets).

Yang F et al. [32] proposed SSD for online pedestrian detection on input fed video using Kalman filter. This method performs post-processing together by combining the fusion module and Kalman filter so that it effectively reduces miss rate and provides better performance at a faster speed. Afifi et al. [33] implemented robust real-time pedestrian detection using YOLOv3 on collected aerial images from Embedded Real-Time Inference (ERTI) Challenge on Jetson TX2 and achieved more than 5 frames per second (fps). Real-time pedestrian detection using a robust Enhanced Tiny-YOLOv3 network was proposed by C.B Murthy [36]. This method introduces an anti-residual module to improve the network's feature extraction and bounding box loss error is minimized but fails to detect severely occluded and denser pedestrians in real-time. Real-time pedestrian detection using the Improved Tiny Yolov3 network was proposed by Yi [35]. This method applies the K-means clustering algorithm on the training dataset to find the best prior bounding boxes to achieve better detection accuracy. But this algorithm fails to detect denser pedestrians in real-time during low light and darker pictures. So, to overcome this problem, the proposed model adopts a feature fusion concatenation module to improve detection accuracy while detecting denser pedestrians in real-time.

3 Methods and materials

3.1 Single shot detector (SSD)

Figure 1 shows the SSD + VGG backbone network for input image size 300 × 300. SSD is referred as regression-based object detector. The model can solve the conflict caused by translational invariance and variability and can achieve better detection accuracy as well as speed.

Each selected feature map which consists of K frames varies in size and width to height ratio. Each frame is termed as anchor box. Figure 1 shows bounding boxes on feature maps of different convolution layers. B class score and four position parameters are predicted by default at every bounding box. B x K x w x h class score and 4 × K x w x h position parameters need to be detected for the w–h feature image. It requires a size 3 × 3(B + 4) x K number of w x h convolution kernel for the processing of the feature map. The convolution result is considered to be the last feature for bounding box and classification regression. The mathematical expression for bounding boxes for every feature map is:

$$S_K = \frac{S_{Min} + (S_{Max} - S_{Min}) * (K - 1)}{(M - 1)}, \text{(K varies from [1, M])} \quad (1)$$

M indicates the count of feature maps, and S_{Min} and S_{Max} are settable parameters. The same five types of aspect ratios

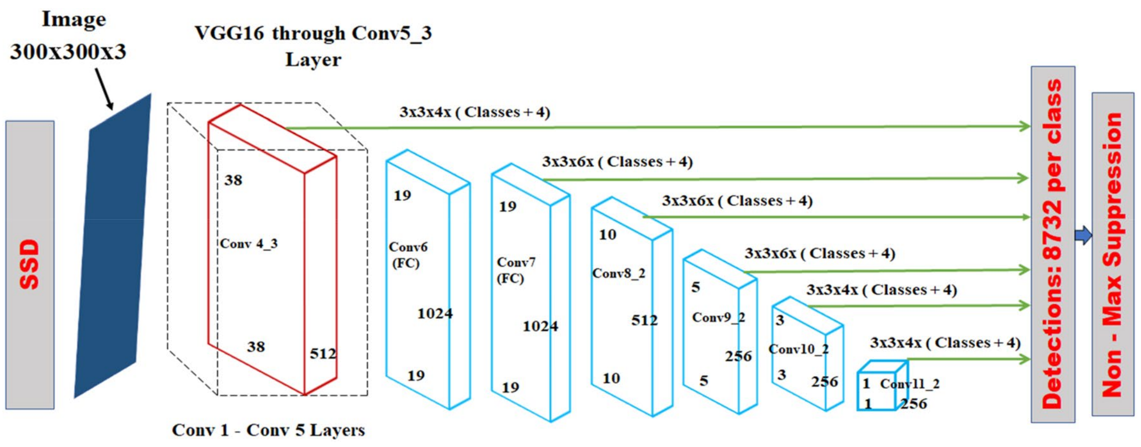

Fig. 1 SSD300 with VGG-16 backbone network through Conv5_3 layer

a = {1, 2, 3, 0.5, 0.33} generate anchor boxes to tune the fairness of feature vectors in training as well as while testing experiments. So, each bounding box can be mathematically expressed as:

$$h_K^a = \frac{S_K}{\sqrt{a_r}} \tag{2}$$

$$w_k^a = S_k\sqrt{a_r} \tag{3}$$

Here, h_k^a and w_k^a represent the height and width of the corresponding bounding box, respectively.

When the aspect ratio is one, a bounding box $s_k^{'} = \sqrt{s_k}.\sqrt{s_{k+1}}$ should be added. The centre of every bounding box varies from $(\frac{(i+0.5)}{|f_k|}, \frac{(j+0.5)}{|f_k|})$ and if $|f_k|$ represents the size of Kth feature unit, i, j € [0, $|f_k|$]. Figure 2 shows that a dog is perfectly matched to a bounding box in the 4 × 4 feature map, while it is not matched to any of the bounding boxes in the 8 × 8 feature map. Since the bounding boxes with different scales do not match with the dog box, they were considered as negatives during training.

The IoU, i.e. the intersection over union, can be mathematically expressed as:

$$\text{IoU} = \frac{(\text{AREA (C)} \cap \text{AREA (D)})}{(\text{AREA (C)} \cup \text{AREA (D)})} \tag{4}$$

If the calibration and bounding box intersection over union value exceeds 0.5, it indicates that for the respective category, the bounding box matches the calibration box.

The sum of bounding box regression's position loss, i.e. $L_{loc}(r, l, g)$ and the classification regression confidence loss is referred to as total loss function and is expressed as shown below:

$$L(s, r, c, l, g) = \frac{1}{N}.\ (L_{conf}(s, c) + \alpha L_{loc}(r, l, g)) \tag{5}$$

where "r" and "s" are eigenvectors of position loss and confident loss, respectively, α is a parameter to tune both position and confidence loss; 'I' is the offset including the scaling offset of the height and width and translational offset of the centre of the predicted boxes, 'N' represents the number of bounding boxes matching the calibration box for a given category, and 'g' is the calibration box of the target actual position.

3.2 Jetson Nano evaluation board

Nvidia Jetson Nano board is an embedded developer kit mainly meant for developing embedded systems that need high processing power for CV, deep learning, machine learning and image/video processing applications. Jetson Nano consists of 128 CUDA cores Maxwell GPU, quad-core ARM A57 CPU works at 1.43 GHz with 4 GB of LPDDR4 memory and has a processing power of 472 GFLOPS. Since the Jetson Nano board consumes power which is less than 5 watts, has in-built GPU cores and is low cost compared to other embedded boards. So, therefore, it is a popular board for implementing real-time pedestrian detection algorithms.

4 Proposed methodology

4.1 Optimized MobileNet + SSD network

MobileNet model was proposed by Google and is a type of base architecture highly suitable for embedded-based vision applications with less computing power. The MobileNet architecture uses depthwise separable convolutions instead of standard convolution. This reduces the number of parameters significantly as compared to the network with normal convolution with the same amount of depth in the network, which results in lightweight deep neural networks. The activation function "ReLU" is replaced by "ReLU6", and the "Batch Normalization" layer was included in each layer of the newly appended structure to prevent the gradient's disappearance. MobileNet is easy to train and takes relatively

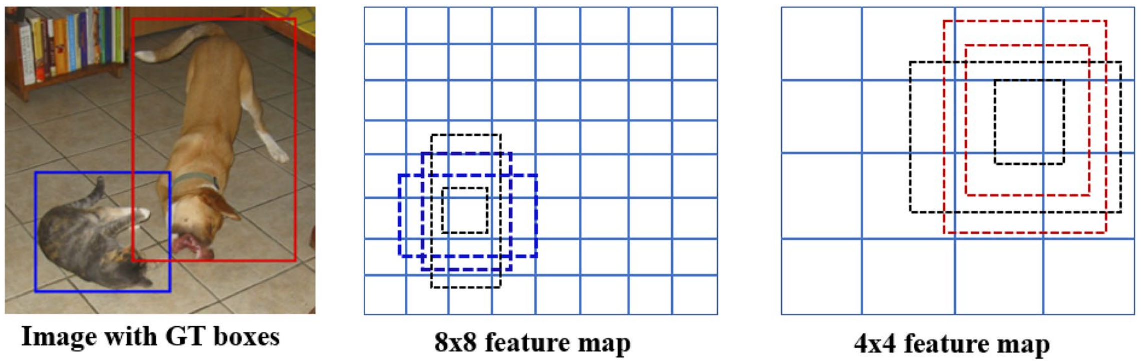

Fig. 2 SSD multiple bounding boxes for localization and confidence

less time while training, which is highly desired for real-time implementation. This makes the network more reliable compared to VGG-16 and other available architectures.

Figure 3 shows that the Optimized MobileNet + SSD network is composed of 21 convolutional layers. The target feature layers for detection used are Conv 4_3, Conv 13, Conv 14_2, Conv 15_2, Conv 16_2 and Conv 17_2. This network enhances the information of the newly added shallow convolutional layer Conv 5_3 to detect smaller and denser objects. Since Conv 4_3 and Conv 5_3 feature maps are different in terms of size, to make the same size, the Conv 5_3 layer is followed by a deconvolution layer (2 × up-sampled). On both layers, 3 × 3x256 convolution layer and applied normalization with 10, 20 different scales were used incorporating the better features to fuse. Finally, both convolutional layers are concatenated before by applying a 1×1×256 convolutional layer which reduces dimensions and feature recombination to generate the final fusion feature map as shown in Fig. 4.

We introduce a feature fusion concatenation module, to inject contextual information into the shallower layer Conv 4_3 since this layer lacks semantic information and it is an important supplement for detecting small-scale and dense pedestrians. Therefore, the detection performance of small-scale pedestrians is improved by passing the captured semantic information in convolutional forward computation back towards the shallower layers. While designing the most effective feature fusion concatenation module, we explored different layer feature fusion trials. Finally, we selected Conv 4_3 and Conv 5_3 shallow layers to fuse which would introduce less background noise while detecting small-scale

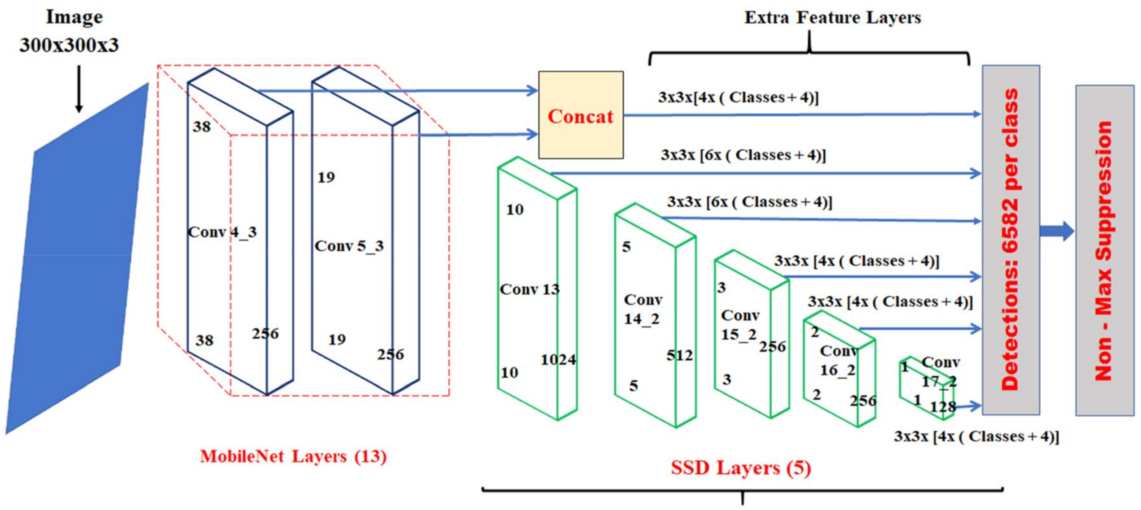

Fig. 3 Optimized MobileNet + SSD backbone network through Conv4_3 layer (proposed method)

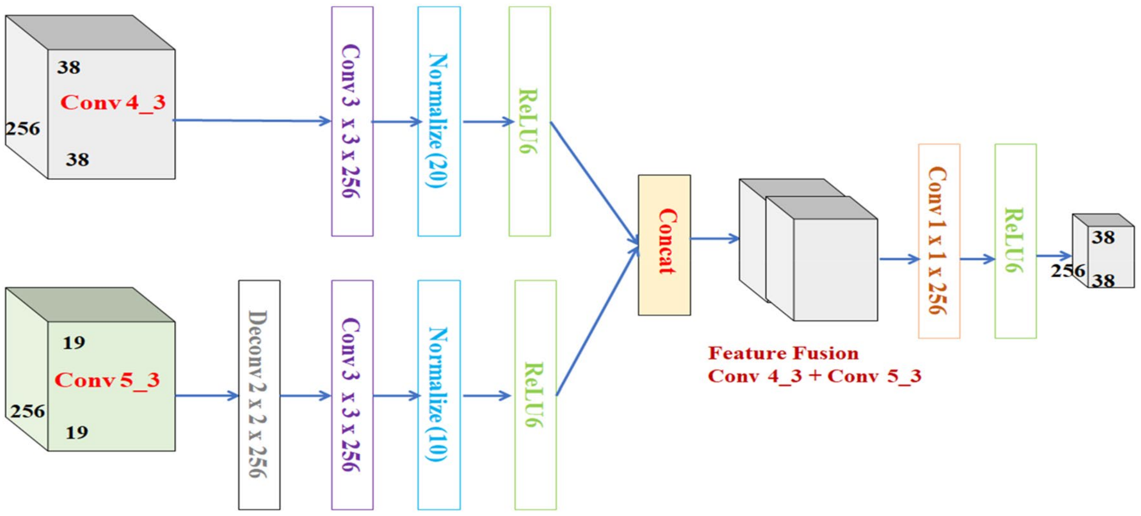

Fig. 4 Feature fusion concatenation module

pedestrians. Higher shallower layers after Conv 5_3 possess large receptive fields and would introduce more background noises while detecting small-scale pedestrians. Figure 5 shows the framework of the proposed Optimized MobileNet + SSD network.

Following pointwise convolution and depthwise convolution together is termed as depthwise separable convolution.

Figure 6 shows MobileNet's standard convolution layers (left side) and depthwise and pointwise separable convolutional layers (right side). Conv_D_w_P_w is a deep and separable convolution structure that consists of two layers, namely pointwise (P_w) and depthwise layers (D_w). The D_w layer uses 3×3 kernels, and the P_w layer uses 1×1 kernels which are also called deep convolutional layers and common convolutional layers, respectively. So, the result of each convolution is then processed by batch normalization (BN) algorithm and ReLU6 activation.

The activation function ReLU6 (Rectified Linear Unit6) is non-linear and also better than the sigmoid function in terms of performance. This algorithm supports the adjustment of data distribution automatically. Here, the activation function is limited to a maximum size of 6, due to increased robustness when used with low precision computation. MobileNet substantially reduces complexity, amount of calculation and also speeds up the training process. ReLU6 activation function is expressed as

$$r(z) = \min(\max(0, z), 6) \tag{6}$$

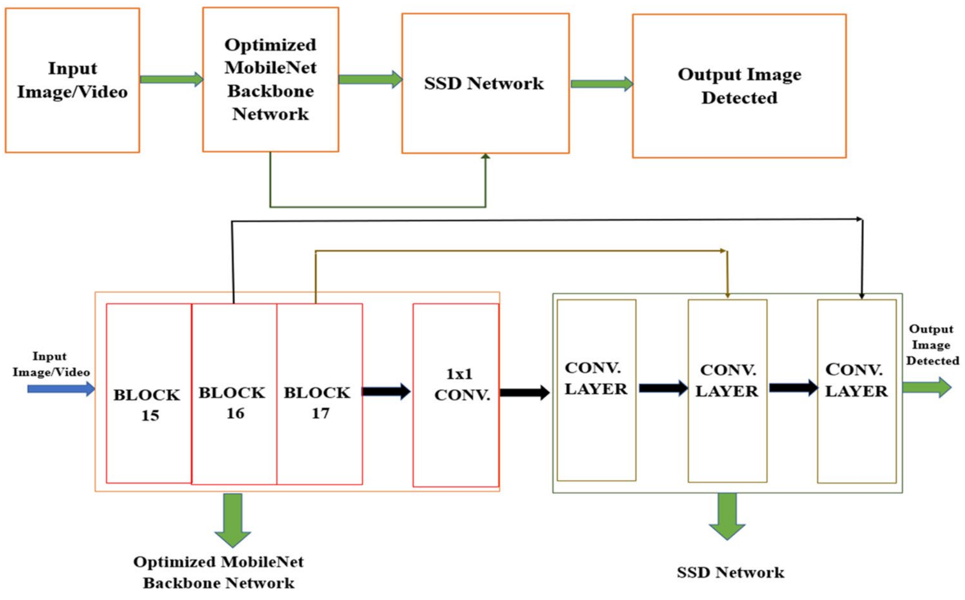

Fig. 5 Proposed Optimized MobileNet + SSD network framework

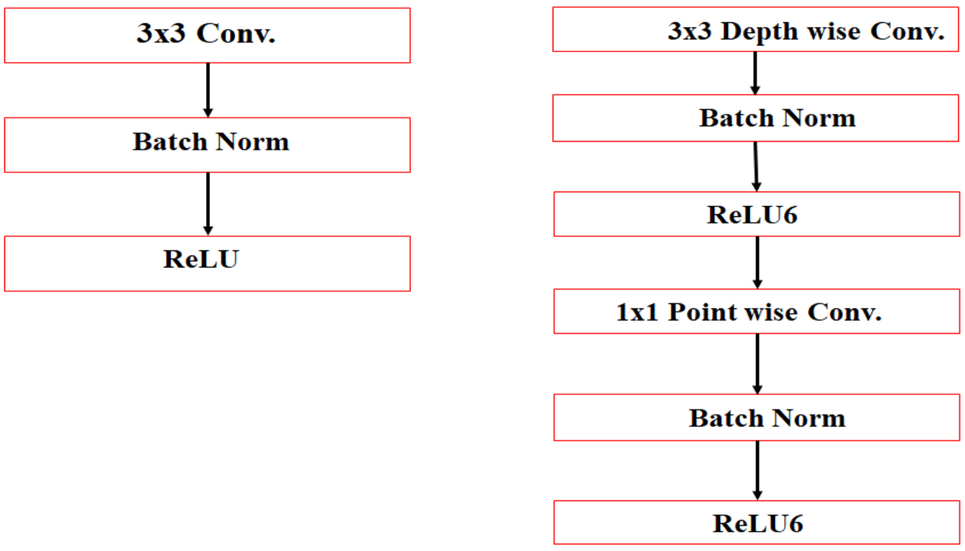

Fig. 6 Standard convolution (left) and depthwise convolution modules (right) with Batch Norm and ReLU6

Algorithm The flow of the proposed algorithm is as shown in Fig. 7.

4.2 Standard convolution

The working of a standard convolutional (regular) layer is similar to the convolution process in signals analysis. The process forms a kernel or sends a filter to all the given channels of the image. It iterates the kernel across the whole image and during each iteration performs a weighted pixels sum under the filter across all input channels.

The important feature of the convolution feature is that it combines the values of all input channels. For example, if there are three input channels and a single convolution kernel is applied, it would still result in an output image with only one channel for every pixel. No matter how many channels it has, for each input pixel, the convolution writes a new output pixel with only a single channel. In normal convolution, if F is the input feature map by applying a convolutional kernel of size K, an output feature map G is produced.

The standard convolution output feature map is computed as:

$$G_Q = \Sigma_Q K_{Q,R} \times F_Q \tag{7}$$

where $K_{Q,R}$ is the filter, Q is the number of input channels, and R is the number of output channels. Standard convolution consists of an input image that includes a featured image (F_Q) and uses the "fill style of zero padding".

When the sizes are $D_F x D_F$ and channels of input images are Q, it is necessary to have R filters with Q channels and the size of $D_K x D_K$ before outputting G feature images of size $D_K \times D_K$. The total computational cost of standard convolution is $D_K \times D_K \times Q \times R \times D_F \times D_F$.

The depthwise convolution with one filter per input channel formula is expressed as:

$$G'_Q = \sum K'_{1,Q} \times F_Q \tag{8}$$

where $K'_{1,Q}$ is the filter.

4.3 Depthwise separable convolution (DSC)

The MobileNet architecture uses depthwise separable convolution (DSC). It uses the standard convolution but just once on the very first layer. After the first layer, all further layers have DSC.

Depthwise separable convolution (DSC) module is just a combination of depthwise + pointwise convolution. The main difference between standard and depthwise separable

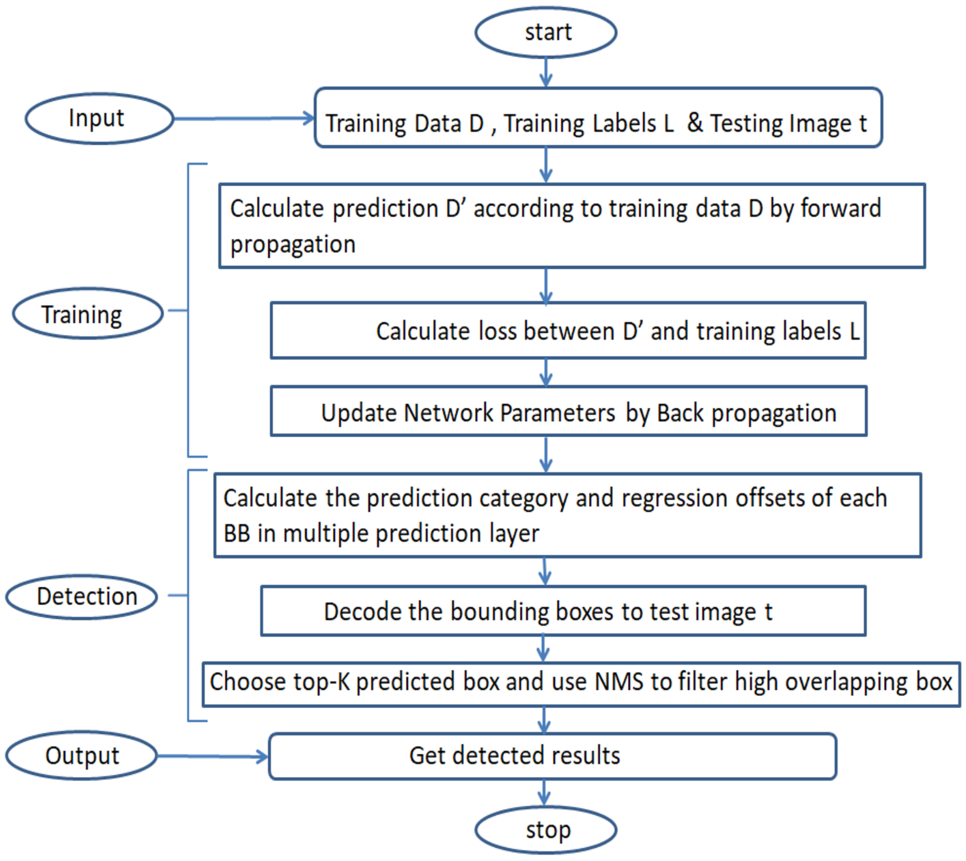
start
Input
Training Data D , Training Labels L & Testing Image t
Training
Calculate prediction D' according to training data D by forward propagation
Calculate loss between D' and training labels L
Update Network Parameters by Back propagation
Detection
Calculate the prediction category and regression offsets of each BB in multiple prediction layer
Decode the bounding boxes to test image t
Choose top-K predicted box and use NMS to filter high overlapping box
Output
Get detected results
stop

Fig. 7 Flowchart of the proposed algorithm

is that unlike the former, the latter performs convolution on each image channel separately. For an image having three image channels, it performs convolution separately and creates an output image that also has image channels. Each channel has then a separate set of weights. The main motive of depthwise operation is applied to a single channel at a time. This results in more precision in edge detection, color filtering, etc. In depthwise separable convolution, if the dimensions of the input feature map ($D_F \times D_F \times Q$) are applied to a filter of kernel size ($D_K \times D_K \times 1$), it produces an output feature map ($D_F \times D_F \times P$). This is then processed by pointwise convolution (1×1convolution) on P channels. The final output feature map generated is $D_k \times D_k \times Q \times D_F \times D_F$.

The objective of doing pointwise convolution is to combine separate channels in the output of depthwise convolution to create new features. Using the two methods results in separate filtering and combining unlike in standard convolution where the two are one process. Another main advantage of depthwise separable over standard is that even though they both finally result in filtering the data and making new features, standard convolution needs much more computational work to get the result and hence needs to learn more weights. Meanwhile, DSC implements convolution operations much more efficiently and uses fewer parameters. Table 1 shows the computation cost and the number of parameters required for both standard convolution and depthwise separable convolution.

Therefore, the reduction of computation cost obtained is:

$$= \frac{D_k \text{ x } D_k \text{ x } Q \text{ x } D_F \text{ x } D_F + Q \text{ x } R \text{ x } D_F \text{ x } D_F}{D_K \text{ x } D_K \text{ x } Q \text{ x } R \text{ x } D_F \text{ x } D_F} = \frac{1}{R} + \frac{1}{D_K^2} \tag{9}$$

Therefore, the parameters reduce to:

$$= \frac{D_k \text{ x } D_k \text{ x } Q + 1 \text{ x } 1 \text{ x } Q \text{ x } R}{D_K \text{ x } D_K \text{ x } Q \text{ x} R} = \frac{1}{R} + \frac{1}{D_K^2} \tag{10}$$

When width multiplier α is applied then the computation cost of DSC is given by:

$$D_K \times D_K \times \alpha Q \times D_F \times D_F + \alpha Q \times \alpha R \times D_F \times D_F \tag{11}$$

Experiments have proved that using a 3×3 kernel and saving computation time by the new approach is about 9 times faster without making any difference in detection. MobileNet uses up to 13 depthwise convolutions in a row.

5 Experiments and results

5.1 Datasets

The proposed methodology was trained on Pascal Voc-2007 [22] trainVal dataset and tested on both Pascal Voc-2007 and Caltech pedestrian test datasets [23].

5.1.1 Pascal VOC dataset

The "Pascal Visual Object Classes Dataset" is Pascal Voc-2007 test dataset that is a collection of images of around 20 classes. During the training, the model used Pascal Voc-2007 trainVal (5011 images). The "Test" image set contains around 4952 images. Since the Pascal dataset background is more complicated, the degree of occlusion and human postures is different, the size of the humans is not the same. So, therefore, several images were used to improve the generalization of the trained network to meet complex real-time traffic scene environment.

5.1.2 Caltech pedestrian dataset

This dataset contains a set of video sequences of 640×480 size. It includes some train (set 00 to set 05) subsets and test (set 06 to set 10) subsets. There are about 350,000 bounding boxes in 250,000 frames and 2300 pedestrians were annotated and only "person" and "people" were used in our experiments. The dataset was challenging due to the small size of pedestrians and different occlusion cases.

5.2 Experimental setup

The experiments were carried out on a workstation during the training phase, and finally, the testing phase was performed both on the workstation and Jetson Nano evaluation board. Figure 8 shows the experimental setup and captured real-time pedestrian detection on the Jetson Nano evaluation board.

Table 1 Computation cost and no. of parameters required for standard convolution and DSC

Layer	Parameter size	Computation cost
Standard Conv	$D_k \times D_k \times Q \times R$	$D_k \times D_k \times Q \times R \times D_F \times D_F$
Depthwise Conv	$D_k \times D_k \times Q$	$D_k \times D_k \times Q \times D_F \times D_F$
Depthwise Separable Conv	$D_k \times D_k \times Q + 1 \times 1 \times Q \times R$	$D_k \times D_k \times Q \times D_F \times D_F + Q \times R \times D_F \times D_F$

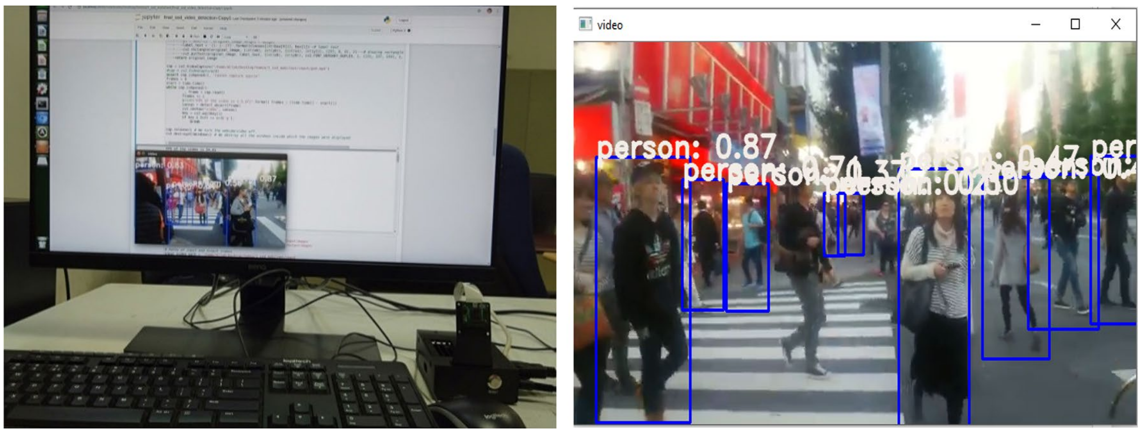

Fig. 8 Experimental setup and captured results on Jetson Nano Evaluation board

Names	Experimental configuration
OS	Windows 10 Pro
CPU/GHz	Intel Xeon 64 bit CPU @3.60
RAM/GB	64
GPU	NVIDIA Quadro P4000, 8 GB, 1792 Cuda cores
GPU acceleration library	CUDA10.0, CUDNN7.4
Tensor flow	2.×
Keras	2.2.×

5.3 Training and evaluation metrics

The model training was performed on trainVal (5011) images of the Pascal VOC-2007 dataset and tested on Pascal Voc-2007 test (4952) images. The input image size was set to 300×300, and various data augmentation techniques were applied such as flipping, cropping and random sampling to enhance the training process. We followed the standard evaluation methods used in [13]. The proposed method used the ADAM (Adaptive Moment Estimation) optimizer instead of SGD [13] optimizer while training the Pascal Voc-2007 dataset. The hyper-parameter values used, while training the proposed model is listed below.

Parameters	Values
Input size	300×300
Optimization method	Adam optimizer
Batch size	32
Weight decay	0.0005
Epsilon	1e-9
Beta1, Beta2	0.9,0.999
Iteration steps	150
Learning rate	0.001
Epochs	110

6 Recall vs. precision

To evaluate the robustness of the proposed model, the commonly used evaluation metrics for real-time pedestrian detection such as recall, precision, average precision (AP), speed (fps) and memory footprint were calculated.

Recall: Recall is defined as the % of total relevant results correctly classified by the algorithm.

$$\text{Recall} = \frac{\text{True Negative}}{\text{Predicted Results}} or \frac{\text{True Positive}}{\text{True Positive} + \text{False Negative}} \tag{12}$$

Precision: This is defined as % of results that are relevant, i.e. represents the accuracy of prediction.

$$\text{Precision} = \frac{\text{True Positive}}{\text{Actual Results}} or \frac{\text{True Positive}}{\text{True Positive} + \text{False Positive}} \tag{13}$$

Average Precision (AP): It is the area under the precision–recall curve, and it shows the correlation between precision and recall at a different level of confidence scores.

$$\text{Accuracy} = \frac{\text{True Positive} + \text{True Negative}}{\text{Total}} \tag{14}$$

Figure 9 shows the precision versus recall curve obtained on Pascal Voc-2007 test dataset. The graph depicts that with an increase in recall value at the convergence point, the precision starts gradually decreasing. The predictor built on Keras runs predictions over the entire dataset, matches predictions to ground-truth boxes, computes precision–recall curves for pedestrian class and samples 11 equidistant points

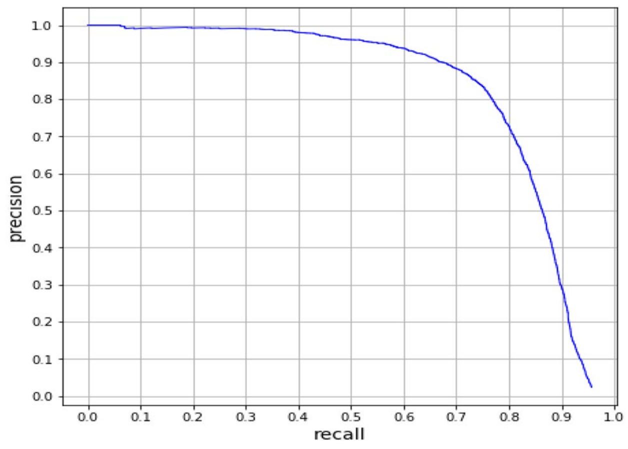

Fig. 9 PR curve of Optimized MobileNet + SSD network

Table 2 Comparison of average precision (AP) with the existing models. (IoU@0.5)

Model	AP (%)	Speed (Quadro P4000)	Weight (MB)
Fast RCNN [9]	68.4	0.5	513
Faster RCNN [10]	70.4	7	522
YOLO [11]	57.9	45	753
YOLOv2 [12]	76.8	38	203
SSD 300 (VGG16) [13]	71.4	46	100
SSD 512 (VGG16) [13]	76.8	19	103
100 Hz DPM [25]	16	100	–
30 Hz DPM [25]	26.1	33	–
Yolov3 [34]	78.12	35	247
Tiny Yolov3 [34]	68.54	**220**	33.9
Improved Tiny Yolov3 [35]	73.98	206	33.1
Proposed method	**80.04**	155	**23.6**

Bold values indicate the best result of the corresponding model

from the precision–recall curves to compute the average precision for the pedestrian class, giving AP value.

Table 2 shows the comparison of average precision (AP) with state-of-the-art (SOTA) models obtained on the Pascal Voc-2007 test dataset.

Comparing the results of Table 2 with existing SOTA models, the proposed model achieves better detection performance; the AP value reaches 80.04% on Pascal Voc-2007 test dataset, which is + 3.24%, + 1.92%, + 11.5% and + 6.06% higher compared to SSD 500 [13], Yolov3 [34], Tiny Yolov3 [34] and Improved Tiny Yolov3 [35], respectively. From the evaluation results, it is clear that the speed of Tiny-Yolov3 [34] is 220 fps, while that of the proposed model is only 155 fps. At the same time, the proposed model file is 23.6 MB, which is much smaller compared to Tiny Yolov3 [34] model file. The proposed model surpasses Improved Tiny Yolov3 [35] model both in terms of accuracy and weight file. To test the robustness of the proposed model, it was also tested on the Caltech pedestrian test dataset and achieved competitive results.

For real-time implementation, the proposed model was tested on a low-cost edge device Jetson Nano board. After training the proposed model on Quadro P4000 GPU, the whole model was tested on the Nvidia Jetson Nano evaluation board with the same system environment. Generally, more CUDA cores represent higher computational power, with the same memory and frequency conditions. The number of cores on Jetson Nano is 256 which is only 1/7th of Quadro P4000 (1792) GPU. But Jetson Nano consumes less energy and has much lower computational power. To test the validity of the proposed algorithm more intuitively, we captured real-time road video under low light conditions and fed it for detection verification. Figure 10 shows a comparison of the proposed model detection speed (fps) with existing SOTA models on the Pascal Voc-2007 test dataset.

By using the same video for verification, the detection speed of all SOTA detectors on Jetson Nano was far slower compared to on Quadro P4000 GPU. From Fig. 10, it is clear that the proposed model runs with a speed of 34.01 fps on Jetson Nano which is quite higher compared to SSD 512 [13], Yolov3 [34], Tiny Yolov3 [34] and Improved Tiny Yolov3 [35] SOTA models.

6.1 Detection results

The model is trained on low-resolution images with a size of 300 × 300. This results in a fall in accuracy on low-resolution images. Nevertheless, with Optimized MobileNet as a Backbone model, our proposed model can detect and identify pedestrian class with an appreciable amount of accuracy. Figure 11 shows samples of detected images of both Pascal Voc-2007 and Caltech pedestrian test datasets. This model accurately detected different samples varying from a few people to several. The model perfectly segregates different persons without intermixing them, giving precise detection results.

The proposed model is tested on low-resolution images, and the detected results are shown in Fig. 12. So this model has a better detection effect while detecting dense and smaller pedestrians during low light and darker pictures, while tiny-yolov3 [34] fails to detect pedestrians in darker pictures.

Since Pascal Voc-2007 dataset contains many small objects. Since our concern is only the pedestrian class, we manually gathered around 300 images that cover mainly smaller pedestrians for testing the performance of the proposed model. Detection results of the original SSD, YOLOv3 and Optimized MobileNet + SSD models are shown in Fig. 13.

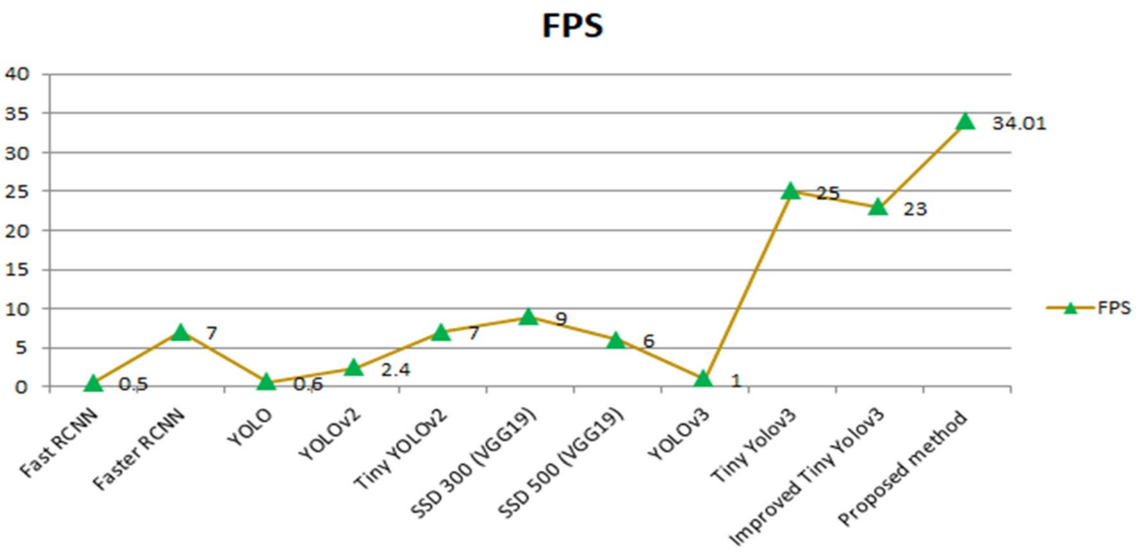

Fig. 10 Comparison of detection speed of the proposed model with the existing SOTA methods. (Tested on Jetson Nano board)

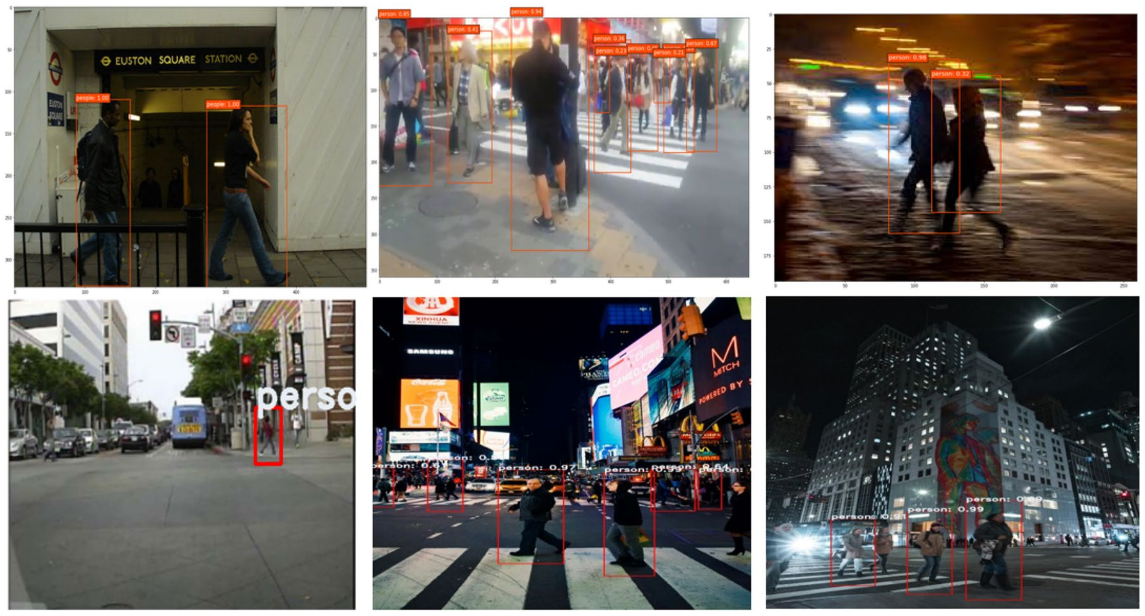

Fig. 11 Detection examples on sample images from Pascal Voc-2007 and Caltech datasets

Figure 13 shows clearly that the proposed model performs better than compared to the original SSD [13], YOLOv3 [34] while detecting small-scale and denser pedestrians in real-time.

To test the validity of the proposed algorithm more intuitively, we captured real-time road video under low light conditions and test results of randomly selected frames 498, 520 and 798 on Jetson Nano board, shown in Fig. 14. Therefore, the proposed algorithm has good adaptability to detect pedestrians under complex environments for real-time video.

From Fig. 14, it is clear that the proposed model when tested on Jetson Nano works better while detecting small-scale and denser pedestrians in real-time but fails to detect both occluded and distant smaller pedestrians.

7 Conclusion and future work

It is a quite challenging task to reliably detect multi-scale pedestrians on a low-end edge device due to their limited resolution and information in images. This is because the

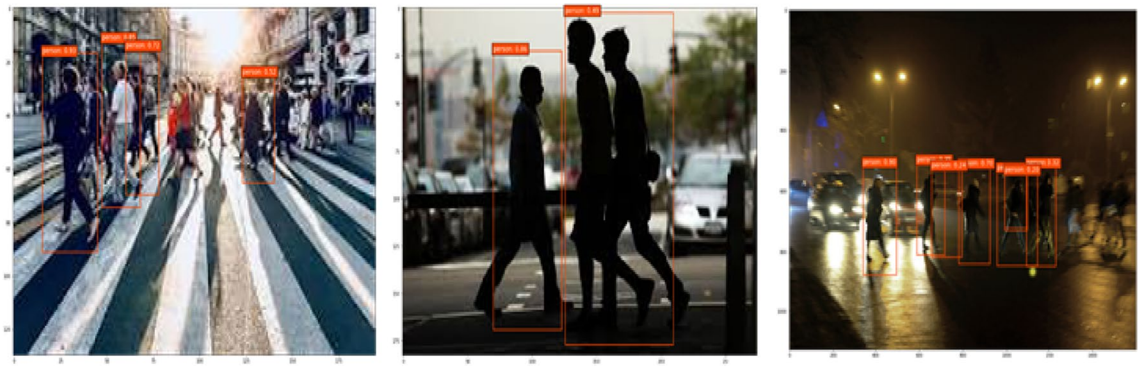

Fig. 12 Detection examples on sample images from Pascal Voc-2007 and Caltech datasets on low-resolution images

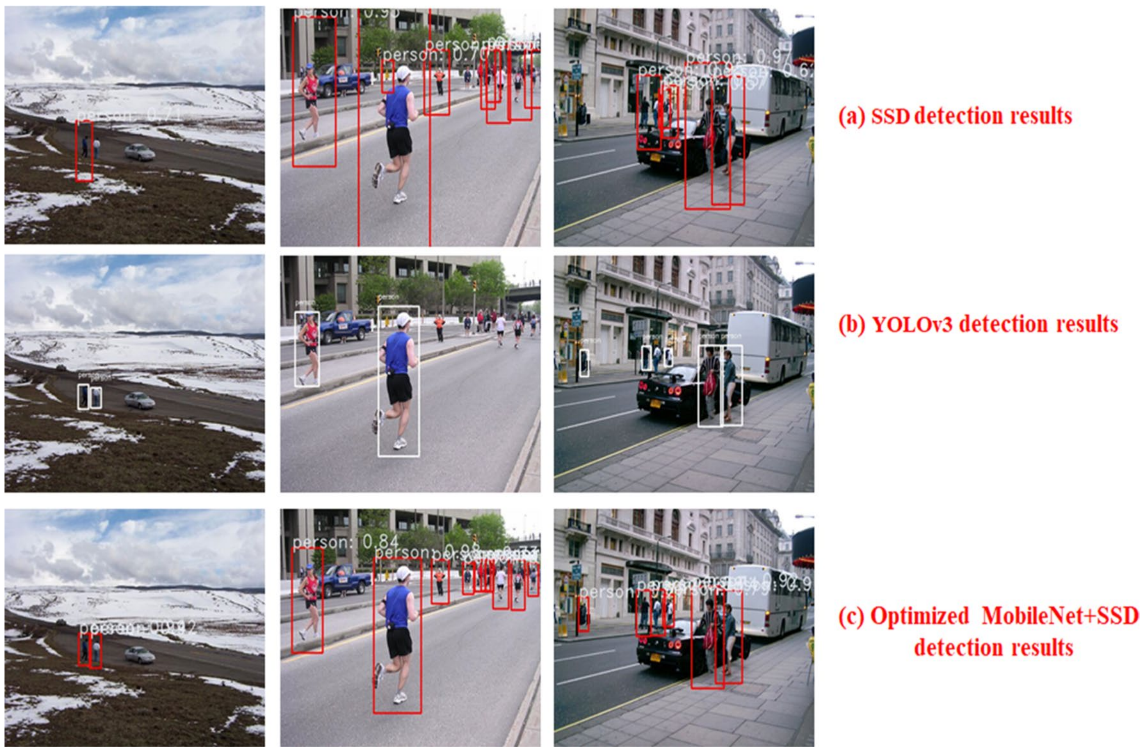
(a) SSD detection results
(b) YOLOv3 detection results
(c) Optimized MobileNet+SSD detection results

Fig. 13 Detection results of **a** Original SSD **b** YOLOv3 **c** Optimized MobileNet + SSD

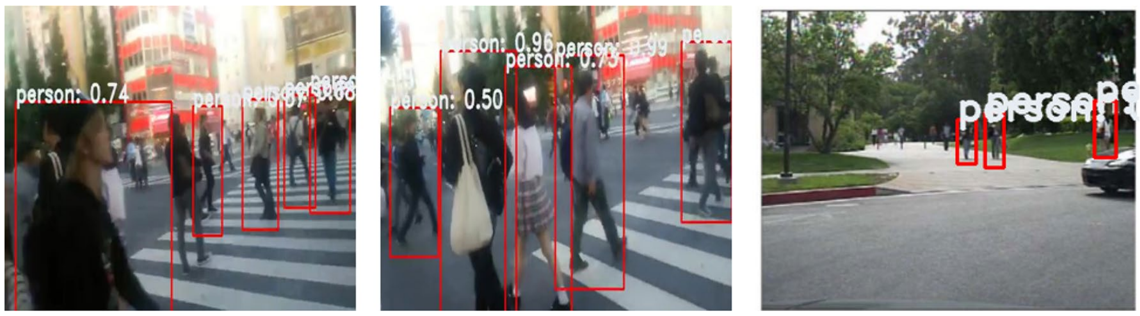

Fig. 14 Detection effect of the Optimized MobileNet + SSD Network on Jetson Nano board

existing SOTA models fail to improve real-time pedestrian detection accuracy. By exploiting the contextual information, considerable improvement is achieved in the proposed model. Therefore, a feature fusion concatenation module is introduced for adding contextual information in the Optimized MobileNet + SSD network to improve pedestrian detection accuracy in real-time. In using the proposed model, the number of network parameters is decreased, while the detection accuracy is improved when compared to state-of-the-art pedestrian detectors.

The proposed model for real-time pedestrian detection is implemented effectively by running it on the Jetson Nano evaluation board. Experimental results show that the proposed model achieves 80.4% average precision on Pascal Voc-2007 test dataset, which is + 1.92%, + 11.5% and + 6.06%, respectively, and much higher compared to Yolov3, Tiny Yolov3 and Improved Tiny Yolov3. Since the proposed model weight file is 23.6 MB and runs at 155 fps on Quadro P4000 GPU, it is more adaptable for low-end edge devices. The proposed model runs at a speed 5 × faster and holds 10.5 × times smaller weight file than the current SOTA real-time detector YoloV3. The proposed model runs in real-time with a speed of 34.01 fps when tested on the Jetson Nano board; however, the proposed model is still flawed, as it fails to detect occluded and distant smaller pedestrians in some frames. Still, the proposed model achieves competitive results on the Caltech pedestrian test dataset.

Declarations

Conflict of interest The authors declare that they have no conflict of interest.

References

1. Krizhevsky A, Sutskever I, & Hinton GE (2012) “Imagenet classification with deep convolutional neural networks,” In Advances in neural information processing systems. 1097–1105. doi: https://doi.org/10.1145/3065386
2. Simonyan K, & Zisserman (2014) “A, Very deep convolutional networks for large-scale image recognition,”arXiv preprint arXiv: 1409.1556
3. Chollet F (2017) "Xception: deep learning with depthwise separable convolutions,"2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR). Honolulu, HI, pp. 1800–1807, https://doi.org/10.1109/CVPR.2017.195.
4. Howard AG, Zhu M, Chen B, Kalenichenko D, Wang W, Weyand T& Adam H (2017) “Mobilenets: Efficient convolutional neural networks for mobile vision applications,” arXiv preprint arXiv: 1704.04861
5. Huang G, Liu Z, Van Der Maaten L, and Weinberger KQ (2017) "Densely Connected Convolutional Networks,"2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR). Honolulu, HI, 2261–2269, https://doi.org/10.1109/CVPR.2017.243.
6. Cao G, Xie X, Yang W, Liao Q, Shi G, & Wu J (2017) “Feature-fused SSD: Fast detection for small objects,” In Ninth International Conference on Graphic and Image Processing (ICGIP), Vol. 10615, p. 106151E. International Society for Optics and Photonics.
7. Dollár P, Appel R, Belongie S, Perona P (2014) Fast feature pyramids for object detection. IEEE Trans Pattern Anal Mach Intell 36(8):1532–1545. https://doi.org/10.1109/TPAMI.2014.2300479
8. Gkioxari G, Girshick R, and Malik J (2015) "Contextual Action Recognition with R*CNN,"2015 IEEE International Conference on Computer Vision (ICCV), Santiago, pp. 1080–1088, doi: https://doi.org/10.1109/ICCV.2015.129.
9. Girshick R (2015) "Fast R-CNN," 2015 IEEE International Conference on Computer Vision (ICCV), Santiago, pp. 1440–1448, https://doi.org/10.1109/ICCV.2015.169.
10. Ren S, He K, Girshick R, & Sun J (2015) “Faster r-cnn: Towards real-time object detection with region proposal networks,” In Advances in neural information processing systems. 91–99
11. Redmon J, Divvala S, Girshick R and Farhadi A (2016) "You Only Look Once: Unified, Real-Time Object Detection," 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Las Vegas, NV, pp. 779–788, https://doi.org/10.1109/CVPR.2016.91.
12. Redmon J and Farhadi A (2017) "YOLO9000: Better, Faster, Stronger," 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Honolulu, HI, pp. 6517–6525, https://doi.org/10.1109/CVPR.2017.690.
13. Liu W, Anguelov D, Erhan D, Szegedy C, Reed S, Fu CY, & Berg AC (2016) “Ssd: Single shot multibox detector,” In European conference on computer vision pp. 21–37, Springer, Cham. https://doi.org/10.1007/978-3-319-46448-0_2
14. He K, Gkioxari G, Dollár P and Girshick R (2017) "Mask R-CNN," 2017 IEEE International Conference on Computer Vision (ICCV), Venice, pp. 2980–2988, https://doi.org/10.1109/ICCV.2017.322.
15. Murthy CB, Hashmi MF, Bokde ND, & Geem ZW (2020) “Investigations of Object Detection in Images/Videos Using Various Deep Learning Techniques and Embedded Platforms-A Comprehensive Review,”Appl Sci. 10(9), 3280. https://doi.org/10.3390/app10093280
16. Zhao Z, Zheng P, Xu S, Wu X (2019) Object detection with deep learning: a review. IEEE Transactions on Neural Networks and Learning Systems 30(11):3212–3232. https://doi.org/10.1109/TNNLS.2018.2876865
17. Dalal N and Triggs B (2005) "Histograms of oriented gradients for human detection," 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR’05), San Diego, CA, USA, pp. 886–893 vol. 1. https://doi.org/10.1109/CVPR.2005.177.
18. Viola, Jones and Snow (2003) "Detecting pedestrians using patterns of motion and appearance," Proceedings Ninth IEEE International Conference on Computer Vision, Nice, France, pp. 734–741 vol.2. https://doi.org/10.1109/ICCV.2003.1238422.
19. Felzenszwalb PF, Girshick RB, McAllester D, Ramanan D (2010) Object detection with discriminatively trained part-based models. IEEE Trans Pattern Anal Mach Intell 32(9):1627–1645. https://doi.org/10.1109/TPAMI.2009.167
20. Keras. https://keras.io/. Accessed: 2019–12–31.
21. Abadi M, Barham P, Chen J, Chen Z, Davis A, Dean J & Kudlur M (2016) Tensorflow: “A system for large-scale machine learning,” In 12th {USENIX} Symposium on Operating Systems Design and Implementation ({OSDI}, pp. 265–283
22. Everingham M, Eslami SA, Van Gool L, Williams CK, Winn J & Zisserman A (2015) “The pascal visual object classes challenge: A retrospective,” International journal of computer vision. 111(1), pp.98–136.

23. Dollar P, Wojek C, Schiele B and Perona P (2009) "Pedestrian detection: A benchmark," 2009 IEEE Conference on Computer Vision and Pattern Recognition, Miami, FL, pp. 304–311. doi: https://doi.org/10.1109/CVPR.2009.5206631.
24. Szarvas M, Yoshizawa A, Yamamoto M, Ogata J (2005) "Pedestrian detection with convolutional neural networks," IEEE Proceedings. Intelligent Vehicles Symposium 2005. Las Vegas, NV, USA 3:224–229. https://doi.org/10.1109/IVS.2005.1505106
25. Sadeghi MA & Forsyth D (2014) "30hz object detection with dpm v5," In European Conference on Computer Vision, pp. 65–79, Springer, Cham. doi: https://doi.org/10.1007/978-3-319-10590-1_5
26. Fukui H, Yamashita T, Yamauchi Y, Fujiyoshi H, Murase H (2015) "Pedestrian detection based on deep convolutional neural network with ensemble inference network," 2015 IEEE Intelligent Vehicles Symposium (IV). Seoul. 9:223–228. https://doi.org/10.1109/IVS.2015.7225690
27. Ouyang W, Zhou H, Li H, Li Q, Yan J, Wang X (2018) Jointly learning deep features, deformable parts, occlusion and classification for pedestrian detection. IEEE Transactions on Pattern Analysis and Machine Intelligence 40(8):1874–1887. https://doi.org/10.1109/TPAMI.2017.2738645
28. Zhang S, Yang J and Schiele B (2018) "Occluded pedestrian detection through guided attention in CNNs," 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition, Salt Lake City, UT, pp. 6995–7003, doi: https://doi.org/10.1109/CVPR.2018.00731.
29. Kuang P, Ma T, Li F, & Chen Z (2018) "Real-time pedestrian detection using convolutional neural networks," International Journal of Pattern Recognition and Artificial Intelligence, vol; 32, no.11, 1856014. doi: https://doi/abs/https://doi.org/10.1142/S0218001418560141
30. Liu W, Liao S, Ren W, Hu W and Yu Y (2019) "High-level semantic feature detection: a new perspective for pedestrian detection," 2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Long Beach, CA, USA, pp. 5182–5191, doi: https://doi.org/10.1109/CVPR.2019.00533.
31. Cheng Y, Chen C, & Gan Z (2019) "Enhanced single shot multibox detector for pedestrian detection," In Proceedings of the 3rd International Conference on Computer Science and Application Engineering, pp. 1–7. doi: https://doi.org/10.1145/3331453.3361665
32. Yang F, Chen H, Li J, Li F, Wang L, Yan X (2019) Single shot multibox detector with kalman filter for online pedestrian detection in video. IEEE Access 7:15478–15488. https://doi.org/10.1109/ACCESS.2019.2895376
33. Afifi, Mohamed, Yara Ali, Karim Amer (2019) Mahmoud Shaker and Mohamed ElHelw, "Robust real-time pedestrian detection in aerial imagery on Jetson TX2," arXiv preprint arXiv: 1905.06653.
34. Redmon, Joseph and Ali Farhadi (2018) "Yolov3: An incremental improvement," arXiv preprint arXiv: 1804.02767.
35. Yi Z, Yongliang S, Jun Z (2019) An improved tiny-yolov3 pedestrian detection algorithm. Optik 183:17–23. https://doi.org/10.1016/j.ijleo.2019.02.038
36. Murthy CB, Farukh Hashmi M (2020) Real time pedestrian detection using robust enhanced tiny-YOLOv3. 2020 IEEE 17th India Council International Conference (INDICON). New Delhi, India 6:1–5. https://doi.org/10.1109/INDICON49873.2020.9342082