Journal of Theoretical and Applied Information Technology
15th August 2020. Vol.98. No 15

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

E-SSD: EMBEDDED DEEP CNN-BASED MODEL FOR CAR LOCALIZATION IN AUTONOMOUS VEHICLE SYSTEMS BASED ON LIGHTWEIGHT DEEP NETWORK

HOANH NGUYEN
Faculty of Electrical Engineering Technology, Industrial University of Ho Chi Minh City, Ho Chi Minh City, Vietnam
E-mail: nguyenhoanh@iuh.edu.vn

ABSTRACT

Single Shot Multibox Detector (SSD) is the leading one-stage object detection method. However, the process of detecting objects at different scales with different levels of semantic information makes it difficult to fuse high-level features from deeper convolution layers with low-level features from shallower convolution layers. To mitigate the fusion process of different feature layers and create enhanced semantic information feature maps, this paper proposes a lightweight and efficient multi-feature fusion module. The proposed multi-feature fusion module includes concatenation operation to concatenate the features at different scales in a simple and efficient way, point-wise convolution layer to reduce the feature dimension, and bilinear interpolation to upsample the size of feature maps. Furthermore, ESPv2 network, a lightweight and efficient deep convolutional neural architecture, is adopted as the base network for generating base convolution layers from input image. The proposed multi-feature fusion module and base network make a significant improvement in both detection accuracy and inference speed. Experimental results on KITTI dataset show that the proposed model outperforms SSD in terms of detection accuracy while maintaining inference speed. In addition, an embedded system implemented on an NVIDIA Jetson TX2 is used for detecting cars in traffic scenes that shows the effectiveness of the proposed detector.

Keywords: *Car Detection, Convolutional Neural Network, Intelligent Transportation System, Object Detection, Embedded System*

1. INTRODUCTION

1.1 Car Detection

Vision-based car detection, which uses a camera to acquire visual information from the driving environment, is an important part of an advanced driver assistance system. Using a camera as sensor to capture image, the information contained in images is abundant, including lanes, vehicles, pedestrians, and traffic lights, which helps the system to adapt to different difficult conditions. In addition, a camera is cheaper than other sensors, such as radar and lidar. With these advantages, vision-based car detection has attracted the attention of researchers. Many car detection methods have been proposed in recent years. Traditional methods are usually based on hand craft feature such as colour, shape, energy, and so on to locate cars in image. Li et al. [17] proposed a method of learning reconfigurable hierarchical and-or models to integrate context and occlusion for car detection. Wu et al. [18] learnt the structure of the and-or model with three components, and the model parameters are jointly trained using Weak-Label Structural SVM. Chen et al. [19] proposed a method based on background Gaussian Mixture Model and shadow removal method to deal with sudden illumination changes and camera vibration. Cui et al. [20] used Haar and Adaboost algorithm to detect the vehicle, and a simplified Lucas-Kanade algorithm was used to remove false positive detection. Slimani et al. [12] proposed to use two-dimensional discrete wavelet transform for extracting features from the images, and background subtraction followed by the connected components method to detect vehicles.

Recently, deep convolutional neural network (CNN)-based methods have become the leading method for high quality general object detection. Faster region-based convolutional neural network [9] defined a region proposal network (RPN) for generating region proposals and a network using these proposals to detect objects. RPN shares full-image convolutional features with the detection network, thus enabling nearly cost-free region

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

proposals. This method has achieved state-of-the-art detection performance and become a commonly employed paradigm for general object detection. SSD detector [5] predicted category scores and box offsets for a fixed set of default bounding boxes using small convolutional filters applied to different scales from feature maps of different scales, and explicitly separate predictions by aspect ratio. SSD achieved high inference speed with comparable performance compared with other state-of-the-art methods. Most of recent state-of-the-art deep learning models target general object detection including cars. To better handle the detection problem of vehicles in complex conditions, Zhou et al. [22] proposed a fast vehicle proposal network for vehicle-like objects extraction, and an attribute learning network aiming to verify each proposal. The author in [23] proposed a multi-scale feature map generation module to fuse different convolution layers at different scales of the base network. In addition, the information surrounding a given object proposal was exploited to enhance the feature representation of proposals. Dong et al. [24] proposed a vehicle type classification method using a semi-supervised convolutional neural network from vehicle frontal-view images. Hu et al. [25] proposed a scale-insensitive convolutional neural network for fast detecting vehicles with a large variance of scales. Deep CNN-based car detection methods have achieved great success in recent years. However, real-time car detection in driving environment is still very challenging. One of the main challenges is that CNN models are sensitive to scales while it is quite common that in-car videos or transportation surveillance videos contain vehicles with a large variance of scales. Current methods are based on modifying the popular CNN detectors to enhance the performance of detection results. These methods focus on making the network fit different scales by utilizing input images with multiple resolutions. However, these methods introduce expensive computational overhead and thus are still incapable of fast vehicle detection, which is essential for autonomous driving systems, real time surveillance and prediction systems.

1.2 Lightweight Deep Convolutional Neural Network (CNN) Architecture

Recent deep CNN-based architectures require a large amount of computational cost. While these architectures achieved high performance on high-end hardware machines, they are too expensive for resource constrained devices such as mobile devices and embedded computers. It is required that the deep CNN architecture should be lightweight and efficient while achieving comparable accuracy to implement on resource constrained devices. Thus, many enhanced networks for mobile devices have been introduced recently. Mobilenets [26] used depth-wise separable convolutions that factor a convolution into two steps to reduce computational complexity: depth-wise convolution that performs light-weight filtering by applying a single convolutional kernel per input channel and pointwise convolution that usually expands the feature map along channels by learning linear combinations of the input channels. Mobilenetsv2 [27] proposed a lightweight network based on an inverted residual structure where the shortcut connections are between the thin bottleneck layers. The intermediate expansion layer uses lightweight depthwise convolutions to filter features as a source of non-linearity. Shufflenet [28] and Shufflenet v2 [29] proposed new architecture that utilizes two new operations, pointwise group convolution and channel shuffle, to greatly reduce computation cost while maintaining accuracy.

2. METHODOLOGY

The proposed approach is based on SSD detector [5], which includes a base network for generating convolution layers and a detection subnetwork for locating objects at different layers. In this paper, a lightweight and efficient structure is adopted as the base network, which significantly improves inference speed while maintaining accuracy. Furthermore, a multi-feature fusion module is designed to fuse low-level feature map and high-level feature map, thus enhancing detection accuracy. The following subsections will elaborate each module in detail.

2.1 The Feature Extraction Subnet

This section elaborates the structure of the feature extraction subnet. To increase inference speed of the proposed model on embedded system and improve the detection performance, ESPNetv2 [1] is adopted as the base network for feature extraction. ESPNetv2 is a lightweight structure that can be easily deployed on mobile devices with limited hardware parts. The core of ESPNetv2 is the EESP unit and the Strided EESP unit, which combine group point-wise convolution layers and depth-wise dilated separate convolution layers. The

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

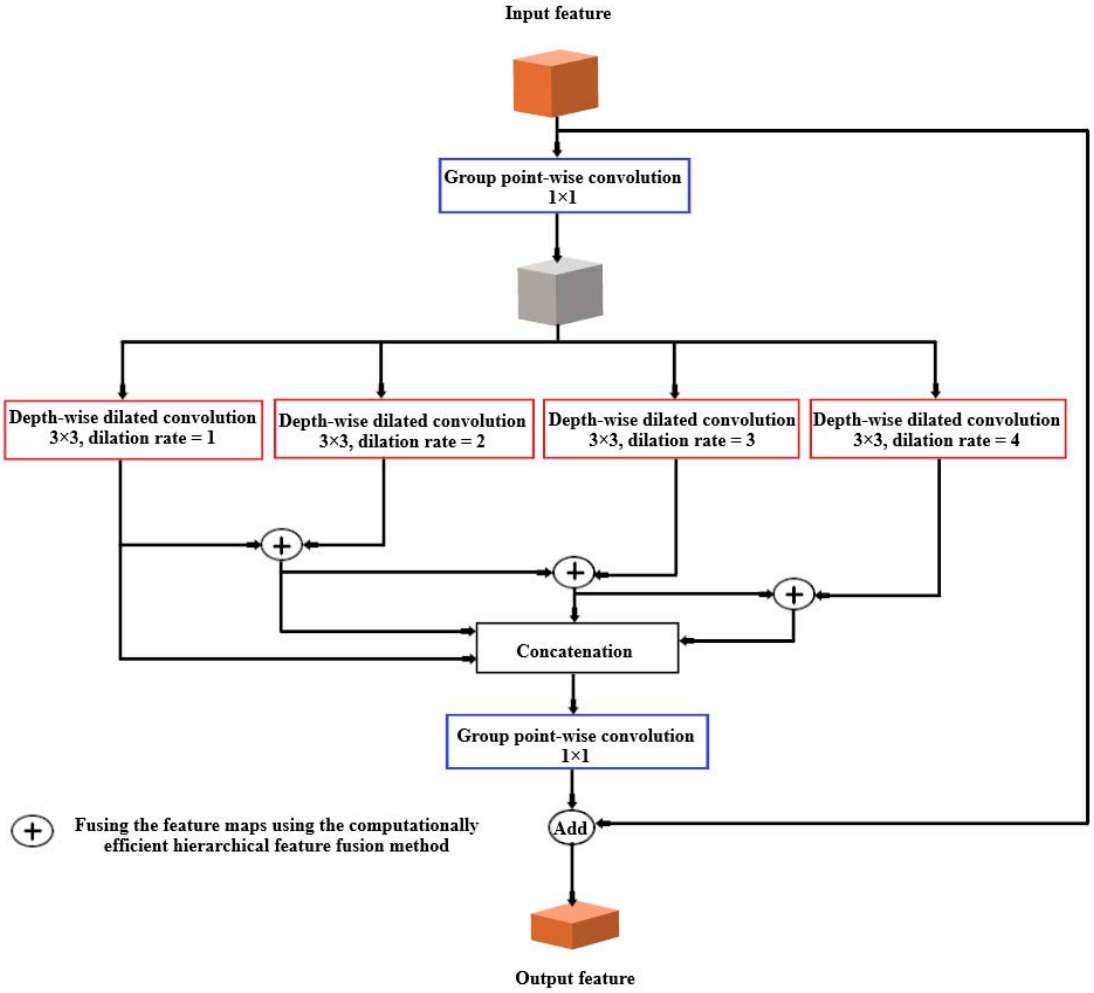

Figure 1: The Structure of The EESP Unit.

structure of the EESP and Strided EESP units are illustrated in the next sub-sections.

2.1.1 The EESP Unit

The EESP unit (Extremely Efficient Spatial Pyramid of Depth-wise Dilated Separable Convolutions), which is motivated by the ESPNet architecture [2], is specifically designed for mobile devices. Figure 1 shows the structure of the EESP unit. In the EESP unit, the high-dimensional input feature maps are first projected into low-dimensional space by using a 1×1 group point-wise convolution layer. The representations in low-dimensional feature maps are then learned in parallel by using 3×3 depth-wise dilated separable convolution layers with different dilation rates. Depth-wise dilated separable convolutions are efficient and can learn representations from large effective receptive fields. Depth-wise dilated separable convolutions apply a lightweight filtering by factoring a standard convolution into two layers: depth-wise dilated convolution layer and point-wise convolution layer. Note that different dilation rates in each branch allow the EESP unit to learn the representations from a large effective receptive field, thus enhancing the detection performance. By learning the representations in a low-dimensional space, the EESP units are efficient with small computational costs. In addition, the feature maps generated by depth-wise dilated separable convolution layers are fused by using the computationally efficient hierarchical feature fusion method [2] to remove the gridding artifacts caused by dilated convolutions.

2.1.2 The Strided EESP Unit

The Strided EESP Unit was designed to learn representations efficiently at multiple scales. Figure 2 shows the structure of the Strided EESP unit. As

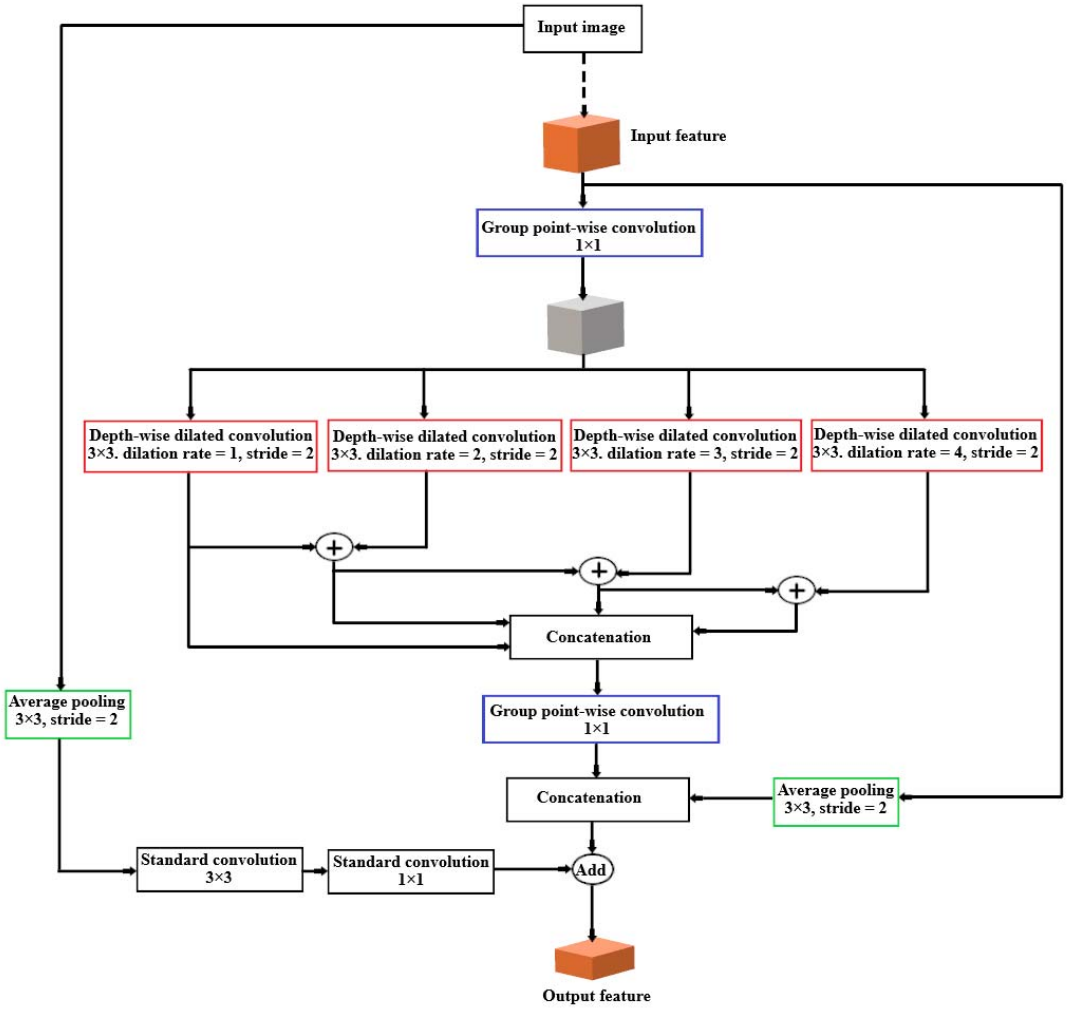

Figure 2: The Structure of The Strided EESP Unit.

shown, depth-wise dilated separable convolutions in the EESP unit is replaced by depth-wise dilated separable convolutions with stride = 2. In addition, an average pooling layer with kernel size 3×3 and stride = 2 is adopted after input feature maps to reduce the dimensions of input feature map, and concatenation operation is used instead of element-wise addition operation to fuse input feature map and feature map generated by depth-wise dilated separable convolutions. Furthermore, to preserve spatial information during down-sampling and convolution operations and learn representations efficiently, an efficient long-range shortcut connection between the input image and the current down-sampling unit is added in each of the Strided EESP unit. In this connection, the input image is first down-samples to the same size of that of the feature map by using a 3×3 average pooling layer with stride = 2. The representations are then learned by using a stack of two convolutions: a standard 3×3 convolution that learns the spatial representations and a 1×1 point-wise convolution that learns linear combinations between the input, and projects it to a high-dimensional space.

2.1.3 The Feature Extraction Subnet Architecture

The feature extraction subnet is based on ESPNetv2 architecture, which is built on the EESP units and the Strided EESP units. Figure 3 illustrates the architecture of the feature extraction subnet. At each spatial level, the feature extraction subnet repeats the EESP unit several times to increase the depth of the network. In the EESP units and the Strided EESP units, batch normalization [3] and PReLU [4] are added after every convolutional layer with an exception to the last group-wise convolutional layer where PReLU is applied after

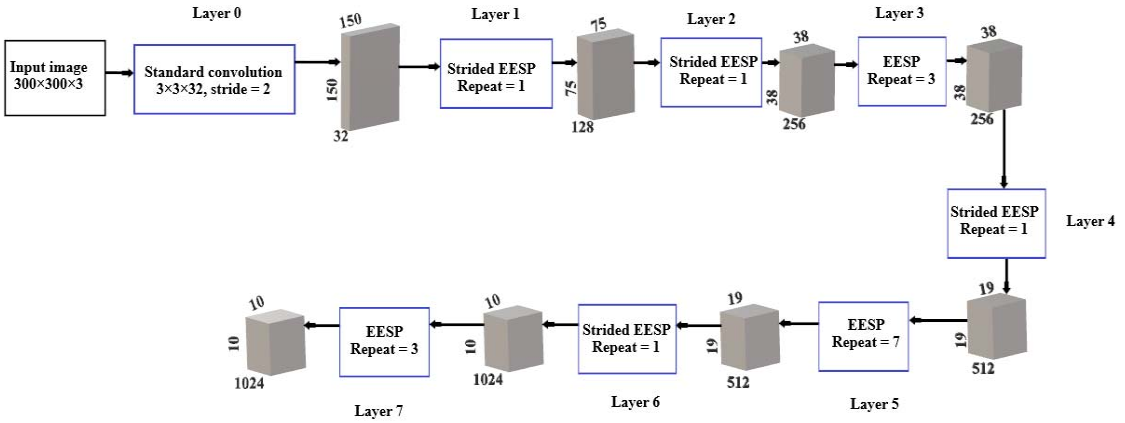

Figure 3: The Architecture of The Feature Extraction Subnet.

element-wise sum operation. To maintain the same computational complexity at each spatial-level, the feature maps are doubled after every down-sampling operation.

2.2 Multi-Feature Fusion Module

The original SSD detector [5] adopted feature maps at different convolution layers of the base network to directly generate object detection results. This approach makes SSD detector lack the capability to capture both the local detailed features and global semantic features. FPN [6] and DSSD [7] proposed to fuse low-level feature maps at shallower convolution layers with high-level feature maps at deeper convolution layers to improve the semantic information of each feature map. These approaches showed better detection performance compared with the original SSD framework. However, these approaches require multiple feature merging processes, thus increasing computational cost. To increase detection accuracy and maintain computational cost, this paper proposes a lightweight and efficient multi-feature fusion module. Figure 4 illustrates the structure of the proposed multi-feature fusion module. In original SSD detector, VGG-16 [8] is used as the base network. SSD used Conv4_3, Fc_7 of the base network and newly added layers, including Conv6_2, Conv7_2, Conv8_2, and Conv9_2 layer, to generate object detection results by the detection subnetwork. The corresponding feature size of these layers is 38×38, 19×19, 10×10, 5×5, 3×3 and 1×1. In this paper, the feature extraction subnet is designed to generate convolution layers. The multi-feature fusion module takes feature maps at layer 3, layer 5 and layer 7 as the input feature maps. The size of feature map at layer 3 is set as the basic feature map, which is 38×38×256. Next, in order to concatenate the features with different scales in a simple and efficient way, a 1×1 standard convolution layer is adopted after layer 5 and layer 7 to reduce the feature dimension firstly. Then bilinear interpolation is used to upsample the size of these feature map to the same size with feature map at layer 3. In this way, all the features have the same size on spatial dimension. Finally, concatenation operation is adopted to merge different feature maps together and generate fused feature map. Based on the fused feature map, this paper adds several convolution layers, including Conv8_2, Conv9_2, Conv10_2, Conv11_2, and Conv12_2, as in SSD. The size of feature maps generated by these convolution layers is 38×38, 19×19, 10×10, 5×5, 3×3 and 1×1 respectively. All these feature maps are fed to the detection subnetwork as in SSD.

2.3 Training

This paper follows the same training policy as in SSD. First, this paper matches a set of default boxes to target ground truth boxes. For each ground truth box, this paper matches it with the best overlapped default box and any default boxes whose Jaccard overlap is larger than 0.5. Among the non-matched default boxes, this paper selects certain boxes as negative samples based on the confidence loss so that the ratio with the matched ones is 3:1. Then this paper minimizes the joint localization loss and confidence loss.

2.3.1 Loss Function

The binary logistic loss is used here for box classification, and smooth L1 loss [9] is employed for box regression. The multi-task loss function used for training network is defined as follow:

$$L = \frac{1}{N_M}\sum_{i=1}^{N_M} L_{cls}(y_i, y_i') + \frac{1}{N_P}\sum_{i=1}^{N_P} L_{reg}(r_i, r_i') \quad (1)$$

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

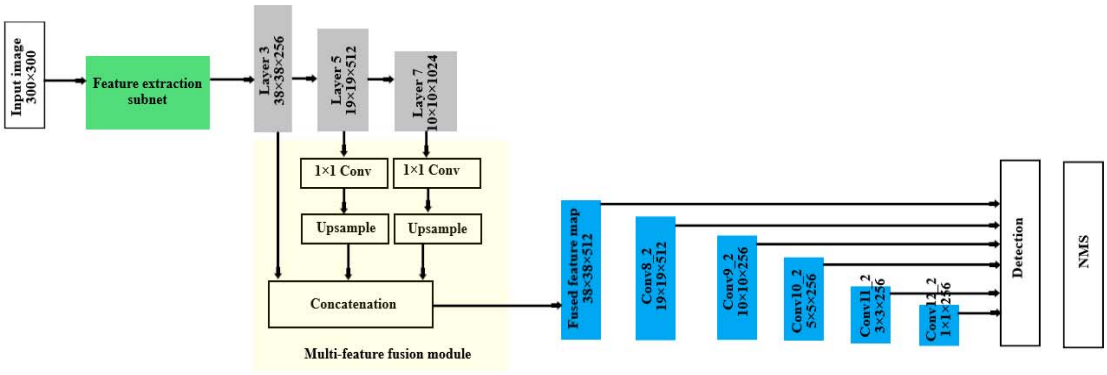

Figure 4: The Structure of The Proposed Multi-Feature Fusion Module.

where N_M is the size of default boxes; N_P is the number of matched default boxes; y_i is the predicted probability of box i being a car; y_i' is the corresponding ground-truth label (1 for positive box, 0 for negative box); r_i is the predicted coordinate offsets (x, y, w, h) for box i; r_i' is the associated offsets for box i relative to the ground-truth.
In (1), the binary logistic loss L_{cls} is defined as follow:

$$L_{cls}(y_i, y_i') = -\log(f_i^{y_i'}) \quad (2)$$

and the smooth L1 loss is defined as follow:

$$L_{reg}(r_i, r_i') = 0.25 * smooth_{L1}(r_i - r_i') \quad (3)$$

where

$$smooth_{L1}(x) = \begin{cases} 0.5x^2 & if\ |x| < 1 \\ |x| - 0.5 & otherwise \end{cases} \quad (4)$$

2.3.2 Choosing Default Boundary Boxes

SSD defines a scale value for each feature map layer. The layer Conv4_3 detects objects at the smallest scale 0.2 and then increases linearly to the rightmost layer at a scale of 0.9. Combining the scale value with the target aspect ratios, the width and the height of the default boxes can be computed. For layers making 6 predictions, SSD starts with 5 target aspect ratios: 1, 2, 3, 1/2 and 1/3. Then the width and the height of the default boxes are calculated as follows:

$$w = scale.\sqrt{ratio} \quad (5)$$

$$h = \frac{scale}{\sqrt{ratio}} \quad (6)$$

This paper has made a minor change in the prior box aspect ratio setting. Scene cars have rectangle shape or square shape, this paper uses three aspect ratios: 1, 2, 1/2 and uses these aspect ratios at every prediction layer.

2.3.3 Hard Negative Mining

As discussed above, there are much more negative matches than positive matches. This creates a class imbalance which hurts training. Thus, instead of using all the negative boxes, these negative boxes are sorted by their calculated confidence loss. As in SSD, this paper picks the negative boxes with the top loss and makes sure the ratio between the picked negative boxes and positive boxes is at most 3:1. This leads to a faster and more stable training.

2.3.4 Data Augmentation

Data augmentation is important in improving accuracy of the network. To handle variants in various object sizes and shapes, each training image is randomly sampled by one of the following options: Use the original; sample a patch with IoU of 0.1, 0.3, 0.5, 0.7 or 0.9; randomly sample a patch. The sampled patch will have an aspect ratio between 1/2 and 2. Then it is resized to a fixed size and this paper flips one-half of the training data.

3. EXPERIMENTS AND RESULTS

3.1 Dataset

To evaluate the detection performance of the proposed method, this paper conducts experiments on the KITTI dataset [13]. KITTI dataset is a large public dataset for evaluating the performance of different vehicle detection methods. It contains various scales of vehicles in different traffic scenes. The size of images in this dataset is 3840×1280 pixels. The dataset consists of 7481 images for training and 7518 images for testing. According to size, occlusion and truncation of vehicles in images, the dataset is classified into three difficulty level groups: easy, moderate and hard. Table 1 presents detailed information of each group.

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

Table 1: Detailed Information of Each Group.

Group	Detailed information		
	Height	Occlusion	Truncation
Easy	> 40 pixels	Fully visible	< 15%
Moderate	> 25 pixels	Partly occluded	< 30%
Hard	> 25 pixels	Difficult to see	< 50%

3.2 Evaluation Metrics

Precision and recall are usually used in the evaluation process of an object detection approach. The precision represents the proportion of the correctly detected cars in the predicted cars, and the recall represents the proportion of the correctly detected cars in all dataset. The precision and recall metrics are computed as follows:

$$precision = \frac{N_{TP}}{N_{TP}+N_{FP}} \quad (7)$$

$$recall = \frac{N_{TP}}{N_{TP}+N_{FN}} \quad (8)$$

where N_{TP} represents the number of true positives, which indicates the number of the correctly detected cars. N_{FP} represents the number of false positives, which indicates the number of the error detected cars (misjudge the background as a target). N_{FN} represents the number of false negatives, which indicates the number of miss detected cars. If the IoU between the predicted bounding box and the ground truth bounding box exceeds 0.5, the detection is regarded as true positive, otherwise, as a false positive. The IoU between the predicted bounding box (B_P) and the ground truth bounding box (B_G) is defined as follow:

$$IoU = \frac{area(B_P \cap B_G)}{area(B_P \cup B_G)} \quad (9)$$

If there are multiple predicted bounding boxes overlap the same ground truth bounding box, then only one is considered as true positive, while others are considered as false positive. The higher precision rate and recall rate, the better detection performance. However, the precision rate is usually balanced against the recall rate. When the recall rate increases, the precision rate will decrease accordingly. Therefore, the average precision (AP) of the precision-recall curve is usually used to evaluate the detection performance. The AP is the area under the precision-recall curve. Here, the average precision is obtained by calculating the average value of the corresponding precision when the recall rate changes from 0 to 1. In this paper, the average precision is calculated by the method used in the PASCAL VOC Challenge [14], which calculates the average precision by taking the mean of the precision rate of the points at all different recall rates on the P-R curve. Furthermore, the inference speed is calculated to compare the effectiveness of different methods.

3.3 Implementation Details

The proposed approach is implemented on a Window system machine with CPU Core i7-8700 @3.2GHz, GPU NVIDIA GTX 1080, RAM @ 12GB DDR4. The code is written in Python with Pytorch deep learning library [11]. In addition, tkinter library [10] is adopted to create graphical user interface.

For the base network, all the proposed experiments are all based on ESPNetv2 [1], which is pre-trained on the ImageNet 1000-way classification dataset [12]. This paper removes all the layers after layer 7 of the ESPNetv2. All of the raw features are converted to 256 channels with a 1×1 convolutional layer. Feature maps from layer 5 and layer 7 are upsampled to 38×38 by bilinear interpolation. Then the transformed feature maps are concatenated together followed by a batch normalization layer to normalize the feature values. Then several down-sampling blocks (including one 3×3 convolutional layer with stride 2 and one ReLU layer) are appended one by one to generate the pyramid features. This paper trains the proposed network with batch size 32 for 120k iterations. The initial learning rate is set to 0.001 and then divided by 10 at step 80k, 100k and 120k. Following the training strategy in SSD, the weight decay is set to 0.0005. This paper adopts SGD with momentum 0.9 to optimize the proposed network.

3.4 Results on KITTI Dataset

To evaluate the detection performance of the proposed method, this paper conducts experiments on KITTI dataset and compare the detection results with that of recent methods, including Faster R-CNN [9], SSD [5], YOLOv2 [15], and MS-CNN [16]. Faster R-CNN introduced a Region Proposal Network (RPN) that shares full-image convolutional features with the detection network, thus enabling

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

Table 2: Performance Comparison of Recent Methods and Proposed Method on KITTI Dataset.

Method	AP (%)			Inference time (s)
	Easy	Moderate	Hard	
Faster R-CNN [9]	87.90	79.11	79.19	2
SSD [5]	83.89	67.17	59.09	0.06
YOLOv2 [15]	28.37	19.31	15.94	0.02
MS-CNN [16]	90.46	88.83	74.76	0.4
E-SSD	89.24	77.60	70.02	0.06

Figure 5: Examples of Detection Results of The Proposed Method on KITTI Dataset.

nearly cost-free region proposals. The RPN is a fully convolutional network that simultaneously predicts object bounds and objectness scores at each position. SSD proposed a method for detecting objects in images using a single deep neural network. YOLOv2 proposed various improvements to the YOLO detection method and achieved significant improvement over YOLO. MS-CNN proposed to use the proposal sub-network to perform detection at multiple output layers, so that receptive fields match objects of different scales.

Table 2 presents the comparison of detection results. As shown, the proposed approach outperforms SSD in all three difficult level groups. More specific, the proposed method improves the AP by 5.35%, 10.43%, and 10.93% in easy, moderate, and hard group respectively compared with SSD. These results show that the multi-feature fusion module based on efficient network has

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

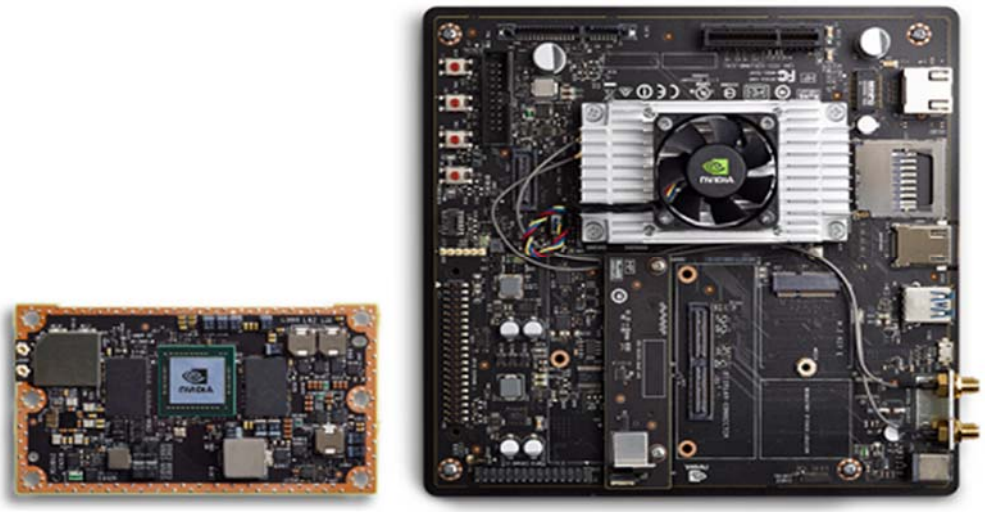

Figure 6: Image of The NVIDIA Jetson TX2 Embedded Board (Left) and Developer Board (Right).

Table 3: Technical Specifications of The NVIDIA Jetson TX2 Embedded Board.

Components	Specification
CPU	Dual-Core NVIDIA Denver 2 64-Bit CPU Quad-Core ARM® Cortex®-A57 MPCore
GPU	256-core NVIDIA Pascal™ GPU architecture with 256 NVIDIA CUDA cores
Memory	8GB 128-bit LPDDR4 Memory 1866 MHx - 59.7 GB/s
Operating system	Ubuntu Linux 14.04 LTS
Camera	5MP CSI camera module (with Omnivision OV5693)
Connectivity	802.11a/b/g/n/ac 2×2 867Mbps WiFi
Storage	32GB eMMC 5.1

Table 4: Running Time of The Proposed Detector and Other Detectors on NVIDIA Jetson TX2.

Model	Base network	Running time (s)
Faster R-CNN	VGG-16	-
SSD	VGG-16	-
E-SSD	ESPNetv2	0.125

improved the detection accuracy of the proposed model. Moreover, compared with YOLOv2, the proposed method surpasses in all difficult level groups. Comparing with Faster R-CNN, the proposed method achieves better result in easy group and comparable results in moderate and hard group. For the inference speed, E-SSD maintains the same speed as SSD detector. E-SSD takes 0.06 second for processing an image. This result shows the efficient of the base network and the feature fusion module. YOLOv2 achieves the best inference time with only 0.02 second. However, YOLOv2 shows the worst detection accuracy. According to the detection results in Table 2, it can be observed that MS-CNN achieves the best detection accuracy in all difficult level groups. However, MS-CNN has the second longest inference time with 0.4 second, which is about seven times E-SSD. Figure 5 shows some examples of detection results of the proposed method on KITTI dataset. As shown, the proposed method can detect cars with different scales in difficult environment conditions.

3.5 Application in Embedded System

To evaluate the effectiveness of the proposed model in embedded systems, this paper implements the proposed model on embedded board for cars

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

detection. For the hardware components of the proposed embedded system, this paper uses the fastest, most power-efficient embedded AI computing device, the Jetson TX2 board. Figure 6 shows the image of the development board, and Table 3 presents the main technical specifications of the board. The Jetson TX2 device is a technology developed by NVIDIA in the embedded system category. This device delivers the performance required for the latest visual computing applications, especially in deep learning. It is built based on NVIDIA Pascal-family GPU architecture with 256 CUDA cores providing greater than 1TFLOPS of FP16 compute performance in less than 7.5 watts of power, 64-bit CPUs, and a 5MP CSI camera module. To run the proposed model on the NVIDIA Jetson TX2 device, this paper adopts the Pytorch deep learning framework [11] compiled for GPU and Python programming language. This paper explores the running of the proposed deep CNN architectures for car detection on the NVIDIA Jetson TX2 embedded platform. In Table 4, this paper summarizes the running time of the proposed detector on NVIDIA Jetson TX2 and the performance of different deep architectures explored in the proposed work. According to Table 4, the proposed detector with ESPv2 architecture as the base network can process eight frames per second. In addition, as mentioned in Table 4, the NVIDIA Jetson TX2 embedded platform cannot run with the VGG16 deep architecture. Thus, the original Faster R-CNN and SSD detector cannot be implemented on the Jetson TX2 embedded platform.

4. CONCLUSIONS

This paper proposed a lightweight and efficient deep CNN-based detector for car detection based on SSD. In this paper, ESPNetv2, a lightweight and efficient architecture, is adopted as the base network, which significantly improves inference speed while maintaining accuracy. Furthermore, a multi-feature fusion module is designed to fuse low-level feature map and high-level feature map, thus enhancing detection accuracy. The proposed multi-feature fusion module and base network in this paper make a significant improvement in both detection accuracy and inference speed on car detection. Experimental results on KITTI dataset show that the performance of the proposed method outperforms SSD. In addition, an embedded system implemented on an NVIDIA Jetson TX2 is used for detecting cars in traffic scenes that shows the effectiveness of the proposed detector.

REFERENCES:

[1] Mehta, Sachin, Mohammad Rastegari, Linda Shapiro, and Hannaneh Hajishirzi. "Espnetv2: A light-weight, power efficient, and general purpose convolutional neural network." In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 9190-9200. 2019.

[2] Mehta, Sachin, Mohammad Rastegari, Anat Caspi, Linda Shapiro, and Hannaneh Hajishirzi. "Espnet: Efficient spatial pyramid of dilated convolutions for semantic segmentation." In *Proceedings of the european conference on computer vision (ECCV)*, pp. 552-568. 2018.

[3] Ioffe, Sergey, and Christian Szegedy. "Batch normalization: Accelerating deep network training by reducing internal covariate shift." *arXiv preprint arXiv:1502.03167* (2015).

[4] He, Kaiming, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. "Delving deep into rectifiers: Surpassing human-level performance on imagenet classification." In *Proceedings of the IEEE international conference on computer vision*, pp. 1026-1034. 2015.

[5] Liu, Wei, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, and Alexander C. Berg. "Ssd: Single shot multibox detector." In *European conference on computer vision*, pp. 21-37. Springer, Cham, 2016.

[6] Lin, Tsung-Yi, Piotr Dollár, Ross Girshick, Kaiming He, Bharath Hariharan, and Serge Belongie. "Feature pyramid networks for object detection." In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 2117-2125. 2017.

[7] Fu, Cheng-Yang, Wei Liu, Ananth Ranga, Ambrish Tyagi, and Alexander C. Berg. "Dssd: Deconvolutional single shot detector." *arXiv preprint arXiv:1701.06659* (2017).

[8] Simonyan, Karen, and Andrew Zisserman. "Very deep convolutional networks for large-scale image recognition." *arXiv preprint arXiv:1409.1556* (2014).

[9] Ren, Shaoqing, Kaiming He, Ross Girshick, and Jian Sun. "Faster r-cnn: Towards real-time object detection with region proposal networks." In *Advances in neural information processing systems*, pp. 91-99. 2015.

[10] Lundh, Fredrik. "An introduction to tkinter." *URL: www. pythonware.*

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

com/library/tkinter/introduction/index.htm (1999).

[11] Ketkar, Nikhil. "Introduction to pytorch." In *Deep learning with python*, pp. 195-208. Apress, Berkeley, CA, 2017.

[12] Russakovsky, Olga, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang et al. "Imagenet large scale visual recognition challenge." *International journal of computer vision* 115, no. 3 (2015): 211-252.

[13] Geiger, Andreas, Philip Lenz, and Raquel Urtasun. "Are we ready for autonomous driving? the kitti vision benchmark suite." In *2012 IEEE Conference on Computer Vision and Pattern Recognition*, pp. 3354-3361. IEEE, 2012.

[14] Everingham, Mark, Luc Van Gool, Christopher KI Williams, John Winn, and Andrew Zisserman. "The pascal visual object classes (voc) challenge." *International journal of computer vision* 88, no. 2 (2010): 303-338.

[15] Redmon, Joseph, and Ali Farhadi. "YOLO9000: better, faster, stronger." In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 7263-7271. 2017.

[16] Cai, Zhaowei, Quanfu Fan, Rogerio S. Feris, and Nuno Vasconcelos. "A unified multi-scale deep convolutional neural network for fast object detection." In *European conference on computer vision*, pp. 354-370. Springer, Cham, 2016.

[17] Li, Bo, Tianfu Wu, and Song-Chun Zhu. "Integrating context and occlusion for car detection by hierarchical and-or model." In *European Conference on Computer Vision*, pp. 652-667. Springer, Cham, 2014.

[18] Wu, Tianfu, Bo Li, and Song-Chun Zhu. "Learning and-or model to represent context and occlusion for car detection and viewpoint estimation." *IEEE transactions on pattern analysis and machine intelligence* 38, no. 9 (2015): 1829-1843.

[19] Chen, Zezhi, Tim Ellis, and Sergio A. Velastin. "Vehicle detection, tracking and classification in urban traffic." In *2012 15th International IEEE Conference on Intelligent Transportation Systems*, pp. 951-956. IEEE, 2012.

[20] Cui, Jianzhu, Fuqiang Liu, Zhipeng Li, and Zhen Jia. "Vehicle localisation using a single camera." In *2010 IEEE Intelligent Vehicles Symposium*, pp. 871-876. IEEE, 2010.

[21] Slimani, Ibtissam, Abdelmoghit Zaarane, Abdellatif Hamdoun, and Issam Atouf. "Traffic surveillance system for vehicle detection using discrete wavelet transform." *Journal of Theoretical & Applied Information Technology* 96, no. 17 (2018).

[22] Zhou, Yi, Li Liu, Ling Shao, and Matt Mellor. "DAVE: A unified framework for fast vehicle detection and annotation." In *European Conference on Computer Vision*, pp. 278-293. Springer, Cham, 2016.

[23] NGUYEN, HOANH. "A MULTI-SCALE DEEP LEARNING NETWORK FOR VEHICLE DETECTION." *Journal of Theoretical and Applied Information Technology* 97, no. 24 (2019).

[24] Dong, Zhen, Yuwei Wu, Mingtao Pei, and Yunde Jia. "Vehicle type classification using a semisupervised convolutional neural network." *IEEE transactions on intelligent transportation systems* 16, no. 4 (2015): 2247-2256.

[25] Hu, Xiaowei, Xuemiao Xu, Yongjie Xiao, Hao Chen, Shengfeng He, Jing Qin, and Pheng-Ann Heng. "SINet: A scale-insensitive convolutional neural network for fast vehicle detection." *IEEE transactions on intelligent transportation systems* 20, no. 3 (2018): 1010-1019.

[26] Howard, Andrew G., Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam. "Mobilenets: Efficient convolutional neural networks for mobile vision applications." *arXiv preprint arXiv:1704.04861* (2017).

[27] Sandler, Mark, Andrew Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen. "Mobilenetv2: Inverted residuals and linear bottlenecks." In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 4510-4520. 2018.

[28] Zhang, Xiangyu, Xinyu Zhou, Mengxiao Lin, and Jian Sun. "Shufflenet: An extremely efficient convolutional neural network for mobile devices." In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 6848-6856. 2018.

[29] Ma, Ningning, Xiangyu Zhang, Hai-Tao Zheng, and Jian Sun. "Shufflenet v2: Practical guidelines for efficient cnn architecture design." In *Proceedings of the European Conference on Computer Vision (ECCV)*, pp. 116-131. 2018.