Journal of Theoretical and Applied Information Technology
15th August 2020. Vol.98. No 15
© 2005 – ongoing JATIT & LLS

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

A LIGHTWEIGHT AND EFFICIENT DEEP CONVOLUTIONAL NEURAL NETWORK BASED ON DEPTHWISE DILATED SEPARABLE CONVOLUTION

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

Deep convolutional neural networks (CNNs) have achieved significant improvements in different vision tasks, including classification, detection and segmentation. However, the increasing model size and computation makes it difficult to implement DNNs on embedded systems with limited hardware resources. Many approaches proposed to build a lightweight network and have achieved comparable performance, such as MobileNets, ShuffleNet, and ESPNet. This paper proposes a lightweight and efficient network based on depthwise dilated separable convolution and MobileNetv2 architecture. Depthwise dilated convolution in depthwise dilated separable convolution module effectively enlarge the field of view of filters to incorporate larger context without increasing the number of parameters or the amount of computation. Furthermore, instead of using a convolution with 3×3 kernel size for each depthwise separable convolution block in MobileNetv2, this paper uses dilated convolutions with different dilation rates to learn the representations in parallel. The proposed model is evaluated on two public datasets. The results show that the proposed model achieves better classification accuracy compared with MobileNetv2. In addition, a simple object detection framework based on the proposed model is designed and conducted on an embedded system. Experiment results show the effectiveness of the proposed model in different vision tasks.

Keywords: *Deep Convolutional Networks, MobileNetv2, Image Classification, Object Detection, Depthwise Separable Convolution*

1. INTRODUCTION

In recent years, deep CNNs have shown significant improvements in many computer vision tasks such as image classification, object detection, and image segmentation. For example, AlexNet [2], which has 60 million parameters and 500,000 neurons and consists of five convolutional layers and two globally connected layers with a final 1000-way softmax, has won the ImageNet Large Scale Visual Recognition Challenge 2012 (ILSVRC2012). VGG-16 [15] includes 13 convolutional and 3 fully-connected layers. VGG-16 architecture stacks more layers onto AlexNet and uses smaller filters size (2×2 kernel and 3×3 kernel). The network consists of 138M parameters and takes up about 500MB of storage space. Inception-v1 [16] designed a 22-layer architecture with 5M parameters. The network is built using dense blocks, where blocks are stacked instead of stacking convolutional layers. The network has won the ImageNet Large-Scale Visual Recognition Challenge 2014 (ILSVRC 2014). Inception-v3 [17] is a successor to Inception-v1. The motivation for Inception-v3 is to avoid representational bottlenecks and have more efficient computations by using factorization methods. Xception [18] is an adaptation from Inception, where the Inception modules have been replaced with depthwise separable convolutions. ResNeXt-50 [19] presented a simple, highly modularized network architecture for image classification. The network is constructed by repeating a building block that aggregates a set of transformations with the same topology.

Most of above deep CNNs require large amounts of computational resources, including memory and power, which makes it difficult to apply the model to portable mobile devices with limited hardware resources. In order to apply deep CNN model to real-time applications on low-spec devices,

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

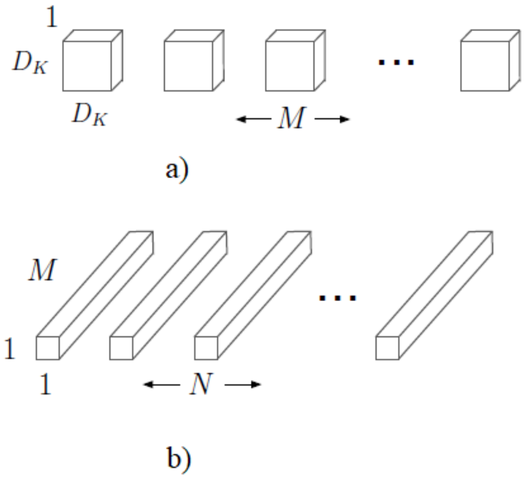

Figure 1: Depthwise Separable Convolution Includes Depthwise Convolution (a) and Pointwise Convolution (b).

many approaches proposed to compress existing architecture or design an efficient architecture. For compressing existing architecture, most of approaches applied this approach improves the inference efficiency of CNNs via weight quantization and/or weight pruning. Hubara et al. [14] proposed a neural network with binary weights and activations at run-time. At train time, the binary weights and activations are used for computing the parameter gradients. During the forward pass, the proposed network drastically reduces memory size and accesses, and replace most arithmetic operations with bit-wise operations, which is expected to substantially improve power-efficiency. Rastegari et al. [20] proposed two efficient approximations to standard convolutional neural networks, including binary-weight-networks and XNOR-networks. He et al. [21] introduced a new channel pruning method to accelerate very deep convolutional neural networks. The compressing approaches are effective, because CNNs have a substantial number of redundant weights. For designing efficient architectures, many approaches have explored efficient CNNs that can be trained end-to-end. MobileNetv1 [1], Mobilenetv2 [3] proposed to use depthwise separable convolutions that factor a convolution into two steps to reduce computational complexity as shown in Figure 1, including depthwise convolution that performs lightweight 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. Huang et al. [22] introduced DenseNet, which connects each layer to every other layer in a feed-forward fashion. Squeezenet [23] propose a small DNN architecture which achieved AlexNet-level accuracy on ImageNet with 50x fewer parameters. The efficient architectures greatly reduce computational requirements without significantly reducing accuracy.

2. METHODOLOGY

2.1 Depthwise Dilated Separable Convolution

To mitigate the computational cost and complexity in deep CNN architecture, many approaches proposed to replace standard convolution layers by different types of convolution layers, such as depthwise separable convolution layer [1] and group convolution layer [2]. This section elaborates the depth-wise dilated separable convolution in detail and compares with other efficient convolutions.

A standard convolution layer takes an input feature map with size $W \times H \times C$ and generates an output feature map with size $W' \times H' \times C'$ by using convolution kernel with size $n \times n \times C \times C'$. Here, $W \times H$ and $W' \times H'$ are the spatial width and height of input feature map and output feature map respectively; C and C' are the input and output channels respectively. The number of learning parameters is n^2CC'. In depth-wise separable convolution [1], standard convolution is replaced by depth-wise separable convolution, which includes depthwise convolution and pointwise convolution. Depthwise convolution applies a single filter per each input channel and pointwise convolution applies 1×1 convolution to create a linear combination of the output of the depthwise layer. In this paper, depth-wise dilated separable convolution is used to replace standard convolution. Depth-wise

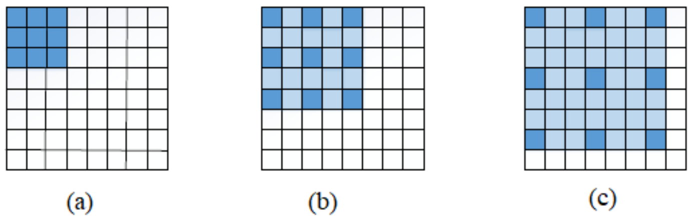

Figure 2: (a) Standard Convolution; (b) Dilated Convolution with Dilation Rate = 2; (c) Dilated Convolution with Dilation Rate = 3.

Table 1: Comparison of The Number of Parameters Between Different Types of Convolutions.

Type of convolution	Receptive field	Number of Parameters
Standard convolution	$n \times n$	n^2CC'
Group convolution	$n \times n$	$\frac{n^2CC'}{g}$
Depth-wise separable convolution	$n \times n$	$n^2C{+}CC'$
Depth-wise dilated separable convolution	$n_r \times n_r$	$n^2C{+}CC'$

dilated separable convolution includes two layers: depth-wise dilated convolution layer and point-wise convolution layer. Depth-wise dilated convolution layer applies dilated convolution per input channel with a dilation rate of r, thus enabling the convolution to learn representations from an effective receptive field of $n_r \times n_r$, where n_r is calculated as follow:

$$n_r = (n-1).r + 1 \tag{1}$$

Dilated convolution with dilation rate of r introduces $r-1$ zeros between consecutive filter values, effectively enlarging the kernel size of a $n \times n$ filter to $n_r \times n_r$ by using the equation (1) without increasing the number of parameters or the amount of computational cost. Figure 2(a) illustrates standard convolution with 3×3 convolution kernel. Figure 2(b) shows dilated convolution where convolution kernel changed to 5×5 with dilation rate = 2. Figure 2(c) shows dilated convolution where convolution kernel changed to 7×7 with dilation rate = 3.

Depth-wise dilated convolution layer only filters input channels, it does not combine them to create new features. Thus, point-wise convolution with 1×1 kernel size is used to learn linear combinations of input channels. The combination of depth-wise dilated convolution and point-wise convolution reduces the computational cost by a factor of Ω as follow:

$$\Omega = \frac{n^2CC'}{n^2C+CC'} \tag{2}$$

Table 1 shows the comparison of the number of parameters between different types of convolutions. As shown, depth-wise dilated separable convolution is efficient and can learn representations from a large and efficient receptive field.

2.2 MobileNetv2 Architecture

MobileNetv2 [3] is an improved network based on MobileNetv1. Each improved module makes MobileNetv2 even more efficient and powerful. In the MobileNetv2 architecture, the depth-wise separable convolution has been modified as shown in Figure 3. There are three convolutional layers in the modified depth-wise separable convolution block. The first layer is a 1×1 point-wise convolution layer which is used to expand the number of channels of the input feature map before it goes into the depth-wise convolution layer. This layer is also called as expansion layer where the number of channels of input feature is always smaller than that of output feature. An expansion factor is defined to show how much the data gets expanded. By default, the expansion factor is set at six. The middle layer is a 3×3 depth-wise convolution layer that filters the input feature map as in the MobileNetv1 network. The final layer is a 1×1 point-wise convolution layer. This final convolution layer is used to project data with a high number of channels into a tensor with a much lower number of channels, thus making the number of channels of the input feature map smaller. The final convolution layer is also called a bottleneck layer because it reduces the amount of data that flows through the network. Figure 4 shows an example of filtering step in depth-wise separable convolution block used in MobileNetv2 where a

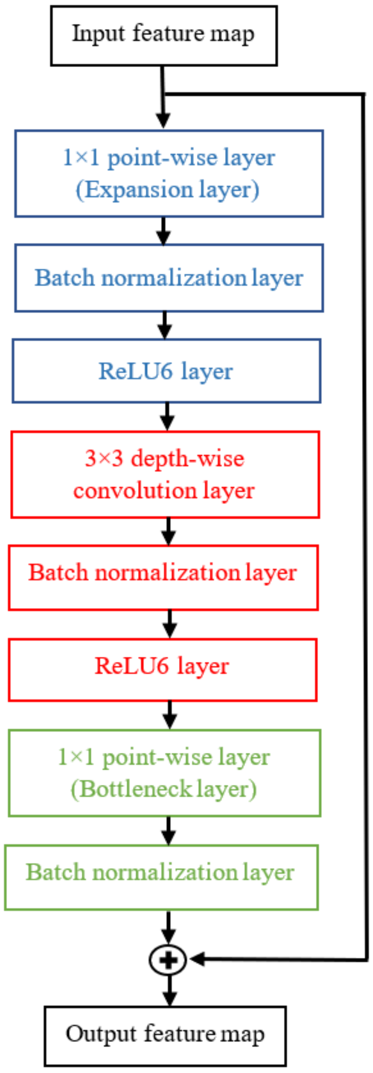

Figure 3: Depth-Wise Separable Convolution Used in MobileNetv2 Network.

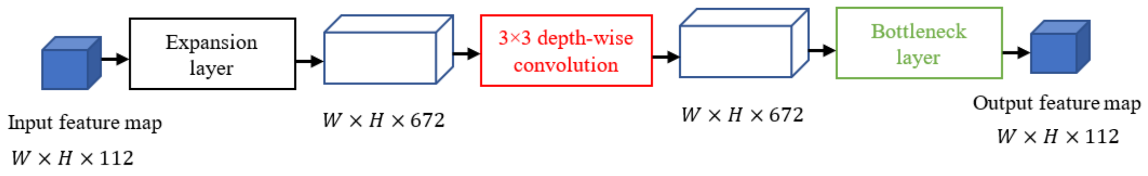

Figure 4: An Example of Filtering Step in Depth-Wise Separable Convolution Block.

feature map with 112 channels is fed into depth-wise separable convolution block. First, the expansion layer converts input feature map into a new feature map with 112*6 = 672 channels. Next, the depth-wise convolution layer applies filters to 673 channels feature. Finally, the final bottleneck layer projects the 672 channels feature map back to smaller channel.

In addition, the residual connection as in ResNet [4] is adopted in the depth-wise separable convolution block to help with the flow of gradients through the network. The residual connection is adopted when the number of channels going into the block is the same as the number of channels coming out of the block. Each layer in the modified depth-wise separable convolution is followed by batch normalization [5] and ReLU6 as activation function (except the last bottleneck layer since using a non-linearity after this layer destroyed useful information). Based on the modified depth-wise

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

Table 2: MobileNetv2 Architecture.

Layer	Type	Kernel size, Stride	Input size	Output size	Expansion factor
0	Standard convolution	3×3×32, 2	224×224×3	112×112×32	-
1	Depth-wise separable convolution	-, 1	112×112×32	112×112×16	1
2	Depth-wise separable convolution	-, 2	112×112×16	56×56×24	6
3	Depth-wise separable convolution	-, 1	56×56×24	56×56×24	6
4	Depth-wise separable convolution	-, 2	56×56×24	28×28×32	6
5	Depth-wise separable convolution	-, 1	28×28×32	28×28×32	6
6	Depth-wise separable convolution	-, 1	28×28×32	28×28×32	6
7	Depth-wise separable convolution	-, 2	28×28×32	14×14×64	6
8	Depth-wise separable convolution	-, 1	14×14×64	14×14×64	6
9	Depth-wise separable convolution	-, 1	14×14×64	14×14×64	6
10	Depth-wise separable convolution	-, 1	14×14×64	14×14×64	6
11	Depth-wise separable convolution	-, 1	14×14×64	14×14×96	6
12	Depth-wise separable convolution	-, 1	14×14×96	14×14×96	6
13	Depth-wise separable convolution	-, 1	14×14×96	14×14×96	6
14	Depth-wise separable convolution	-, 2	14×14×96	7×7×160	6
15	Depth-wise separable convolution	-, 1	7×7×160	7×7×160	6
16	Depth-wise separable convolution	-, 1	7×7×160	7×7×160	6
17	Depth-wise separable convolution	-, 1	7×7×160	7×7×320	6
18	Standard convolution	1×1×1280, 1	7×7×320	7×7×1280	-
19	Average pooling	7×7	7×7×1280	1×1×1280	-
20	Standard convolution	1×1×k, 1	1×1×1280	1×k	-

separable convolution block, the full MobileNetv2 architecture is shown in Table 2. As shown in Table 2, MobileNetv2 consists of 17 of the modified depthwise separable convolution blocks followed by a regular 1×1 convolution layer. The first layer is a regular 3×3 convolution layer with 32 channels.

2.3 Proposed Network Architecture Based on MobileNetv2 and Depth-wise Dilated Separable Convolution

Taking advantage of depth-wise dilated separable convolution and MobileNetv2 architecture, this paper introduces pyramid depth-wise dilated separable convolution by integrating depth-wise dilated convolution into depth-wise separable convolution in original MobileNetv2 network. Figure 5 presents the structure of pyramid depth-wise dilated separable convolution. In the pyramid depth-wise dilated separable convolution, low-dimensional input feature map is first projected to high-dimensional feature map by using pointwise convolution layer as in depth-wise dilated separable convolution. Then, the representations in high-dimensional feature map are learned in parallel by using dilated convolutions with different dilation rates. Different dilation rates in each branch allow the pyramid depth-wise dilated separable convolution to learn the representations from a large effective receptive field. This factorization allows the pyramid depth-wise dilated separable convolution to be efficient. Next, the output features of each branch are fused by using concatenation operation. Finally, the final bottleneck layer projects the high-dimensional fused feature map back to smaller channel.

Based on the proposed pyramid depth-wise dilated separable convolution and MobileNetv2 architecture, the architecture of the proposed network is shown in Table 3. At each spatial level, the proposed network repeats pyramid depth-wise dilated separable convolution several times to increase the depth of the network. In addition, PReLU [6] is used after every convolution layer with an exception to the last point-wise convolution layer.

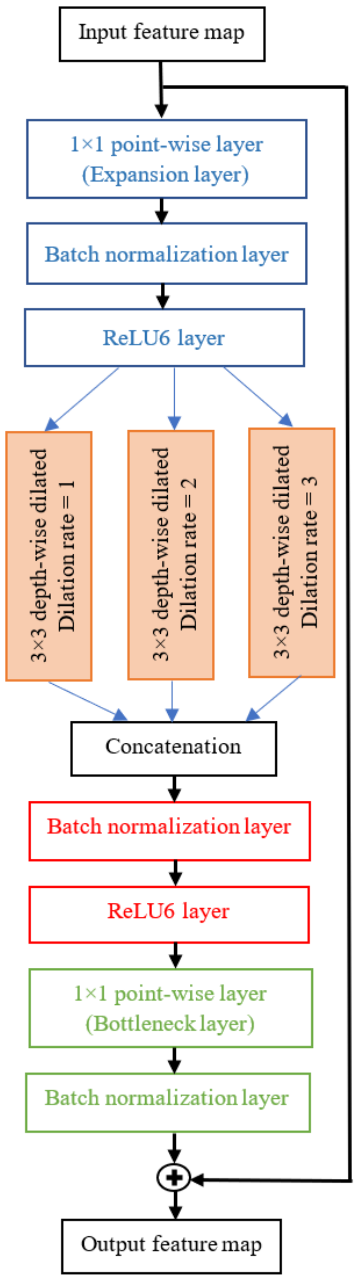

Figure 5: The Structure of Pyramid Depth-Wise Dilated Separable Convolution.

3. EXPERIMENTS AND RESULTS

3.1 Dataset

To evaluate the effectiveness of the proposed model, this paper conducts experiments on the Caltech-101 dataset [7] and Caltech-256 dataset [8]. Caltech-101 dataset contains 9,145 images with 102 classes, which are 101 object classes and one background class. The number of images in each class is between 40 to 800. Caltech-256 dataset is based on Caltech-101 dataset with more images and classes. There are 30,607 images in this dataset with 257 classes, which are 256 object classes and one background class. Figure 6 presents some images in Caltech-101 dataset (a) and Caltech-256 dataset (b). In each dataset, 2,000 images are randomly selected as testing images. The remaining labeled images are used for training.

3.2 Implementation Details

The proposed model is trained and tested using Python with Pytorch deep learning framework [9].

Table 3: The Architecture of The Proposed Network.

Layer	Type	Kernel size, Stride	Input size	Output size	# Repeat
0	Standard convolution	3×3×32, 2	224×224×3	112×112×32	1
1	Pyramid depth-wise dilated separable convolution	-, 1	112×112×32	112×112×16	1
2	Pyramid depth-wise dilated separable convolution	-, 2	112×112×16	56×56×24	1
3	Pyramid depth-wise dilated separable convolution	-, 1	56×56×24	56×56×24	1
4	Pyramid depth-wise dilated separable convolution	-, 2	56×56×24	28×28×32	1
5	Pyramid depth-wise dilated separable convolution	-, 1	28×28×32	28×28×32	2
7	Pyramid depth-wise dilated separable convolution	-, 2	28×28×32	14×14×64	1
8	Pyramid depth-wise dilated separable convolution	-, 1	14×14×64	14×14×96	6
14	Pyramid depth-wise dilated separable convolution	-, 2	14×14×96	7×7×160	1
15	Pyramid depth-wise dilated separable convolution	-, 1	7×7×160	7×7×320	3
18	Standard convolution	1×1×1280, 1	7×7×320	7×7×1280	1
19	Average pooling	7×7	7×7×1280	1×1×1280	1
20	Standard convolution	1×1×k, 1	1×1×1280	1×k	1

The proposed model is implemented on a Window system machine with CPU Intel Core i7-8700 @3.2 GHz, GPU Nvidia GTX 1080, RAM 12GB DDR4, and CUDA 10.1 with cuDNN back-ends. To initialize the weights of the network, this paper uses the method described in [6]. The network is trained with a batch size of 512 for 300 epochs by optimizing the cross-entropy loss. For faster convergence, this paper decays the learning rate by a factor of two at the following epoch intervals: {50, 100, 130, 160, 190, 220, 250, 280}, and the initial learning rate is set at 0.045.

3.3 Analysis of Experimental Results

In this section, this paper conducts experiments on Caltech-101 dataset and Caltech-256 dataset and compares the classification results with other state-of-the-are models, including MobileNetv1 [1], MobileNetv2 [3], and ShuffleNet [10].

Table 4 provides a performance comparison between the proposed model and other state-of-the-art models on Caltech-101 dataset. It can be observed that the accuracy of classification models has reached a balance after 25,000 training steps. For the classification accuracy, the proposed model outperforms other models on Caltech-101 dataset with the same training step. More specific, compared with MobileNetv1 network, the accuracy of the proposed model is improved by 2.5% after 25,000 training steps. Compared with MobileNetv2 and Shufflenet network, the accuracy of the proposed model is improved by 0.3% and 1.4% respectively after 25,000 training steps. When increasing the training step to 45,000, the accuracy of the proposed model increases by 0.7%, while the accuracy of MobileNetv1 decreases by 0.2%. For other models, the accuracy of MobileNetv2 increases to 79.8%, and the accuracy of Shufflenet increases to 78.7% at this training step.

Table 5 presents the classification accuracy of four classification models on Caltech-256 dataset. As shown, the classification accuracy of all models is almost steady after 30,000 training steps. Compared with other models, the proposed model achieves the best classification results. With training step at 30,000, the accuracy of the proposed model is improved by 7.5%, 0.9%, and 2.7% compared with MobileNetv1, MobileNetv2, and Shufflenet respectively. After 50,000 training steps, the accuracy of the proposed model is improved by 7.8%, 0.1%, and 2% compared with MobileNetv1, MobileNetv2, and Shufflenet respectively.

The classification results on two datasets show the effectiveness of the proposed model. With depth-

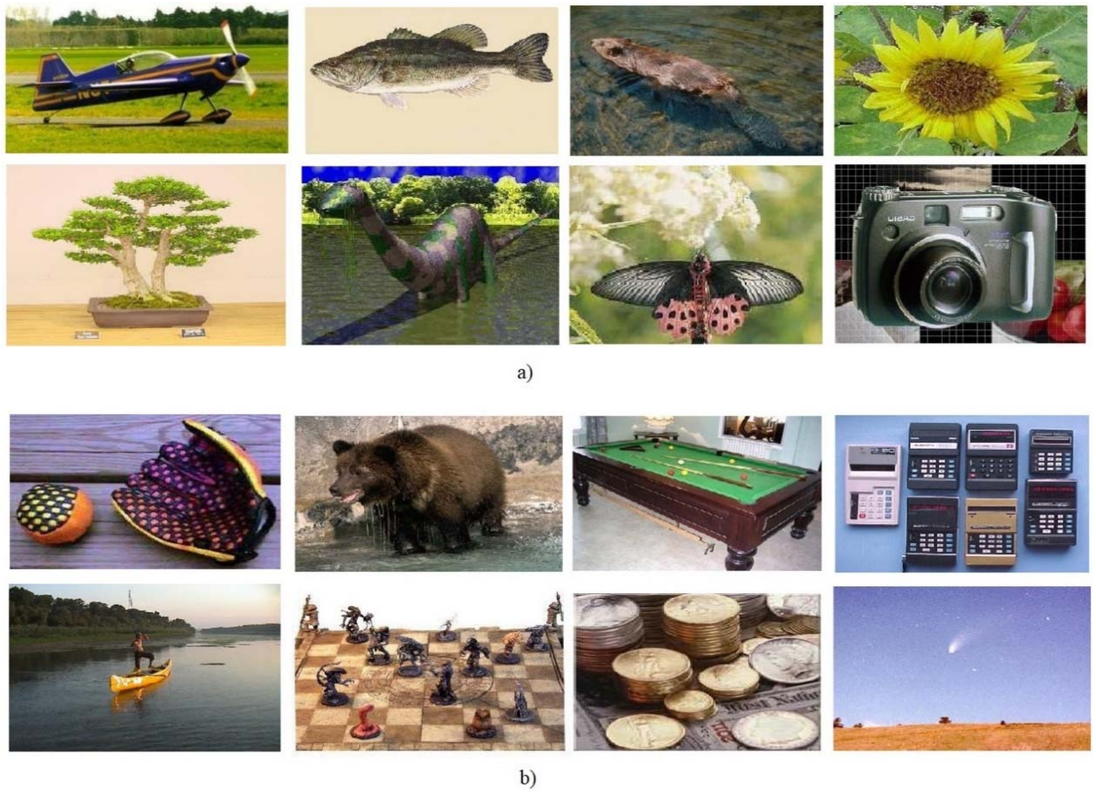

Figure 6: Example Images in Caltech-101 Dataset (a) and Caltech-256 Dataset (b).

Table 4: Classification Accuracy (%) on The Caltech-101 Dataset.

Model	Number of steps				
	25,000	30,000	35,000	40,000	45,000
MobileNetv1	76.6	75.7	76.8	76.8	76.4
MobileNetv2	79.3	79.5	79.8	79.8	79.8
Shufflenet	77.3	78.4	78.1	79.2	78.7
Proposed model	79.4	79.5	79.9	80.1	80.1

Table 5: Classification Accuracy (%) on The Caltech-256 Dataset.

Model	Number of steps				
	30,000	35,000	40,000	45,000	50,000
MobileNetv1	64.6	64.6	64.5	64.7	64.6
MobileNetv2	71.2	71.2	72.1	72.3	72.3
Shufflenet	69.4	70.2	70.5	70.4	70.4
Proposed model	72.1	72.3	72.3	72.4	72.4

wise dilated separable convolution, the receptive field in each layer is enlarged while maintaining the computation cost. Thus, the accuracy of the model is improved compared with depth-wise separable convolution in original models.

3.4 Application of Object Detection on Embedded System

To evaluate the effectiveness of the proposed model in object detection, this paper implements the proposed model for vehicle detection on embedded system. For vehicle detection framework, this paper adopts RPN [11] to design a lightweight and

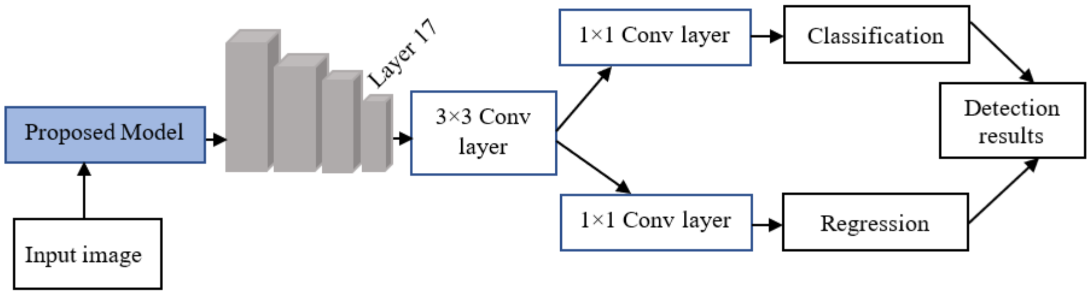

Figure 7: The Structure of The Proposed Detector.

Figure 8: Jetson TX2 Board.

Table 6: 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

efficient detector. Figure 7 presents the structure of the proposed detector. As shown, the proposed lightweight model is used as the base network. The RPN includes a 3×3 convolution layer followed by two 1×1 convolution layers. The base network is used to generate base feature maps, and the RPN is used to generate vehicle proposals and classify and regress the coordinates of detected vehicles. Feature map generated after Layer 17 of the base network is adopted to detect vehicle.

For the hardware components of the proposed embedded system, this paper uses the most power-efficient embedded AI computing device, the Jetson TX2 board as shown in Figure 8. Table 6 presents the main technical specifications of the board. The Jetson TX2 is a recent embedded board developed

Table 7: Detection Results of The Proposed Detector on KITTI Dataset with NVIDIA Jetson TX2.

Model	Base network	mAP (%)	Inference time (s)
SSD	VGG-16	68.4	0.4
Proposed detector	Proposed model	65.8	0.1

by NVIDIA in the embedded system category. This mini-computer provides 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 implement the proposed detector on the NVIDIA Jetson TX2 board, this paper uses the Pytorch deep learning framework [9] compiled for GPU and Python programming language.

For the experimental dataset, this paper adopts KITTI dataset [12] for vehicle detection. KITTI dataset is a recent public dataset for evaluating the performance of different vehicle detection approaches. The dataset 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. For evaluating the results, this paper adopts mean average precision (mAP), which is calculated by averaging the average precision of all three groups.

In Table 7, this paper presents the mAP and the inference time of the proposed detector and SSD [13] on NVIDIA Jetson TX2 with KITTI dataset. According to Table 7, the proposed detector with the proposed lightweight architecture as the base network achieves mAP at 65.8% on KITTI dataset. Because SSD detector used detection network on different layers of the based network, this detector achieves better result compared with the proposed detector. However, the proposed detector takes 0.1 second to process an image, while SSD detector takes up to 0.4 second. These results show the effectiveness of the proposed model on different visual tasks with a trade-off between accuracy and inference speed.

4. CONCLUSIONS

This paper proposes a lightweight and efficient deep convolutional neural network based on depthwise dilated separable convolution and MobileNetv2 architecture. In the proposed model, depthwise dilated convolution is used to replace standard depthwise convolution. Depthwise dilated convolution effectively enlarge the field of view of filters to incorporate larger context without increasing the number of parameters or the amount of computation. In addition, a pyramid depth-wise dilated separable convolution is introduced, where feature maps are learned in parallel by using dilated convolutions with different dilation rates. Experimental results on Caltech-101 dataset and Caltech-256 dataset show that the proposed model achieves better accuracy compared with MobileNetv2. Furthermore, an object detection framework based on the proposed model is designed and implemented on the NVIDIA Jetson TX2. Experiment results show the effectiveness of the proposed model on different vision tasks.

REFERENCES:

[1] 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).

[2] Krizhevsky, Alex, Ilya Sutskever, and Geoffrey E. Hinton. "Imagenet classification with deep convolutional neural networks." In *Advances in neural information processing systems*, pp. 1097-1105. 2012.

[3] 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.

[4] He, Kaiming, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. "Deep residual learning for image recognition." In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770-778. 2016.

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

[6] 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.

[7] Fei-Fei, Li, Rob Fergus, and Pietro Perona. "Learning generative visual models from few training examples: An incremental bayesian approach tested on 101 object categories." In *2004 conference on computer vision and pattern recognition workshop*, pp. 178-178. IEEE, 2004.

[8] Griffin, Gregory, Alex Holub, and Pietro Perona. "Caltech-256 object category dataset." (2007).

[9] Paszke, Adam, Sam Gross, Soumith Chintala, and Gregory Chanan. "Pytorch: Tensors and dynamic neural networks in python with strong gpu acceleration." *PyTorch: Tensors and dynamic neural networks in Python with strong GPU acceleration* 6 (2017).

[10] 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.

[11] 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.

[12] 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.

[13] 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.

[14] Hubara, Itay, Matthieu Courbariaux, Daniel Soudry, Ran El-Yaniv, and Yoshua Bengio. "Binarized neural networks." In *Advances in neural information processing systems*, pp. 4107-4115. 2016.

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

[16] Szegedy, Christian, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. "Going deeper with convolutions." In Proceedings of the IEEE conference on computer vision and pattern recognition, pp. 1-9. 2015.

[17] Szegedy, Christian, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna. "Rethinking the inception architecture for computer vision." In Proceedings of the IEEE conference on computer vision and pattern recognition, pp. 2818-2826. 2016.

[18] Chollet, François. "Xception: Deep learning with depthwise separable convolutions." In Proceedings of the IEEE conference on computer vision and pattern recognition, pp. 1251-1258. 2017.

[19] Xie, Saining, Ross Girshick, Piotr Dollár, Zhuowen Tu, and Kaiming He. "Aggregated residual transformations for deep neural networks." In Proceedings of the IEEE conference on computer vision and pattern recognition, pp. 1492-1500. 2017.

[20] Rastegari, Mohammad, Vicente Ordonez, Joseph Redmon, and Ali Farhadi. "Xnor-net: Imagenet classification using binary convolutional neural networks." In *European conference on computer vision*, pp. 525-542. Springer, Cham, 2016.

[21] He, Yihui, Xiangyu Zhang, and Jian Sun. "Channel pruning for accelerating very deep neural networks." In *Proceedings of the IEEE International Conference on Computer Vision*, pp. 1389-1397. 2017.

[22] Huang, Gao, Zhuang Liu, Laurens Van Der Maaten, and Kilian Q. Weinberger. "Densely connected convolutional networks." In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 4700-4708. 2017.

[23] Iandola, Forrest N., Song Han, Matthew W. Moskewicz, Khalid Ashraf, William J. Dally, and Kurt Keutzer. "SqueezeNet: AlexNet-level accuracy with 50x fewer parameters and< 0.5 MB model size." *arXiv preprint arXiv:1602.07360* (2016).