

# DiCENet: Dimension-Wise Convolutions for Efficient Networks

Sachin Mehta<sup>ID</sup>, *Student Member, IEEE*, Hannaneh Hajishirzi<sup>ID</sup>, and Mohammad Rastegari

**Abstract**—We introduce a novel and generic convolutional unit, DiCE unit, that is built using dimension-wise convolutions and dimension-wise fusion. The dimension-wise convolutions apply light-weight convolutional filtering across each dimension of the input tensor while dimension-wise fusion efficiently combines these dimension-wise representations; allowing the DiCE unit to efficiently encode spatial and channel-wise information contained in the input tensor. The DiCE unit is simple and can be seamlessly integrated with any architecture to improve its efficiency and performance. Compared to depth-wise separable convolutions, the DiCE unit shows significant improvements across different architectures. When DiCE units are stacked to build the DiCENet model, we observe significant improvements over state-of-the-art models across various computer vision tasks including image classification, object detection, and semantic segmentation. On the ImageNet dataset, the DiCENet delivers 2-4 percent higher accuracy than state-of-the-art manually designed models (e.g., MobileNetv2 and ShuffleNetv2). Also, DiCENet generalizes better to tasks (e.g., object detection) that are often used in resource-constrained devices in comparison to state-of-the-art separable convolution-based efficient networks, including neural search-based methods (e.g., MobileNetv3 and MixNet).

**Index Terms**—Deep convolutional neural network, image classification, object detection, semantic segmentation, efficient networks

## 1 INTRODUCTION

THE basic building layer at the heart of convolutional neural networks (CNNs) is a convolutional layer that encodes spatial and channel-wise information simultaneously [1], [2], [3]. Learning representations using this layer is computationally expensive. Improving the efficiency of CNN architectures as well as convolutional layers is an active area of research. Most recent attempts have focused on improving the efficiency of CNN architectures using compression- and quantization-based methods (e.g., [4], [5]). Recently, several factorization-based methods have been proposed to improve the efficiency of standard convolutional layers (e.g., [6], [7], [8]). In particular, depth-wise separable convolutions [7], [9] have gained a lot of attention (see Fig. 1a). These convolutions have been used in several efficient state-of-the-art architectures, including neural search-based architectures [10], [11], [12], [13], [14].

Separable convolutions factorize the standard convolutional layer in two steps: (1) a light-weight convolutional filter is applied to each input channel using depth-wise convolutions [9] to learn spatial representations, and (2) a point-wise ( $1 \times 1$ ) convolution is applied to learn linear combinations between spatial representations. Though depth-wise convolutions are efficient, they do not encode channel-wise relationships. Therefore, separable convolutions rely on point-wise convolutions to encode channel-wise relationships. This puts a significant computational

load on point-wise convolutions and makes them a computational bottleneck. For example, point-wise convolutions account for about 90 percent of total operations in ShuffleNetv2 [11] and MobileNetv2 [10].

In this paper, we introduce DiCENet, Dimension-wise Convolutions for Efficient Networks that encodes spatial and channel-wise representations efficiently. Our main contribution is the novel and generic module, the DiCE unit (Fig. 1a), that is built using Dimension-wise Convolutions (DimConv) and Dimension-wise Fusion (DimFuse). The DimConv applies a light-weight convolutional filter across “each dimension” of the input tensor to learn local dimension-wise representations while DimFuse efficiently combines these dimension-wise representations to incorporate global information.

With DimConv and DimFuse, we build an efficient convolutional unit, the DiCE unit, that can be easily integrated into existing or new CNN architectures to improve their performance and efficiency. Fig. 1 shows that the DiCE unit is effective in comparison to widely-used separable convolutions. Compared to state-of-the-art manually designed networks (e.g., MobileNet [7], MobileNetv2 [10], and ShuffleNetv2 [11]), DiCENet delivers significantly better performance. For example, for a network with about 300 million floating point operations (FLOPs), DiCENet is about 4 percent more accurate than MobileNetv2. Importantly, DiCENet, a manually designed network, delivers similar or better performance than neural architecture search (NAS) based methods. For example, DiCENet is 1.7 percent more accurate than MNASNet for a network with about 300 MFLOPs.

We empirically demonstrate in Sections 6 and 7 that the DiCENet network, built by stacking DiCE units, achieves significant improvements on standard benchmarks across different tasks over existing networks. Compared to existing efficient networks, DiCENet generalizes better to tasks (e.g.,

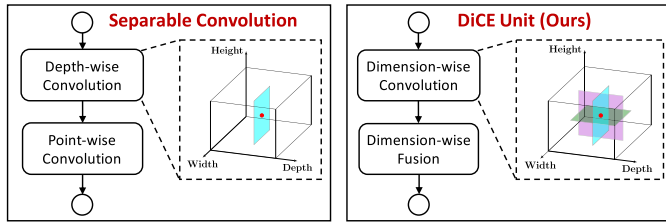
• The authors are with the University of Washington, Seattle, WA 98195 USA. E-mail: {sacmehta, hannaneh, mrast}@cs.washington.edu.

Manuscript received 26 Mar. 2020; revised 11 Oct. 2020; accepted 29 Nov. 2020. Date of publication 2 Dec. 2020; date of current version 1 Apr. 2022.

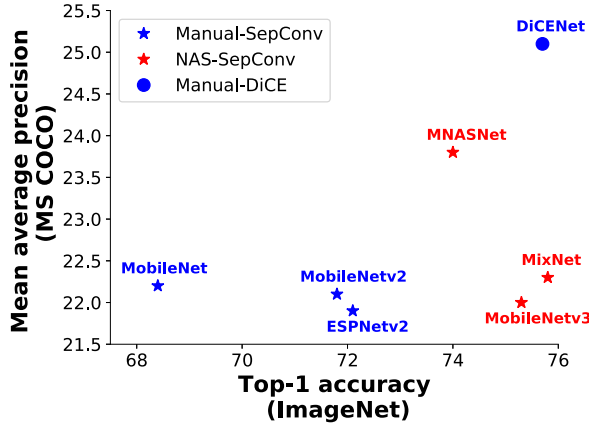
(Corresponding author: Sachin Mehta.)

Recommended for acceptance by R. Feris.

Digital Object Identifier no. 10.1109/TPAMI.2020.3041871



(a) Block of separable convolutions [7] and the DiCE unit. Convolutional kernels are highlighted in color (depth-, width-, and height-wise).



(b) Network with the DiCE unit (DiCENet) has better task-level generalization properties compared to networks with separable convolutions (MobileNet [7], MobileNetV2 [10], MobileNetV3 [15], MixNet [16], MNASNet [13]). Here, Manual-SepConv and NAS-SepConv represents the models that uses separable convolution without and with neural architecture search (NAS), respectively. Manual-DiCE represents the DiCENet without NAS. On the ImageNet dataset, these networks have about 200-300 million floating point operations. See Section 6 for more details.

Fig. 1. Separable convolutions versus the DiCE unit.

object detection) that are often used in resource-constrained devices. For instance, DiCENet achieves about 3 percent higher mean average precision than MobileNetV3 [15] and MixNet [16] on the MS-COCO object detection task with SSD [17] as a detection pipeline (Fig. 1b). Our source code in PyTorch is open-source and is available at <https://github.com/sacmehta/EdgeNets/>.

## 2 RELATED WORK

**CNN Architecture Designs.** Recent success in visual recognition tasks, including classification, detection, and segmentation, can be attributed to the exploration of different CNN designs (e.g., [1], [2], [3], [7]). The basic building layer in these networks is a standard convolutional layer, which is computationally expensive. Factorization-based methods improve the efficiency of these layers. For instance, flattened convolutions [6] approximate a standard convolutional layer with three point-wise convolutions that are applied sequentially, one point-wise convolution per tensor dimension. These convolutions ignore spatial relationships between pixels and do not generalize across a wide variety of computer vision tasks (e.g., detection and segmentation) and large scale datasets (e.g., ImageNet and MS-COCO). To improve the efficiency of standard convolutions while maintaining the performance and generalization ability at scale, depth-wise separable convolutions [7] are proposed that factorizes the standard convolutional layer into depth-wise and point-wise convolution

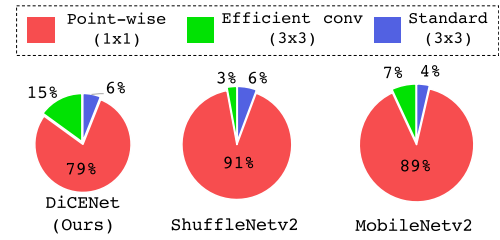


Fig. 2. Convolution-wise distribution of FLOPs for different networks with similar accuracy. The size of pie charts is scaled with respect to MobileNetV2's FLOPs. In DiCENet, efficient conv's correspond to dimension-wise convolutions while in other networks, they correspond to depth-wise convolutions.

layers. Most of the efficient CNN architectures are built using these separable convolutions, including MobileNets [7], [10], ShuffleNets [11], [18], and ESPNetV2 [12]. In this work, we introduce dimension-wise convolutions that generalize depth-wise convolutions to all dimensions of the input tensor. We also introduce an efficient way for combining these dimension-wise representations. As confirmed by our experiments in Sections 5, 6, and 7, the DiCE unit is more effective than separable convolutions.

**Neural Architecture Search.** Recently, neural architecture search-based methods have been proposed to automatically construct network architectures (e.g., [13], [14], [15], [19], [20]). These methods search over a large network space (e.g., MNASNet [13] searches over 8K different design choices) using a dictionary of pre-defined search space parameters, including different types of convolutional layers and kernel sizes, to find a heterogeneous network structure that satisfies optimization constraints, such as inference time. The proposed unit is novel and cannot be discovered using existing neural search-based methods (e.g., [13], [14], [21]). However, we believe that better neural architectures can be discovered by adding the DiCE unit in neural search dictionary.

**Quantization, Compression, and Distillation.** Network quantization-based approaches [22], [23], [24], [25] approximate 32-bit full precision convolution operations with fewer bits. This improves inference speed and reduces the amount of memory required for storing network weights. Network compression-based approaches [4], [5], [26], [27], [28] improve the efficiency of a network by removing redundant weights and connections. Unlike network quantization and compression, distillation-based approaches [29], [30], [31] improve the accuracy of (usually shallow) networks by supervising the training with large pre-trained networks. These approaches are effective for improving the efficiency of a network, including efficient architecture designs (e.g., [32], [33]) and are orthogonal to our work. We believe that the efficiency of DiCENet can be further improved using these methods.

## 3 DICENET

Standard convolutions encode spatial and channel-wise information *simultaneously*, but they are computationally expensive. To improve the efficiency of standard convolutions, separable (or depth-wise separable) convolutions are introduced [7], where spatial and channel-wise information is encoded *separately* using depth-wise and point-wise convolutions, respectively. Though this factorization is effective,

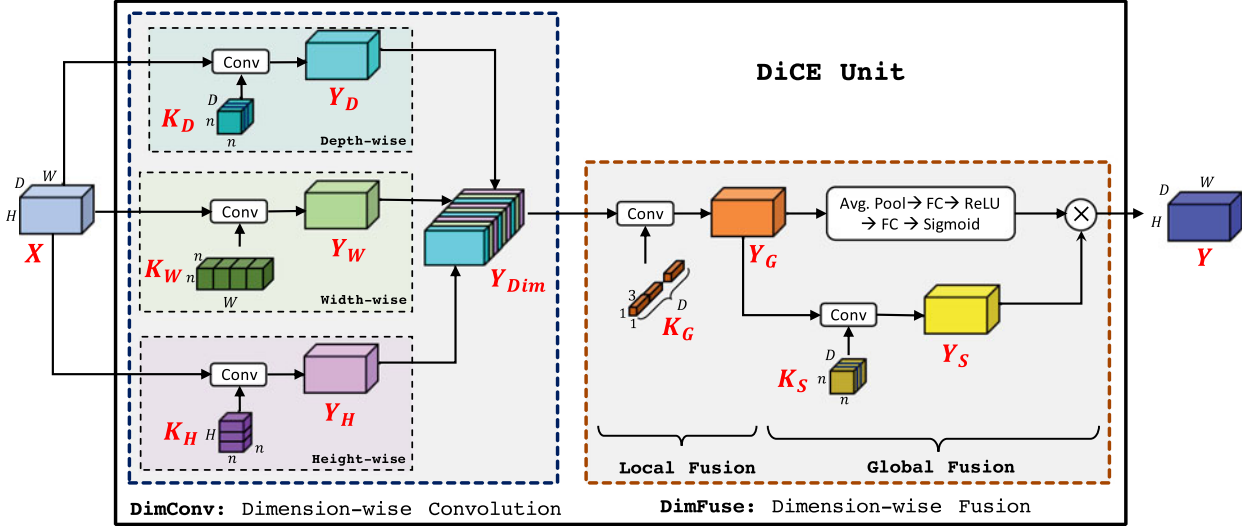


Fig. 3. *DiCE* unit efficiently encodes the spatial and channel-wise information in the input tensor  $X$  using dimension-wise convolutions (DimConv) and dimension-wise fusion (DimFuse) to produce an output tensor  $Y$ . For simplicity, we show kernel corresponding to each dimension independently. However, in practice, these three kernels are executed simultaneously, leading to faster run-time. See Section 3.4 and 6 for more details.

it puts a significant computational load on point-wise convolutions and makes them a computational bottleneck (see Fig. 2).

To encode spatial and channel-wise information efficiently, we introduce the *DiCE* unit and is shown in Fig. 3. The *DiCE* unit factorizes standard convolution using Dimension-wise Convolution (DimConv, Section 3.1) and Dimension-wise Fusion (DimFuse, Section 3.2). DimConv applies a light-weight filtering across each dimension of the input tensor to learn local dimension-wise representations. DimFuse efficiently combines these representations from different dimensions and incorporates global information. The ability to encode local spatial and channel-wise information from all dimensions using DimConv enables the *DiCE* unit to use DimFuse instead of computationally expensive point-wise convolutions.

### 3.1 Dimension-Wise Convolution (DimConv)

We use dimension-wise convolutions (DimConv) to encode depth-, width-, and height-wise information *independently*. To achieve this, DimConv extends depth-wise convolutions to *all dimensions* of the input tensor  $X \in \mathbb{R}^{D \times H \times W}$ , where  $W$ ,  $H$ , and  $D$  corresponds to width, height, and depth of  $X$ . As illustrated in Fig. 3, DimConv has three branches, one branch per dimension. These branches apply  $D$  depth-wise convolutional kernels  $k_D \in \mathbb{R}^{1 \times n \times n}$  along depth,  $W$  width-wise convolutional kernels  $k_W \in \mathbb{R}^{n \times n \times 1}$  along width, and  $H$  height-wise convolutional kernels  $k_H \in \mathbb{R}^{n \times 1 \times n}$  kernels along height to produce outputs  $Y_D, Y_W$ , and  $Y_H \in \mathbb{R}^{D \times H \times W}$  that encode information from all dimensions of the input tensor. The outputs of these independent branches are concatenated along the depth dimension, such that the first spatial plane of  $Y_D, Y_W$ , and  $Y_H$  are put together and so on, to produce the output  $Y_{Dim} = \{Y_D, Y_W, Y_H\} \in \mathbb{R}^{3D \times H \times W}$ .

### 3.2 Dimension-Wise Fusion (DimFuse)

The dimension-wise convolutions encode local information from different dimensions of the input tensor, but do not

capture global information. A standard approach to combine features globally in CNNs is to use a point-wise convolution [3], [7]. A point-wise convolutional layer applies  $D$  point-wise kernels  $k_p \in \mathbb{R}^{3D \times 1 \times 1}$  and performs  $3D^2HW$  operations to combine dimension-wise representations of  $Y_{Dim} \in \mathbb{R}^{3D \times H \times W}$  and produce an output  $Y \in \mathbb{R}^{D \times H \times W}$ . This is computationally expensive. Given the ability of DimConv to encode spatial and channel-wise information (though independently), we introduce a fusion module, Dimension-wise fusion (DimFuse), that allows us to combine representations of  $Y_{Dim}$  efficiently. As illustrated in Fig. 3, DimFuse factorizes the point-wise convolution in two steps: (1) local fusion and (2) global fusion.

$Y_{Dim} \in \mathbb{R}^{3D \times H \times W}$  concatenates spatial planes along depth dimension from  $Y_D, Y_W$ , and  $Y_H$  (see Fig. 3). Therefore,  $Y_{Dim}$  can be viewed as a tensor with  $D$  groups, each group with three spatial planes (one from each dimension). DimFuse uses a group point-wise convolutional layer to combine dimension-wise information contained in  $Y_{Dim}$ . In particular, this group convolutional layer applies  $D$  point-wise convolutional kernels  $k_G \in \mathbb{R}^{3 \times 1 \times 1}$  to  $Y_{Dim}$  and produces an output  $Y_G \in \mathbb{R}^{D \times H \times W}$ . Since  $D$  kernels in  $k_G$  operates independently on  $D$  groups in  $Y_{Dim}$ , we call this *local* fusion operation.

To efficiently encode the global information in  $Y_G$ , DimFuse learns spatial and channel-wise representations independently and then propagate channel-wise encodings to spatial encodings using an element-wise multiplication. Specifically, DimFuse encodes spatial representations by applying  $D$  depth-wise convolutional kernels  $k_S \in \mathbb{R}^{1 \times n \times n}$  to  $Y_G$  to produce an output  $Y_S$ .<sup>1</sup> Motivated by Squeeze-Excitation (SE) unit [34], we squeeze spatial dimensions of  $Y_G$  and encode channel-wise representations using two fully connected (FC) layers. The first FC layer reduces the input dimension from  $D$  to  $\frac{D}{4}$  while the second FC layer expands

1. When the depth of  $Y_{Dim}$  is different from  $Y$ , then  $k_S$  is a group convolution, where number of groups is the greatest common divisor between the depth of  $Y_{Dim}$  and  $Y$ .



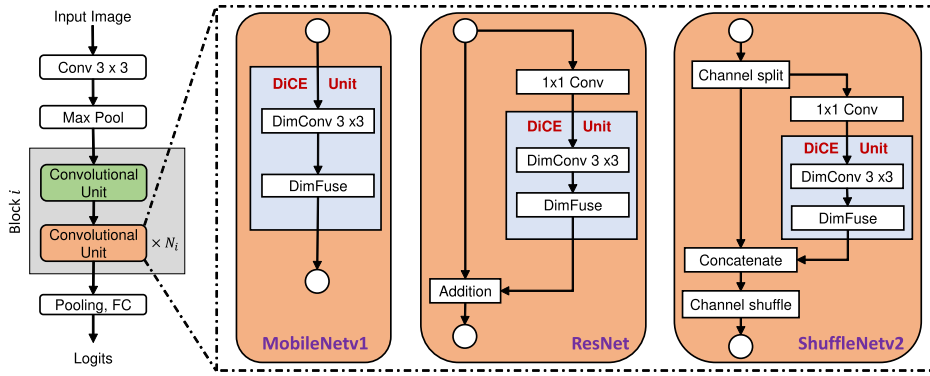


Fig. 4. *DiCE* unit in different architecture designs for the task of image classification on the ImageNet dataset. Green and boxes are with and without stride, respectively. Here,  $N_i = \{3, 7, 3\}$  for  $i = \{1, 2, 3\}$ . See supplementary material, which can be found on the Computer Society Digital Library at <http://doi.ieeecomputersociety.org/10.1109/TPAMI.2020.3041871>, for detailed architecture specification at different complexity levels.

dimensionality from  $\frac{D}{4}$  to  $D$ . To allow these fully connected layers to learn non-linear representations, a ReLU activation is added in between these two layers. Similar to the SE unit, spatial representations  $\mathbf{Y}_G$  are then scaled using these channel-wise representations to produce output  $\mathbf{Y}$ .

The computational cost of *DimFuse* is  $HWD(3 + n^2 + D)$ . Effectively, *DimFuse* reduces the computational cost of point-wise convolutions by a factor of  $\frac{3D}{3+n^2+D}$ . *DimFuse* uses  $n = 3$ , so the computational cost is approximately  $3\times$  smaller than that of the point-wise convolution.

### 3.3 DiCE Unit for Arbitrary Sized Inputs

The *DiCE* unit stacks *DimConv* and *DimFuse* to encode spatial and channel-wise information in the input tensor efficiently. However, the two kernels (i.e.,  $\mathbf{k}_H$  and  $\mathbf{k}_W$ ) in *DimConv* unit correspond to spatial dimensions of the input tensor. This may pose a challenge when spatial dimensions of the input tensor are different from the ones with which the network is trained. To make *DiCE* units invariant to spatial dimensions of the input tensor, we dynamically scale (either up-sample or down-sample) the height or width dimension of the input tensor to the height or width of the input tensor used in the pretrained network. The resultant tensors are then scaled (either down-sampled or up-sampled) back to their original size before being fed to *DimFuse*; this makes the *DiCE* unit invariant to input tensor size. Fig. 5 sketches the *DiCE* unit with dynamic scaling. Results in Section 7, especially object detection and semantic segmentation, show that the *DiCE* unit can handle arbitrary sized inputs.

### 3.4 DiCENet Architecture

*DiCE* units are generic and can be easily integrated in any existing network. Fig. 4 visualizes the *DiCE* unit with different architectures: (1) *MobileNet* [7] stacks separable convolutions

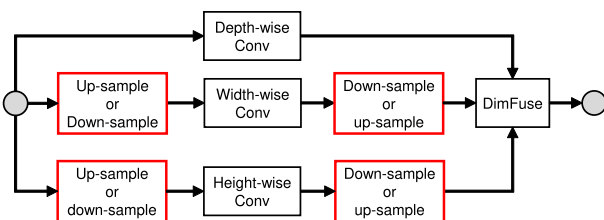


Fig. 5. *DiCE* unit for arbitrary sized input.

(depth-wise convolution followed by point-wise convolution) to learn representations. (2) *ResNet* [3] introduces the bottleneck unit with residual connections to train very deep networks. The bottleneck unit is a stack of three convolutional layers: one  $3 \times 3$  depth-wise convolutional layer<sup>2</sup> surrounded by two point-wise convolutions. This block can be viewed as a point-wise convolution followed by separable convolution. (3) *ShuffleNetv2* [11] is a state-of-the-art efficient network that outperforms other efficient networks, including *MobileNetv2* [10]. *ShuffleNetv2*'s unit stacks a point-wise convolution and separable convolution. It also uses channel split and shuffle to promote feature reuse.

To illustrate the performance benefits and generic nature of the *DiCE* unit over separable convolutions, we replace separable convolutions with the *DiCE* unit in these architectures. Our empirical results in Section 5 shows that the *DiCE* unit with *ShuffleNetv2*'s architecture delivers the best performance. Therefore, we choose the *ShuffleNetv2* [11] architecture and call the resultant network *DiCENet* (*ShuffleNetv2* with the *DiCE* unit).

**CUDA Implementation.** *DimConv* applies  $D$ ,  $W$ , and  $H$  depth-wise, width-wise and height-wise convolutional kernels to the input tensor  $\mathbf{X}$  to aggregate information from different dimensions of the tensor, respectively. A standard solution would be to apply each kernel independently to the tensor and then concatenate their results, as shown in Fig. 6a. Another solution would be to apply all kernels simultaneously, as shown in Fig. 6b. Compared to three CUDA kernel calls in the former solution, the later one requires one kernel call, thus reducing the kernel launch time. Also, each thread in the CUDA kernel process  $3n^2$  elements compared to  $n^2$  elements in the former solution for  $n \times n$  convolutional kernels. This maximizes the work done per thread and improves speed. Our results in Section 6 shows that *DiCENet* is accurate and fast compared to state-of-the-art methods, including neural search-based methods.

## 4 EXPERIMENTAL SET-UP

Following most architecture designs (e.g., [3], [7], [12]), we evaluate the generic nature of the *DiCE* unit on the ImageNet dataset in Section 5. We integrate the *DiCE* unit in different

2. Our focus is on efficient network. Therefore, we replace standard convolutional layer with depth-wise convolutional layer.

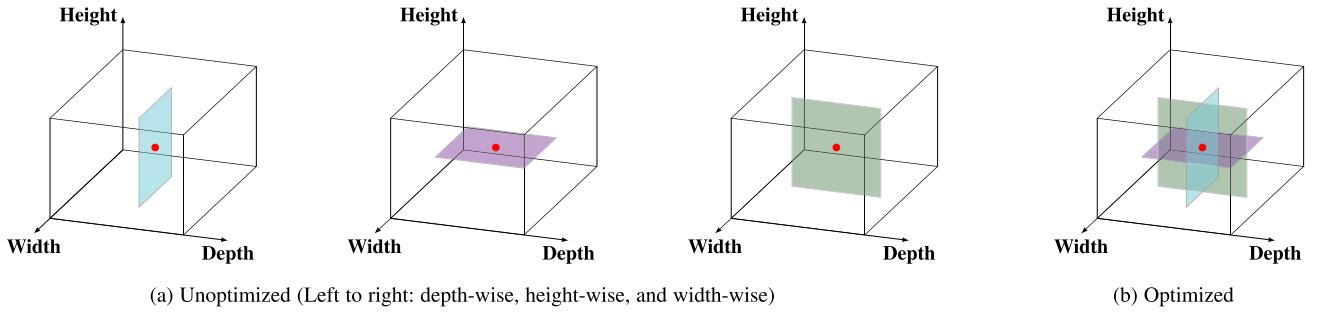


Fig. 6. Implementation of dimension-wise convolution (*DimConv*). In (a), each kernel is applied to a pixel (represented by red dot) independently. In (b), all kernels are applied to a pixel simultaneously, allowing us to aggregate the information from tensor efficiently. Convolutional kernels are highlighted in color (**depth**-, **width**-, and **height**-wise).

image classification architectures (Fig. 4) and study the impact on efficiency and accuracy. We also study the importance of the two main components of the DiCE unit, i.e., *DimConv* and *DimFuse*, and show that DiCE units are more effective than separable convolutions [7]. In Section 6, we evaluate the image classification performance of DiCENet on the ImageNet dataset and show that DiCENet delivers similar or better performance than state-of-the-art efficient networks, including neural search-based methods. In Section 7, we evaluate task-level generalization ability of DiCENet on three different visual recognition tasks, i.e., object detection, semantic segmentation, and multi-object classification, that are often used in resource-constrained devices. We demonstrate that DiCENet generalizes better than existing efficient networks that are built using separable convolutions.

*Datasets.* We use following datasets in our experiments.

*Image classification:* For single label image classification, we use ImageNet-1K classification dataset [35]. This dataset consists of 1.28M training and 50K validation images. All networks on this dataset are trained from scratch. For multi-label classification, we use MS-COCO dataset [36] that has 2.9 labels (on an average) per image. We use the same training and validation splits as in [12].

*Object detection:* We use MS-COCO [36] and PASCAL VOC 2007 [37] datasets for evaluating on the task of object detection. Following a standard convention for training on PASCAL VOC 2007 dataset, we augment it with PASCAL VOC 2012 [38] and PASCAL VOC 2007 *trainval* set for training and evaluate the performance on PASCAL VOC 2007 *test* set.

*Semantic Segmentation.* We use PASCAL VOC 2012 [38] dataset for this task. Following a standard convention, we use additional images for training from [39] and [36]. Similar to MobileNetV2 [10], we evaluate the performance on the validation set.

*Efficiency Metric.* We measure efficiency in terms of the number of floating point operations (FLOPs) and inference time. We use PyTorch for training our networks.

## 5 EVALUATING DiCE UNIT ON IMAGENET

We first evaluate the two important properties of the DiCE unit, i.e., generic and efficiency, in Section 5.1. To evaluate this, we replace separable convolutions [7] in different architectures with the DiCE unit (Fig. 4). We then study the importance of each component of the DiCE unit,

*DimConv* and *DimFuse*, in Section 5.2. Recent studies (e.g., MobileNetV3 [15] and MNASNet [13]) uses several different methods, such as exponential moving average (EMA) and large batch sizes, to improve the performance. In Section 5.3, we study the effect of these methods on the performance of DiCENet.

In these experiment, we use the ImageNet dataset [35]. In Sections 5.1 and 5.2, we follow the experimental setup of ESPNetV2 [12] and ShuffleNetV2 [11] (fewer training epochs with smaller batch size) while in Section 5.3, we follow experimental set-up similar to MobileNets [10], [15] (longer training with larger batch size).

### 5.1 DiCE Unit versus Separable Convolutions

Table 1 shows the performance of the DiCE unit with different architectures at different FLOP ranges. When separable convolutions are replaced with the DiCE unit in MobileNet architecture, we observe significant gains in performance both in terms of accuracy and efficiency. Since this architecture does not employ any advanced methods (e.g., residual connections and channel shuffle) to improve performance, it allows us to understand the “true” gains of the DiCE unit over separable convolutions.

TABLE 1  
Comparison Between the DiCE Unit and Separable Convolutions on the ImageNet Dataset Across Different Architectures

FLOP Range (in millions)	Separable conv (SC)		DiCE unit (DU)		Absolute difference (DU - SC)	
	Top-1	FLOPs	Top-1	FLOPs	Top-1	FLOPs
<b>MobileNet [7]</b>						
25-60	49.80	41 M	<b>52.55</b>	29 M	<b>+2.75</b>	<b>-12 M</b>
120-170	65.30	<b>162 M</b>	<b>69.05</b>	167 M	<b>+3.75</b>	<b>+5 M</b>
270-320	68.40	317 M	<b>70.83</b>	<b>277 M</b>	<b>+2.43</b>	<b>-40 M</b>
<b>ResNet [3]</b>						
25-60	59.30	59 M	<b>61.35</b>	52 M	<b>+2.05</b>	<b>-7 M</b>
120-170	67.80	142 M	<b>67.90</b>	122 M	<b>+0.10</b>	<b>-20 M</b>
270-320	70.67	302 M	<b>71.80</b>	<b>300 M</b>	<b>+1.13</b>	<b>-2 M</b>
<b>ShuffleNetV2 [11]</b>						
25-60	59.69	<b>41 M</b>	<b>62.80</b>	46 M	<b>+3.11</b>	<b>+5 M</b>
120-170	68.14	146 M	<b>68.21</b>	122 M	<b>+0.07</b>	<b>-24 M</b>
270-320	71.80	<b>292 M</b>	<b>72.90</b>	298 M	<b>+1.10</b>	<b>+6 M</b>

Models with the DiCE unit requires fewer channels compared to models with separable convolution in order to obtain similar performance. Thus, models with the DiCE unit has fewer FLOPs compared to separable convolutions.

TABLE 2  
Evaluating DiCE Unit on the ImageNet Dataset

Layer	ResNet		ShuffleNetv2	
	FLOPs	Top-1	FLOPs	Top-1
DWise + Point-wise	142 M	67.80	146 M	68.14
DimConv + Point-wise	<b>132 M</b>	<b>68.10</b>	<b>135 M</b>	<b>68.45</b>

(a) Importance of DimConv. DWise denotes depth-wise conv.

Layer	ResNet		ShuffleNetv2	
	FLOPs	Top-1	FLOPs	Top-1
DWise + Point-wise (Separable)	142 M	67.80	146 M	68.14
DWise + SE	137 M	63.90	140 M	64.70
DWise + Point-wise + SE	142 M	68.20	146 M	68.60
DWise + DimFuse	136 M	65.90	139 M	66.80
DimConv + Point-wise	132 M	<b>68.10</b>	135 M	<b>68.45</b>
DimConv + SE	134 M	64.80	138 M	65.40
DimConv + Point-wise + SE	132 M	67.90	135 M	68.30
DimConv + DimFuse (DiCE unit)	<b>122 M</b>	67.90	<b>122 M</b>	68.21

(b) Importance of DimFuse. DWise denotes depth-wise conv.

Top-1 accuracy is reported on the validation set. Models with DiCE unit requires fewer channels compared to models with separable convolution in order to obtain similar performance. Therefore, models with DiCE unit has fewer FLOPs compared to separable convolutions.

When we replace separable convolutions with the DiCE unit in ResNet and ShuffleNetv2, we observe significant improvements especially for small-(25-60 MFLOPs) and medium-sized (120-170 MFLOPs) models. For instance, the DiCE unit improved the performance of ShuffleNetv2 by about 3 percent for small-sized model (about 40 MFLOPs). Similarly, ShuffleNetv2 with the DiCE unit requires 24 million fewer FLOPs to achieve the same accuracy as with separable convolutions for medium-sized model (120-170 MFLOPs). These results suggests that the DiCE unit is generic and learns better representations than separable convolutions.

## 5.2 Importance of DimConv and DimFuse

To understand the significance of each component of the DiCE unit, we replace DimConv with depth-wise convolution and DimFuse with different fusion methods, including point-wise convolutions and squeeze-excitation (SE) unit [34] and study their combinations for two architectures (ResNet and ShuffleNetv2).<sup>3</sup> In these experiments, we study efficient models by restricting the computational budget between 120 and 150 MFLOPs.

*Importance of DimConv.* We replace depth-wise convolutional layers with DimConv in ResNet and ShuffleNetv2 architectures. Table 2 a shows that these networks with DimConv require about 10-11 million fewer FLOPs to achieve similar accuracy as the depth-wise convolution. These results suggest that encoding spatial and channel-wise information independently in DimConv helps learning better representations compared to encoding only spatial information in depth-wise convolution.

*Importance of DimFuse* To understand the effect of DimFuse, we replace DimFuse with two widely used fusion operations, i.e., point-wise convolution and SE unit. Table 2 b summarizes the results. Compared to the widely used

3. MobileNet's performance is significantly lower than ResNet and ShuffleNetv2, therefore, we do not use MobileNet for these experiments.

TABLE 3  
Impact of Replacing All Pointwise Convolutions With DimFuse. Here, DWise Denotes Depth-Wise Convolution

Row #	Layer	Network Width	ResNet		ShuffleNetv2	
			FLOPs	Top-1	FLOPs	Top-1
R1	Point-wise + DWise + DimFuse	1×	136 M	65.90	139 M	66.80
R2	DimFuse + DWise + DimFuse	1×	78 M	60.10	78 M	61.80
R3	DimFuse + DWise + DimFuse	4×	141 M	66.20	140 M	66.90
R4	Point-wise + DimConv + DimFuse	1×	122 M	67.90	122 M	68.21
R5	DimFuse + DimConv + DimFuse	1×	<b>72 M</b>	62.10	<b>72 M</b>	63.70
R6	DimFuse + DimConv + DimFuse	4×	129 M	<b>68.20</b>	132 M	<b>69.20</b>

combination of depth-wise and point-wise convolutions (or separable convolution), the combination of DimConv and DimFuse (or the DiCE unit) is the most effective and improves the efficiency of networks by 15-20 percent with little or no impact on accuracy.

The combination of depth-wise and DimFuse is not as effective as DimConv and DimFuse. DimConv encodes local spatial and channel-wise information, which enables the DiCE unit to use a less complex fusion method (DimFuse) for encoding global information. Unlike DimConv, depth-wise convolutions only encode local spatial information and require computationally expensive point-wise convolutions to encode global information.

When point-wise convolutions are replaced with SE unit, the performance of networks with depth-wise and DimConv convolutions dropped significantly. This is because SE unit relies on an existing convolutional unit, such as ResNext [40], to encode global spatial and channel-wise information. When SE unit is used as a *replacement* for point-wise convolutions, it fails to effectively encode this information; resulting in significant performance drop.

*DimFuse Replacing all Point-Wise Convolutions.* The first layer in ShuffleNetv2 and ResNet is a point-wise convolution (Fig. 4; Section 3.4). Table 2 b shows that DimFuse is an effective replacement for point-wise convolutions. A natural question arises if we can replace the first point-wise convolutional layer in these architectures with DimFuse. Table 3 shows the effect of replacing point-wise convolutions with DimFuse. For a fixed network width, the number of FLOPs are reduced by 45 percent when point-wise convolutions are replaced with DimFuse, however, the accuracy drops by about 5-7 percent. For similar number of FLOPs, networks with DimFuse achieves higher accuracy. However, such networks (R3 and R6) are about 4× wider than the networks with point-wise convolutions (R1 and R4); this poses memory constraints for resource-constrained devices. Therefore, we only replace separable convolutions with the DiCE unit while keeping the remaining architecture intact.

## 5.3 Effect of MobileNet's Training Hyper-Parameters

In previous experiments, we follow the experimental setup similar to ESPNetv2 and ShuffleNetv2 i.e., each model is trained for 150 epochs using a batch size of 512 and minimizes cross-entropy loss using SGD. Recent efficient models (e.g., MobileNetv3 and MNASNet) are trained longer (600 epochs) with extremely large batch size (4096) using cross-entropy with label smoothing (CE-LS) and exponential moving



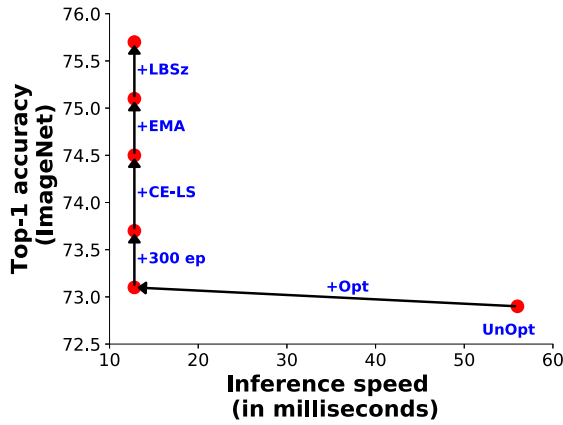


Fig. 7. Impact of different components in the training of *DiCENet*. With +, we indicate that the component is added to the previous configuration. Here, **UnOpt** represents the un-optimized *DiCENet* trained for 150 epochs with a batch size of 512 with cross entropy and **Opt** represents *DiCENet* with custom CUDA kernel. **300 ep** denotes that model is trained for 300 epochs, **CE-LS** denotes that label-smooth cross-entropy is used, **EMA** denotes that exponential moving average is used, and **LBSz** denotes that large batch size (2048 images) is used for training. Here, inference time is measured on NVIDIA GTX 1080 Ti GPU and is an average across 100 trials for a batch of 32 RGB images, each with a spatial dimension of  $224 \times 224$ .

average (EMA). We also trained *DiCENet* with CE-LS and EMA, with an exception to batch size (2048) and number of epochs (300). For training with larger batch size, we accumulated the gradients for 4 iterations. This resulted in an effective batch size of 2048 (128 images per NVIDIA GTX 1080 Ti GPU  $\times$  4 GPUs  $\times$  accumulation frequency of 4). Fig. 7 shows the effect of these changes. Similar to state-of-the-art models (e.g., MobileNetv2, MobileNetv3, MixNet, and MNASNet), *DiCENet* also benefits from these hyper-parameters and yields a top-1 accuracy of 75.7 on the ImageNet.

Importantly, the optimized CUDA kernel (Fig. 6) improved the inference speed drastically over the unoptimized version. This is because the optimized kernel launches one kernel for these three branches compared to one per branch in the unoptimized one and also, maximizes work done per CUDA thread. This reduces latency. With

optimized CUDA kernels, *DiCENet* models (10-300 MFLOPs) takes between one and three days for training on the ImageNet dataset on 4 NVIDIA GTX 1080 Ti GPUs with an effective batch size of 2048.

## 6 EVALUATING *DiCENet* ON THE IMAGENET

In this section, we evaluate the performance of *DiCENet* on the ImageNet dataset and show that *DiCENet* delivers significantly better performance than state-of-the-art efficient networks, including neural search architectures. Recall that the *DiCENet* is ShuffleNetv2 with the DiCE unit (Section 3.4).

*Implementation Details.* We scale the number of output channels by a width scaling factor  $s$  to obtain *DiCENet* models at different complexity levels, ranging from 6 MFLOPs to 500+ MFLOPs (see supplementary material, available online, for details).

*Evaluation Metrics and Baselines.* We use  $224 \times 224$  single crop top-1 accuracy to evaluate the performance on the validation set. The performance of *DiCENet* is compared with state-of-the-art manually designed efficient networks (MobileNets [7], [10], ShuffleNetv2 [11], CondenseNet [41], and ESPNetv2 [12]) and automatically designed networks (MNASNet [13], FBNet [14], MixNet [16], and MobileNetv3 [15]).

*Results.* Recent studies (e.g., MobileNetv3) have shown that longer training with extremely large batch sizes improves performance. To have fair comparisons with state-of-the-art methods, we report the performance of *DiCENet* on two settings. The first setting, *DiCENet*-E150-B512, is similar to networks like ESPNetv2, ShuffleNetv2, and CondenseNet where *DiCENet* is trained for fewer epochs (150) with a smaller batch size (512) without EMA and label smoothing. The second setting, *DiCENet*-E300-B2048, is similar to networks like MobileNets, MNASNet, and MixNet, where *DiCENet* is trained for 300 epochs with a batch size of 2048. Table 4 compares the performance of *DiCENet* with state-of-the-art efficient architectures at different FLOP ranges.

Compared to networks that are trained with smaller batch sizes and fewer epochs, e.g., ESPNetv2 (epochs: 300; batch size: 512) and ShuffleNetv2 (epochs: 240 and batch

TABLE 4  
Results on the ImageNet Dataset

Network	Type	FLOP ranges (in millions)						
		< 10 M	10-20 M	21-60 M	61-90 M	91-130 M	131-170 M	200 -320 M
MobileNet [7]	Manual		41.5 (14)	56.3 (49)	59.1 (77)	61.7 (110)	65.3 (162)	68.4 (317)
MobileNetv2 [10]	Manual		45.5 (11)	61.0 (50)	63.9 (71)	66.4 (107)	68.7 (153)	71.8 (300)
ESPNetv2 [12]	Manual				66.1 (86)	67.9 (124)		72.1 (284)
CondenseNet [41]	Manual							71.0 (274)
ShuffleNetv2 [11]	Manual	39.1 (8.0)		59.7 (41)			68.1 (142)	71.8 (292)
MNASNet [13]	NAS				62.4 (76)	67.3 (103)		74.0 (317)
FBNet [14]	NAS				65.3 (72)	67.0 (92)		74.1 (295)
MobileNetv3 [15]	NAS				67.4 (66)			75.2 (219)
MixNet [16]	NAS							75.8 (256)
<b>DiCENet-E150-B512 (Ours)</b>	Manual	40.6 (6.5)	46.2 (14)	62.8 (46)	66.5 (70)	67.8 (98)	69.5 (139)	72.9 (298)
<b>DiCENet-E300-B2048 (Ours)</b>	Manual	43.1 (6.5)	48.2 (14)	65.1 (46)	68.5 (70)	69.3 (98)	72.0 (139)	75.7 (298)

*DiCENet* delivers similar or better performance than state-of-the-art methods, including neural architecture search (NAS)-based methods. Here, each entry is represented as top-1 accuracy and FLOPs within parentheses. *DiCENet*-E150-B512 models are trained for 150 epochs with a batch size of 512 (without EMA and label smoothing) while *DiCENet*-E300-B2048 models are trained for 300 epochs with an effective batch size of 2048 (with EMA and label smoothing).

TABLE 5  
Inference Speed

Model	Network statistics			Device: GTX-960 M (Memory = 4GB)		Device: GTX-1080 Ti (Memory = 11 GB)		
	# Params	# FLOPs	Top-1 Accuracy	Batch size = 1	Batch size = 32	Batch size = 1	Batch size = 32	Batch size = 64
<b>MobileNetv2</b>	3.5 M	300 M	71.8	<b>5.6 ± 0.2 ms</b>	114 ± 0.1 ms	<b>5.9 ± 0.1 ms</b>	22.8 ± 0.1 ms	44.3 ± 0.8 ms
<b>ShuffleNetv2</b>	3.5 M	300 M	71.8	<b>5.8 ± 0.2 ms</b>	80.7 ± 0.6 ms	<b>5.8 ± 0.2 ms</b>	<b>12.9 ± 0.1 ms</b>	<b>24.1 ± 0.4 ms</b>
<b>MobileNetv3</b>	5.5 M	220 M	75.2	8.5 ± 0.2 ms	Out-of-memory	9.0 ± 0.3 ms	20.9 ± 0.1 ms	40.2 ± 0.2 ms
<b>DiCENet (Ours)</b>	5.1 M	297 M	<b>75.7</b>	<b>5.9 ± 0.1 ms</b>	<b>79 ± 0.1 ms</b>	<b>5.7 ± 0.1 ms</b>	<b>12.8 ± 0.1 ms</b>	<b>24.1 ± 0.6 ms</b>

*DiCENet and ShuffleNetv2 are comparatively faster than MobileNetv2 and MobileNetv3 on both devices (Mobile GPU: GTX-960M and Desktop GPU: GTX-1080 Ti). Inference results are an average over 100 trials for RGB input images of size  $224 \times 224$ . We used PyTorch with CUDA 10.2 for measuring speed. MobileNetv2 and ShuffleNetv2 implementations are taken from official PyTorch repository while MobileNetv3's implementation is taken from [42]. Since efficient implementations of EESP module in ESPNetv2 and mixed depth-wise convolution in MixNet are not available in PyTorch, we do not compare with these works.*

size: 1024), DiCENet delivers better performance across different FLOP ranges. Similarly, when DiCENet is trained for longer with larger batch sizes, it delivered similar or better performance than state-of-the-art methods, including neural architecture search (NAS)-based methods. For about 300 MFLOPs, DiCENet is 4 percent more accurate than MobileNetv2. Specifically, we observe that DiCENet is very effective when model size is small (FLOPs < 150 M). For example, DiCENet outperforms MNASNet [13], FBNet [14], and MobileNetv3 [15] by 6.1, 3.2, and 1.1 percent for network size of about 70 MFLOPs, respectively.

Overall, these results shows that DiCE unit learns better representations than separable convolutions. We believe that incorporating the DiCE unit with NAS would yield better network.

*Inference Speed.* We measure the inference time on two GPUs: (1) embedded or mobile GPU (NVIDIA GTX 960M) and (2) desktop GPU (NVIDIA GTX 1080 Ti).<sup>4</sup> DiCENet is as fast as ShuffleNetv2 but delivers better performance. Compared to MobileNetv2 and MobileNetv3, DiCENet has low latency while delivering similar or better performance. We observe that MobileNetv2 and MobileNetv3 models are slow in comparison to ShuffleNetv2 and DiCENet when the batch size increases. This is because the number of channels in the depth-wise convolution in the inverted residual block of MobileNetv2 and MobileNetv3 are very large as compared to DiCENet and ShuffleNetv2. For example, the maximum number of channels in depth-wise convolution in MobileNetv2 (300 MFLOPs) are 960 while the maximum number of channels in DimConv and depth-wise convolution in DiCENet (298 MFLOPs) and ShuffleNetv2 (292 MFLOPs) are 576 and 352, respectively.

## 7 TASK-LEVEL GENERALIZATION OF DICENET

Several previous works (e.g., [43], [44]) have shown that high accuracy on the Imagenet dataset does not necessarily correlates with high accuracy on visual scene understanding tasks (e.g., object detection and semantic segmentation). Since these tasks are widely used in real-world applications (e.g., autonomous wheel chair and robots) and often run on resource-constrained devices (e.g., embedded devices), it is important that efficient networks generalizes well on these tasks.

4. We do not measure the inference speed on Smartphones because efficient implementations of DiCE unit are not yet available for such devices.

Therefore, we evaluate the performance of DiCENet on three different tasks: (1) object detection (Section 7.1), (2) semantic segmentation (Section 7.2), and (3) multi-object classification (Section 7.3). Compared to existing efficient networks that are built using separable convolutions (e.g., MobileNets [7], [10], [15], MixNet [16], and ESPNetv2 [12]), DiCENet delivers better performance.

### 7.1 Object Detection on VOC and MS-COCO

*Implementation Details.* For object detection, we use Single Shot object Detection (SSD) [17] pipeline. We use DiCENet (298 MFLOPs) pretrained on the ImageNet as a base feature extractor instead of VGG [2]. We fine-tune our network using SGD with smooth L1 and cross-entropy losses for object localization and classification, respectively.

*Evaluation Metrics and Baselines.* We evaluate the performance using mean Average Precision (mAP). For MS-COCO, we report mAP@IoU of 0.50:0.95. For SSD as a detection pipeline, we compare DiCENet's performance with two types of base feature extractors: (1) *manual* (VGG [2], MobileNet [7], MobileNetv2 [10], and ESPNetv2 [12]) and (2) *NAS-based* (MNASNet [13], MixNet [16], and MobileNetv3 [15]).

*Results.* Table 6 compares quantitative results of SSD with different backbone networks on the PASCAL VOC 2007 and the MS-COCO datasets. DiCENet significantly improves the performance of SSD-based object detection pipeline and

TABLE 6  
Object Detection Results of SSD [17] With Different Backbones on PASCAL VOC 2007 and MS-COCO

SSD backbone	Image size	VOC07		MS-COCO	
		FLOPs	mAP	FLOPs	mAP
VGG [2]	512x512	90.2 B	74.9	99.5 B	26.8
	300x300	31.3 B	72.4	35.2 B	23.2
MobileNet [7]	320x320	–	–	1.3 B	22.2
MobileNetv2 [10]	320x320	–	–	<b>0.8 B</b>	22.1
ESPNetv2 [12]	512x512	2.5 B	75.0	3.2 B	26.0
	256x256	0.9 B	70.3	1.1 B	21.9
DiCENet (Ours)	512x512	2.0 B	<b>77.2</b>	2.6 B	<b>28.0</b>
DiCENet (Ours)	300x300	<b>0.7 B</b>	71.9	0.9 B	25.1
MobileNetv3 [15] (NAS)	–	–	–	0.6 B	22.0
MixNet [16] (NAS)	–	–	–	0.9 B	22.3
MNASNet [13] (NAS)	–	–	–	0.8 B	23.0

*On the MS-COCO dataset, total network parameters in SSD with different backbones are about 5 million, except VGG.*



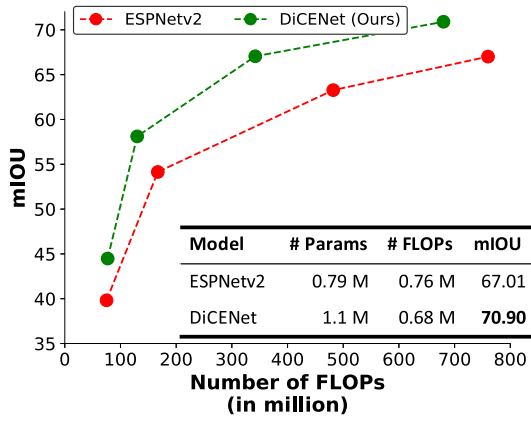


Fig. 8. Semantic segmentation results on the PASCAL VOC 2012 validation set. Here, mIOU represents mean intersection over union.

delivers 1-4 percent higher mAP than other existing efficient variants of SSD, including NAS-based backbones such as MobileNetv3 (22.0 versus 25.1) and MixNet (22.3 versus 25.1). Compared to standard SSD with VGG as backbone, DiCENet achieves higher mAP while being 45 $\times$  and 38 $\times$  more efficient on the PASCAL VOC 2007 and the MS-COCO dataset, respectively.

## 7.2 Semantic Segmentation on PASCAL VOC

*Implementation Details.* We adapt DiCENet to ESPNetv2's [12] encoder-decoder architecture. We choose this network because it delivers competitive performance to existing methods even with low resolution images (e.g.,  $256 \times 256$  versus  $512 \times 512$ ). We replace the encoder in ESPNetv2 (pretrained on ImageNet) with the DiCENet and follow the same training procedure for fine-tuning as ESPNetv2. We do not change the decoder.

*Evaluation Metrics and Baselines.* The performance at different complexity levels (FLOPs) is evaluated using mean intersection over union (mIOU).

*Results.* Fig. 8 compares the performance of DiCENet with ESPNetv2 on the PASCAL VOC 2012 validation set. DiCENet significantly improves the segmentation performance i.e., for similar FLOPs, DiCENet achieves higher mIOU while for similar mIOU, DiCENet requires significantly fewer FLOPs.

## 7.3 Multi-Object Classification on MS-COCO

*Implementation Details.* Following [12], we fine-tune DiCENet using the binary cross-entropy loss.

*Evaluation Metrics and Baselines.* Similar to [12], we evaluate the performance using overall and per-class F1 score and compare with two efficient architectures, i.e., ESPNetv2 and ShuffleNetv2.

*Results.* Table 7 shows that DiCENet outperforms existing efficient networks by a significant margin (e.g., ESPNetv2 and ShuffleNetv2 by 4.1 and 5.8 percent respectively) on this task.

## 8 CONCLUSIONS

We introduce a novel and generic convolutional unit, the DiCE unit, that uses dimension-wise convolutions and

TABLE 7  
Multi-Object Classification Results on the MS-COCO Dataset

Network	# Params	# FLOPs	F1-score	
			Class-wise	Overall
ShuffleNetv2 [11] <sup>†</sup>	3.5 M	300 M	60.42	67.58
ESPNetv2 [12] <sup>†</sup>	3.5 M	284 M	63.41	69.23
DiCENet (ours)	5.1 M	298 M	<b>66.92</b>	<b>73.41</b>

Here, <sup>†</sup> indicates that results are from [12].

dimension-wise fusion module to learn spatial and channel-wise representations efficiently. Our empirical results suggest that the DiCE unit is more effective than separable convolutions. Moreover, when we stack DiCE units to build DiCENet model, we observe significant improvements across different computer vision tasks. We have shown that the DiCE unit is effective. Future work involves adding the DiCE unit in neural search space to discover a better neural architecture, particularly with [13], [15], [16].

## ACKNOWLEDGMENTS

This work was supported by ONR N00014-18-1-2826, DARPA N66001-19-2-403, NSF (IIS-1616112, IIS1252835), an Allen Distinguished Investigator Award, Samsung GRO and gifts from Allen Institute for AI, Google, and Amazon. The authors would also like to thank the members of the PRIOR team at the Allen Institute for Artificial Intelligence (AI2), the H2Lab at the University of Washington, Seattle, and anonymous reviewers for their valuable feedback and comments.

## REFERENCES

- [1] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet classification with deep convolutional neural networks," in *Proc. Advances Neural Inf. Process. Syst.*, 2012, pp. 1097–1105.
- [2] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," 2014, *arXiv:1409.1556*.
- [3] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2016, pp. 770–778.
- [4] C. Li and C. R. Shi, "Constrained optimization based low-rank approximation of deep neural networks," in *Proc. Eur. Conf. Comput. Vis.*, 2018, pp. 746–761.
- [5] Y. He, J. Lin, Z. Liu, H. Wang, L.-J. Li, and S. Han, "AMC: Automl for model compression and acceleration on mobile devices," in *Proc. Eur. Conf. Comput. Vis.*, 2018, pp. 815–832.
- [6] J. Jin, A. Dundar, and E. Culurciello, "Flattened convolutional neural networks for feedforward acceleration," 2014, *arXiv:1412.5474*.
- [7] A. G. Howard *et al.*, "MobileNets: Efficient convolutional neural networks for mobile vision applications," 2017, *arXiv:1704.04861*.
- [8] S. Mehta, M. Rastegari, A. Caspi, L. Shapiro, and H. Hajishirzi, "Espnet: Efficient spatial pyramid of dilated convolutions for semantic segmentation," in *Proc. Eur. Conf. Comput. Vis.*, 2018, pp. 552–568.
- [9] F. Chollet, "Xception: Deep learning with depthwise separable convolutions," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2017, pp. 1800–1807.
- [10] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen, "Mobilenetv2: Inverted residuals and linear bottlenecks," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2018, pp. 4510–4520.
- [11] N. Ma, X. Zhang, H.-T. Zheng, and J. Sun, "Shufflenet v2: Practical guidelines for efficient CNN architecture design," in *Proc. Eur. Conf. Comput. Vis.*, 2018, pp. 4510–4520.
- [12] S. Mehta, M. Rastegari, L. Shapiro, and H. Hajishirzi, "Espnetv2: A light-weight, power efficient, and general purpose convolutional neural network," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2019, pp. 9182–9192.

- [13] M. Tan *et al.*, "MnasNet: Platform-aware neural architecture search for mobile," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2019, pp. 2815–2823, doi: [10.1109/CVPR.2019.00293](https://doi.org/10.1109/CVPR.2019.00293)
- [14] B. Wu *et al.*, "FBNet: Hardware-aware efficient convNet design via differentiable neural architecture search," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2019, pp. 10734–10742.
- [15] A. Howard *et al.*, "Searching for mobileNetV3," in *Proc. IEEE/CVF Int. Conf. Comput. Vis.*, 2019, pp. 1314–1324.
- [16] M. Tan and Q. V. Le, "Mixconv: Mixed depthwise convolutional kernels," in *Proc. British Mach. Vis. Conf.*, 2019, pp. 1–13. [Online]. Available: <https://bmvc2019.org/wp-content/uploads/papers/0583-paper.pdf>
- [17] W. Liu *et al.*, "SSD: Single shot multibox detector," in *Proc. Eur. Conf. Comput. Vis.*, 2016, pp. 21–37.
- [18] X. Zhang, X. Zhou, M. Lin, and J. Sun, "Shufflenet: An extremely efficient convolutional neural network for mobile devices," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2018, pp. 6848–6856.
- [19] B. Zoph and Q. V. Le, "Neural architecture search with reinforcement learning," in *Proc. Int. Conf. Learn. Representations*, 2017. [Online]. Available: <https://openreview.net/forum?id=r1Ue8Hcxg>
- [20] B. Zoph, V. Vasudevan, J. Shlens, and Q. V. Le, "Learning transferable architectures for scalable image recognition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2018, pp. 8697–8710.
- [21] M. Wortsman, A. Farhadi, and M. Rastegari, "Discovering neural wirings," in *Proc. Conf. Neural Inf. Process. Syst.*, 2019, pp. 2684–2694.
- [22] M. Rastegari, V. Ordonez, J. Redmon, and A. Farhadi, "XNOR-net: Imagenet classification using binary convolutional neural networks," in *Proc. Eur. Conf. Comput. Vis.*, 2016, pp. 525–542.
- [23] J. Wu, C. Leng, Y. Wang, Q. Hu, and J. Cheng, "Quantized convolutional neural networks for mobile devices," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2016, pp. 4820–4828.
- [24] I. Hubara, M. Courbariaux, D. Soudry, R. El-Yaniv, and Y. Bengio, "Binarized neural networks," in *Proc. Advances Neural Inf. Process. Syst.*, 2016, pp. 4114–4122.
- [25] R. Andri, L. Cavigelli, D. Rossi, and L. Benini, "YodaNN: An architecture for ultralow power binary-weight CNN acceleration," *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst.*, vol. 37, no. 1, pp. 48–60, Jan. 2018.
- [26] S. Han, H. Mao, and W. J. Dally, "Deep compression: Compressing deep neural networks with pruning, trained quantization and Huffman coding," in *Proc. Int. Conf. Learn. Representations*, 2016.
- [27] W. Wen, C. Wu, Y. Wang, Y. Chen, and H. Li, "Learning structured sparsity in deep neural networks," in *Proc. Advances Neural Inf. Process. Syst.*, 2016, pp. 2082–2090.
- [28] A. Veit and S. Belongie, "Convolutional networks with adaptive inference graphs," in *Proc. Eur. Conf. Comput. Vis.*, 2018, pp. 730–741.
- [29] G. Hinton, O. Vinyals, and J. Dean, "Distilling the knowledge in a neural network," in *Proc. NIPS Deep Learn. Representation Learn. Workshop*, 2015. [Online]. Available: <https://research.google/pubs/pub44873/>
- [30] S. Gupta, J. Hoffman, and J. Malik, "Cross modal distillation for supervision transfer," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2016, pp. 2827–2836.
- [31] J. Yim, D. Joo, J. Bae, and J. Kim, "A gift from knowledge distillation: Fast optimization, network minimization and transfer learning," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2017, pp. 4133–4141.
- [32] B. Jacob *et al.*, "Quantization and training of neural networks for efficient integer-arithmetic-only inference," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2018, pp. 2704–2713.
- [33] K. Wang, Z. Liu, Y. Lin, J. Lin, and S. Han, "HAQ: Hardware-aware automated quantization with mixed precision," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2019, pp. 8612–8620.
- [34] J. Hu, L. Shen, and G. Sun, "Squeeze-and-excitation networks," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2018, pp. 7132–7141.
- [35] O. Russakovsky *et al.*, "ImageNet large scale visual recognition challenge," *Int. J. Comput. Vis.*, vol. 115, pp. 211–252, 2015.
- [36] T.-Y. Lin *et al.*, "Microsoft coco: Common objects in context," in *Proc. Eur. Conf. Comput. Vis.*, 2014, pp. 740–755.
- [37] M. Everingham *et al.*, "The PASCAL visual object classes (VOC) challenge," *Int. J. Comput. Vis.*, vol. 88, pp. 303–338, 2010.
- [38] M. Everingham, S. A. Eslami, L. Van Gool, C. K. Williams, J. Winn, and A. Zisserman, "The pascal visual object classes challenge: A retrospective," *Int. J. Comput. Vis.*, vol. 111, pp. 98–136, 2015.
- [39] B. Hariharan, P. Arbeláez, L. Bourdev, S. Maji, and J. Malik, "Semantic contours from inverse detectors," in *Proc. Int. Conf. Comput. Vis.*, 2011, pp. 991–998.
- [40] S. Xie, R. Girshick, P. Dollár, Z. Tu, and K. He, "Aggregated residual transformations for deep neural networks," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2017, pp. 5987–5995.
- [41] G. Huang, S. Liu, L. van der Maaten, and K. Q. Weinberger, "Condensenet: An efficient densenet using learned group convolutions," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2018, pp. 2752–2761.
- [42] R. Wightman, "PyTorch image models," Accessed: Dec. 8, 2020. [Online]. Available: <https://github.com/rwightman/pytorch-image-models>
- [43] J. Long, E. Shelhamer, and T. Darrell, "Fully convolutional networks for semantic segmentation," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2015, pp. 3431–3440.
- [44] Z.-Q. Zhao, P. Zheng, S.-T. Xu, and X. Wu, "Object detection with deep learning: A review," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 30, no. 11, pp. 3212–3232, Nov. 2019.



**Sachin Mehta** (Student Member, IEEE) is currently working toward the PhD degree from the University of Washington, Seattle. His research interests include intersection of computer vision, NLP, and machine learning, especially in designing fast, light-weight, power efficient, and memory efficient neural architectures that can be used for modeling visual and textual data on resource-constrained devices across different domains, including computer vision for accessible technologies and health care.



**Hannaneh Hajishirzi** is currently an assistant professor at the Allen School for Computer Science and Engineering, University of Washington. Her research interests include NLP, AI, and machine learning, particularly designing algorithms for semantic understanding, reasoning, question answering, and knowledge extraction from multimodal data. She is the recipient of multiple awards, including the Allen Distinguished Investigator Award, the SigDial best paper award, and the Sloan Research Fellowship.



**Mohammad Rastegari** received the PhD degree from the University of Maryland. He is currently a senior member of AI/ML at Apple Inc. and an affiliate assistant professor at the Allen School for Computer Science and Engineering, University of Washington. Prior to joining Apple, he was a co-founder and the Chief Technology Officer (CTO) at Xnor.ai and also a research scientist with Allen Institute for AI (AI2). His main area of research and expertise relies on efficient methods for machine learning and computer vision. Previously, he was a research scholar with the University of California, Berkeley and a Facebook fellow in the artificial intelligent team (FAIR).

► For more information on this or any other computing topic, please visit our Digital Library at [www.computer.org/csdl](http://www.computer.org/csdl).