ESPNetv2: A Light-weight, Power Efficient, and General Purpose Convolutional Neural Network

Sachin Mehta[♠], Mohammad Rastegari[♡♣], Linda Shapiro[♠], and Hannaneh Hajishirzi[♠♡]

[♠] University of Washington [♡] Allen Institute for AI (AI2) [♣] XNOR.AI
{*sacmehta, shapiro, hannaneh*}*@cs.washington.edu* *mohammadr@allenai.org*

Abstract

We introduce a light-weight, power efficient, and general purpose convolutional neural network, `ESPNetv2`*, for modeling visual and sequential data. Our network uses group point-wise and depth-wise dilated separable convolutions to learn representations from a large effective receptive field with fewer FLOPs and parameters. The performance of our network is evaluated on four different tasks: (1) object classification, (2) semantic segmentation, (3) object detection, and (4) language modeling. Experiments on these tasks, including image classification on the ImageNet and language modeling on the PenTree bank dataset, demonstrate the superior performance of our method over the state-of-the-art methods. Our network outperforms ESPNet by 4-5% and has* $2-4\times$ *fewer FLOPs on the PASCAL VOC and the Cityscapes dataset. Compared to YOLOv2 on the MS-COCO object detection,* `ESPNetv2` *delivers 4.4% higher accuracy with* $6\times$ *fewer FLOPs. Our experiments show that* `ESPNetv2` *is much more power efficient than existing state-of-the-art efficient methods including ShuffleNets and MobileNets. Our code is open-source and available at* https://github.com/sacmehta/ESPNetv2.

1. Introduction

The increasing programmability and computational power of GPUs have accelerated the growth of deep convolutional neural networks (CNNs) for modeling visual data [16, 22, 34]. CNNs are being used in real-world visual recognition applications such as visual scene understanding [62] and bio-medical image analysis [42]. Many of these real-world applications, such as self-driving cars and robots, run on resource-constrained edge devices and demand online processing of data with low latency.

Existing CNN-based visual recognition systems require large amounts of computational resources, including memory and power. While they achieve high performance on high-end GPU-based machines (e.g. with NVIDIA TitanX), they are often too expensive for resource constrained edge devices such as cell phones and embedded compute platforms. As an example, ResNet-50 [16], one of the most well known CNN architecture for image classification, has 25.56 million parameters (98 MB of memory) and performs 2.8 billion high precision operations to classify an image. These numbers are even higher for deeper CNNs, e.g. ResNet-101. These models quickly overtax the limited resources, including compute capabilities, memory, and battery, available on edge devices. Therefore, CNNs for real-world applications running on edge devices should be light-weight and efficient while delivering high accuracy.

Recent efforts for building light-weight networks can be broadly classified as: (1) *Network compression-based methods* remove redundancies in a pre-trained model in order to be more efficient. These models are usually implemented by different parameter pruning techniques [24, 55]. (2) *Low-bit representation-based methods* represent learned weights using few bits instead of high precision floating points [20, 39, 47]. These models usually do not change the structure of the network and the convolutional operations could be implemented using logical gates to enable fast processing on CPUs. (3) *Light-weight CNNs* improve the efficiency of a network by factoring computationally expensive convolution operation [17, 18, 29, 32, 44, 60]. These models are computationally efficient by their design i.e. the underlying model structure learns fewer parameters and has fewer floating point operations (FLOPs).

In this paper, we introduce a light-weight architecture, `ESPNetv2`, that can be easily deployed on edge devices. The main contributions of our paper are: (1) A general purpose architecture for modeling both visual and sequential data efficiently. We demonstrate the performance of our network across different tasks, ranging from object classifica-

tion to language modeling. (2) Our proposed architecture, ESPNetv2, extends ESPNet [32], a dilated convolution-based segmentation network, with depth-wise separable convolutions; an efficient form of convolutions that are used in state-of-art efficient networks including MobileNets [17, 44] and ShuffleNets [29, 60]. Depth-wise dilated separable convolutions improves the accuracy of ESPNetv2 by 1.4% in comparison to depth-wise separable convolutions. We note that ESPNetv2 achieves better accuracy (72.1 with 284 MFLOPs) with fewer FLOPs than dilated convolutions in the ESPNet [32] (69.2 with 426 MFLOPs). (3) Our empirical results show that ESPNetv2 delivers similar or better performance with fewer FLOPS on different visual recognition tasks. On the ImageNet classification task [43], our model outperforms all of the previous efficient model designs in terms of efficiency and accuracy, especially under small computational budgets. For example, our model outperforms MobileNetv2 [44] by 2% at a computational budget of 28 MFLOPs. For semantic segmentation on the PASCAL VOC and the Cityscapes dataset, ESPNetv2 outperforms ESPNet [32] by 4-5% and has $2-4\times$ fewer FLOPs. For object detection, ESPNetv2 outperforms YOLOv2 by 4.4% and has $6\times$ fewer FLOPs. We also study a cyclic learning rate scheduler with warm restarts. Our results suggests that this scheduler is more effective than the standard fixed learning rate scheduler.

2. Related Work

Efficient CNN architectures: Most state-of-the-art efficient networks [17, 29, 44] use depth-wise separable convolutions [17] that factor a convolution into two steps to reduce computational complexity: (1) depth-wise convolution that performs light-weight filtering by applying a single convolutional kernel per input channel and (2) point-wise convolution that usually expands the feature map along channels by learning linear combinations of the input channels. Another efficient form of convolution that has been used in efficient networks [18,60] is group convolution [22], wherein input channels and convolutional kernels are factored into groups and each group is convolved independently. The ESPNetv2 network extends the ESPNet network [32] using these efficient forms of convolutions. To learn representations from a large effective receptive field, ESPNetv2 uses depth-wise "dilated" separable convolutions instead of depth-wise separable convolutions.

In addition to convolutional factorization, a network's efficiency and accuracy can be further improved using methods such as channel shuffle [29] and channel split [29]. Such methods are orthogonal to our work.

Neural architecture search: These approaches search over a huge network space using a pre-defined dictionary containing different parameters, including different convolutional layers, different convolutional units, and different filter sizes [4, 52, 56, 66]. Recent search-based methods [52, 56] have shown improvements for MobileNetv2. We believe that these methods will increase the performance of ESPNetv2 and are complementary to our work.

Network compression: These approaches improve the inference of a pre-trained network by pruning network connections or channels [12, 13, 24, 53, 55]. These approaches are effective, because CNNs have a substantial number of redundant weights. The efficiency gain in most of these approaches are due to the sparsity of parameters, and are difficult to efficiently implement on CPUs due to the cost of look-up and data migration operations. These approaches are complementary to our network.

Low-bit representation: Another approach to improve inference of a pre-trained network is low-bit representation of network weights using quantization [1, 9, 20, 39, 47, 57, 64]. These approaches use fewer bits to represent weights of a pre-trained network instead of 32-bit high-precision floating points. Similar to network compression-based methods, these approaches are complementary to our work.

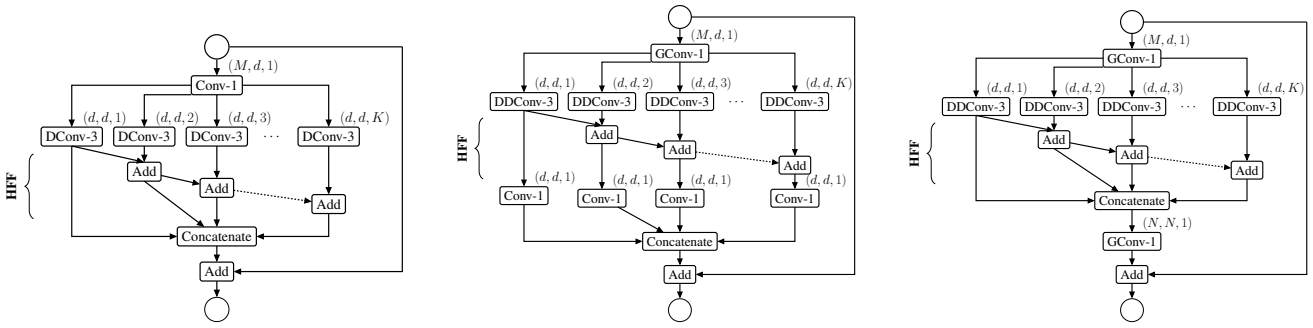
3. ESPNetv2

This section elaborates the ESPNetv2 architecture in detail. We first describe *depth-wise dilated separable convolutions* that enables our network to learn representations from a large effective receptive field efficiently. We then describe the core unit of the ESPNetv2 network, the EESP unit, which is built using group point-wise convolutions and depth-wise dilated separable convolutions.

3.1. Depth-wise dilated separable convolution

Convolution factorization is the key principle that has been used by many efficient architectures [17, 29, 44, 60]. The basic idea is to replace the full convolutional operation with a factorized version such as depth-wise separable convolution [17] or group convolution [22]. In this section, we describe depth-wise dilated separable convolutions and compare with other similar efficient forms of convolution.

A standard convolution convolves an input $\mathbf{X} \in \mathbb{R}^{W \times H \times c}$ with convolutional kernel $\mathbf{K} \in \mathbb{R}^{n \times n \times c \times \hat{c}}$ to produce an output $\mathbf{Y} \in \mathbb{R}^{W \times H \times \hat{c}}$ by learning $n^2 c\hat{c}$ parameters from an effective receptive field of $n \times n$. In contrast to standard convolution, depth-wise dilated separable convolutions apply a light-weight filtering by factoring a standard convolution into two layers: 1) depth-wise dilated convolution per input channel with a dilation rate of r; enabling the convolution to learn representations from an effective receptive field of $n_r \times n_r$, where $n_r = (n-1) \cdot r + 1$ and 2) point-wise convolution to learn linear combinations of input. This factorization reduces the computational cost by



(a) ESP (b) EESP-A (c) EESP

Figure 1: This figure visualizes the building blocks of the ESPNet, the ESP unit in (a), and the `ESPNetv2`, the EESP unit in (b-c). We note that EESP units in (b-c) are equivalent in terms of computational complexity. Each convolutional layer (Conv-n: $n \times n$ standard convolution, GConv-n: $n \times n$ group convolution, DConv-n: $n \times n$ dilated convolution, DDConv-n: $n \times n$ depth-wise dilated convolution) is denoted by (# input channels, # output channels, and dilation rate). Point-wise convolutions in (b) or group point-wise convolutions in (c) are applied after HFF to learn linear combinations between inputs.

Convolution type	**Parameters**	**Eff. receptive field**
Standard	$n^2c\hat{c}$	$n \times n$
Group	$\frac{n^2c\hat{c}}{g}$	$n \times n$
Depth-wise separable	$n^2c + c\hat{c}$	$n \times n$
Depth-wise dilated separable	$n^2c + c\hat{c}$	$n_r \times n_r$

Table 1: Comparison between different type of convolutions. Here, $n \times n$ is the kernel size, $n_r = (n-1) \cdot r + 1$, r is the dilation rate, c and $\hat{c}$ are the input and output channels respectively, and g is the number of groups.

a factor of $\frac{n^2c\hat{c}}{n^2c+c\hat{c}}$. A comparison between different types of convolutions is provided in Table 1. Depth-wise dilated separable convolutions are efficient and can learn representations from large effective receptive fields.

3.2. EESP unit

Taking advantage of depth-wise dilated separable and group point-wise convolutions, we introduce a new unit EESP, Extremely Efficient Spatial Pyramid of Depth-wise Dilated Separable Convolutions, which is specifically designed for edge devices. The design of our network is motivated by the ESPNet architecture [32], a state-of-the-art efficient segmentation network. The basic building block of the ESPNet architecture is the ESP module, shown in Figure 1a. It is based on a *reduce-split-transform-merge* strategy. The ESP unit first projects the high-dimensional input feature maps into low-dimensional space using point-wise convolutions and then learn the representations in parallel using dilated convolutions with different dilation rates. Different dilation rates in each branch allow the ESP unit to learn the representations from a large effective receptive field. This factorization, especially learning the representations in a low-dimensional space, allows the ESP unit to be efficient.

To make the ESP module even more computationally efficient, we first replace point-wise convolutions with group point-wise convolutions. We then replace computationally expensive 3×3 dilated convolutions with their economical counterparts i.e. depth-wise dilated separable convolutions. To remove the gridding artifacts caused by dilated convolutions, we fuse the feature maps using the computationally efficient hierarchical feature fusion (HFF) method [32]. This method additively fuses the feature maps learned using dilated convolutions in a hierarchical fashion; feature maps from the branch with lowest receptive field are combined with the feature maps from the branch with next highest receptive field at each level of the hierarchy[1]. The resultant unit is shown in Figure 1b. With group point-wise and depth-wise dilated separable convolutions, the total complexity of the ESP block is reduced by a factor of $\frac{Md+n^2d^2K}{\frac{Md}{g}+(n^2+d)dK}$, where K is the number of parallel branches and g is the number of groups in group point-wise convolution. For example, the EESP unit learns $7\times$ fewer parameters than the ESP unit when M=240, g=K=4, and d=$\frac{M}{K}$=60.

We note that computing K point-wise (or 1×1) convolutions in Figure 1b independently is equivalent to a single group point-wise convolution with K groups in terms of complexity; however, group point-wise convolution is more efficient in terms of implementation, because it launches one convolutional kernel rather than K point-wise convolutional kernels. Therefore, we replace these K point-wise convolutions with a group point-wise convolution, as shown in Figure 1c. We will refer to this unit as EESP.

[1] Other existing works [54,59] add more convolutional layers with small dilation rates to remove gridding artifacts. This increases the computational complexity of the unit or network.

Layer	Output Size	Kernel size / Stride	Repeat	Output channels for different ESPNetv2 models					
Convolution	112 × 112	3 × 3 / 2	1	16	32	32	32	32	32
Strided EESP (Fig. 2)	56 × 56		1	32	64	80	96	112	128
Strided EESP (Fig. 2)	28 × 28		1	64	128	160	192	224	256
EESP (Fig. 1c)	28 × 28		3	64	128	160	192	224	256
Strided EESP (Fig. 2)	14 × 14		1	128	256	320	384	448	512
EESP (Fig. 1c)	14 × 14		7	128	256	320	384	448	512
Strided EESP (Fig. 2)	7 × 7		1	256	512	640	768	896	1024
EESP (Fig. 1c)	7 × 7		3	256	512	640	768	896	1024
Depth-wise convolution	7 × 7	3 × 3		256	512	640	768	896	1024
Group convolution	7 × 7	1 × 1		1024	1024	1024	1024	1280	1280
Global avg. pool	1 × 1	7 × 7							
Fully connected				1000	1000	1000	1000	1000	1000
Complexity				28 M	86 M	123 M	169 M	224 M	284 M
Parameters				1.24 M	1.67 M	1.97 M	2.31 M	3.03 M	3.49 M

Table 2: The ESPNetv2 network at different computational complexities for classifying a 224 × 224 input into 1000 classes in the ImageNet dataset [43]. Network's complexity is evaluated in terms of total number of multiplication-addition operations (or FLOPs).

Strided EESP with shortcut connection to an input image: To learn representations efficiently at multiple scales, we make following changes to the EESP block in Figure 1c: 1) depth-wise dilated convolutions are replaced with their strided counterpart, 2) an average pooling operation is added instead of an identity connection, and 3) the element-wise addition operation is replaced with a concatenation operation, which helps in expanding the dimensions of feature maps efficiently [60].

Spatial information is lost during down-sampling and convolution (filtering) operations. To better encode spatial relationships and learn representations efficiently, we add an efficient long-range shortcut connection between the input image and the current down-sampling unit. This connection first down-samples the image to the same size as that of the feature map and then learns the representations using a stack of two convolutions. The first convolution is a standard 3 × 3 convolution that learns the spatial representations while the second convolution is a point-wise convolution that learns linear combinations between the input, and projects it to a high-dimensional space. The resultant EESP unit with long-range shortcut connection to the input is shown in Figure 2.

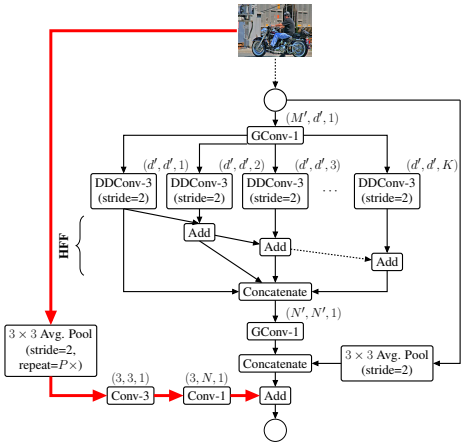
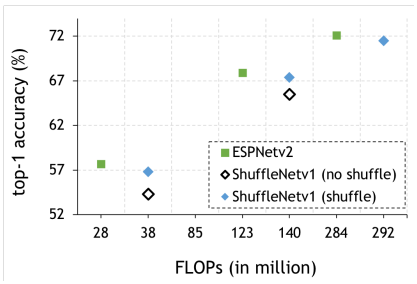


Figure 2: Strided EESP unit with shortcut connection to an input image (highlighted in red) for down-sampling. The average pooling operation is repeated $P\times$ to match the spatial dimensions of an input image and feature maps.

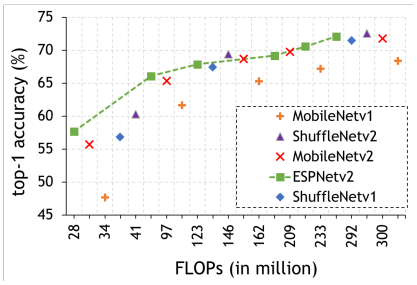
3.3. Network architecture

The ESPNetv2 network is built using EESP units. At each spatial level, the ESPNetv2 repeats the EESP units several times to increase the depth of the network. In the EESP unit (Figure 1c), we use batch normalization [21] and PReLU [15] after every convolutional layer with an exception to the last group-wise convolutional layer where PReLU is applied after element-wise sum operation. To maintain the same computational complexity at each spatial-level, the feature maps are doubled after every down-sampling operation [16, 46].

In our experiments, we set the dilation rate r proportional to the number of branches in the EESP unit (K). The effective receptive field of the EESP unit grows with K. Some of the kernels, especially at low spatial levels such as 7 × 7, might have a larger effective receptive field than the size of the feature map. Therefore, such kernels might not contribute to learning. In order to have meaningful kernels, we limit the effective receptive field at each spatial level l with spatial dimension $W^l \times H^l$ as:



(a)



(b)

Network	# Params	FLOPs	Top-1
MobileNetv1 [17]	2.59 M	325 M	68.4
CondenseNet [18]	–	274 M	71.0
IGCV3 [49]	–	318 M	72.2
Xception† [7]	–	305 M	70.6
DenseNet† [19]	–	295 M	60.1
ShuffleNetv1 [60]	3.46 M	292 M	71.5
MobileNetv2 [44]	3.47 M	300 M	71.8
	6.9 M	585 M	74.7
ShuffleNetv2 [29]	3.5 M	299 M	72.6
	7.4 M	591 M	74.9
ESPNetv2 (Ours)	3.49 M	284 M	72.1
	5.9 M	602 M	74.9

(c)

Figure 3: Performance comparison of different efficient networks on the ImageNet validation set: (a) ESPNetv2 vs. ShuffleNetv1 [60], (b) ESPNetv2 vs. efficient models at different network complexities, and (c) ESPNetv2 vs. state-of-the-art for a computational budget of approximately 300 million FLOPs. We count the total number of multiplication-addition operations (FLOPs) for an input image of size 224×224. Here, † represents that the performance of these networks is reported in [29]. Best viewed in color.

$\mathrm{n}_d^l(Z^l) = 5 + \frac{Z^l}{7}, Z^l \in \{W^l, H^l\}$ with the effective receptive field $(n_d \times n_d)$ corresponding to the lowest spatial level (i.e. 7×7) as 5×5. Following [32], we set $K = 4$ in our experiments. Furthermore, in order to have a homogeneous architecture, we set the number of groups in group point-wise convolutions equal to number of parallel branches ($g = K$). The overall ESPNetv2 architectures at different computational complexities are shown in Table 2.

4. Experiments

To showcase the power of the ESPNetv2 network, we evaluate and compare the performance with state-of-the-art methods on four different tasks: (1) object classification, (2) semantic segmentation, (3) object detection, and (3) language modeling.

4.1. Image classification

Dataset: We evaluate the performance of the ESPNetv2 on the ImageNet 1000-way classification dataset [43] that contains 1.28M images for training and 50K images for validation. We evaluate the performance of our network using the single crop top-1 classification accuracy, i.e. we compute the accuracy on the center cropped view of size 224×224.

Training: The ESPNetv2 networks are trained using the PyTorch deep learning framework [38] with CUDA 9.0 and cuDNN as the back-ends. For optimization, we use SGD [50] with *warm restarts*. At each epoch t, we compute the learning rate η_t as:

$$\eta_t = \eta_{max} - (t \bmod T) \cdot \eta_{min} \quad (1)$$

where η_{max} and η_{min} are the ranges for the learning rate and T is the cycle length after which learning rate will restart. Figure 4 visualizes the learning rate policy for three

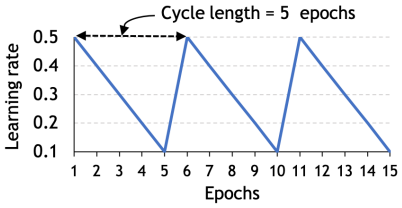
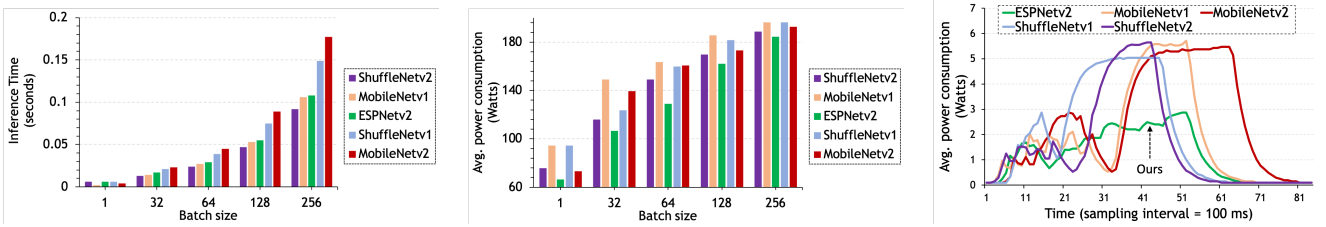


Figure 4: Cyclic learning rate policy (see Eq.1) with linear learning rate decay and warm restarts.

cycles. This learning rate scheme can be seen as a variant of the cosine learning policy [28], wherein the learning rate is decayed as a function of cosine before warm restart. In our experiment, we set $\eta_{min} = 0.1$, $\eta_{max} = 0.5$, and $T = 5$. We train our networks with a batch size of 512 for 300 epochs by optimizing the cross-entropy loss. For faster convergence, we decay the learning rate by a factor of two at the following epoch intervals: {50, 100, 130, 160, 190, 220, 250, 280}. We use a standard data augmentation strategy [16, 51] with an exception to color-based normalization. This is in contrast to recent efficient architectures that uses less scale augmentation to prevent underfitting [29, 60]. The weights of our networks are initialized using the method described in [15].

Results: Figure 3 provides a performance comparison between ESPNetv2 and state-of-the-art efficient networks. We observe that: (1) Like ShuffleNetv1 [60], ESPNetv2 also uses group point-wise convolutions. However, ESPNetv2 does not use any channel shuffle which was found to be very effective in ShuffleNetv1 and delivers better performance than ShuffleNetv1. (2) Compared to MobileNets, ESPNetv2 delivers better performance especially under small computational budgets. With 28 million FLOPs, ESPNetv2 outperforms MobileNetv1 [17] (34



(a) Inference time vs. batch size (1080 Ti) (b) Power vs. batch size (1080 Ti) (c) Power consumption on TX2

Figure 5: Performance analysis of different efficient networks (computational budget ≈ 300 million FLOPs). Inference time and power consumption are averaged over 100 iterations for a 224 × 224 input on a NVIDIA GTX 1080 Ti GPU and NVIDIA Jetson TX2. We do not report execution time on TX2 because there is not much substantial difference. Best viewed in color.

million FLOPs) and MobileNetv2 [44] (30 million FLOPs) by 10% and 2% respectively. (3) ESPNetv2 delivers comparable accuracy to ShuffleNetv2 [29] without any channel split, which enables ShuffleNetv2 to deliver better performance than ShuffleNetv1. We believe that such functionalities (channel split and channel shuffle) are orthogonal to ESPNetv2 and can be used to further improve its efficiency and accuracy. (4) Compared to other efficient networks at a computational budget of about 300 million FLOPs, ESPNetv2 delivered better performance (e.g. 1.1% more accurate than the CondenseNet [18]).

Multi-label classification: To evaluate the generalizability for transfer learning, we evaluate our model on the MSCOCO multi-object classification task [25]. The dataset consists of 82,783 images, which are categorized into 80 classes with 2.9 object labels per image. Following [65], we evaluated our method on the validation set (40,504 images) using class-wise and overall F1 score. We finetune ESPNetv2 (284 million FLOPs) and Shufflenetv2 [29] (299 million FLOPs) for 100 epochs using the same data augmentation and training settings as for the ImageNet dataset, except η_{max}=0.005, η_{min}=0.001 and learning rate is decayed by two at the 50th and 80th epochs. We use binary cross entropy loss for optimization. Results are shown in Figure 6. ESPNetv2 outperforms ShuffleNetv2 by a large margin, especially when tested at image resolution of 896 × 896; suggesting large effective receptive fields of the EESP unit help ESPNetv2 learn better representations.

Performance analysis: Edge devices have limited computational resources and restrictive energy overhead. An efficient network for such devices should consume less power and have low latency with a high accuracy. We measure the efficiency of our network, ESPNetv2, along with other state-of-the-art networks (MobileNets [17, 44] and ShuffleNets [29, 60]) on two different devices: 1) a high-end graphics card (NVIDIA GTX 1080 Ti) and 2) an embedded device (NVIDIA Jetson TX2). For a fair comparison, we use PyTorch as a deep-learning framework. Figure 5 compares the inference time and power consumption while

Figure 6: Performance improvement in F1-score of ESPNetv2 over ShuffleNetv2 on MS-COCO multi-object classification task when *tested* at different image resolutions. Class-wise/overall F1-scores for ESPNetv2 and ShuffleNetv2 for an input of 224 × 224 on the validation set are 63.41/69.23 and 60.42/67.58 respectively.

networks complexity along with their accuracy are compared in Figure 3. The inference speed of ESPNetv2 is slightly lower than the fastest network (ShuffleNetv2 [29]) on both devices, however, it is much more power efficient while delivering similar accuracy on the ImageNet dataset. This suggests that ESPNetv2 network has a good trade-off between accuracy, power consumption, and latency; a much desirable property for any network running on edge devices.

4.2. Semantic segmentation

Dataset: We evaluate the performance of the ESPNetv2 on two datasets: (1) the Cityscapes [8] and (2) the PASCAL VOC 2012 dataset [10]. The Cityscapes dataset consists of 5,000 finely annotated images (training/validation/test: 2,975/500/1,525). The task is to segment an image into 19 classes that belongs to 7 categories. The PASCAL VOC 2012 dataset provide annotations for 20 foreground objects and has 1.4K training, 1.4K validation, and 1.4K test images. Following a standard convention [5, 63], we also use additional images from [14, 25] for training our networks.

Training: We train our network in two stages. In the first stage, we use a smaller image resolution for training (256 ×

256 for the PASCAL VOC 2012 dataset and 512 × 256 for the CityScapes dataset). We train ESPNetv2 for 100 epochs using SGD with an initial learning rate of 0.007. In the second stage, we increase the image resolution (384 × 384 for the PASCAL VOC 2012 and 1024 × 512 for the Cityscapes dataset) and then finetune the ESPNetv2 from first stage for 100 epochs using SGD with initial learning rate of 0.003. For both these stages, we use cyclic learning schedule discussed in Section 4.1. For the first 50 epochs, we use a cycle length of 5 while for the remaining epochs, we use a cycle length of 50 i.e. for the last 50 epochs, we decay the learning rate linearly. We evaluate the accuracy in terms of mean Intersection over Union (mIOU) on the private test set using *online evaluation server*. For evaluation, we up-sample segmented masks to the same size as of the input image using nearest neighbour interpolation.

Results: Figure 7 compares the performance of ESPNetv2 with state-of-the-methods on both the Cityscapes and the PASCAL VOC 2012 dataset. We can see that ESPNetv2 delivers a competitive performance to existing methods while being very efficient. Under the similar computational constraints, ESPNetv2 outerperforms existing methods like ENet and ESPNet by large margin. Notably, ESPNetv2 is 2-3% less accurate than other efficient networks such as ICNet, ERFNet, and ContextNet, but has $9 - 12\times$ fewer FLOPs.

4.3. Object detection

Dataset and training details: For object detection, we replace VGG with ESPNetv2 in single shot object detector. We evaluate the performance on two datasets: (1) the PASCAL VOC 2007 and (2) the MS-COCO dataset. For the PASCAL VOC 2007 dataset, we also use additional images from the PASCAL VOC 2012 dataset. We evaluate the performance in terms of mean Average Precision (mAP). For the COCO dataset, we report mAP @ IoU of 0.50:0.95. For training, we use the same learning policy as in Section 4.2.

Network	FLOPs	mIOU
SegNet [2]	82 B	57.0
ContextNet [37]	33 B	66.1
ICNet [61]	31 B	69.5
ERFNet [41]	26 B	69.7
MobileNetv2** [44]	21 B	**70.7**
RTSeg- MobileNet [45]	13.8 B	61.5
RTSeg-ShuffleNet [45]	6.2 B	58.3
ESPNet [32]	4.5 B	60.3
ENet [36]	3.8 B	58.3
ESPNetv2-val (Ours)	**2.7 B**	66.4
ESPNetv2-test (Ours)	**2.7 B**	66.2

(a) Cityscapes

Network	FLOPs	mIOU
FCN-8s [27]	181 B	62.2
DeepLabv3 [6]	81 B	**80.49**
SegNet [2]	31 B	59.1
MobileNetv1 [17]	14 B	75.29
MobileNetv2 [44]	5.8 B	75.7
ESPNet [32]	2.2 B	63.01
ESPNetv2 - val	0.76 B	67.0
ESPNetv2 - test	**0.76 B**	68.0

(b) PASCAL VOC 2012

Figure 7: Semantic segmentation results on (a) the Cityscapes dataset and (b) the PASCAL VOC 2012 dataset. For a fair comparison, we report FLOPs at the same image resolution which is used for computing the accuracy.

** [44] uses additional data from [25]

Network	VOC07		COCO	
	FLOPs	mAP	FLOPs	mAP
SSD-512 [26]	90.2 B	**74.9**	99.5 B	**26.8**
SSD-300 [26]	31.3 B	72.4	35.2 B	23.2
YOLOv2 [40]	6.8 B	69.0	17.5 B	21.6
MobileNetv1-320 [17]	–	–	1.3 B	22.2
MobileNetv2-320 [44]	–	–	0.8 B	22.1
ESPNetv2-512 (Ours)	2.5 B	68.2	2.8 B	26.0
ESPNetv2-384 (Ours)	1.4 B	65.6	1.6 B	23.2
ESPNetv2-256 (Ours)	**0.6 B**	63.8	**0.7 B**	21.9

Table 3: Object detection results on the PASCAL VOC 2007 and the MS-COCO dataset.

Results: Table 3 compares the performance of ESPNetv2 with existing methods. ESPNetv2 provides a good trade-off between accuracy and efficiency. Notably, ESPNetv2 delivers the same performance as YOLOv2, but has 25× fewer FLOPs. Compared to SSD, ESPNetv2 delivers a very competitive performance while being very efficient.

4.4. Language modeling

Dataset: The performance of our unit, the EESP, is evaluated on the Penn Treebank (PTB) dataset [30] as prepared by [35]. For training and evaluation, we follow the same splits of training, validation, and test data as in [34].

Language Model: We extend LSTM-based language models by replacing linear transforms for processing the input vector with the EESP unit inside the LSTM cell[2]. We call this model ERU (Efficient Recurrent Unit). Our model uses 3-layers of ERU with an embedding size of 400. We use standard dropout [48] with probability of 0.5 after embedding layer, the output between ERU layers, and the output of final ERU layer. We train the network using the same learning policy as [34]. We evaluate the performance in terms of perplexity; a lower value of perplexity is desirable.

Results: Language modeling results are provided in Table 4. ERUs achieve similar or better performance than state-of-the-art methods while learning fewer parameters. With similar hyper-parameter settings such as dropout, ERUs deliver similar (only 1 point less than PRU [32]) or better performance than state-of-the-art recurrent networks while learning fewer parameters; suggesting that the introduced EESP unit (Figure 1c) is efficient and powerful, and can be applied across different sequence modeling tasks such as question answering and machine translation. We note that our smallest language model with 7 million parameters outperforms most of state-of-the-art language models (e.g. [3, 11, 58]). We believe that the performance of

[2] We replace 2D convolutions with 1D convolutions in the EESP unit.

Language Model	# Params	Perplexity
Variational LSTM [11]	20 M	78.6
SRU [23]	24 M	60.3
Quantized LSTM [58]	–	89.8
QRNN [3]	18 M	78.3
Skip-connection LSTM [33]	24 M	58.3
AWD-LSTM [34]	24 M	57.3
PRU [31] (with standard dropout [48])	19 M	62.42
AWD-PRU [31] (with weight dropout [34])	19 M	**56.56**
ERU-Ours (with standard dropout [48])	**7 M**	73.63
	15 M	63.47

Table 4: This table compares single model word-level perplexity of our model with state-of-the-art on test set of the Penn Treebank dataset. Lower perplexity value represents better performance.

ERU can be further improved by rigorous hyper-parameter search [33] and advanced dropouts [11, 34].

5. Ablation Studies on the ImageNet Dataset

This section elaborate on various choices that helped make ESPNetv2 efficient and accurate.

Impact of different convolutions: Table 5 summarizes the impact of different convolutions. Clearly, depth-wise dilated separable convolutions are more effective than dilated and depth-wise convolutions.

Impact of hierarchical feature fusion (HFF): In [32], HFF is introduced to remove gridding artifacts caused by dilated convolutions. Here, we study their influence on object classification. The performance of the ESPNetv2 network with and without HFF are shown in Table 6 (see R1 and R2). HFF improves classification performance by about 1.5% while having no impact on the network's complexity. This suggests that the role of HFF is dual purpose. First, it removes gridding artifacts caused by dilated convolutions (as noted by [32]). Second, it enables sharing of information between different branches of the EESP unit (see Figure 1c) that allows it to learn rich and strong representations.

Impact of long-range shortcut connections with the input: To see the influence of shortcut connections with the input image, we train the ESPNetv2 network with and without shortcut connection. Results are shown in Table 6 (see R2 and R3). Clearly, these connections are effective and efficient, improving the performance by about 1% with a little (or negligible) impact on network's complexity.

Convolution	FLOPs	top-1
Dilated (standard)	478 M	69.2
Depth-wise separable	123 M	66.5
Depth-wise dilated separable	123 M	67.9

Table 5: ESPNetv2 with different convolutions. ESPNetv2 with standard dilated convolutions is the same as ESPNet.

	Network properties		Learning schedule		Performance		
	HFF	LRSC	Fixed	Cyclic	# Params	FLOPs	Top-1
R1	✗	✗	✓	✗	1.66 M	84 M	58.94
R2	✓	✗	✓	✗	1.66 M	84 M	60.07
R3	✓	✓	✓	✗	1.67 M	86 M	61.20
R4	✓	✓	✗	✓	1.67 M	86 M	62.17
R5†	✓	✓	✗	✓	1.67 M	86 M	66.10

Table 6: Performance of ESPNetv2 under different settings. Here, HFF represents hierarchical feature fusion and LRSC represents long-range shortcut connection with an input image. We train ESPNetv2 for 90 epochs and decay the learning rate by 10 after every 30 epochs. For fixed learning rate schedule, we initialize learning rate with 0.1 while for cyclic, we set η_{min} and η_{max} to 0.1 and 0.5 in Eq. 1 respectively. Here, † represents that the learning rate schedule is the same as in Section 4.1.

Fixed vs cyclic learning schedule: A comparison between fixed and cyclic learning schedule is shown in Table 6 (R3 and R4). With cyclic learning schedule, the ESPNetv2 network achieves about 1% higher top-1 validation accuracy on the ImageNet dataset; suggesting that cyclic learning schedule allows to find a better local minima than fixed learning schedule. Further, when we trained ESPNetv2 network for longer (300 epochs) using the learning schedule outlined in Section 4.1, performance improved by about 4% (see R4 and R5 in Table 6).

6. Conclusion

We introduce a light-weight and power efficient network, ESPNetv2, which better encode the spatial information in images by learning representations from a large effective receptive field. Our network is a general purpose network with good generalization abilities and can be used across a wide range of tasks, including sequence modeling. Our network delivered state-of-the-art performance across different tasks such as object classification, detection, segmentation, and language modeling while being more power efficient.

Acknowledgement: This research was supported by the Intelligence Advanced Research Projects Activity (IARPA) via Interior/Interior Business Center (DOI/IBC) contract number D17PC00343, NSF III (1703166), Allen Distinguished Investigator Award, Samsung GRO award, and gifts from Google, Amazon, and Bloomberg. We also thank Rik Koncel-Kedziorski, David Wadden, Beibin Li, and Anat Caspi for their helpful comments. The U.S. Government is authorized to reproduce and distribute reprints for Governmental purposes notwithstanding any copyright annotation thereon. Disclaimer: The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing endorsements, either expressed or implied, of IARPA, DOI/IBC, or the U.S. Government.

References

[1] Renzo Andri, Lukas Cavigelli, Davide Rossi, and Luca Benini. Yodann: An architecture for ultralow power binary-weight cnn acceleration. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 2018. 2

[2] Vijay Badrinarayanan, Alex Kendall, and Roberto Cipolla. Segnet: A deep convolutional encoder-decoder architecture for image segmentation. *TPAMI*, 2017. 7

[3] James Bradbury, Stephen Merity, Caiming Xiong, and Richard Socher. Quasi-recurrent neural networks. In *ICLR*, 2017. 8

[4] Han Cai, Ligeng Zhu, and Song Han. ProxylessNAS: Direct neural architecture search on target task and hardware. In *ICLR*, 2019. 2

[5] Liang-Chieh Chen, George Papandreou, Iasonas Kokkinos, Kevin Murphy, and Alan L Yuille. Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs. *TPAMI*, 2018. 6

[6] Liang-Chieh Chen, George Papandreou, Florian Schroff, and Hartwig Adam. Rethinking atrous convolution for semantic image segmentation. *arXiv preprint arXiv:1706.05587*, 2017. 7

[7] François Chollet. Xception: Deep learning with depthwise separable convolutions. In *CVPR*, 2017. 5

[8] Marius Cordts, Mohamed Omran, Sebastian Ramos, Timo Rehfeld, Markus Enzweiler, Rodrigo Benenson, Uwe Franke, Stefan Roth, and Bernt Schiele. The cityscapes dataset for semantic urban scene understanding. In *CVPR*, 2016. 6

[9] Matthieu Courbariaux, Itay Hubara, Daniel Soudry, Ran El-Yaniv, and Yoshua Bengio. Binarized neural networks: Training neural networks with weights and activations constrained to+ 1 or- 1. *arXiv preprint arXiv:1602.02830*, 2016. 2

[10] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman. The PASCAL Visual Object Classes Challenge 2012 (VOC2012) Results. http://www.pascal-network.org/challenges/VOC/voc2012/workshop/index.html. 6

[11] Yarin Gal and Zoubin Ghahramani. A theoretically grounded application of dropout in recurrent neural networks. In *NIPS*, 2016. 8

[12] Song Han, Huizi Mao, and William J Dally. Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding. *arXiv preprint arXiv:1510.00149*, 2015. 2

[13] Song Han, Jeff Pool, John Tran, and William Dally. Learning both weights and connections for efficient neural network. In *NIPS*, 2015. 2

[14] Bharath Hariharan, Pablo Arbeláez, Lubomir Bourdev, Subhransu Maji, and Jitendra Malik. Semantic contours from inverse detectors. In *ICCV*, 2011. 6

[15] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *ICCV*, 2015. 4, 5

[16] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, 2016. 1, 4, 5

[17] Andrew G Howard, 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. 1, 2, 5, 6, 7

[18] Gao Huang, Shichen Liu, Laurens van der Maaten, and Kilian Q Weinberger. Condensenet: An efficient densenet using learned group convolutions. In *CVPR*, 2018. 1, 2, 5, 6

[19] Gao Huang, Zhuang Liu, Laurens van der Maaten, and Kilian Q Weinberger. Densely connected convolutional networks. In *CVPR*, 2017. 5

[20] Itay Hubara, Matthieu Courbariaux, Daniel Soudry, Ran El-Yaniv, and Yoshua Bengio. Quantized neural networks: Training neural networks with low precision weights and activations. *arXiv preprint arXiv:1609.07061*, 2016. 1, 2

[21] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167*, 2015. 4

[22] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *NIPS*, 2012. 1, 2

[23] Tao Lei, Yu Zhang, and Yoav Artzi. Training rnns as fast as cnns. In *EMNLP*, 2018. 8

[24] Chong Li and CJ Richard Shi. Constrained optimization based low-rank approximation of deep neural networks. In *ECCV*, 2018. 1, 2

[25] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft coco: Common objects in context. In *ECCV*, 2014. 6, 7

[26] Wei Liu, 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*, pages 21–37. Springer, 2016. 7

[27] Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully convolutional networks for semantic segmentation. In *CVPR*, 2015. 7

[28] Ilya Loshchilov and Frank Hutter. Sgdr: Stochastic gradient descent with warm restarts. In *ICLR*, 2017. 5

[29] Ningning Ma, Xiangyu Zhang, Hai-Tao Zheng, and Jian Sun. Shufflenet v2: Practical guidelines for efficient cnn architecture design. In *ECCV*, 2018. 1, 2, 5, 6

[30] Mitchell P Marcus, Mary Ann Marcinkiewicz, and Beatrice Santorini. Building a large annotated corpus of english: The penn treebank. *Computational linguistics*, 1993. 7

[31] Sachin Mehta, Rik Koncel-Kedziorski, Mohammad Rastegari, and Hannaneh Hajishirzi. Pyramidal recurrent unit for language modeling. In *EMNLP*, 2018. 8

[32] Sachin Mehta, Mohammad Rastegari, Anat Caspi, Linda Shapiro, and Hannaneh Hajishirzi. Espnet: Efficient spatial pyramid of dilated convolutions for semantic segmentation. In *ECCV*, 2018. 1, 2, 3, 5, 7, 8

[33] Gábor Melis, Chris Dyer, and Phil Blunsom. On the state of the art of evaluation in neural language models. In *ICLR*, 2018. 8

[34] Stephen Merity, Nitish Shirish Keskar, and Richard Socher. Regularizing and optimizing lstm language models. In *ICLR*, 2018. 1, 7, 8

[35] Tomáš Mikolov, Martin Karafiát, Lukáš Burget, Jan Černockỳ, and Sanjeev Khudanpur. Recurrent neural network based language model. In *Eleventh Annual Conference of the International Speech Communication Association*, 2010. 7

[36] Adam Paszke, Abhishek Chaurasia, Sangpil Kim, and Eugenio Culurciello. Enet: A deep neural network architecture for real-time semantic segmentation. *arXiv preprint arXiv:1606.02147*, 2016. 7

[37] Rudra PK Poudel, Ujwal Bonde, Stephan Liwicki, and Christopher Zach. Contextnet: Exploring context and detail for semantic segmentation in real-time. In *BMVC*, 2018. 7

[38] PyTorch. Tensors and Dynamic neural networks in Python with strong GPU acceleration. http://pytorch.org/. Accessed: 2018-11-15. 5

[39] Mohammad Rastegari, Vicente Ordonez, Joseph Redmon, and Ali Farhadi. Xnor-net: Imagenet classification using binary convolutional neural networks. In *ECCV*, 2016. 1, 2

[40] Joseph Redmon and Ali Farhadi. Yolo9000: better, faster, stronger. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 7263–7271, 2017. 7

[41] Eduardo Romera, José M Alvarez, Luis M Bergasa, and Roberto Arroyo. Erfnet: Efficient residual factorized convnet for real-time semantic segmentation. *IEEE Transactions on Intelligent Transportation Systems*, 2018. 7

[42] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *MICCAI*, 2015. 1

[43] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. *IJCV*, 2015. 2, 4, 5

[44] Mark Sandler, Andrew Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen. Mobilenetv2: Inverted residuals and linear bottlenecks. In *CVPR*, 2018. 1, 2, 5, 6, 7

[45] M. Siam, M. Gamal, M. Abdel-Razek, S. Yogamani, and M. Jagersand. rtseg: Real-time semantic segmentation comparative study. In *2018 25th IEEE International Conference on Image Processing (ICIP)*. 7

[46] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. In *ICLR*, 2014. 4

[47] Daniel Soudry, Itay Hubara, and Ron Meir. Expectation backpropagation: Parameter-free training of multilayer neural networks with continuous or discrete weights. In *NIPS*, 2014. 1, 2

[48] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *JMLR*, 2014. 7, 8

[49] Ke Sun, Mingjie Li, Dong Liu, and Jingdong Wang. Igcv3: Interleaved low-rank group convolutions for efficient deep neural networks. In *BMVC*, 2018. 5

[50] Ilya Sutskever, James Martens, George Dahl, and Geoffrey Hinton. On the importance of initialization and momentum in deep learning. In *ICML*, 2013. 5

[51] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *CVPR*, 2015. 5

[52] Mingxing Tan, Bo Chen, Ruoming Pang, Vijay Vasudevan, and Quoc V Le. Mnasnet: Platform-aware neural architecture search for mobile. *arXiv preprint arXiv:1807.11626*, 2018. 2

[53] Andreas Veit and Serge Belongie. Convolutional networks with adaptive inference graphs. In *ECCV*, 2018. 2

[54] Panqu Wang, Pengfei Chen, Ye Yuan, Ding Liu, Zehua Huang, Xiaodi Hou, and Garrison Cottrell. Understanding convolution for semantic segmentation. In *WACV*, 2018. 3

[55] Wei Wen, Chunpeng Wu, Yandan Wang, Yiran Chen, and Hai Li. Learning structured sparsity in deep neural networks. In *NIPS*, 2016. 1, 2

[56] Bichen Wu, Xiaoliang Dai, Peizhao Zhang, Yanghan Wang, Fei Sun, Yiming Wu, Yuandong Tian, Peter Vajda, Yangqing Jia, and Kurt Keutzer. Fbnet: Hardware-aware efficient convnet design via differentiable neural architecture search. *arXiv preprint arXiv:1812.03443*, 2018. 2

[57] Jiaxiang Wu, Cong Leng, Yuhang Wang, Qinghao Hu, and Jian Cheng. Quantized convolutional neural networks for mobile devices. In *CVPR*, 2016. 2

[58] Chen Xu, Jianqiang Yao, Zhouchen Lin, Wenwu Ou, Yuanbin Cao, Zhirong Wang, and Hongbin Zha. Alternating multi-bit quantization for recurrent neural networks. In *ICLR*, 2018. 8

[59] Fisher Yu, Vladlen Koltun, and Thomas A Funkhouser. Dilated residual networks. In *CVPR*, 2017. 3

[60] Xiangyu Zhang, Xinyu Zhou, Mengxiao Lin, and Jian Sun. Shufflenet: An extremely efficient convolutional neural network for mobile devices. In *CVPR*, 2018. 1, 2, 4, 5, 6

[61] Hengshuang Zhao, Xiaojuan Qi, Xiaoyong Shen, Jianping Shi, and Jiaya Jia. Icnet for real-time semantic segmentation on high-resolution images. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 405–420, 2018. 7

[62] Hengshuang Zhao, Jianping Shi, Xiaojuan Qi, Xiaogang Wang, and Jiaya Jia. Pyramid scene parsing network. In *CVPR*, 2017. 1

[63] Hengshuang Zhao, Jianping Shi, Xiaojuan Qi, Xiaogang Wang, and Jiaya Jia. Pyramid scene parsing network. In *CVPR*, 2017. 6

[64] Shuchang Zhou, Yuxin Wu, Zekun Ni, Xinyu Zhou, He Wen, and Yuheng Zou. Dorefa-net: Training low bitwidth convolutional neural networks with low bitwidth gradients. *arXiv preprint arXiv:1606.06160*, 2016. 2

[65] Feng Zhu, Hongsheng Li, Wanli Ouyang, Nenghai Yu, and Xiaogang Wang. Learning spatial regularization with image-level supervisions for multi-label image classification. *CVPR*, 2017. 6

[66] Barret Zoph and Quoc V Le. Neural architecture search with reinforcement learning. *arXiv preprint arXiv:1611.01578*, 2016. 2