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

Sachin Mehta[1], Mohammad Rastegari[2,3], Linda Shapiro[1], and Hannaneh Hajishirzi[1,2]

[1]*University of Washington* [2]*Allen Institute for AI (AI2)* [3]*XNOR.AI*

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 three different tasks: (1) object classification, (2) semantic segmentation, and (3) 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 has better generalization properties than ShuffleNetv2 when tested on the MSCOCO multi-object classification task and the Cityscapes urban scene semantic segmentation task. 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 [12, 19, 30]. CNNs are being used in real-world visual recognition applications such as visual scene understanding [52] and bio-medical image analysis [36]. 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 [12], 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 [21, 47]. (2) *Low-bit representation-based methods* represent learned weights using few bits instead of high precision floating points [16, 34, 40]. 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 [13, 14, 25, 28, 38, 51]. 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. Our model extends ESPNet [28], a light-weight semantic segmentation network, by using group point-wise and depth-wise "dilated" separable convolutions instead of computationally expensive point-wise and dilated convolutions. This reduces network parameters and complexity while maintaining high accuracy. The core builiding block of our network, the EESP unit, is general and can be used across wide range of visual and sequence modeling tasks. Our approach is orthogonal to the current state-of-the-art ef-

ficient models [25,51] yet reaches higher performance without any channel shuffle or channel split, which have been shown to be very effective for improving the accuracy of light-weight models.

To show the generalizability of our model, we evaluate our network across three different tasks: (1) object classification, (2) semantic segmentation, and (3) language modeling. On the ImageNet classification task [37], 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 [38] by 2% at a computational budget of 28 million FLOPs. Our most efficient model learns 3.5 million parameters and has 284 million FLOPs while delivering a top-1 classification accuracy of 72.1% on the ImageNet classification task. Our network has better generalization properties than ShuffleNetv2 [25] when tested on the MSCOCO multi-object classification task [22] and the Cityscapes urban scene semantic segmentation task [6]. Furthermore, we show that the introduced EESP unit can be used as a drop-in replacement for recurrent neural networks and delivers state-of-the-art performance while learning fewer parameters. Our experiments also show that our network is much more power efficient than existing state-of-the-art efficient methods including ShuffleNets [25, 51] and MobileNets [13, 38].

We also propose to use SGD with a cyclic learning rate schedule and warm restarts. In each cycle, the learning rate is initialized to its maximum value and is then scheduled to decrease linearly to its minimum value. Our experimental results on the ImageNet dataset suggest that such a learning policy helps the network avoid saddle points and reach higher accuracy in comparison to widely used step-wise learning policies. Our code is open-source and available at https://github.com/sacmehta/ESPNetv2.

2. Related Work

This section briefly reviews different methods for building efficient networks.

Efficient CNN architectures: Most state-of-the-art efficient networks [13, 25, 38] use depth-wise separable convolutions [13] 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 [14,51] is group convolution [19], wherein input channels and convolutional kernels are factored into groups and each group is convolved independently. The ESPNetv2 network extends the ESPNet network [28] 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 amd accuracy can be further improved using methods such as channel shuffle [25] and channel split [25]. Such methods are orthogonal to our work.

Network compression: These approaches improve the inference of a pre-trained network by pruning network connections or channels [9, 10, 21, 45, 47]. 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, 7, 16, 34, 40, 48, 53]. 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.

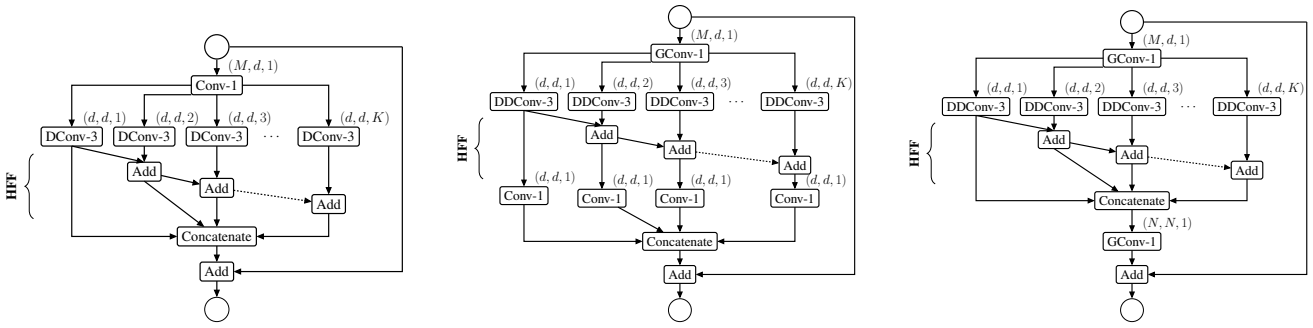
3. ESPNetv2

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 [13, 25, 38, 51]. The basic idea is to replace the full convolutional operation with a factorized version such as depth-wise separable convolution [13] or group convolution [19]. 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



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

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	$n^2 c\hat{c}$	$n \times n$
Group	$\frac{n^2 c\hat{c}}{g}$	$n \times n$
Depth-wise separable	$n^2 c + c\hat{c}$	$n \times n$
Depth-wise dilated separable	$n^2 c + c\hat{c}$	$n_r \times n_r$

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.

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 factor of $\frac{n^2 c\hat{c}}{n^2 c + c\hat{c}}$. 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 [28], 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 [28]. 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 $\frac{Md + n^2 d^2 K}{\frac{Md}{g} + (n^2 + d)dK}$, 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

[1] Other existing works [46,50] 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 [37]. Network's complexity is evaluated in terms of total number of multiplication-addition operations (or FLOPs).

convolutions with a group point-wise convolution, as shown in Figure 1c. We will refer to this unit as EESP.

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 [51].

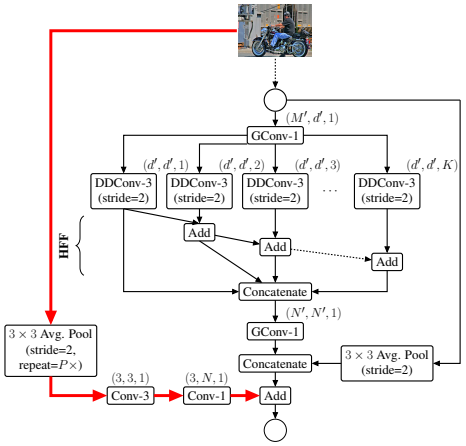


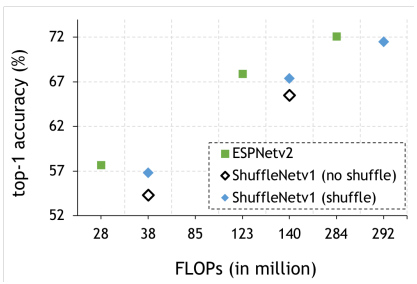
Figure 2: Strided EESP unit with shortcut connection to an input image (highlighted in red) for down-sampling. The average pooling operation is repeated $P\times$ to match the spatial dimensions of an input image and feature maps.

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.

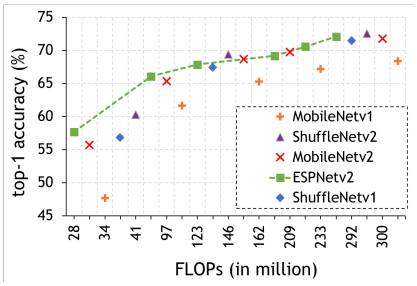
3.3. Network architecture

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 [17] and PReLU [11] 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 [12, 39].

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



(a)



(b)

Network	# Params	FLOPs	Top-1
MobileNetv1 [13]	**2.59 M**	325 M	68.4
MobileNetv2 [38]	3.47 M	300 M	71.8
CondenseNet [14]	–	**274 M**	71.0
IGCV3 [42]	–	318 M	72.2
Xception† [5]	–	305 M	70.6
DenseNet† [15]	–	295 M	60.1
ShuffleNetv1 [51]	3.46 M	292 M	71.5
ShuffleNetv2 [25]	3.5 M	299 M	**72.6**
ESPNetv2 (Ours)	3.49 M	284 M	72.1

(c)

Figure 3: Performance comparison of different efficient networks on the ImageNet validation set: (a) ESPNetv2 vs. ShuffleNetv1 [51], (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 [25]. Best viewed in color.

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: $\mathrm{n}_d^l(Z^l) = 5 + \frac{Z^l}{7}, Z^l \in \{W^l, H^l\}$ with the effective receptive field $(n_d \times n_d)$ corresponding to the lowest spatial level (i.e. 7×7) as 5×5. Following [28], 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 three different tasks: (1) object classification, (2) semantic segmentation, and (3) language modeling.

4.1. Image classification

Dataset: We evaluate the performance of the ESPNetv2 on the ImageNet 2012 dataset [37] that contains 1.28 million images for training and 50,000 images for validation. The task is to classify an image into 1,000 categories. 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 [33] with CUDA 9.0 and cuDNN as the back-ends. For optimization, we use SGD [43] 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 cycles. This learning rate scheme can be seen as a variant of the cosine learning policy [24], 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 [12, 44] with an exception to color-based normalization. This is in contrast to recent efficient architectures that uses less scale augmentation to prevent underfitting [25, 51]. The weights of our networks are initialized using the method described in [11].

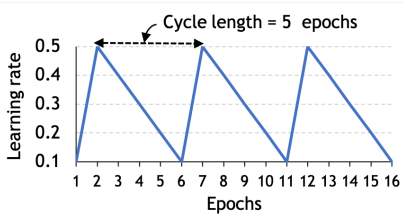
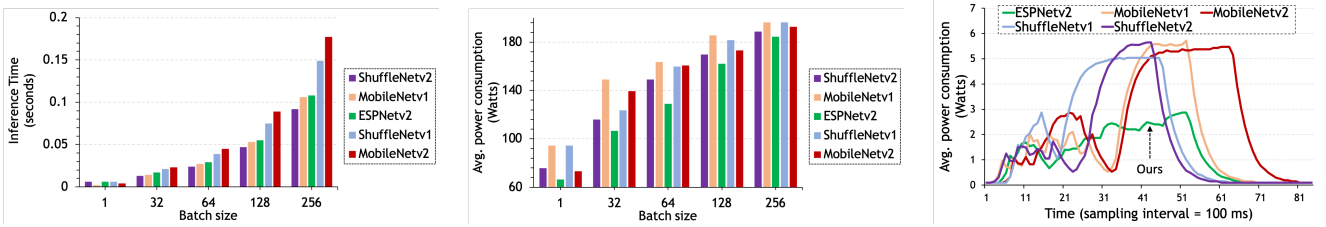


Figure 4: Cyclic learning rate policy (see Eq.1) with linear learning rate decay and warm restarts.

Results: Figure 3 provides a performance comparison between ESPNetv2 and state-of-the-art efficient networks. We observe that

1. ESPNetv2 outperforms ShuffleNetv1 [51] with or without channel shuffle; suggesting that our architecture enables learning of efficient representations.
2. ESPNetv2 outperforms MobileNets [13, 38], especially under small computational budgets. With 28 million FLOPs, ESPNetv2 outperforms MobileNetv1 [13]



(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.

(34 million FLOPs) and MobileNetv2 [38] (30 million FLOPs) by 10% and 2% respectively.

3. ShuffleNetv2 [25] extends ShuffleNetv1 [51] by adding channel split functionality, which enables it to deliver better performance than ShuffleNetv1. ESPNetv2 delivers comparable accuracy to ShuffleNetv2 without any channel split or shuffle. We believe that such functionalities are orthogonal to our network and can further improve its efficiency and accuracy.
4. Compared to other efficient networks at a computational budget of approximately 300 million FLOPs, ESPNetv2 delivered better performance (e.g. 1.1% more accurate than the CondenseNet [14]).

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.

Generalizability: To evaluate the generalizability for transfer learning, we evaluate our model on the MSCOCO multi-object classification task [22]. The dataset consists of 82,783 images, which are categorized into 80 classes with 2.9 object labels per image. Following [54], 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 [25] (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 and generalizable 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 [13, 38] and ShuffleNets [25, 51]) 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 networks complexity along with their accuracy are compared in Figure 3. The inference speed of ESPNetv2 is slightly lower than the fastest network (ShuffleNetv2 [25]) 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.

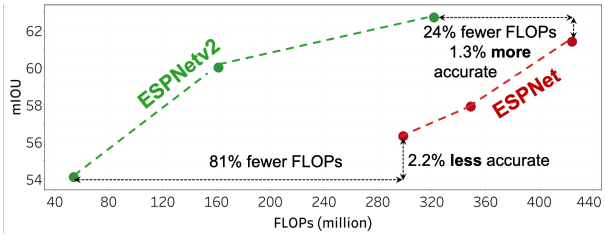
4.2. Semantic segmentation

Dataset: We evaluate the performance of the ESPNetv2 on an urban scene semantic segmentation dataset, the Cityscapes [6]. The dataset is collected across 50 cities in different environmental conditions such as weather and season. It 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.

Training: We train our network for 300 epochs using ADAM [18] with an initial learning rate of 0.0005 and polynomial rate decay with a power of 0.9. Standard data

augmentation strategies, such as scaling, cropping and flipping, are used while training the networks. For training, we sub-sample the images by a factor of 2 (or image size of 1024×512). 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 (i.e. 2048×1024) using nearest neighbour interpolation.

Results: The performance of the ESPNetv2 with the ESPNet [28] is compared in Figure 7a. Clearly, the ESPNetv2 is much more efficient and accurate than the ESPNet. When the base segmentation network, ESPNetv2 , is replaced with ShuffleNetv2 (with the same computational complexity), the performance of the segmentation network is dropped by about 2%; suggesting that ESPNetv2 has better generalization properties (see Figure 7b). Furthermore, Figure 7c provides a comparison between the ESPNetv2 network and state-of-the-art networks. Under the same computational constraints, ESPNetv2 is 4% and 2% more accurate than ENet [32] and ESPNet [28] respectively.



(a) ESPNet vs. ESPNetv2 (validation set)

Base network	**mIOU** (val)
ESPNetv2 (284 M FLOPs)	62.7
ShuffleNetv2 (299 M FLOPs)	60.3

(b) Performance with different efficient base networks

Network	**# Params**	**FLOPs**	**Speed (FPS)**	**mIOU**
PSPNet [52]	67 M	82.78 B	5	**78.4**
FCN-8s [23]	134 M	62.71 B	15	65.3
DeepLab-v2 [4]	44 M	37.51 B	6	70.4
SegNet [2]	29.45 M	31 B	17	57.0
ERFNet [35]	2.06 M	2.45 B	48	68.0
ESPNet [28]	364 K	424 M	112	60.3
ENet [32]	364 K	345 M	88	58.3
ESPNetv2 (Ours)	725 K	322 M	83	62.1
	99 K	**54 M**	**142**	54.7

(c) Comparison with state-of-the-art networks on test set

Figure 7: Performance on the Cityscapes validation and test sets. Here, performance is measured in terms of class-wise mean intersection over union (mIOU). We measure FLOPs for a 224×224 inputs and inference speed (averaged over 100 iterations) in terms of frames processed per second (FPS) for a 1024×512 inputs on a NVIDIA TitanX GPU.

Language Model	**# Params**	**Perplexity**
Variational LSTM [8]	20 M	78.6
SRU [20]	24 M	60.3
Quantized LSTM [49]	–	89.8
QRNN [3]	18 M	78.3
Skip-connection LSTM [29]	24 M	58.3
AWD-LSTM [30]	24 M	57.3
PRU [27] (with standard dropout [41])	19 M	62.42
AWD-PRU [27] (with weight dropout [30])	19 M	**56.56**
ERU-Ours (with standard dropout [41])	**7 M**	73.63
	15 M	63.47

Table 3: 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.

4.3. Language modeling

Dataset: The performance of our unit, the EESP, is evaluated on the Penn Treebank (PTB) dataset [26] as prepared by [31]. For training and evaluation, we follow the same splits of training, validation, and test data as in [30].

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 [41] 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 [30]. We evaluate the performance in terms of perplexity; a lower value of perplexity is desirable.

Results: Language modeling results are provided in Table 3. 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 [28]) 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, 8, 49]). We believe that the performance of ERU can be further improved by rigorous hyper-parameter search [29] and advanced dropouts [8, 30].

5. Ablation Studies on the ImageNet Dataset

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

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

	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 4: 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.

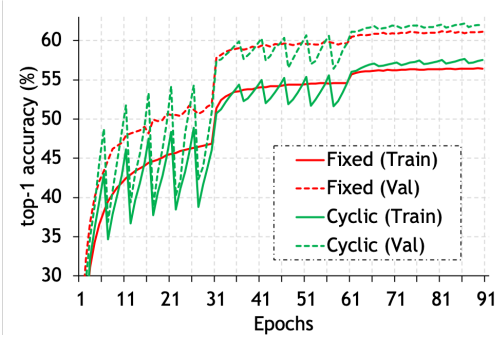
Impact of hierarchical feature fusion (HFF): In [28], HFF is introduced to remove gridding artifacts caused by dilated convolutions. Here, we study their influence on object classification. The performance of the ESPNetv2 network with and without HFF are shown in Table 4 (see R1 and R2). HFF improves 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 [28]). 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 4 (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.

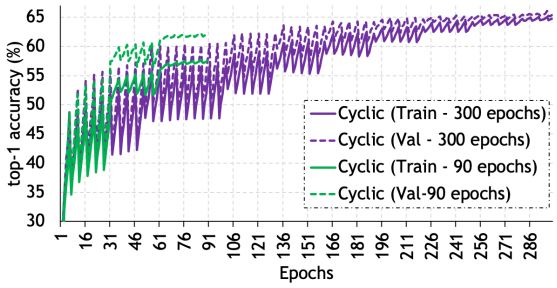
Fixed vs cyclic learning schedule: A comparison between fixed and cyclic learning schedule is shown in Figure 8a and Table 4 (R3 and R4). With cyclic learning schedule, the ESPNetv2 network achieves about 1% higher top-1 validation accuracy on the ImageNet dataset; suggesting that cyclic learning schedule 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 4 and Figure 8b). We observed similar gains when we trained ShuffleNetv1 [51]; top-1 accuracy improved by 3.2% to 65.86 over fixed learning schedule[3].

Group point-wise convolutions for dimensionality reduction: We replace group point-wise convolutions in Figure 1c (or Figure 1b) with point-wise convolutions for dimensionality reduction. Results are shown in Table 5. With group point-wise convolutions, the ESPNetv2 network is able to achieve similar performance as with point-wise convolutions, but more efficiently.

[3]We note that the top-1 accuracy is lower than the reported accuracy of 67.2; likely due to different scale augmentation.



(a) Fixed vs. cyclic (# epochs=90)



(b) Impact of training for longer duration (90 vs. 300 epochs).

Figure 8: Impact of different learning schedules on the performance of ESPNetv2 (FLOPs = 86 M). Best viewed in color.

Group point-wise ($g = 4$)			Point-wise		
# Params	FLOPs	Top-1	# Params	FLOPs	Top-1
1.24 M	28 M	57.7	1.30 M	39 M	57.14
1.67 M	86 M	66.1	1.92 M	127 M	67.14

Table 5: Impact of group point-wise and point-wise convolutions on the performance of the ESPNetv2 network. Top-1 accuracy is measured on the ImageNet validation dataset. Training policy is the same as discussed in Section 4.1.

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, semantic segmentation,

and language modeling while being much 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] R. Andri, L. Cavigelli, D. Rossi, and L. 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] V. Badrinarayanan, A. Kendall, and R. Cipolla. Segnet: A deep convolutional encoder-decoder architecture for image segmentation. *TPAMI*, 2017. 7

[3] J. Bradbury, S. Merity, C. Xiong, and R. Socher. Quasi-recurrent neural networks. In *ICLR*, 2017. 7

[4] L.-C. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. L. Yuille. Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs. *TPAMI*, 2018. 7

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

[6] M. Cordts, M. Omran, S. Ramos, T. Rehfeld, M. Enzweiler, R. Benenson, U. Franke, S. Roth, and B. Schiele. The cityscapes dataset for semantic urban scene understanding. In *CVPR*, 2016. 2, 6

[7] M. Courbariaux, I. Hubara, D. Soudry, R. El-Yaniv, and Y. Bengio. Binarized neural networks: Training neural networks with weights and activations constrained to+ 1 or- 1. *arXiv preprint arXiv:1602.02830*, 2016. 2

[8] Y. Gal and Z. Ghahramani. A theoretically grounded application of dropout in recurrent neural networks. In *NIPS*, 2016. 7

[9] S. Han, H. Mao, and W. J. Dally. Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding. *arXiv preprint arXiv:1510.00149*, 2015. 2

[10] S. Han, J. Pool, J. Tran, and W. Dally. Learning both weights and connections for efficient neural network. In *NIPS*, 2015. 2

[11] K. He, X. Zhang, S. Ren, and J. Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *ICCV*, 2015. 4, 5

[12] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *CVPR*, 2016. 1, 4, 5

[13] A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, and H. Adam. Mobilenets: Efficient convolutional neural networks for mobile vision applications. *arXiv preprint arXiv:1704.04861*, 2017. 1, 2, 5, 6

[14] G. Huang, S. Liu, L. van der Maaten, and K. Q. Weinberger. Condensenet: An efficient densenet using learned group convolutions. In *CVPR*, 2018. 1, 2, 5, 6

[15] G. Huang, Z. Liu, L. van der Maaten, and K. Q. Weinberger. Densely connected convolutional networks. In *CVPR*, 2017. 5

[16] I. Hubara, M. Courbariaux, D. Soudry, R. El-Yaniv, and Y. Bengio. Quantized neural networks: Training neural networks with low precision weights and activations. *arXiv preprint arXiv:1609.07061*, 2016. 1, 2

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

[18] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. In *ICLR*, 2015. 6

[19] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *NIPS*, 2012. 1, 2

[20] T. Lei, Y. Zhang, and Y. Artzi. Training rnns as fast as cnns. In *EMNLP*, 2018. 7

[21] C. Li and C. R. Shi. Constrained optimization based low-rank approximation of deep neural networks. In *ECCV*, 2018. 1, 2

[22] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick. Microsoft coco: Common objects in context. In *ECCV*, 2014. 2, 6

[23] J. Long, E. Shelhamer, and T. Darrell. Fully convolutional networks for semantic segmentation. In *CVPR*, 2015. 7

[24] I. Loshchilov and F. Hutter. Sgdr: Stochastic gradient descent with warm restarts. In *ICLR*, 2017. 5

[25] N. Ma, X. Zhang, H.-T. Zheng, and J. Sun. Shufflenet v2: Practical guidelines for efficient cnn architecture design. In *ECCV*, 2018. 1, 2, 5, 6

[26] M. P. Marcus, M. A. Marcinkiewicz, and B. Santorini. Building a large annotated corpus of english: The penn treebank. *Computational linguistics*, 1993. 7

[27] S. Mehta, R. Koncel-Kedziorski, M. Rastegari, and H. Hajishirzi. Pyramidal recurrent unit for language modeling. In *EMNLP*, 2018. 7

[28] S. Mehta, M. Rastegari, A. Caspi, L. Shapiro, and H. Hajishirzi. Espnet: Efficient spatial pyramid of dilated convolutions for semantic segmentation. In *ECCV*, 2018. 1, 2, 3, 5, 7, 8

[29] G. Melis, C. Dyer, and P. Blunsom. On the state of the art of evaluation in neural language models. In *ICLR*, 2018. 7

[30] S. Merity, N. S. Keskar, and R. Socher. Regularizing and optimizing lstm language models. In *ICLR*, 2018. 1, 7

[31] T. Mikolov, M. Karafiát, L. Burget, J. Černockỳ, and S. Khudanpur. Recurrent neural network based language model. In *Eleventh Annual Conference of the International Speech Communication Association*, 2010. 7

[32] A. Paszke, A. Chaurasia, S. Kim, and E. Culurciello. Enet: A deep neural network architecture for real-time semantic segmentation. *arXiv preprint arXiv:1606.02147*, 2016. 7

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

[34] M. Rastegari, V. Ordonez, J. Redmon, and A. Farhadi. Xnor-net: Imagenet classification using binary convolutional neural networks. In *ECCV*, 2016. 1, 2

[35] E. Romera, J. M. Alvarez, L. M. Bergasa, and R. Arroyo. Erfnet: Efficient residual factorized convnet for real-time semantic segmentation. *IEEE Transactions on Intelligent Transportation Systems*, 2018. 7

[36] O. Ronneberger, P. Fischer, and T. Brox. U-net: Convolutional networks for biomedical image segmentation. In *MICCAI*, 2015. 1

[37] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. *IJCV*, 2015. 2, 4, 5

[38] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen. Mobilenetv2: Inverted residuals and linear bottlenecks. In *CVPR*, 2018. 1, 2, 5, 6

[39] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. In *ICLR*, 2014. 4

[40] D. Soudry, I. Hubara, and R. Meir. Expectation backpropagation: Parameter-free training of multilayer neural networks with continuous or discrete weights. In *NIPS*, 2014. 1, 2

[41] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *JMLR*, 2014. 7

[42] K. Sun, M. Li, D. Liu, and J. Wang. Igcv3: Interleaved low-rank group convolutions for efficient deep neural networks. In *BMVC*, 2018. 5

[43] I. Sutskever, J. Martens, G. Dahl, and G. Hinton. On the importance of initialization and momentum in deep learning. In *ICML*, 2013. 5

[44] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich. Going deeper with convolutions. In *CVPR*, 2015. 5

[45] A. Veit and S. Belongie. Convolutional networks with adaptive inference graphs. In *ECCV*, 2018. 2

[46] P. Wang, P. Chen, Y. Yuan, D. Liu, Z. Huang, X. Hou, and G. Cottrell. Understanding convolution for semantic segmentation. In *WACV*, 2018. 3

[47] W. Wen, C. Wu, Y. Wang, Y. Chen, and H. Li. Learning structured sparsity in deep neural networks. In *NIPS*, 2016. 1, 2

[48] J. Wu, C. Leng, Y. Wang, Q. Hu, and J. Cheng. Quantized convolutional neural networks for mobile devices. In *CVPR*, 2016. 2

[49] C. Xu, J. Yao, Z. Lin, W. Ou, Y. Cao, Z. Wang, and H. Zha. Alternating multi-bit quantization for recurrent neural networks. In *ICLR*, 2018. 7

[50] F. Yu, V. Koltun, and T. A. Funkhouser. Dilated residual networks. In *CVPR*, 2017. 3

[51] X. Zhang, X. Zhou, M. Lin, and J. Sun. Shufflenet: An extremely efficient convolutional neural network for mobile devices. In *CVPR*, 2018. 1, 2, 4, 5, 6, 8

[52] H. Zhao, J. Shi, X. Qi, X. Wang, and J. Jia. Pyramid scene parsing network. In *CVPR*, 2017. 1, 7

[53] S. Zhou, Y. Wu, Z. Ni, X. Zhou, H. Wen, and Y. Zou. Dorefa-net: Training low bitwidth convolutional neural networks with low bitwidth gradients. *arXiv preprint arXiv:1606.06160*, 2016. 2

[54] F. Zhu, H. Li, W. Ouyang, N. Yu, and X. Wang. Learning spatial regularization with image-level supervisions for multi-label image classification. *CVPR*, 2017. 6