DivNet: Efficient Convolutional Neural Network via Multilevel Hierarchical Architecture Design

BACHIR KADDAR\textsuperscript{1,2}, HADRIA FIZAZI\textsuperscript{2}, MIGUEL HERNÁNDEZ-CABRONERO\textsuperscript{3}, VICTOR SANCHEZ\textsuperscript{4}, (Member, IEEE), AND JOAN SERRA-SAGRISTÀ\textsuperscript{3}, (Senior Member, IEEE)

\textsuperscript{1}Department of Natural and Life Sciences, University Ibn Khaldoun, Tiaret 14000, Algeria
\textsuperscript{2}Department of Computer Sciences, Université des Sciences et de la Technologie d’Oran Mohamed Boudiaf (USTO-MB), Bir El Djir, Oran 31000, Algeria
\textsuperscript{3}Department of Information and Communications Engineering, Universitat Autònoma de Barcelona, Cerdanyola del Vallès, 08193 Bellaterra, Spain
\textsuperscript{4}Department of Computer Science, University of Warwick, Coventry CV4 7AL, U.K.

Corresponding author: Bachir Kaddar (bachir.kaddar@univ-usto.dz)

This work was supported in part by the Secretary of Universities and Research (Government of Catalonia) under Grant 2017SGR-463, in part by the Postdoctoral Fellowship Program Beatriu de Pinós under Grant 2018-BP-00008, in part by the Horizon 2020 Program of Research and Innovation of the European Union under the Marie Skłodowska-Curie Grant under Agreement 801370, and in part by the Spanish Ministry of Economy and Competitiveness and the Fondo Europeo de Desarrollo Regional (MINECO/FEDER, EU) under Grant RTI2018-095287-B-I00.

\textbf{ABSTRACT} Designing small and efficient mobile neural networks is difficult because the challenge is to determine the architecture that achieves the best performance under a given limited computational scenario. Previous lightweight neural networks rely on a cell module that is repeated in all stacked layers across the network. These approaches do not permit layer diversity, which is critical for achieving strong performance. This paper presents an experimental study to develop an efficient mobile network using a hierarchical architecture. Our proposed mobile network, called Diversity Network (DivNet), has been shown to perform better than the basic architecture generally employed by the best high-efficiency models –with simply stacked layers–, regarding complexity cost and performance. A set of architectural design decisions are described that reduce the proposed model size while yielding a significant performance improvement. Our experiments on image classification show that compared to, respectively, MobileNetV2, SqueezeNet, and ShuffleNetV2, our proposal DivNet can improve accuracy by 2.09\%, 0.76\%, and 0.66\% on the CIFAR100 dataset, and by 0.05\%, 4.96\%, and 1.13\% on the CIFAR10 dataset. On more complex datasets, e.g., ImageNet, our proposal DivNet achieves 70.65\% Top-1 accuracy and 90.23\% Top-5 accuracy, still better than other small models like MobilNet, SqueezeNet, ShuffleNet.

\textbf{INDEX TERMS} Deep neural network, mobile network, network compression.

I. INTRODUCTION

Convolutional neural networks have been widely used in many computer vision applications in recent years, achieving state-of-the-art results in numerous tasks such as image classification [1], object detection [2], video analysis [3], and others [4]. The performance on these different computer vision tasks is dramatically boosted by sophisticated architectures such as AlexNet [5], Network In Network (NIN) [6], VGG-Net [7], Inception Network [8], ResNet [9], and others [10].

Since 2012, the overall trend has been to increase the depth and width of the networks in order to improve performance [11]. However, such improvements often result in a very large number of parameters [9]. Over-parameterized models have high computational complexity, which may be prohibitive especially with embedded sensors and mobile devices where computational and power resources are usually limited [12]. For many recent applications, such as Augmented Reality (AR), Self-Driving Cars, and Autonomous Devices, an important challenge is to handle recognition tasks under computationally restricted resources. On the other hand, real-time classification and search require fast inference models enabling quick and easy data processing. For these applications, in addition to accuracy, computational efficiency and small network sizes are crucially enabling factors. This is leading researchers to re-examine the issue of network complexity and propose more efficient architectures with fewer parameters and convolution operations.
MobileNet is an example of an efficient model with fewer parameters [13]. Specifically, MobileNet uses depthwise separable convolutions [15], [16] to build lightweight neural networks that trade off a reasonable amount of accuracy over model size and latency. These convolutional filters are the main component of subsequent works focusing on fast and lightweight networks. However, one drawback of these filters is the fact that the expensive pointwise convolutions result in a limited number of feature channels, i.e., feature maps, to meet feature-diversity requirements, which may significantly hinder performance. Additionally, these filters become less efficient in extremely small networks because of the costly dense $1 \times 1$ convolutions. In [14], a simple design called group-wise convolution is proposed to effectively reduce the number of parameters, while allowing sufficient feature diversity. The authors introduce a novel channel shuffle operation to help the flowing of information across feature channels and overcome the side effects brought by group convolutions. MobileNetV2 [17] achieves the best results by using resource-efficient inverted residuals and linear bottlenecks. Recently, [18] proposes a series of practical guidelines for efficient CNN design and achieves state-of-the-art results among mobile networks. Despite all these efforts, the poor performance of these models compared to that of state-of-the-art deep CNNs remains an unresolved issue [19].

In this paper, we propose to overcome some drawbacks of the previously discussed approaches. We introduce a novel CNN model based on a hierarchical architecture, called DivNet. The goal is to train the network to extract relevant features of high diversity that best characterise the complexity of any input image. To achieve this, we consider a hierarchical structure that has been shown to perform better than the standard architecture—with simply stacked layers—in terms of complexity and performance [7], [8], [41]. Our DivNet model is built on top of a novel cell module, called Diversity Module (DivMod), designed to extract more discriminative and diverse features. This module takes as an input $c$ feature channels that are first filtered with a standard convolutional filter. Features are subsequently split into $k$ groups, each with the same number of channels. One group remains the identity, i.e., unmodified, while the remaining groups go through the same set of convolution operations. This efficient design can reduce the number of learnable parameters by ensuring that each set of convolutions are applied only to the corresponding input channel group. The resulting features are subsequently concatenated followed by a $1 \times 1$ convolution. The purpose of this second convolution is to recover the channel group dimensions so they match those of the identity group after a shortcut connection. After that, the $k$ groups are concatenated and the number of channels remains the same. A channel shuffle operation [14] is then used to enable information flow among different groups. The batch normalization (BN) [16] and ReLU nonlinearity [17] are used before the split operator to prevent information collapse.

In summary, the contributions of our research are threefold. First, we propose DivMod, a novel cell module designed to extract more discriminative and diverse features. DivMod has an efficient lightweight design with high discriminative capacity and low computational cost. Second, based on DivMod, we propose DivNet, a hierarchical architecture that allows to construct efficient lightweight CNNs. This CNN model is particularly suitable for mobile designs. Third, we thoroughly evaluate DivNet to highlight its advantages, in terms of performance and computational cost, and compare its classification performance against several state-of-the-art lightweight neural networks. As evidenced in Section IV, the lightweight neural network we propose can offer promising results in terms of performance as well as efficiency.

The remainder of this paper is organized as follows. Section II briefly reviews the most successful works on mobile architectures. The proposed DivNet network is described in detail in Section III. Experimental results on several image classification datasets are given in Section IV. Finally, Section V concludes the paper.

II. RELATED WORK

Most successful works on developing efficient lightweight architectures fall into two main categories: one focuses on obtaining small networks from existing complex architectures, while the other focuses on developing novel lightweight architectures from scratch.

A. SIMPLIFYING COMPLEX ARCHITECTURES

Recent works on CNNs have yielded powerful deep and wide architectures compared to the early proposed deep neural networks, which use the highly parameterized fully connected layers. In this context, [6], [9] propose to use the average pooling layers to significantly reduce the number of parameters. Another technique used by the GoogLeNet's inception module [8] consists of adding $1 \times 1$ convolutions for dimensionality reduction. [20] proposes to downsample the image at an early stage to reduce the size of the feature channels. Other approaches consist in taking an existing complex CNN model and compressing it in a lossy fashion. Several compression techniques have been introduced in the literature, including those based on parameter quantization [21], binarization [22]–[24], and pruning [25]. For example, [26] reports highly competitive compression ratios on AlexNet [5] and VGG-Net [7], without significantly hindering performance, by pruning those weights with small magnitudes and then retraining the network.

Other approaches to speeding up CNNs and work with smaller models are factorization [27], [28], and distillation [29]. The latter involves using a large network to teach a small network. Low-bit networks [30], [31] are emerging architectures that include a quantization process to map continuous real values to discrete integers, thus simplifying computation. Flattened networks [32] rely on fully factorized convolutions and exploit the potential of extremely factorized networks [33]. Subsequently, the Xception network [34] has demonstrated how to scale up depthwise separable filters to outperform Inception V3 networks.
B. Kaddar et al.: DivNet: Efficient Convolutional Neural Network via Multilevel Hierarchical Architecture Design

The approaches discussed before focus primarily at optimizing large models to achieve state-of-the-art performance. However, such approaches have shown limits with regard to the difficulty of training and the rate at which performance drops when compression is increased [26]. The point at which accuracy starts to rapidly drop occurs even when the model is still very large [49]. Thus, the drawback when deployed to resource-constrained environments is that they often struggle to scale down sufficiently. To achieve this objective, expert knowledge and trial-and-error are required. Additionally, as the CNN architectures continue to become deeper, the computational costs of the convolutional layers remain high. Thus, models such as those above [8], [21]–[25], still contain millions of parameters and billions of FLOPs even after compression, which require significant computational resources to execute, and exhibit diminishing returns when scaling. Lightweight architectures based on simplifying complex models are, therefore, out of the scope of this paper. Thus, we focus on efficient models developed from the ground up [51].

**B. LIGHTWEIGHT ARCHITECTURES**

Recently, a few works attempting to design small and efficient lightweight architectures for mobile applications have emerged. For instance, [13] presents an efficient model called MobileNet (see Fig. 1). MobileNet uses depthwise separable convolutions [15], [16] to build lightweight neural networks by trading off a reasonable amount of accuracy for a model size reduction. In mathematical notation, let $g(T, \gamma(T), \Theta(T), \ast)$ be a function defining a standard convolutional layer, where $X^{(l)} \in \mathbb{R}^{c_{in} \times d_{h} \times d_{w}}$ is the set of $c_{in}$ input feature channels, each with a spatial size of $D_{h} \times D_{w}$, $\gamma^{(l)} \in \mathbb{R}^{c_{out} \times D_{h} \times D_{w}}$ is the corresponding set of $c_{out}$ output feature channels, each with a spatial size of $D_{h} \times D_{w}$, $\Theta^{(l)} \in \mathbb{R}^{D_{k} \times D_{k} \times D_{h} \times D_{w}}$ is a set of filter weights, and $\ast$ denotes the convolution operation. $T$ is an index $\in \{0, \ldots, l - 1\}$ such that $T = 0$ indicates the input layer in which $X^{(0)}$ is the input image, $l$ indicates the total number of layers, and $D_{l}$, $D_{k}$ and $D_{c}$ are the spatial dimension of each kernel, the total number kernels and the number of feature channels, respectively. The computational cost (i.e., the computational complexity), $C_{r}$, of a standard convolutional layer with stride one can then be defined as:

$$
C_{r} = (D_{l} \times D_{l}) \times c_{in} \times c_{out} \times (D_{h} \times D_{w}).
$$

As one can observe, the cost depends multiplicatively on the number of input channels, $c_{in}$, the number of output channels, $c_{out}$, the kernel size, $D_{l} \times D_{l}$, and the input channel size, $D_{h} \times D_{w}$. The cost $C_{dp}$ of the depthwise separable convolution, which is the combination of a depthwise convolution and a $1 \times 1$ pointwise convolution, is [13]:

$$
C_{dp} = (D_{l} \times D_{l}) \times c_{in} \times (D_{h} \times D_{w})
+ c_{in} \times c_{out} \times (D_{h} \times D_{w}),
$$

where $(D_{l} \times D_{l}) \times c_{in}$ is the depthwise convolutional kernel size.

One drawback of the depthwise separable convolution is that it becomes less efficient in extremely small networks because of the costly dense $1 \times 1$ convolution [34], [35]. Additionally, the expensive pointwise convolutions result in a limited number of feature channels to meet feature-diversity requirements, which may significantly hinder performance. To address this issue, [14] proposes a new architecture based on a low-cost pointwise group convolution and channel shuffle operation (see Fig. 1b).

The group convolution significantly reduces the computational cost by ensuring that each convolution operates only on the corresponding input channel group. The size $S_{g}$ of a standard convolutional filter with $n$ input feature channels and $m$ output feature channels is:

$$
S_{g} = (D_{l} \times D_{l}) \times n \times m.
$$

Now, let us consider the group convolution introduced in [14] with $k$ groups. The size $S_{g}$ of each group filter is:

$$
S_{g} = (D_{l} \times D_{l}) \times \frac{n}{k} \times \frac{m}{k}.
$$

As a result, the cost of each group convolution, $C_{g}$, is:

$$
C_{g} = (D_{l} \times D_{l}) \times \frac{n}{k} \times (D_{h} \times D_{w})
+ \frac{n}{k} \times \frac{m}{k} \times (D_{h} \times D_{w}).
$$

One limitation of the group convolution is that the output feature channels are only derived from a small fraction of the...
input features channels. To address this issue, [14] proposes a channel shuffle operation that allows information flow among channel groups and improve representation.

Another small network is MobileNetV2 [17], which has been shown to improve upon the state-of-the-art by using resource-efficient inverted residuals and linear bottlenecks (see Fig. 2). In turn, SqueezeNet [19] uses a bottleneck approach to design a very small network (see Fig. 2b). The authors propose a Fire module that consists of a low cost 1 × 1 convolutional layer to reduce the size of feature channels and an expanded layer that has a mix of 1 × 1 and 3 × 3 convolutional filters. This allows achieving AlexNet-level accuracy with 50 times fewer parameters. Recently, [18] proposes a series of practical guidelines for efficient CNN design and has been shown to achieve state-of-the-art results among mobile networks.

In contrast to MobileNet, MobileNetV2, and ShuffleNet, which are dependent on depth-wise separable convolutions, Wang et al. [48] propose a variant of DenseNet named PelleNet, which instead uses conventional convolutions. PelleNet consists of a stem block and four stages of feature extraction. The stem block can effectively improve the feature extraction capabilities without adding too much computational cost. The authors use two stacked convolutional layers with a 3 × 3 kernel to get different scales of the receptive field. Another important contribution is that the number of channels in the bottleneck layer varies according to the input shape. Another recently proposed model is IoTNet [49], which improves the trade-off between accuracy and computational cost by factorizing the 3 × 3 standard convolution commonly found in large models into pairs of 1 × 3 and 3 × 1 convolutions, rather than using depth-wise separable convolutions. In a similar way, the authors of EffNet [51] address the computational complexity associated with bottleneck layers by replacing 3 × 3 convolutions with pairs of 1 × 3 and 3 × 1 convolutions performed as a depth-wise separable operation. A weakness of such an approach is that the computational complexity savings of performing a 1 × 3 convolution as a depth-wise operation is less than the cost of a 3 × 3 convolution. On the other hand, LiteNet [50] takes an inception block, which contains 1 × 1, 1 × 2 and 1 × 3 standard convolutions arranged side by side, and makes modifications by replacing half of the 1 × 2 and 1 × 3 standard convolutions with their depth-wise equivalents. Their proposed block, therefore, contains a mix of both standard and depth-wise separable convolutions. Their work also makes use of a SqueezeNet fire block [19] to further reduce the total number of learnable parameters. The drawback of their proposed model is the side-by-side structure employed, which increases the total number of parameters according to the depth.

More recently, smaller and faster neural network architectures have been developed in a fully automated manner [36], [37]. These methods focus on cell level architecture search. Once a cell structure is determined, it is used in all layers across the network. One drawback of these approaches is that only a small dataset is used on the optimization process, which often results in a highly complex model [38].

III. FRAMEWORK
This section first details the different components of the proposed cell module, DivMod, followed by a description of our DivNet architecture.

A. PROPOSED CELL MODULE
State-of-the-art networks, such as SqueezeNet [19], MobileNet [13], MobileNetV2 [17], and ShuffleNet [14], introduce efficient cell modules into the building blocks to provide an excellent trade-off between feature representation capability and computational cost. Here, the challenge is to design a cell module that achieves the best performance under limited computational resources [39]. For this purpose, there exist two main techniques used by many efficient neural network architectures: depth-wise separable convolutions and group-wise convolutions [13], [14], [17], [34]. Depth-wise separable convolutions decompose the standard convolution into two separate operations: depthwise convolution and pointwise convolution to reduce the computational complexity of the 1 × 1 convolutions (see Fig. 3). Group-wise convolution [14] is used to further reduce the model size.

Following [18], we propose a novel cell unit, called DivMod, based on an efficient channel split technique. DivMod aims to increase the feature diversity, which is one of the main factors influencing the discriminative ability of CNNs. DivMod should be seen as complementary to the cell modules used by MobileNetV2 [17], ShuffleNet [18], and ShuffleNetV2 [14], as it allows to further improve performance.
FIGURE 3. An example of a depthwise separable convolutional network.

FIGURE 4. Our DivMod cell unit. BN: Batch Normalization.

FIGURE 5. Group convolutions with shuffle operation.

As illustrated in Fig. 4, DivMod contains two $1 \times 1$ convolutions, one depthwise $3 \times 3$ convolution, and one ReLU activation function. A $1 \times 1$ convolution is first used to reduce the dimensionality of the data before the expensive depthwise $3 \times 3$ convolutions. The ReLU activation function follows the first $1 \times 1$ convolution and is applied before the split operation to avoid information collapse [14]. Experiments illustrate that using ReLU outside the convolution operations is crucial to prevent non-linearities from destroying information [17]. The $c$ feature channels resulting from the $1 \times 1$ convolution operation are split into $k$ groups with the same number of channels each. In our experiments, we fix $k = 4$ as a trade-off between the model size and performance, thus reducing the number of parameters by 4 times compared to the case of not using such a group convolution. One group is used as the identity, and each of the remaining $k - 1$ groups go through a $3 \times 3$ depthwise convolution. Then, the $k - 1$ groups are concatenated and another $1 \times 1$ convolution is applied. Two sets of features are then concatenated into one: the first set is the identity group (see skip connection in Fig. 4), and the second one is that resulting from concatenating the remaining $k - 1$ groups. The two $1 \times 1$ convolutions have the same number of input and output feature channels and hence the number of input and output channels over the whole cell module is the same [14]. The channel shuffle operation is then used to enable information flow among the $k$ groups (see Fig. 5). Specifically, first we divide the channels in each group into several subgroups. Then we feed each group with different subgroups. It is important to note that the shuffle operation does not take any additional parameters and is computationally efficient.

B. PROPOSED ARCHITECTURE

Recent lightweight mobile models [13], [14], [17], [19] focus on cell level architecture search. Once the cell structure is determined, it is used in all layers across the network. These approaches do not permit layer diversity, which has been shown to be critical for achieving both consistent performance and lower latency of the overall network [39]–[41]. As suggested in [42], there is a need to produce feature channels with high diversity across layers to obtain better performance-latency trade-offs. However, designing deeper and more complicated networks does not necessarily lead to more efficient models with respect to size and speed [39]. To overcome this drawback, we propose DivNet, an efficient architecture based on a multi-level hierarchical design. Fig. 6b shows the baseline structure of DivNet for classification. The architecture of DivNet is summarized in TABLE 1. The basic building block is our DivMod cell unit, discussed in the previous section. DivNet contains an initial standard convolutional layer with 32 filters, followed by 4 DivMod cell layers. Each DivMod cell layer consists of a number of DivMod blocks. Each DivMod block comprises a number of DivMod cell units, each with the same architecture as explained before. Unlike [18], the number of DivMod blocks in each layer is fixed to generate networks with a progressive complexity. Earlier layers in CNNs usually process large amounts of data and thus have a much higher impact on inference latency than later layers. On the other hand, mid-level and last-level features have been shown to considerably
FIGURE 6. (a) Sample architecture of a typical CNN and (b) our DivNet model for classification.

TABLE 1. Overall architecture of DivNet model.

| Layer Type          | Input size     | Output size     | Stride |
|---------------------|----------------|-----------------|--------|
| Conv                | 224 x 224 x 3 | 112 x 112 x 32 | 2      |
| Split - S1          | 112 x 112 x 32| 4 x 112 x 112 x 8|        |
| DivMod cell layer 1 | 4 x 112 x 112 x 8| 4 x 56 x 56 x 24| 2      |
| Concatenate & shuffle 1 | 4 x 56 x 56 x 24| 56 x 56 x 96 |        |
| Split - S2          | 56 x 56 x 96 | 4 x 56 x 56 x 24 |        |
| DivMod cell layer 2 | 4 x 56 x 56 x 24| 4 x 28 x 28 x 96| 2      |
| Concatenate & shuffle 2 | 4 x 28 x 28 x 96| 28 x 28 x 28 |        |
| Split - S3          | 28 x 28 x 28 | 4 x 28 x 28 x 96 |        |
| DivMod cell layer 3 | 4 x 28 x 28 x 96| 4 x 14 x 14 x 24| 2      |
| Concatenate & shuffle 3 | 4 x 14 x 14 x 24| 14 x 14 x 1536 |        |
| Split - S4          | 14 x 14 x 1536| 8 x 14 x 14 x 284| 2      |
| DivMod cell layer 4 | 8 x 14 x 14 x 284| 8 x 7 x 7 x 256| 2      |
| Concatenate          | 8 x 7 x 7 x 256| 7 x 7 x 2104 |        |
| Conv                 | 7 x 7 x 2104 | 7 x 7 x 1024 | 1      |
| Global AvgPool       | 7 x 7 x 1024 | 1 x 1 x 1024 |        |
| Conv                 | 1 x 1 x 1024 | 1 x 1 x 100 |        |

IV. EXPERIMENTS

In this section we discuss experiments conducted to evaluate the classification performance of the proposed DivNet model on the CIFAR10, CIFAR100, and ImageNet 2012 datasets [5], [45], often employed in the literature. First, we evaluate the effect of various DivNet architecture design decisions on performance. We analyze the trade-offs between accuracy and computational cost in terms of model size and floating-point operations (FLOPS) by tuning several hierarchical structure designs. Next, we compare the effectiveness of DivNet against the popular network models MobileNet [13], MobileNetV2 [17], SqueezeNet [19], ShuffleNet [14], and ShuffleNetV2 [18].

A. TRAINING SETUP

Our experiments are performed on the CIFAR10, CIFAR100 [45] and ImageNet 2012 datasets [5]. CIFAR10 consists of 60,000 32 x 32 color images divided in 10 classes, with 6,000 images per class. There are 50,000 training images and 10,000 test images. CIFAR100 consists of 60,000 32 x 32 color images divided in 100 classes, with 600 images per class. ImageNet is organized according to the WordNet hierarchy, in which each node of the hierarchy is depicted by several images. We train our models using Tensorflow [46]. We use the standard NADAM optimizer [47] with default parameter values.

B. PERFORMANCE EVALUATION METRICS

To evaluate the performance of DivNet, we employ the Top-1 accuracy, the Top-5 accuracy, the number of FLOPS, and the number of multiply accumulate operations (MAC).
The Top-1 accuracy is based on the number of true positives (TP), false positives (FP), true negatives (TN) and false negatives (FN). The Top-1 accuracy can then be defined as the percentage of correctly classified instances:

$$\text{Top-1 Accuracy} = \frac{\text{Corrected prediction}}{\text{Total prediction}} = \frac{\text{TP}}{\text{TP} + \text{TN}}. \quad (6)$$

The Top-5 accuracy is computed by determining if the correct label is among the top-5 predicted labels.

### C. DivNet Architecture Design

In this section, we design and execute experiments with the goal of providing intuition about the performance of DivNet with respect to the design strategies discussed in Section III. Specifically, our goal is to investigate the impact of design choices on accuracy, model size and computational cost. To this end, we explore the architecture designs related to the high-level connections among DivMod cell units. To build an efficient mobile model, first we evaluate several split operator strategies as shown in Fig. 6b for boosting performance. Second, we evaluate the performance of the model with/without shuffle operations.

#### 1) Split Operator Strategies

TABLE 2 tabulates the performance of three DivNet architectures with various split rates, $S$, i.e., $S_1$, $S_2$, $S_3$, $S_4$. Note that the architecture depicted in Fig. 6b has split rates of $S_1 = S_2 = S_3 = 4$ and $S_4 = 8$. As one can observe, large split rates (e.g., $S = 8$) consistently boost classification scores. Performance improves with split rates progressively increasing from the input to the last layer. Specifically, these results show that using a high split rate for the mid and last layers gives the best performance with low computational cost. For instance, using a split operator $S = 8$ for the mid-layers achieves 60.51% accuracy, higher than using $S = 8$ for early layers ($+4.43\%$) on the CIFAR100 dataset, and 79.85% accuracy on the CIFAR10 dataset, better than using $S = 8$ for early layers ($+9.27\%$). The architecture with a high split rate in the last layer, i.e., $S = 8$, gives the best performance, i.e., 70.17% and 94.35% on the CIFAR100 and CIFAR10 datasets, respectively, and outperforms the first and the second architectures by 23.77% and 14.5% (CIFAR100), and by 14.09% and 9.66% (CIFAR10), respectively. In this case, DivNet only requires 489,897,672 FLOPS, fewer than the first and the second architectures by 193,378,304 FLOPS and 240,694,272 FLOPS, respectively. It is then clear that DivNet benefits from the hierarchical network design.

#### 2) Channel Shuffle Operation

The purpose of the shuffle operation is to encourage the information flow among channels and is important to improve DivNet’s performance of DivNet. TABLE 3 compares the performance of the third DivNet architecture tabulated in TABLE 2 when using/removing the channel shuffle operation. The evaluations are performed under four different setups: Shuffle 1 is applied exclusively, Shuffle 2 is applied exclusively, Shuffle 3 is applied exclusively, and no shuffle operation is applied. As one can observe, using the shuffle operations improves significantly the classification performance on both datasets, i.e., CIFAR10 (94.35 %), CIFAR100 (70.17 %), compared to the case of not using this operation, i.e., CIFAR10 (+2.00%), CIFAR100 (+3.2%). In particular, on the CIFAR10 dataset, DivNet with Shuffle 3 attains 93.65% accuracy, better than that attained with Shuffle 1 (+0.65%) and Shuffle 2 (+0.35%). On the CIFAR100 dataset, DivNet with Shuffle 3 attains 69.07% accuracy, better than that attained with Shuffle 1 (+1.70%) and Shuffle 2 (+1.4%). From these results, it is clear that the proposed DivNet model performs better with the shuffle operation. Additionally, using feature channels with high diversity across mid-level and last-level works especially well for more challenging datasets, i.e., CIFAR100, resulting in important performance improvements.

#### D. Comparison Between DivNet and Large Models

In this section, we compare DivNet against VGG11$_{bn}$ [7], which is a large architecture of high computational cost. In our comparison, we use the third DivNet architecture tabulated in TABLE 2. Results are shown in TABLE 4. As one can observe, DivNet outperforms VGG11$_{bn}$ by a significant margin with a much lower complexity in terms of number of parameters. Convolutional layers in VGG11$_{bn}$ [7] follow state-of-the-art trends in terms of size and complexity (e.g. Total Params $= 28.5$ M), without exploring low complexity alternatives. DivNet, on the other hand, is based on small cell modules with fewer parameters (e.g., Total Params $= 2.8$ M). These results show that DivNet is more efficient than large architectures, like VGG11$_{bn}$, with less computational cost and higher performance, e.g., on the CIFAR100 dataset, there is an improvement of 1.53% in Top-1 accuracy.
TABLE 4. Performance on the CIFAR100 and CIFAR10 datasets.

| Network Model      | Total Parameters (M) | CIFAR100 Top-1 accuracy (%) | CIFAR10 Top-1 accuracy (%) |
|--------------------|----------------------|----------------------------|---------------------------|
| MobileNet [13]     | 3.30                 | 65.98                      | 90.42                     |
| MobileNetV2 [17]   | 2.36                 | 68.08                      | 94.30                     |
| SqueezeNet [19]    | 0.78                 | 69.41                      | 89.39                     |
| ShuffleNet [14]    | 1.00                 | 70.06                      | 91.76                     |
| ShuffleNetV2 [18]  | 1.30                 | 69.51                      | 93.22                     |
| VGG11bN [7]        | 28.50                | 68.64                      | 92.09                     |
| DivNet             | 2.80                 | 70.17                      | 94.35                     |

TABLE 5. Performance on the ImageNet 2012 dataset.

| Network Model      | Total Parameters (M) | FLOPs (M) | CPU Speed (ms) | ImageNet 2012 Top-1 accuracy (%) | Top-5 accuracy (%) |
|--------------------|----------------------|-----------|----------------|----------------------------------|-------------------|
| MobileNet [13]     | 4.20                 | 569       | 113            | 70.60                            | 90.10             |
| MobileNetV2 [17]   | 6.90                 | 585       | 143            | 74.70                            | 92.50             |
| SqueezeNet [19]    | 1.25                 | 235       | 29.0           | 57.50                            | 80.30             |
| ShuffleNet [14]    | 3.74                 | 524       | 75.6           | 67.60                            | 87.09             |
| ShuffleNetV2 [18]  | 2.30                 | 299       | 37.8           | 69.36                            | 88.32             |
| PeleeNet [48]      | 2.80                 | 308       | –              | 72.60                            | 90.60             |
| VGG11bN [7]        | 28.50                | 7640      | –              | 70.40                            | 90.00             |
| DivNet             | 2.80                 | 489       | 70.55          | 70.65                            | 90.23             |

E. COMPARISON BETWEEN DivNet AND OTHER MOBILE NETWORKS

TABLE 4 also tabulates results for other state-of-the-art lightweight neural networks, including MobileNet [13], MobileNetV2 [17], SqueezeNet [19], ShuffleNet [14], and ShuffleNetV2 [18]. From these results, as one can observe, DivNet (with the third architecture in TABLE 2) consistently performs better than the other mobile networks. On the CIFAR100 dataset, DivNet attains 70.17% accuracy, better than MobileNet (+4.19%), MobileNetV2 (+2.09%), SqueezeNet (+0.76%), ShuffleNet (+0.11%), and ShuffleNetV2 (+0.66%). On the CIFAR10 dataset, DivNet attains 94.35% accuracy, better than MobileNet (+3.93%), MobileNetV2 (+0.05%), SqueezeNet (+4.96%), ShuffleNet (+2.59%), and ShuffleNetV2 (+1.13%).

TABLE 5 tabulates the performance of DivNet on the more challenging ImageNet 2012 dataset. In order to handle the complexity of this dataset, we evaluate more complex architectures versions of the state-of-the-art lightweight neural networks tabulated in TABLE 5. For a fair comparison, in addition to the accuracy, we consider two other metrics: FLOPs and the inference time speed (CPU speed). The latter is measured in milliseconds (ms) and depends on the platform characteristics.

As shown in the TABLE 5, our DivNet model achieves 70.65% Top-1 and 90.23% Top-5 accuracy with 70.55ms inference time, 2.80M parameters, and 489M FLOPs. DivNet outperforms the large model VGG11bN, in terms of both Top-1 (+0.25%) and Top-5 (+0.23%) accuracy, with less parameters (~25.7 M). DivNet yields a slightly better accuracy than MobileNet, while requiring 1.4 M less parameters. It outperforms ShuffleNet in terms of both Top-1 (+3.05%) and Top-5 (+3.14%) accuracy, while requiring around 0.94 M less parameters. MobileNetV2 consistently performs better than our model by +4.05% and +2.27% for Top-1 and Top-5 accuracies, respectively; however, such a performance requires an extra 4.1 M parameters. On other hand, DivNet’s FLOP count is only 489M, 80M smaller than MobileNet, 96M smaller than MobileNetV2, and 35M smaller than ShuffleNet. SqueezeNet’s FLOPs are 254M lower, but DivNet’s accuracy is better, i.e., +13.15%(Top-1) and +9.93%(Top-5). DivNet’s inference time speed is 70.55ms, 42.45ms shorter than MobileNet, 72.45ms shorter than MobileNetV2, and 5.05ms shorter than ShuffleNet. PeleeNet outperforms our DivNet model, i.e., +1.95%(Top-1) and +0.37%(Top-5), but with higher FLOPs (+19M). Compared with ShuffleNetV2, DivNet performs better in terms of both Top-1 (+1.29%) and Top-5 (+1.91%) accuracies, requiring 0.5 M less parameters but with higher computational cost, i.e., DivNet’s FLOPs are +190 M greater and inference time speed (+32.75 ms) higher. However, it should be noted that FLOPs is a metric that indirectly approximates the computational complexity and it may not always be equivalent to the direct metric based on CPU speed (as one can confirm by comparing ShuffleNetV2 and DivNet). This discrepancy between the indirect (FLOPs) and direct (speed) metrics can be attributed to the degree of parallelism [18]. A model with high degree of parallelism could be much faster than another one with low degree of parallelism, under the same FLOPs. Therefore, we anticipate that the inference time of our hierarchical structure based-model could be improved if it were implemented on effectively designed parallel algorithms. In particular, the inherent multi-branch nature of DivNet enables efficient inference and supports the parallel execution on limited computing resources. In this sense, we see in TABLE 2 that, not only the number of parameters, but also the architecture is critical to achieve a good performance: the third architecture with a high split rate in the last layer, i.e., $S = 8$, gives the best performance, i.e., 70.17% Top-5 accuracy and 94.35% Top-1 accuracy on the CIFAR100 and CIFAR10 datasets, respectively, and outperforms the second architecture by 14.5% (CIFAR100), and by 9.66% (CIFAR10), with a smaller number of parameters (~59392). DivNet benefits from the hierarchical network design, which is the main contribution.
of our work. On other hand, existing automatic neural architecture search methods for mobile devices are computationally expensive because the design space is combinatorially large. A network with a hierarchical design, in conjunction with an exhaustive search for the most appropriate type and kernel size and number of filters in each layer, is expected to lead to better accuracy and efficiency.

V. CONCLUSION

This paper presented an efficient lightweight CNN based on a hierarchical design, called DivNet. Our goal is to produce feature channels with high diversity across layers based on an efficient channel split operation to obtain better performance-complexity trade-offs. DivNet employs a novel cell unit, called DivMod. The split operation strategy in combination with the DivMod cell units consistently allow extracting relevant features that best characterize the complexity of the input image. Such feature extraction can greatly increase diversity, which is one of the main factors influencing the discriminative ability of neural networks. Experimental results illustrated the potential of DivNet to improve classification performance with low computational cost. On the CIFAR100 dataset, DivNet achieved 70.17% accuracy, whereas only 68.64% and 68.08% accuracies were achieved by VGG11 and MobileNetv2, respectively. Similarly, on the CIFAR10 dataset, DivNet outperformed most of the other evaluated state-of-the-art models and attained 94.35% Top-1 accuracy. On more complex datasets, i.e., ImageNet 2012, DivNet achieved a Top-1 accuracy of 70.65% and Top-5 accuracy of 90.23%, and outperformed other small models such as MobiNet, SqueezeNet and ShuffleNet.

REFERENCES

[1] W. Rawat and Z. Wang, “Deep convolutional neural networks for image classification: A comprehensive review,” Neural Comput., vol. 29, no. 9, pp. 2322–2449, Sep. 2017.
[2] C. Szegedy, A. Toshev, and D. Erhan, “Deep neural networks for object detection,” in Proc. Adv. Neural Inf. Process. Syst., 2013, pp. 2553–2561.
[3] C. Gan, N. Wang, Y. Yang, D.-Y. Yeung, and A. G. Hauptmann, “DivNet: A deep event network for multimedia event detection and evidence recounting,” in Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR), Jun. 2015, pp. 2568–2577.
[4] A. Graves and J. Schmidhuber, “Framework phone classification with bidirectional LSTM and other neural network architectures,” Neural Netw., vol. 18, no. 5, pp. 602–610, 2005.
[5] A. Krizhevsky, I. Sutskever, and G. E. Hinton, “ImageNet classification with deep convolutional neural networks,” in Proc. Adv. Neural Inf. Process. Syst. (NIPS), 2012, pp. 1097–1105.
[6] M. Lin, Q. Chen, and S. Yan, “Network in network,” 2013, arXiv:1312.4400 [Online]. Available: http://arxiv.org/abs/1312.4400
[7] K. Simonyan and A. Zisserman, “Very deep convolutional networks for large-scale image recognition,” 2014, arXiv:1409.1556 [Online]. Available: http://arxiv.org/abs/1409.1556
[8] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, “Going deeper with convolutions,” in Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR), Jun. 2015, pp. 1–9.
[9] K. He, X. Zhang, S. Ren, and J. Sun, “Deep residual learning for image recognition,” in Proc. IEEE Conf. Comput. Vis. Pattern Recognit., Jun. 2016, pp. 770–778.
[10] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, and A. C. Berg, “ImageNet large scale visual recognition challenge,” Int. J. Comput. Vis., vol. 115, no. 3, pp. 211–252, Dec. 2015.
[11] C. Szegedy, S. Ioffe, V. Vanhoucke, and A. A. Alemi, “Inception-v4, inception-resnet and the impact of residual connections on learning,” in Proc. 31st AAAI Conf. Artif. Intell., 2017, pp. 4278–4284.
[12] M. Denil, B. Shakibi, L. Dinh, M. A. Ranzato, and N. De Freitas, “Predicting parameters in deep learning,” in Proc. Adv. Neural Inf. Process. Syst., 2013, pp. 2148–2156.
[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,” 2017, arXiv:1704.04861 [Online]. Available: http://arxiv.org/abs/1704.04861
[14] X. Zhang, X. Zhou, M. Lin, and J. Sun, “ShuffleNet: An extremely efficient convolutional neural network for mobile devices,” in Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit., Jun. 2018, pp. 6848–6856.
[15] L. Sifre and S. Mallat, “Rigid-motion scattering for image classification,” Ph.D. dissertation, Ecole Polytechn., Palaiseau, France, 2014, p. 3, vol. 1.
[16] S. Ioffe and C. Szegedy, “Batch normalization: Accelerating deep network training by reducing internal covariate shift,” 2015, arXiv:1502.03167 [Online]. Available: http://arxiv.org/abs/1502.03167
[17] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen, “MobileNetV2: Inverted residuals and linear bottlenecks,” in Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit., Jun. 2018, pp. 4510–4520.
[18] N. Ma, X. Zhang, H. T. Zheng, and J. Sun, “ShuffleNet V2: Practical guidelines for efficient CNN architecture design,” in Proc. Eur. Conf. Comput. Vis. (ECCV), 2018, pp. 116–131.
[19] F. N. Iandola, S. Han, M. W. Moskewicz, K. Ashraf, W. J. Dally, and K. Keutzer, “SqueezeNet: AlexNet-level accuracy with 50x fewer parameters and <0.5 MB model size,” 2016, arXiv:1602.07360 [Online]. Available: http://arxiv.org/abs/1602.07360
[20] K. He and J. Sun, “Convolutional neural networks at constrained time cost,” in Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR), Jun. 2015, pp. 5353–5360.
[21] J. Wu, C. Leng, Y. Wang, Q. Hu, and J. Cheng, “Quantized convolutional neural networks for mobile devices,” in Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR), Jun. 2016, pp. 4820–4828.
[22] M. Courbariaux, I. Hubara, D. Soudry, R. El-Yaniv, and Y. Bengio, “Binarized neural networks: Training deep neural networks with weights and activations constrained to +1 or −1,” 2016, arXiv:1602.02830 [Online]. Available: http://arxiv.org/abs/1602.02830
[23] I. Hubara, M. Courbariaux, D. Soudry, R. El-Yaniv, and Y. Bengio, “Binarized neural networks,” in Proc. Adv. Neural Inf. Process. Syst., 2016, pp. 4107–4115.
[24] M. Courbariaux, Y. Bengio, and J. P. David, “BinaryConnect: Training deep neural networks with binary weights during propagations,” in Proc. Adv. Neural Inf. Process. Syst., 2015, pp. 3123–3131.
[25] S. Han, J. Pool, J. Tran, and W. Dally, “Learning both weights and connections for efficient neural network,” in Proc. Adv. Neural Inf. Process. Syst., 2015, pp. 1135–1143.
[26] S. Han, H. Mao, and W. J. Dally, “Deep compression: Compressing deep neural networks with pruning, trained quantization and Huffman coding,” 2015, arXiv:1510.00149 [Online]. Available: http://arxiv.org/abs/1510.00149
[27] M. Jaderberg, A. Vedaldi, and A. Zisserman, “Speeding up convolutional neural networks with low rank expansions,” 2014, arXiv:1405.3866 [Online]. Available: http://arxiv.org/abs/1405.3866
[28] V. Lebedev, Y. Ganin, M. Rakhuba, I. Oseledets, and V. Lempitsky, “Speeding-up convolutional neural networks using fine-tuned CP-decomposition,” 2014, arXiv:1412.6553 [Online]. Available: http://arxiv.org/abs/1412.6553
[29] G. Hinton, O. Vinyals, and J. Dean, “Distilling the knowledge in a neural network,” 2015, arXiv:1503.02531 [Online]. Available: http://arxiv.org/abs/1503.02531
[30] M. Courbariaux, Y. Bengio, and J.-P. David, “Training deep neural networks with low precision multiplications,” 2014, arXiv:1412.7024 [Online]. Available: http://arxiv.org/abs/1412.7024
[31] I. Hubara, M. Courbariaux, D. Soudry, R. El-Yaniv, and Y. Bengio, “Quantized neural networks: Training neural networks with low precision weights and activations,” J. Mach. Learn. Res., vol. 18, no. 1, pp. 6869–6898, 2017.
[32] J. Jin, A. Gandar, and E. Culurciello, “Flattened convolutional neural networks for feedforward acceleration,” 2014, arXiv:1412.5474 [Online]. Available: http://arxiv.org/abs/1412.5474
B. Kaddar et al.: DivNet: Efficient Convolutional Neural Network via Multilevel Hierarchical Architecture Design

HADRIA FIZAZI received the Ph.D. degree in industrial computer science and automation engineering from the University of Lille 1, France, in 1987. She is currently a Professor with the Faculty of Computer Science, University of Science and Technology of Oran Mohamed Boudiaf. Her research interests include cover image processing and the application of pattern recognition to remotely sensed data.

MIGUEL HERNÁNDEZ-CABRERO received the dual B.Sc. degree in computer science and mathematics from the Universidad Autónoma de Madrid, Madrid, Spain, in 2010, and the M.Sc. and Ph.D. degrees in computer science from the Universitat Autònoma de Barcelona, Spain, in 2011 and 2015, respectively. He has held post-doctoral research positions at the University of Warwick, Coventry, U.K., and the University of Arizona, Tucson, AZ, USA. Since April 2019, he has been with the Universitat Autònoma de Barcelona. His research interests include data compression, especially medical and remote sensing image coding, and signal processing. He has coauthored multiple articles in these areas. He has also served as a reviewer for several journals and conferences in the fields.

VICTOR SÁNCHEZ (Member, IEEE) received the M.Sc. degree from the University of Alberta, Canada, in 2003, and the Ph.D. degree from The University of British Columbia, Canada, in 2010. From 2011 to 2012, he was with the Video and Image Processing Laboratory, University of California at Berkeley, as a Postdoctoral Researcher. In 2012, he was a Visiting Lecturer with the Group on Interactive Coding of Images, Universitat Autònoma de Barcelona. From 2018 to 2019, he was a Visiting Scholar with the School of Electrical and Information Engineering, The University of Sydney, Australia. He is currently an Associate Professor with the Department of Computer Science, University of Warwick, U.K. His research has been funded by the Consejo Nacional de Ciencia y Tecnologia, Mexico, the Natural Sciences and Engineering Research Council, Canada, the Canadian Institutes of Health Research, the FP7 and the H2020 Programs of the European Union, the Engineering and Physical Sciences Research Council, U.K., and the Defence and Security Accelerator, U.K. He has authored several technical articles, book chapters, and a book in these areas. His main research interests include signal and information processing and machine learning with applications to multimedia analysis, computer vision, image and video coding, security, and communications.

JOAN SERRA-SAGRISTÀ (Senior Member, IEEE) received the Ph.D. degree in computer science from the Universitat Autònoma de Barcelona (UAB), Barcelona, Spain, in 1999. From 1997 to 1998, he was with the University of Bonn, Bonn, Germany, funded by DAAD. He is currently a Full Professor with the Department of Information and Communications Engineering, UAB. He has coauthored more than 100 articles. His research interests include data compression, especially image coding for remote sensing applications. He was a recipient of the Spanish Intensification Young Investigator Award, in 2006. He serves as the Committee Chair for the Data Compression Conference. He also serves as an Associate Editor for IEEE TRANSACTIONS ON GEOSCIENCE AND REMOTE SENSING and IEEE TRANSACTIONS ON IMAGE PROCESSING.