MDGNet: a light-weight, hardware-compliant Convolutional Neural Network for efficient image inference tasks

Van Thien NGUYEN William GUICQUERO

Univ. Grenoble Alpes, CEA, Leti, F-38000 Grenoble, France

Résumé – La conception des réseaux de neurones dédiée à des architectures micro-électroniques frugales est désormais primordiale pour limiter le coût des algorithmes d'inférence implémentés au sein de systèmes embarqués. Dans cet article, nous proposons un modèle de réseau compact, `MDGNet`, dédié à la classification d'images. Pour obtenir ce modèle compressé, `MDGNet` s'appuie sur une combinaison d'éléments de factorisation tels que les convolutions séparables et convolutions par groupe. Par ailleurs, ce modèle tire profit d'interconnexions réalisées à l'aide d'un Multiplexeur contrôlé dynamiquement et permettant une forte compatibilité avec une approche de co-conception matérielle impliquant une forte quantification, comme reporté dans cet article. Les résultats expérimentaux sur deux bases de données (STL-10, CelebA) démontrent une excellente performance de cette approche, pour un coût en mémoire et complexité de calculs réduits.

Abstract – Designing hardware-compliant deep neural networks is ubiquitous to favor low-power but accurate embedded inference. In this paper, we introduce a compact convolutional network architecture, namely `MDGNet`, dedicated to classification tasks. `MDGNet` relies on depth-wise convolutions combined with a cascade of group convolutions to promote light-weight feature processing blocks. On the other hand, a Mux-skip connection (presented in a previous work) is used to fuse these components together in a hardware-compliant manner, compatible with a quantization of the model. Experimental results on two different datasets (STL-10, CelebA) demonstrate that our model allows higher performance at lower model size and computational complexity compared to prior works.

1 Introduction

Convolutional Neural Networks (CNNs) have recently become a standard choice in many computer vision tasks, namely for their shift invariant property. Their great performance strongly relies on increasingly complex model architectures containing a huge amount of parameters and operations that quickly bypasses the limited memory, computation capabilities of devices "at the edge". Therefore, designing Hardware-compatible tiny models –in terms of memory and computational costs– is crucial to enable accurate embedded inference tasks. Thus, research along the line of designing compact networks gains considerable attention. Recent research efforts involves techniques such as efficient architecture design [4,9,16], pruning [8] and weights and activations quantization [7].

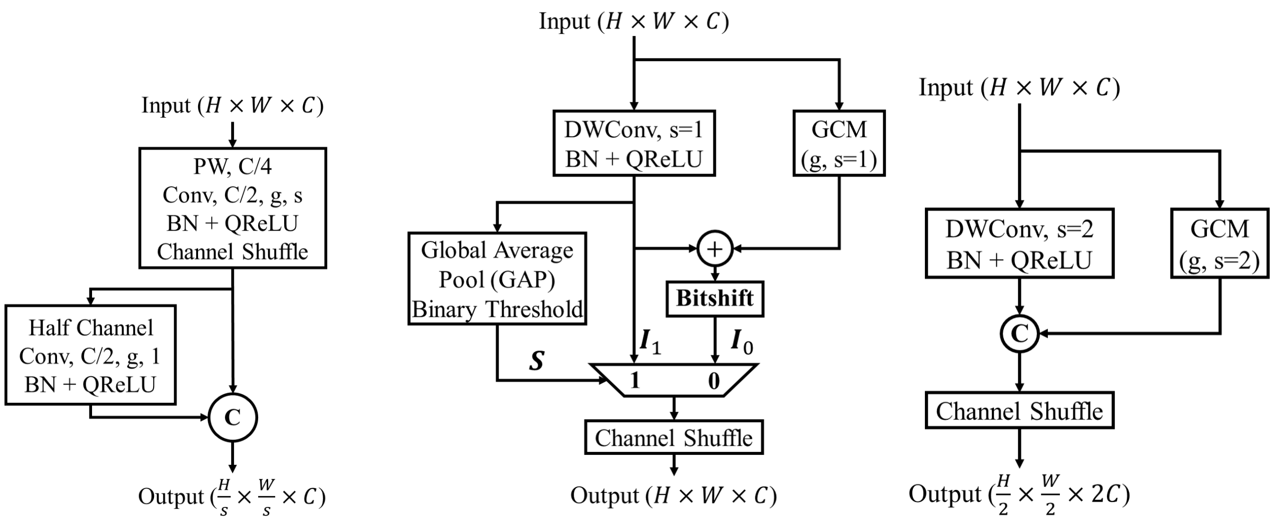
In that context, this paper introduces a novel neural network architecture that is specifically tailored for mobile and resource-constrained systems. This work relies on previously published MOGNET [11], which is composed of a streamlined convolutional factorization and a Multiplexer mechanism to perform the skip connection in a hardware-compliant manner. Based on this architecture, CNN layer factorization is futher improved by a light-weight multi-branch structure consisting of depth-wise, point-wise and group convolutions. Through the Multiplexer (MUX) mechanism, these operations are combined together in an input-dependent manner to provide output channels capturing different levels of channel-wise and spatial information. Additionally, model quantization is applied thanks to a dedicated quantization-aware training to reduce the hardware-related costs. Building upon these components, our tiny model `MDGNet` exhibits an accuracy of 77.18% on the STL-10 dataset, with only 0.07MB parameters.

2 Related works

Efficient architecture design is the straightforward approach for matching CNNs with hardware constraints, simplifying the hardware-expensive standard convolution layers. To this end, simpler convolution layers (*e.g.*, depth-wise, point-wise, group convolution) as well as the residual connections [5] and many other light-weight types of layers have been introduced into the available design space. These architectures can be designed specifically by human intuition and experience (*e.g.*, MobileNets [12]), or automatically under the framework of Neural Architecture Search (*e.g.*, EfficientNet [14]). The proposed `MDGNet` extends MOGNET [11] by leveraging a multi-branch skip-connection strategy. In particular, `MDGNet` integrates a cascade of two group convolutions, enabling to learn representations at different spatial and channel scales.

Model pruning aims at removing redundant operations, neurons and weights to improve the hardware-algorithm compromise of CNNs. The pruning mask can be determined by different criteria with traditionally fixed pattern [6, 8], or in an input-driven manner, where the pruning mask is adaptively changed according to the input data [1]. In our work, the MUX mechanism can be considered as a dynamic pruning strategy, where each output channel of the light-weight depth-wise convolutions is used to determine whether to perform the operations in the main branch of the group convolutions.

Network quantization [7] aims at reducing the precision of weights and activations in order to enable low-bitwidth arithmetics while reducing the overall memory requirements. It consists in applying a mapping of full-precision values onto a quantized representation. Note that, this function is preferably adjusted during training [17], [3] to cap the accuracy lowering.



(a) Group Convolution Module (GCM) (b) Normal block (c) Reduced block

Figure 1 – Details on the three main building blocks of `MDGNet`, with its associated internal hyper-parameters.

3 `MDGNet`

This section elaborates the architecture of `MDGNet` in detail (cf. Table 1). We first describe the Group Convolution Module (GCM) that enables to learn features from both 3×3 and 5×5 receptive fields at the same time. We then describe the building blocks of `MDGNet`, which is built upon a light-weight branch with depth-wise convolution and the GCM.

Table 1 – Top-level architecture description of `MDGNet`

Operator	Input size	Stride	Group	Repeat
Conv2d	$H \times W \times 3$	1	1	1
GCM	$H \times W \times f$	2	g	1
Normal block	$\frac{H}{2} \times \frac{W}{2} \times f$	1	g	2
Reduced block	$\frac{H}{2} \times \frac{W}{2} \times f$	2	g	1
Normal block	$\frac{H}{4} \times \frac{W}{4} \times 2f$	1	g	2
Reduced block	$\frac{H}{4} \times \frac{W}{4} \times 2f$	2	g	1
Normal block	$\frac{H}{8} \times \frac{W}{8} \times 4f$	1	$2g$	2
Reduced block	$\frac{H}{8} \times \frac{W}{8} \times 4f$	2	$2g$	1
Normal block	$\frac{H}{16} \times \frac{W}{16} \times 8f$	1	$4g$	2
Reduced block	$\frac{H}{16} \times \frac{W}{16} \times 8f$	2	$4g$	1
Pointwise Conv	$\frac{H}{32} \times \frac{W}{32} \times 16f$	1	1	1
DepthWise Conv	$\frac{H}{32} \times \frac{W}{32} \times C$	1	1	1

3.1 Group Convolution module (GCM)

Figure 1a depicts the structure of the GCM module, which receives input of size $H \times W \times C$ and computes output of size $\frac{H}{s} \times \frac{W}{s} \times C$. In details, the GCM first projects the high-dimensional input feature maps into a low-dimensional space of only $C/4$ channels using a point-wise convolution, then learns the representation at 3×3 receptive field via the first group convolution of $C/2$ output channels, g groups and stride of s. After that, we apply channel shuffle to force information sharing across groups, before taking only the first $C/4$ channels to pass through the second group convolution of also $C/2$ output channels, g groups and a stride of 1. Each group convolution is then followed by a Batch Normalization (BN) and a QReLU activation (as in [11]). Finally, we concatenate the output of two group convolutions, to obtain the CGM output.

3.2 `MDGNet` building blocks

Similar to previous works in designing efficient model architectures, we also propose two basic building blocks: *normal block* (Figure 1b) with output having the same size as input, and *reduced block* (Figure 1c) for spatial downsampling (*i.e.*, $s = 2$). Both of these blocks include 2 branches based on a depth-wise convolution layer and a GCM.

The core element of the normal block is a 2-input MUX gate, where the input $\mathbf{I}_1$ is the output of the depth-wise convolution branch, and the input $\mathbf{I}_0$ is the result of the addition between two branches followed by a Bitshift operation (division by 2). This MUX gate receives control signal $\mathbf{S}$ as a parameter-free channel attention mechanism driven by a Global Average Pooling (GAP) followed by a binary threshold $T(x) = \mathbb{1}_{\{x>0.5m\}}$. Here m is set to the maximum of the GAP's outputs in the full-precision representation with ReLU activations, and to 1 in the quantized model which is the maximum possible value of the quantized QReLU version. This way, the output of the light-weight depth-wise convolution controls the operation of the Multiplexer module in a channel-wise manner. Concretely, the MUX gate takes $\mathbf{I}_0$ for output feature maps if the corresponding channels of the depth-wise convolution are dominated by small (*i.e.*, zero) values. Otherwise, it provides the straightforward output $\mathbf{I}_1$ of the depth-wise convolution. In this case, the computation in the GCM branch can be advantageously deactivated in a dynamic pruning manner, depending on the position of the channel.

In the case of the reduced block, we set the stride $s = 2$ for both the GCM and the depth-wise convolution. As conventional model architectures usually double the number of channels when performing spatial downsampling, we concatenate the output of the GCM and the depth-wise convolution. Finally, to further favor information sharing across the channel dimension, a Channel Shuffle is added at the end of both normal and reduced blocks.

3.3 Network architecture

The overall architecture of `MDGNet` is depicted in Table 1. We first project the feature maps from the RGB input space into f-dimensional space using a standard convolution followed by a GCM with stride $s = 2$. Next, at each spatial level, the `MDGNet` repeats the normal blocks 2 times before downsampling with a reduced block. There are four spatial levels in total, hence reducing the spatial dimension $32\times$. We gradually double the group parameter g for the last two levels, as these stages involve a larger number of parameters and operations. Finally, we make use of a classifier including a point-wise convolution (with $\mathcal{C}$ output feature maps equal to the number of classes) followed by a full-scale depthwise convolution with a kernel size equal to the input tensor spatial dimension. This is introduced in order to extract a single value for each feature map, while preserving spatial information in a channel-wise manner. Systematically, `MDGNet` is configured through two parameters: a number of feature maps f, and the number of groups g, hence we denote each setting as `MDGNet` (f, g).

4 Experiments

We evaluate and compare the performance of `MDGNet` with state-of-the-art methods on two different benchmarks: (1) image classification on STL-10 dataset [2] of 96×96 RGB images, and (2) facial attributes prediction on CelebA dataset [10] with 218×178 aligned RGB images. For the quantization-aware training strategy, we adopt the same 3-stage training strategy as in [11]: first train the full-precision `MDGNet` from scratch, then fine-tune the weight-ternarized model using BTQ method, and finally fine-tune the fully-quantized model with QReLU activation (4-bit for STL-10 and 3-bit for CelebA). We measure the model's hardware efficiency in terms of the weight-related memory (*i.e.*, model size) and the computational complexity using an estimation of the number of Binary-equivalent Operations (BOPs [15]). For the sake of scientific rigor, the BatchNormalization layers (still present in the W-ter configurations) are taken into account for the BOPs estimation considering 32b gain/offset parameters with 16b inputs. In addition, convolution factorizations (*e.g.*, sequential stacking of PW and Conv in GCM) are also considered with intermediate 16b integer activations.

4.1 Image classification on STL-10

Setting: The target input size is 96×96 with $\mathcal{C} = 10$ classes for the network's output. We apply a simple data augmentation scheme for training: random crop from all-sided 12-pixel padded images combined with random horizontal flip. The model is trained with a batch size of 50 during 300 epochs per stage, with a learning rate initialized at 10^{-3} and exponentially decayed after 150 epochs with a rate of 0.95.

Results: Table 2 shows the comparison between `MDGNet` and two state-of-the-art efficient models: MobileNetV2 [12] and EfficientNet [14], both trained under the same condition with the full-precision `MDGNet` that we discussed above. Generally, the `MDGNet` delivers much better performance although requiring smaller model size compared to existing efficient models. For instance, when compared to the closest peer EfficientNet-B0, our full-precision `MDGNet` (64, 2) achieves 0.37% higher accuracy at only 18% model size but nearly 175% BOPs. This efficiency is even strongly boosted when ternarizing the weights (denoted as **W-ter**), showing that the ternarization helps improving model's generability in datasets like STL-10. Indeed, when ternarizing the weights, we significantly increase the accuracy in both configurations (*i.e.*, $> +3\%$). However, a strong degradation of more than 3% when quantizing the activation to 4-bit shows that high-precision activations play a key role in guaranteeing the representation power of efficient architecture like `MDGNet`.

Table 2 – Comparison with the state-of-the-art efficient networks on STL-10 dataset.

Model	**Params** (MB)	**BOPs** (10^9)	**Accuracy** (%)
MobileNetV2 $0.5\times$	2.8	18.12	72.21
MobileNetV2 $0.75\times$	5.48	39.22	73.98
EfficientNet-B0	16.08	73.01	74.83
`MDGNet` (32,1) (Ours)			
Full-precision	1.15	54.7	71.61
W-ter/A-32b	0.07	4.23	77.18
W-ter/A-4b	0.07	1.43	72.22
`MDGNet` (64,2) (Ours)			
Full-precision	2.88	128.73	75.20
W-ter/A-32b	0.18	9.66	79.28
W-ter/A-4b	0.18	3.00	75.04
`MDGNet` (64,1) (Ours)			
Full-precision	4.45	206.87	76.38
W-ter/A-32b	0.28	14.55	79.62
W-ter/A-4b	0.28	4.53	76.14

4.2 Facial attributes prediction on CelebA

Setting: The target input size is 196×160 and the output has $\mathcal{C} = 40$ attributes. To match the target resolution, we randomly crop training images during the learning process while centrally crop the validation and test images at inference time. The model is trained with a batch size of 128 during 60 epochs per stage, with a learning rate initialized at 2×10^{-3}, 10^{-3} and 10^{-4} for each stage respectively, and exponentially decayed after 30 epochs with a rate of 0.9.

Results: We compare our model with the existing work Slim-CNN [13], and two models MobileNetV2 $0.5\times$, MobileNetV2 $0.75\times$ which are also trained under the same condition with the full-precision `MDGNet`. The average accuracy versus model efficiency trade-offs are reported in Table 3. We can see clearly that the full-precision `MDGNet` (32,1) provides higher average accuracy while requiring smaller model size (*i.e.*, $\approx 50\%$) compared to Slim-CNN and MobileNetV2 $0.5\times$ networks. However, our model requires a higher budget of computation. Besides, when ternarizing the weights, we observe a degradation of 1.56% to obtain a very small-sized model of under 0.08MB while decreasing the BOPs by nearly $13\times$. Finally, when using 3-b QReLU activations, we obtain

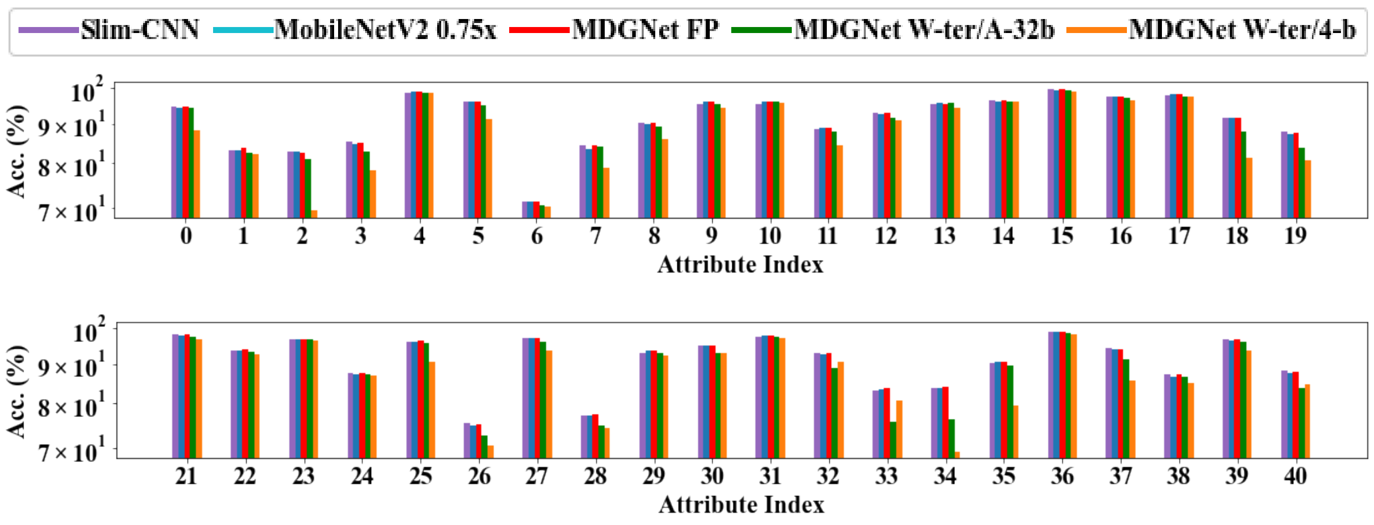


Figure 2 – The accuracy (%) of 40 binary attributes prediction on CelebA dataset by different efficient networks.

87.66% average accuracy with a loss of 2.11% while reducing furthermore the BOPs by 3×. These results suggest that MDGNet is more relevant to target memory-constrained systems. Figure 2 depicts the prediction accuracy of different models over 40 attributes of the CelebA dataset.

Table 3 – Average accuracy of efficient networks on CelebA attributes prediction benchmark.

Model	**Params** (MB)	**BOPs** (10^9)	**Average. Acc.** (%)
Slim-CNN	2.28	N/A	91.24
MobileNetV2 0.5×	2.82	66.70	91.09
MobileNetV2 0.75×	5.37	141.31	91.15
MDGNet (32,1) (Ours)			
Full-precision	1.22	186.62	91.33
W-ternary/A-32b	0.077	14.42	89.77
W-ternary/A-3b	0.077	4.85	87.66

5 Conclusion and Perspectives

This work presents a novel efficient CNN architecture called MDGNet, which tends to better encode the spatial information in image by favoring learning representations from two receptive fields (3 × 3 and 5 × 5) by macro block, with different levels of channel-wise complexity. This is made possible thanks to 2-branch building blocks, involving depth-wise and group convolutions combined via a MUX gate. Moreover, for the sake of hardware-friendliness, MDGNet also enables a dynamic pruning mechanism at inference time, while being compatible to quantization techniques. Note that this paper does not report any estimation of the concrete dynamic power consumption reduction that can be efficiently obtained using such model. We yet empirically shows that our network can be used for various tasks, demonstrating better performances while requiring a notably lower model size compared to existing efficient networks. Future works may focus on increasing the number of cascaded group convolution in the GCM, or channel-grouping the MUX gate to further exploit the benefits of dynamic pruning (via on-line selective hardware execution).

References

[1] Zhourong Chen and Yang et al. Li. You look twice: Gaternet for dynamic filter selection in cnns. In *CVPR*, 2019.

[2] Adam Coates and A. Ng et al. An analysis of single-layer networks in unsupervised feature learning. In *AISTATS*, 2011.

[3] Steven K. Esser and Jeffrey L. McKinstry et al. Learned step size quantization. In *ICLR*, 2020.

[4] Andrew G. Howard et al. Mobilenets: Efficient convolutional neural networks for mobile vision applications. *CoRR*, abs/1704.04861, 2017.

[5] Kaiming He and Xiangyu et al. Zhang. Deep residual learning for image recognition. In *CVPR*, 2016.

[6] Yang He and Guoliang Kang et al. Soft filter pruning for accelerating deep convolutional neural networks. In *IJCAI*, 2018.

[7] Itay Hubara and Matthieu Courbariaux et al. Quantized neural networks: Training neural networks with low precision weights and activations. *ArXiv*, abs/1609.07061, 2017.

[8] Hao Li and Asim Kadav et al. Pruning filters for efficient convnets. In *ICLR*, 2017.

[9] Yunsheng et al. Li. Micronet: Improving image recognition with extremely low flops. In *ICCV*, 2021.

[10] Ziwei Liu and Ping et al. Luo. Deep learning face attributes in the wild. In *ICCV*, December 2015.

[11] Van Thien Nguyen, William Guicquero, and Gilles Sicard. MOGNET: A Mux-residual quantized Network leveraging Online-Generated weights. In *AICAS*, 2022.

[12] Mark Sandler and Andrew et al. Howard. Mobilenetv2: Inverted residuals and linear bottlenecks. In *CVPR*, 2018.

[13] Ankit Kumar Sharma and Hassan Foroosh. Slim-cnn: A lightweight cnn for face attribute prediction. In *2020 15th IEEE International Conference on Automatic Face and Gesture Recognition (FG 2020)*, 2020.

[14] Mingxing Tan and Quoc Le. EfficientNet: Rethinking model scaling for convolutional neural networks. In *ICML*, 2019.

[15] Ying Wang and Yadong Lu et al. Differentiable joint pruning and quantization for hardware efficiency. In *ECCV*, 2020.

[16] Xiangyu et al. Zhang. Shufflenet: An extremely efficient convolutional neural network for mobile devices. In *CVPR*, 2018.

[17] Xiandong Zhao and Ying Wang et al. Linear symmetric quantization of neural networks for low-precision integer hardware. In *ICLR*, 2020.