Supplementary Materials for: CondenseNet V2 Sparse Feature Reactivation for Deep Networks

1. Network Structure

1.1. Implementation details of SFR-ShuffleNetV2 and SFR-ShuffleNetV2+

In this section, we provide more details for building SFR-ShuffleNetV2 and SFR-ShuffleNetV2+ based on ShuffleNetV2 [11] and ShuffleNetV2+[1], respectively. The general network architecture of SFR-ShuffleNetV2 and SFR-ShuffleNetV2+ are shown in Table 1 and Table 3. The ShuffleNet Unit with stride 2 in Table 1 is implemented following [11]. The details of SFR-ShuffleNet Unit is illustrated in Figure 5 of main paper. For SFR-ShuffleNetV2+, we follow the same principle to integrate the SFR modules into different ShuffleNet Units. Moreover, different from CondenseNetV2, the SFR procedure is conducted by 3×3 convolutions in SFR-ShuffleNet. There are four type of ShuffleNet Units in ShuffleNetV2+ (refer to the original implementation of ShuffleNetV2+ for more details). Note that we do not conduct the feature reactivation on the unit with stride equal to 2. Additionally, the network configurations of SFR-ShuffleNetV2 and SFR-ShuffleNetV2+ are provided in Table 2 and 4.

Table 1. Network architecture of SFR-ShuffleNetV2. The number of output channels and the sparse factor of i-th stage are c_i and S_i, respectively. The number of units is denoted by n.

Output size	c	Operator	n	S
224×224	3	Input Image	-	-
112×112	24	Conv2d 3×3 (stride 2)	-	-
56×56	24	MaxPool 3×3 (stride 2)	-	-
28×28	c_1	ShuffleNet Unit (stride 2)	1	-
		SFR-ShuffleNet Unit	3	S_1
14×14	c_2	ShuffleNet Unit (stride 2)	1	-
		SFR-ShuffleNet Unit	7	S_2
7×7	c_3	ShuffleNet Unit (stride 2)	1	-
		SFR-ShuffleNet Unit	3	S_3
7×7	1024	Conv2d 1×1	-	-
1×1	-	AvgPool 7×7	-	-
1×1	1000	FC	-	-

[1] https://github.com/megvii-model/ShuffleNet-Series/tree/master/ShuffleNetV2%2B

Table 2. Network configurations for SFR-ShuffleNetV2 models.

Network	$\{c_i\}$	$\{S_i\}$
SFR-ShuffleNetV2 0.5x	48-96-192	24-48-96
SFR-ShuffleNetV2 1.0x	116-232-464	58-116-232
SFR-ShuffleNetV2 1.5x	176-352-704	88-176-352

Table 4. Network configurations for SFR-ShuffleNetV2+ models. S. and M. represent Small and Medium, respectively.

Network	$\{c_i\}$	$\{S_i\}$
SFR-ShuffleNetV2+ S.	36-104-208-416	18-52-104-213
SFR-ShuffleNetV2+ M.	48-128-256-512	24-64-128-256

Table 5. Network architecture of CondenseNet. The number of layers and the growth rate of i-th dense block are d_i and k_i, respectively.

Input	Operator	d	k
224×224	Conv2d 3×3 (stride 2)	-	-
112×112	DenseLayer	d_1	k_1
112×112	AvgPool 2×2 (stide 2)	-	-
56×56	DenseLayer	d_2	k_2
56×56	AvgPool 2×2 (stride 2)	-	-
28×28	DenseLayer	d_3	k_3
28×28	AvgPool 2×2 (stride 2)	-	-
14×14	DenseLayer	d_4	k_4
14×14	AvgPool 2×2 (stride 2)	-	-
7×7	DenseLayer	d_5	k_5
1×1	AvgPool 7×7	-	-
1×1	FC	-	-

1.2. Implementation details of CondenseNet

In CondenseNet [8], only the network architecture of CondenseNet-C with 300M FLOPs is provided. In order to conduct a more comprehensive comparison, we further design CondenseNet-A/B which are under another two computation levels(50M and 150M). The general network architecture and configurations of CondenseNet are provided in Table 5 and Table 6, respectively. Here, the DenseLayers in Table 5 are implemented with learned group convolutions.

Table 3. Network architecture of SFR-ShuffleNetV2+. The number of output channels and the sparse factor of i-th stage are c_i and S_i, respectively. The number of units is denoted by$n$. SE and HS denote whether using Squeeze-Excitation [7] and Hard-Swish module in this unit.

Output size	c	Operator	n	S	SE	HS
112×112	16	Conv2d 3×3 (stride 2)	1	-	-	1
56×56	c_1	ShuffleNet 3×3 Unit (stride 2)	1	-	-	-
		SFR-ShuffleNet 3×3 Unit	1	S_1	-	-
		SFR-ShuffleNet Xception Unit	1	S_1	-	-
		SFR-ShuffleNet 5×5 Unit	1	S_1	-	-
28×28	c_2	ShuffleNet 5×5 Unit (stride 2)	1	-	-	1
		SFR-ShuffleNet 5×5 Unit	1	S_2	-	1
		SFR-ShuffleNet 3×3 Unit	2	S_2	-	1
14×14	c_3	ShuffleNet 7×7 Unit (stride 2)	1	-	1	1
		SFR-ShuffleNet 3×3 Unit	1	S_3	1	1
		SFR-ShuffleNet 7×7 Unit	1	S_3	1	1
		SFR-ShuffleNet 5×5 Unit	2	S_3	1	1
		SFR-ShuffleNet 3×3 Unit	1	S_3	1	1
		SFR-ShuffleNet 7×7 Unit	1	S_3	1	1
		SFR-ShuffleNet 3×3 Unit	1	S_3	1	1
7×7	c_4	ShuffleNet 7×7 Unit (stride 2)	1	S_4	1	1
		SFR-ShuffleNet 5×5 Unit	1	S_4	1	1
		SFR-ShuffleNet Xception Unit	1	S_4	1	1
		SFR-ShuffleNet 7×7 Unit	1	S_4	1	1
7×7	1280	Conv2d 1×1	-	-	-	1
1×1	-	AvgPool 7×7	-	-	-	-
1×1	1280	SE Module	-	-	1	-
1×1	1280	FC	-	-	-	1
1×1	1000	FC	-	-	-	-

Table 6. Detailed network configurations for CondenseNet models. C denotes the condense factor for learned group convolution.

Network	$\{d_i\}$	$\{k_i\}$	C
CondenseNet-A	2-4-6-8-4	8-8-16-32-64	8
CondenseNet-B	2-4-6-8-6	6-12-24-48-96	6
CondenseNet-C	4-6-8-10-8	8-16-32-64-128	8

posed CondenseNetV2 are superior to many other handcraft-designed efficient deep CNNs significantly. We also observe that the proposed networks outperform the CondenseNets by a large margin, which demonstrates that the effectiveness of our SFR module.

2. A comperhensive study of efficient deep learning models on ImageNet

In this section, we provide more comprehensive comparisons between the proposed network architectures and other state-of-the-art efficient deep learning models. These models are grouped into three levels of computational costs, including 50M, 150M and 300M FLOPs. Our comparisons include the most efficient network architectures: (1) Handcrafted Light-weighted CNN architectures, such as CondenseNet [8], MobileNetV1 [6], MobileNetV2 [16], ShuffleNetV1 [19], ShuffleNetV2 [11], IGCV3 [17], Xception [3] and ESPNetV2 [12], are shown in Table 7. 2) NAS based methods, such as NASNet [21], PNASNet [9], MnasNet [18], ProxylessNas [1], AmoebaNet [15], GhostNet [4], MobileNetV3 [5] and RegXNet [14], are shown in Table 8.

From the results in Table 7, we conclude that the proposed CondenseNetV2 are superior to many other handcraft-designed efficient deep CNNs significantly. We also observe that the proposed networks outperform the CondenseNets by a large margin, which demonstrates that the effectiveness of our SFR module.

As we can see, in Table 8, our CondenseNetV2-C surpass most of the efficient models based on NAS under the computational budget of ∼300M. Note that our CondenseNetV2-C's FLOPs is only half of NASNet-A, AmoebaNet-C and PNASNet-5, however, CondenseNetV2-C still outperform these NAS based models. Although the EfficientNetB0 achieves the best performance when the computational budget is ∼300M, its FLOPs is much larger than CondenseNetV2-C (390M vs 309M). The MobileNetV3 outperforms our models with ∼50M and ∼150M FLOPs which can be due to the effectiveness of NAS. Since our implemented SFR-ShuffleNetV2+ Small can surpass the MobileNetV3 Large 0.75 × by 1.2 percent in terms of Top-1 Error, we believe that CondenseNetV2's performance can be further boosted by NAS algorithms. Therefore, the future work will mainly focus on applying NAS methods on the proposed CondenseNetV2.

Table 7. Comparison of Top-1 and Top-5 classification error rate (%) with state-of-the-art handcraft-designed efficient models on the ILSVRC validation set.

Model	FLOPs	Params	Top-1 err.	Top-5 err.
ShuffleNetV1 0.5×(g=3) [19]	38M	1.0M	41.2	19.0
MobileNetV1-0.25 [6]	41M	–	49.4	–
ShuffleNetV2 0.5× [11]	41M	1.4M	38.9	17.4
MobileNetV2-0.40 [16]	43M	–	43.4	–
CondenseNet-A [8]	56M	0.9M	43.5	20.2
SFR-ShuffleNetV2 0.5×	43M	1.4M	38.3	17.0
CondenseNetV2-A	46M	2.0M	**35.6**	**15.8**
MobileNeXt-0.50 [20]	110M	2.1M	32.3	–
ShuffleNetV1 1.0×(g=3) [19]	138M	1.9M	32.2	12.3
MobileNetV1-0.50 [6]	149M	–	36.3	–
ShuffleNetV2 1.0× [11]	146M	2.3M	30.6	11.1
MobileNetV2-0.75 [16]	145M	–	32.1	–
IGCV3-D-0.70 [17]	210M	2.8M	31.5	–
CondenseNet-B [8]	132M	2.1M	33.9	13.1
SFR-ShuffleNetV2 1.0×	150M	2.3M	29.9	10.9
CondenseNetV2-B	146M	3.6M	**28.1**	**9.7**
Xception 1.5× [3]	305M	–	29.4	–
ShuffleNetV1 1.5× [19]	292M	–	28.5	–
MobileNetV1-0.75 [6]	325M	–	31.6	–
ShuffleNetV2 1.5× [11]	299M	–	27.4	–
MobileNetV2-1.00 [16]	300M	3.4M	28.0	–
MobileNeXt-1.00 [20]	300M	3.4M	26.0	–
IGCV3-D-1.00 [17]	318M	3.5M	27.8	–
FE-Net 1.0× [2]	301M	3.7M	27.1	–
ESPNetV2 [12]	284M	3.5M	27.9	–
CondenseNet-C [8]	274M	2.9M	29.0	10.0
SFR-ShuffleNetV2 1.5×	306M	3.5M	26.5	8.6
CondenseNetV2-C	309M	6.1M	**24.1**	**7.3**

3. Experimental Setup on ImageNet

In our experiments, all our models are conducted under Pytorch Implementation [13]. Our training is based on the open-source code[2] which successfully reproduces the reported performance in MobileNetV3 [5]. We follow most of the training settings used in MobileNetV3 [5], except that we use the stochastic gradient descent (SGD) optimizer with an initial learning rate of 0.4, the cosine learning rate [10], a Nesterov momentum of weight 0.9 without dampening and a weight decay of 4×10^{-5} when batch size is 1024.

References

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

[2] Weijie Chen, Di Xie, Yuan Zhang, and Shiliang Pu. All you need is a few shifts: Designing efficient convolutional neural networks for image classification. In *CVPR*, 2019. 3

[3] François Chollet. Xception: Deep learning with depthwise separable convolutions. *arXiv preprint arXiv:1610.02357*, 2016. 2, 3

[4] Kai Han, Yunhe Wang, Qi Tian, Jianyuan Guo, and Chang Xu. Ghostnet: More features from cheap operations. In *CVPR*, 2020. 2, 4

[5] Andrew Howard, Mark Sandler, Grace Chu, Liang-Chieh Chen, Bo Chen, Mingxing Tan, Weijun Wang, Yukun Zhu, Ruoming Pang, Vijay Vasudevan, Quoc V. Le, and Hartwig Adam. Searching for mobilenetv3. In *ICCV*, 2019. 2, 3, 4

[6] Andrew G Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam. Mobilenets: Efficient convolutional neural networks for mobile vision applications. *arXiv preprint arXiv:1704.04861*, 2017. 2, 3

[2] https://github.com/rwightman/pytorch-image-models/

Table 8. Comparison of Top-1 and Top-5 classification error rate (%) with state-of-the-art NAS based efficient models on the ILSVRC validation set.

Model	FLOPs	Params	Top-1 err.	Top-5 err.
MobileNetV3 Large 0.75× [5]	155M	4.0M	26.7	–
RegNetX [14]	200M	2.7M	31.1	–
GhostNet 1.0× [4]	141M	5.2M	26.1	8.6
SFR-ShuffleNetV2+ Small	161M	5.2M	**25.5**	**8.2**
MobileNetV3 Large 1.00× [5]	219M	5.4M	24.8	–
GhostNet 1.3× [4]	226M	7.3M	24.3	7.3
MnasNet-A1 [18]	312M	3.9M	24.8	7.5
ProxylessNAS[1]	320M	4.1M	25.4	7.8
RegNetX [14]	400M	5.2M	27.3	–
NASNet-A [21]	564M	5.2M	26.0	8.4
AmoebaNet-C [15]	570M	6.4M	24.3	7.6
PNASNet-5 [9]	588M	5.1M	25.8	8.1
SFR-ShuffleNetV2+ Medium	229M	5.7M	**23.9**	**7.3**

[7] Jie Hu, Li Shen, and Gang Sun. Squeeze-and-excitation networks. In *CVPR*, 2018. 2

[8] Gao Huang, Shichen Liu, Laurens Van der Maaten, and Kilian Q Weinberger. Condensenet: An efficient densenet using learned group convolutions. In *CVPR*, 2018. 1, 2, 3

[9] Chenxi Liu, Barret Zoph, Maxim Neumann, Jonathon Shlens, Wei Hua, Li-Jia Li, Li Fei-Fei, Alan Yuille, Jonathan Huang, and Kevin Murphy. Progressive neural architecture search. In *ECCV*, 2018. 2, 4

[10] Ilya Loshchilov and Frank Hutter. Sgdr: Stochastic gradient descent with warm restarts. In *ICLR*, 2017. 3

[11] Ningning Ma, Xiangyu Zhang, Hai-Tao Zheng, and Jian Sun. Shufflenet v2: Practical guidelines for efficient cnn architecture design. In *ECCV*, 2018. 1, 2, 3

[12] Sachin Mehta, Mohammad Rastegari, Linda Shapiro, and Hannaneh Hajishirzi. Espnetv2: A light-weight, power efficient, and general purpose convolutional neural network. In *CVPR*, 2019. 2, 3

[13] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. In *NeurIPS*, pages 8026–8037, 2019. 3

[14] Ilija Radosavovic, Raj Prateek Kosaraju, Ross Girshick, Kaiming He, and Piotr Dollár. Designing network design spaces. In *CVPR*, 2020. 2, 4

[15] Esteban Real, Alok Aggarwal, Yanping Huang, and Quoc V Le. Regularized evolution for image classifier architecture search. In *AAAI*, 2019. 2, 4

[16] Mark Sandler, Andrew Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen. Inverted residuals and linear bottlenecks: Mobile networks for classification, detection and segmentation. In *CVPR*, 2018. 2, 3

[17] Ke Sun, Mingjie Li, Dong Liu, and Jingdong Wang. Igcv3: Interleaved low-rank group convolutions for efficient deep neural networks. In *BMVC*, 2018. 2, 3

[18] Mingxing Tan, Bo Chen, Ruoming Pang, Vijay Vasudevan, Mark Sandler, Andrew Howard, and Quoc V. Le. Mnasnet: Platform-aware neural architecture search for mobile. In *CVPR*, 2019. 2, 4

[19] Xiangyu Zhang, Xinyu Zhou, Mengxiao Lin, and Jian Sun. Shufflenet: An extremely efficient convolutional neural network for mobile devices. In *CVPR*, 2018. 2, 3

[20] Daquan Zhou, Qibin Hou, Yunpeng Chen, Jiashi Feng, and Shuicheng Yan. Rethinking bottleneck structure for efficient mobile network design. In *ECCV*, 2020. 3

[21] Barret Zoph, Vijay Vasudevan, Jonathon Shlens, and Quoc V Le. Learning transferable architectures for scalable image recognition. In *CVPR*, 2018. 2, 4