Rethinking Channel Dimensions for Efficient Model Design – Appendix –

Dongyoon Han Sangdoo Yun Byeongho Heo YoungJoon Yoo

NAVER AI Lab

A. Additional Search Results

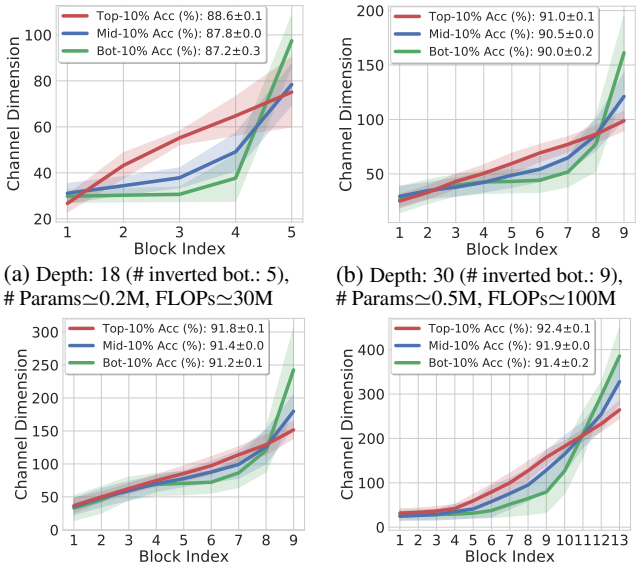
We provide the additional results of searching an effective channel dimension configuration by the proposed search method described in §4 to show the consistent trend of the searched channel configurations over different datasets. We perform searches on the CIFAR-10 dataset [9] with the identical search constraints presented in Table 3 in the main paper.

Following the previous analysis, we collect top-10%, middle-10% (i.e., the models between the top 50% and 60%), and bottom-10% models in terms of the model accuracy from 200 searched models to show the channel configurations with the performance statistics after each search. Figure A1 shows that the searched models as colored with red, which look linear functions, have higher accuracy while maintaining the similar computational costs. These similar trends are regularly observed while searching under the various constraints, and we can parameterize the models with a linear function by the block index again. The models in green have highly reduced the input-side channels and many output-side weight parameters resulting in the loss of accuracy. In addition, blue represents the models at the middle-10% accuracy, which looks similar to the conventional channel configurations such as MobileNetV2's [14].

All the searched channel configurations which can be approximated to linear parameterizations by the block index have higher accuracy (red) then blue ones, which are show the identical trends to Figure 2 in the main paper. It is worth noting that all the red lines in Figure A1 have higher slopes compared to those in Figure 2 in the main paper. This is because CIFAR-100 models have more parameters at the final classifier due to a larger number of classes, so the early layers employ fewer parameters than those of the models trained on the CIFAR-10 dataset.

B. Network Upgrade (cont'd)

In this section, we give further information of ReXNets and introduce our new model rebuilt upon MobileNetV1 [7] called ReXNet (plain) which does not use skip connections [5, 14] at each building block.



(a) Depth: 18 (# inverted bot.: 5), # Params≃0.2M, FLOPs≃30M

(b) Depth: 30 (# inverted bot.: 9), # Params≃0.5M, FLOPs≃100M

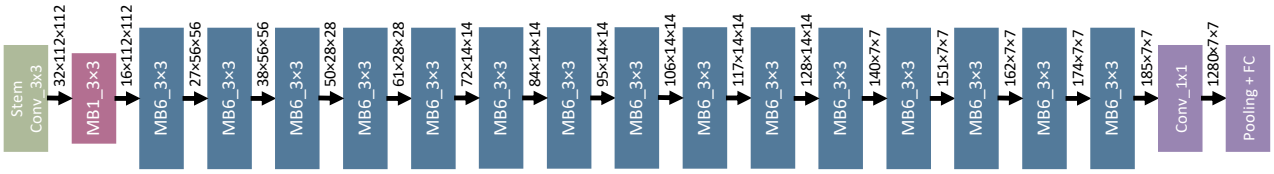
(c) Depth: 30 (# inverted bot.: 9), # Params≃1.0M, FLOPs≃200M

(d) Depth: 42 (# inverted bot.: 13), # Params≃3.0M, FLOPs≃350M

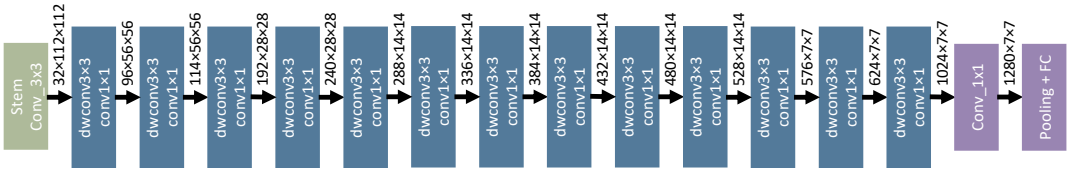
Figure A1. **Visualization of searched models' channel dimension on CIFAR-10**. Red: top 10%-accuracy models; Blue: middle 10% models; Green: bottom 10% models; we plot the averaged channel configurations with the 1-sigma range and report the averaged top-1 accuracy over each searched candidate.

B.1. ReXNet (cont'd)

We have described our upgraded model based on MobileNetV2 [14], which follows the searched linear parameterization on channel dimensions with some minor modifications in §4.4. Here, we illustrate the network architecture of our ReXNet (×1.0) in Figure A2a. We observe ReXNet (×1.0) has the identical block configuration to that of MobileNetV2 where a single-type building block MB6_3x3, which is the original inverted bottleneck [14] with the 3×3 depthwise convolution and the expansion ratio 6 is used as the basic building block except for the first inverted bottleneck. Every inverted bottleneck block that expands the



(a) ReXNet (×1.0)



(b) ReXNet (plain)

Figure A2. **Architectures of ReXNet (×1.0) and ReXNet (plain)**. MB1 and MB6 refer to MobileNetV2 [14]'s inverted bottlenecks with the expansion ratio of 1 and 6, respectively. Each model has almost similar architectural elements compared to the original ones.

channel dimensions (except for the downsampling blocks) has a skip connection where the expanded channel dimensions are padded with zeros.

B.2. ReXNet (plain)

We now present a new model redesigned based on MobileNetV1 [7]. We choose MobileNetV1 as another baseline because we intend to show a network architecture without skip connections (so-called a plain network) is able to be redesigned by following the proposed linear parameterization to show performance improvement. We do not change the depth of MobileNetV1. We use the identical configuration at the stem (i.e., 3×3 convolution with BN and ReLU) and the same large expansion layer at the penultimate layer.

We reassign the output channel dimensions of each 1×1 convolution as we did for ReXNet in §4.4. Following the investigation of single-layer design, we only replace the ReLUs with SiLU [6, 12] after the expansion layers such as all the 1×1 convolutions. We leave the ReLUs right after each depthwise convolution where the channel dimension ratio is 1. All the other channel dimensions including the stem and the penultimate layer are not changed. Since the network is a plain network, we do not adopt SE [8]. Our ReXNet (plain) is illustrated in Figure A2b. We provide the ImageNet performance of ReXNet (plain) in Table A1. We train the model by following the identical training setup in §5.1. As shown in Table A1, ReXNet (plain) does not achieve the best accuracy, but it is extremely faster than ReXNets on both CPU and GPU even with larger FLOPs.

B.3. Overall models

In addition to the models introduced in Table 5 in the main paper, we provide additional models adjusted by different width multipliers. Table A1 shows the ReXNets (×1.1, ×1.2, ×1.4, ×2.2, and ×3.0) and our new model ReXNet (plain) with the corresponding performances.

Network	Top-1	Top-5	FLOPs	Params.	CPU	GPU
ReXNet (plain)	74.8%	91.9%	0.56B	3.4M	22ms	10ms
ReXNet (×0.9)	77.2%	93.5%	0.35B	4.1M	46ms	20ms
ReXNet (×1.0)	77.9%	93.9%	0.40B	4.8M	47ms	21ms
ReXNet (×1.1)	78.6%	94.1%	0.48B	5.6M	51ms	24ms
ReXNet (×1.2)	79.0%	94.3%	0.57B	6.6M	53ms	26ms
ReXNet (×1.3)	79.5%	94.7%	0.66B	7.6M	55ms	28ms
ReXNet (×1.4)	79.8%	94.9%	0.76B	8.6M	57ms	30ms
ReXNet (×1.5)	80.3%	95.2%	0.86B	9.7M	59ms	31ms
ReXNet (×2.0)	81.6%	95.7%	1.5B	16M	69ms	40ms
ReXNet (×2.2)	81.7%	95.8%	1.8B	19M	73ms	46ms
ReXNet (×3.0)	82.8%	96.3%	3.4B	34M	96ms	61ms

Table A1. **Performance of ReXNets**. We report the ImageNet [13] performances of ReXNets. In addition to Table 5 in the main paper, we provide more models including ReXNet (plain) and ReXNets (×1.1, ×1.2, ×1.4, ×2.2, ×3.0). All the models are trained and evaluated with the resolution 224×224.

Nonlinearity	Top-1 (%)	Top-5 (%)	FLOPs	Params.
ReLU6 [14]	77.3	93.5	0.40B	4.8M
Leaky ReLU [11]	77.4	93.6	0.40B	4.8M
Softplus [3]	77.6	93.8	0.40B	4.8M
ELU [1]	77.6	93.7	0.40B	4.8M
SiLU [6, 12]	**77.9**	**93.9**	0.40B	4.8M

Table A2. **Nonlinear functions and ImageNet accuracy**.

C. Further Empirical Studies

C.1. Impact of nonlinear functions.

We have studied how nonlinearity can affect rank in the investigation in S3. We further study the actual impact of them by training the models on ImageNet. We train ReXNet (×1.0) with ELU, SoftPlus, LeakyReLU, ReLU6, and SiLU (Swish-1) with the identical training setup. As shown in Table A2, we obtain the results of top-1 accuracy in the order of SiLU (77.9%), ELU (77.6%), SoftPlus (77.6%), Leaky ReLU (77.4%), and ReLU6 (77.3%), and the trend is similar to the result in the empirical study in §3.2. The result

Model	Input Size	Bbox AP at IOU AP	AP_{50}	AP_{75}	Params	FLOPs
EfficientNet-B0 [15] + SSDLite	320×320	23.6	39.4	23.3	6.2M	0.97B
ReXNet (×0.9) + SSDLite	320×320	24.6	41.2	24.6	5.0M	0.88B
EfficientNet-B1 [15] + SSDLite	320×320	25.6	42.2	25.8	8.7M	1.35B
ReXNet (×1.0) + SSDLite	320×320	25.2	41.9	25.3	5.7M	1.01B
EfficientNet-B2 [15] + SSDLite	320×320	26.4	43.4	26.6	10.0M	1.55B
ReXNet (×1.3) + SSDLite	320×320	27.1	44.7	27.4	8.4M	1.60B

Table A3. **ReXNets vs. Noisy Student EfficientNets on COCO object detection**. We compare our ReXNets trained solely on ImageNet with stronger EfficientNets trained by Noisy Student training method [16] with RandAug [2]. Note that ReXNets here are equivalent to the models in §5.2. We report box APs on `val2017`.

Model	Input Size	Avg. Precision at IOU AP	AP_{50}	AP_{75}	Params.	FLOPs
EfficienetNet-B0 [15] + SSDLite	320x320	23.9	39.6	24.1	6.2M	0.97B
ReXNet (×0.9) + SSDLite	320x320	25.3	41.4	25.9	5.0M	0.88B
EfficienetNet-B1 [15] + SSDLite	320x320	25.6	41.9	26.0	8.7M	1.35B
ReXNet (×1.0) + SSDLite	320x320	25.9	42.6	26.3	5.7M	1.01B
EfficienetNet-B2 [15] + SSDLite	320x320	26.5	43.3	26.7	10.0M	1.55B
ReXNet (×1.3) + SSDLite	320x320	27.7	45.1	28.0	8.4M	1.60B

Table A4. **ReXNets vs. EfficientNets on COCO object detection.**. Note that all the models are trained from scratch with the identical training setup except for the doubled training iterations. We report box APs on `val 2017`.

indicates the quality of different nonlinearities that relates to model expressiveness; SiLU shows the best performance. This may provide a backup for why the recent lightweight models use SiLU (Swish-1) as the nonlinearity.

C.2. COCO object detection (cont'd)

We further provide more comparisons of ReXNets with EfficientNets [15] in SSDLite [14] on object detection. We first replace the EfficientNet backbone used in §5.2 with a stronger EfficientNet [16]. We then compare them trained from scratch on the COCO dataset [10].

Comparison with NoisyStudent EfficientNets. We now compare ReXNets with stronger EfficientNets [16], where the backbones are trained by a self-training method with extra large-scale data and RandAug [2]. Our goal is to show ReXNet's architectural capability over the EfficientNets without using the extra data when applying the backbones to a downstream task. We borrow the AP scores on `val 2017` from the ReXNet+SSDLite models in Table 6 in the main paperand train EfficientNet-B0, B1, and B2 in SSDLite using the pretrained NoisyStudent+RA EfficientNet models which are publicly released[1]. All the AP scores are evaluated by each checkpoint cached at the last iteration. Table A4 shows that ReXNets outperform the counterparts with the comparable computational costs. This indicates ReXNets pretrained on ImageNet are still promising.

Comparison of the models trained from scratch. We aim to verify the model expressiveness itself without using ImageNet-pretrained backbones. This is because one may wonder using a pretrained models trained with different training setups such as optimizer or regularizers affect the performance of downstream tasks. As shown in the work [4], the COCO dataset [10] is able to be trained from scratch, we verify the model's pure expressiveness by training the models from scratch.

We individually train ReXNets (×0.9, ×1.0, and ×1.3) and EfficientNet-B0, B1, and B2 in SSDLite without using ImageNet pretrained backbones. All AP scores are evaluated by each checkpoint cached at the last iteration again. Table A4 shows that ReXNets outperform the counterparts by **+1.4pp**, **+0.3pp**, and **+1.2pp** in AP score. With similar computational demands, ReXNets beat the EfficientNet counterparts by large margins, and surprisingly, the training from scratch makes ReXNet (×1.0) outperforms EfficienetNet-B1 by **+0.3pp** with much less computational costs. This indicates that our models are more powerful in terms of expressiveness even without the aids of the supervision of ImageNet pretrained models.

[1] https://github.com/tensorflow/tpu/tree/master/models/official/efficientnet

References

[1] Djork-Arné Clevert, Thomas Unterthiner, and Sepp Hochreiter. Fast and accurate deep network learning by exponential linear units (elus). In *ICLR*, 2016. 2

[2] Ekin D Cubuk, Barret Zoph, Jonathon Shlens, and Quoc V Le. Randaugment: Practical data augmentation with no separate search. *arXiv preprint arXiv:1909.13719*, 2019. 3

[3] Charles Dugas, Yoshua Bengio, François Bélisle, Claude Nadeau, and René Garcia. Incorporating second-order functional knowledge for better option pricing. In *NIPS*, 2001. 2

[4] Kaiming He, Ross Girshick, and Piotr Dollár. Rethinking imagenet pre-training. In *ICCV*, 2019. 3

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

[6] Dan Hendrycks and Kevin Gimpel. Gaussian error linear units (gelus). *arXiv preprint arXiv:1606.08415*, 2016. 2

[7] Andrew G Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam. Mobilenets: Efficient convolutional neural networks for mobile vision applications. *arXiv preprint arXiv:1704.04861*, 2017. 1, 2

[8] Jie Hu, Li Shen, and Gang Sun. Squeeze-and-excitation networks. In *arXiv:1709.01507*, 2017. 2

[9] A. Krizhevsky. Learning multiple layers of features from tiny images. In *Tech Report*, 2009. 1

[10] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft coco: Common objects in context. In *ECCV*, 2014. 3

[11] Andrew L. Maas, Awni Y. Hannun, and Andrew Y. Ng. Rectifier nonlinearities improve neural network acoustic models. In *ICML Workshop on Deep Learning for Audio, Speech and Language Processing*, 2013. 2

[12] Prajit Ramachandran, Barret Zoph, and Quoc V Le. Searching for activation functions. *arXiv preprint arXiv:1710.05941*, 2017. 2

[13] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. Imagenet large scale visual recognition challenge. *International Journal of Computer Vision*, 115(3):211–252, 2015. 2

[14] Mark Sandler, Andrew Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen. Mobilenetv2: Inverted residuals and linear bottlenecks. In *CVPR*, 2018. 1, 2, 3

[15] Mingxing Tan and Quoc V Le. Efficientnet: Rethinking model scaling for convolutional neural networks. *arXiv preprint arXiv:1905.11946*, 2019. 3

[16] Qizhe Xie, Minh-Thang Luong, Eduard Hovy, and Quoc V Le. Self-training with noisy student improves imagenet classification. In *CVPR*, 2020. 3