A. Appendix

A.1. Training recipe used in Table 1

Both Recipe-1 and Recipe-2 share the same batch size of 256, initial learning rate 0.1, weight decay at 4×10^{-5}, SGD optimizer, and cosine learning rate schedule. Recipe-1 train the model for 30 epochs and Recipe-2 train the model for 90 epochs. We don't introduce training techniques such as dropout, stochastic depth, and mixup in Recipe-1 or Recipe-2.

We make the same observation when training Recipe-1 and Recipe-2 use the same #Epochs but different weight decay: The accuracy of ResNet18 (1.4x width) is 0.25% higher and 0.36% lower than that of ResNet18 (2x depth) when the weight decay is $1e^{-4}$ and $1e^{-5}$, respectively.

A.2. Base architecture in recipe-only search

We show the base architecture (a scaled version of FBNetV2-L2) used in the recipe-only search in Table 10, while the input resolution is 256×256. This is the base architecture used in the training recipe search in Section 4.1. It achieves 79.1% top-1 accuracy on ImageNet with the original training recipe used for FBNetV2. With the searched training recipes, it achieves 79.9% ImageNet top-1 accuracy.

A.3. Search settings and details

In the recipe-only search experiment, we set the early-stop rank correlation threshold to be 0.92, and find the corresponding early-stop epoch to be 103. In the predictor-based evolutionary search, we set the population of the initial generation to be 100 (50 best-performing candidates from constrained iterative optimizationand 50 randomly generated samples). We generate 24 children from each candidate and pick the top 40 candidates for the next generation. Most of the settings are shared by the joint search of architecture and training recipes, except the early-stop epoch to be 108. The accuracy predictor consists of one embedding layer (architecture encoder layer) and one extra hidden layer. The embedding width is 24 for the joint search (note that there is no pretrained embedding layer for the recipe-only search). We set both minimum and maximum FLOPs constraint at 400M and 800M for the joint search, respectively. The selection of m best-performing samples in the constrained iterative optimization involves two steps: (1) equally divide the FLOP range into m bins and (2) pick the sample with the highest predicted score within each bin.

We show the detailed searched training recipe in Table 9. We also release the searched models.

A.4. Comparison between recipe-only search and hyperparameter optimizers

Many well-known hyperparameter optimizers (ASHA, Hyberband, PBT) evaluate on CIFAR10. One exception is [36], which reports a 0.5% gain for ResNet50 on ImageNet by searching optimizers, learning rate, weight decay, and momentum. By contrast, our recipe-only search with the same space (without EMA) increases ResNet50 accuracy by 1.9%, from 76.1% to 78.0%.

Notation	Value
lr	0.026
optim	RMSprop
ema	true
p	0.17
d	0.09
m	0.19
wd	7e-6

Table 9: Searched training recipe.

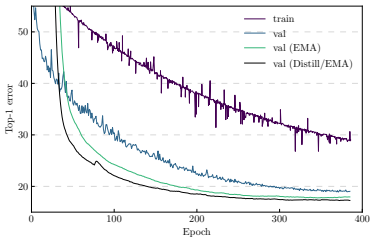

Figure 8: Training curve of the search recipe on FBNetV3-G.

A.5. Training settings and details

We use distributed training with 8 nodes for the final models, and scale up the learning rate by the number of distributed nodes (e.g., 8× for 8-node training). The batch size is set to be 256 per node. We use label smoothing and AutoAugment in the training. Additionally, we set the weight decay and momentum for batch normalization parameters to be zero and 0.9, respectively

We implement the EMA model as a copy of the original network (they share the same weights at $t = 0$). After each backward pass and model weights update, we update the EMA weights as

$$w_{t+1}^{ema} = \alpha w_t^{ema} + (1 - \alpha) w_{t+1} \tag{2}$$

where w_{t+1}^{ema}, w_t^{ema}, and w_{t+1} refer to the EMA weight at step $t + 1$, EMA weight at step t, and model weight at $t + 1$. We use an EMA decay α of 0.99985, 0.999, and 0.9998 in our experiments on ImageNet, CIFAR-10, and COCO, respectively. We further provide the training curves of FBNetV3-G in Fig. 8.

The baseline models (e.g., AlexNet, ResNet, DenseNet, and ResNeXt) are adopted from PyTorch open-source implementation without any architecture change. The input resolution is 224×224.

block	k	e	c	n	s	se	act.
Conv	1	3	16	1	2	-	hswish
MBConv	3	1	16	2	1	N	hswish
MBConv	5	5.46	24	1	2	N	hswish
MBConv	5	1.79	24	1	1	N	hswish
MBConv	3	1.79	24	1	1	N	hswish
MBConv	5	1.79	24	2	1	N	hswish
MBConv	5	5.35	40	1	2	Y	hswish
MBConv	5	3.54	32	1	1	Y	hswish
MBConv	5	4.54	32	3	1	Y	hswish
MBConv	5	5.71	72	1	2	N	hswish
MBConv	3	2.12	72	1	1	N	hswish
Skip	-	-	72	-	-	-	hswish
MBConv	3	3.12	72	1	1	N	hswish
MBConv	3	5.03	128	1	1	N	hswish
MBConv	5	2.51	128	1	1	Y	hswish
MBConv	5	1.77	128	1	1	Y	hswish
MBConv	5	2.77	128	1	1	Y	hswish
MBConv	5	3.77	128	4	1	Y	hswish
MBConv	3	5.57	208	1	2	Y	hswish
MBConv	5	2.84	208	2	1	Y	hswish
MBConv	5	4.88	208	3	1	Y	hswish
Skip	-	-	248	-	-	-	hswish
MBPool	-	6	1984	1	-	-	hswish
FC	-	-	1000	1	-	-	-

Table 10: Baseline architecture used in the recipe-only search. The block notations are identical to Table 2. Skip block refers to an identity connection if the input and output channel are equal otherwise a 1×1 conv.

Model	Distillation	FLOPs	Acc. (%)	Δ
EfficientNetB2	N	1050	80.3	0.0
FBNetV3-E	N	762	80.4	+0.1
FBNetV3-E	Y	762	81.3	+1.0

Table 11: Model comparison w/ and w/o distillation.

A.6. More discussions on training tricks

We acknowledge EfficientNet does not use distillation. For fair comparison, we report FBNetV3 accuracy without distillation. We provide an example in Table 11: Without distillation, FBNetV3 achieves higher accuracy with 27% less FLOPs, compared to EfficientNet. However, all our training tricks (including EMA and distillation) are used in the other baselines, including BigNAS and OnceForAll.

Generality of *stochastic weight averaging via EMA*. We observe that *stochastic weight averaging via EMA* yields significant accuracy gain for the classification tasks, as has been noted prior [3, 14]. We hypothesize that such a mechanism could be used as a general technique to improve other DNN models. To validate this, we employ it to train a RetinaNet [27] on COCO object detection [28] with ResNet50 and ResNet101 backbones. We follow most of the default training settings but introduce EMA and Cosine learning rate. We observe similar training curves and behavior as the classification tasks, as shown in Fig. 9. The generated RetinaNets

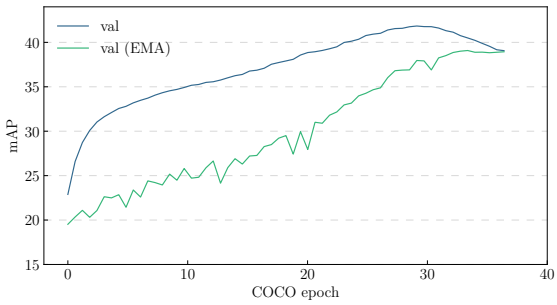

Figure 9: Training curves for RetinaNet with ResNet101 backbone on COCO object detection.

with ResNet50 and ResNet101 backbones achieve 40.3 and 41.9 mAP, respectively, both substantially outperform the best reported values in [54] (38.7 and 40.4 for ResNet50 and ResNet101, respectively). A promising future direction is to study such techniques and extend it to other DNNs and applications.

A.7. Further improvements on FBNetV3

We demonstrate that using a teacher model with higher accuracy leads to further accuracy gain on FBNetV3. We use RegNetY-32G FLOPs (top-1 accuracy 84.5%) [10] as the teacher model, and distill all the FBNetV3 models. We show all the derived models in Fig. 10, where we observe a

consistent accuracy gain at 0.2% - 0.5% for all the models.

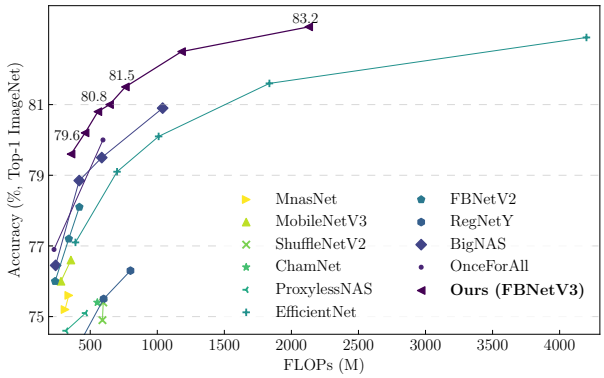

Figure 10: ImageNet accuracy vs. model FLOPs comparison of FBNetV3 (distilled from giant RegNet-Y models) with other efficient convolutional neural networks.