Rethinking Vision Transformers for MobileNet Size and Speed – Supplementary Material

Yanyu Li
Snap Inc., Northeastern University
li.yanyu@northeastern.edu

Ju Hu
Snap Inc.
jhu3@snap.com

Yang Wen
Snap Inc.
yangwenca@gmail.com

Georgios Evangelidis
Snap Inc.
gevangelidis@snap.com

Kamyar Salahi
UC Berkeley
kam.salahi@berkeley.edu

Yanzhi Wang
Northeastern University
yanz.wang@northeastern.edu

Sergey Tulyakov
Snap Inc.
stulyakov@snap.com

Jian Ren
Snap Inc.
jren@snap.com

1. More Experimental Details and Results

Training hyper-parameters. We provide the detailed training hyper-parameters for the ImageNet-1K [2] classification task in Tab. 1, which is a similar recipe following DeiT [8], LeViT [3], and EfficientFormer [5] for fair comparisons.

Table 1. Training hyper-parameters for ImageNet-1K classification task. The drop path rate is for the [S0, S1, S2, L] model series.

Hyperparameters	Config
optimizer	AdamW
learning rate	0.001×(BS/1024)
LR schedule	cosine
warmup epochs	5
training epochs	300
weight decay	0.025
augmentation	RandAug(9, 0.5)
color jitter	0.4
gradient clip	0.01
random erase	0.25
label smooth	0.1
mixup	0.8
cutmix	1.0
drop path	$[0, 0, 0.02, 0.1]$

Results without distillation. We provide our models trained *without* distillation in Table. 2. Compared with representative works trained without distillation, *e.g.*, MobileNetV2 [7], MobileFormer [1] (trained with longer epochs), EdgeViT [6], and PoolFormer [9], our models still achieve better latency-accuracy trade-offs.

Analysis on attention bias.

Table 2. Results without distillation.

Model	Params (M)	MACs (G)	Latency (ms)	Epochs	Top-1 (%)
MobileNetV2 1.0	3.5	0.30	0.9	300	71.8
EfficientFormerV2-S0	3.5	0.40	0.9	300	73.7
EdgeViT-XS	6.7	1.10	3.6	300	77.5
EfficientFormerV2-S1	6.1	0.65	1.1	300	77.9
MobileFormer-508M	14.0	0.51	6.6	450	79.3
PoolFormer-s12	12.0	2.0	1.5	300	77.2
EfficientFormerV2-S2	12.6	1.25	1.6	300	80.4

Table 3. Analysis of explicit position encoding (Attention Bias). We use EfficientFormerV2-S1 for the experiments.

Params (M)	Epoch	Attention Bias	Top-1 (%)
6.10	300	Y	79.0
6.08	300	N	78.8
6.10	450	Y	79.7
6.08	450	N	79.5

Attention Bias is employed to serve as explicit position encoding. On the downside, attention bias is resolution sensitive, making the model fragile when migrating to downstream tasks. By deleting attention bias, we observe 0.2% drop in accuracy for both 300 and 450 training epochs (Attention Bias as Y *vs.* N in Tab. 3), showing that EfficientFormerV2 can still preserve a reasonable accuracy without explicit position encoding.

2. More Ablation Analysis of Search Algorithm

Importance of Expansion Ratios. We first discuss the necessity to search for expansion ratios on top of width. As in Tab. 4, we show that, by adjusting width to maintain an identical budget, *i.e.*, the same number of parameters for each model, varying the expansion ratio incurs considerable difference in performance. As a result, we can not obtain

Pareto optimality by solely searching for width while setting a fixed expansion ratio.

Table 4. Ablation analysis on expansion ratios. Varying expansion ratios lead to different results even with the same number of parameters. Latency is obtained on iPhone 12.

Expansion ratio	Params (M)	Latency (ms)	Top-1 (%)
4	13.4	1.6	81.8
2	13.4	1.6	81.6
1	13.4	1.6	81.1

Analysis of Searching the Expansion Ratios. We verify the performance of different search algorithms in Tab. 5. We obtain the baseline result using the search pipeline in EfficientFormer [5] to search only for the depth and width. With a budget of 7M parameters, we obtain a subnetwork with 79.2% top-1 accuracy on ImageNet-1K. Then, we apply a simple magnitude-based pruning to determine expansion ratios in a fine-grained manner. Unfortunately, the performance is not improved. Though searching for expansion ratios is important (Tab. 4), it is non-trivial to achieve Pareto optimality with simple heuristics. Finally, we apply our fine-grained search method and obtain a subnetwork with 79.4% top-1 accuracy, demonstrating the effectiveness of our approach.

Table 5. Ablation on search methods for depth, width, and expansion ratios. EfficientFormer [5] merely searches for depth and width. On top of EfficientFormer [5], we perform network pruning to decide channel numbers for stage width and expansion ratios. Finally, we show the results of our search algorithm for jointly optimizing depth, width, and expansions.

Method	Params (M)	Latency (ms)	Top-1 (%)
From EfficientFormer [5]	7.0	1.15	79.2
From EfficientFormer [5] + Pruning	7.0	1.15	79.2
Ours	7.0	1.15	79.4

Analysis of Different $\alpha_{latency}$ and α_{size} in Eqn. 1. Here, we provide the results for analyzing how different values of $\alpha_{latency}$ and α_{size} can impact the search results in Tab. 6. Our search algorithm is stable to different α settings. Increasing the weight of size (α_{size}) leads to slower models. Our current setting ($\alpha_{latency}$ as 1.0 and α_{size} as 0.5) is determined by aligning with recent works, *e.g.*, EdgeViT, UniNet, etc, to make fair comparisons.

Table 6. Analysis of the $\alpha_{latency}$ and α_{size} in search algorithm.

$\alpha_{latency}$	α_{size}	Params (M)	Latency	Top-1 (%)
1.0	1.0	3.5	1.1 ms	77.0
0.5	1.0	3.5	1.3 ms	77.3
1.0	0.5	3.5	0.9 ms	75.7

Visualization of Search Results. In Fig. 1, we visualize the performance of the searched subnetworks, including the networks obtained by using the search algorithm from EfficinetFormer[5] and networks found by our fine-grained joint search. We employ `MES` as an efficiency measurement and plot in logarithmic scale. The results demonstrate the advantageous performance of our proposed search method.

Table 7. Generalization of design choices on detection and instance segmentation. Configuration matches Tab.1 in paper. For instance, Sec.3.1 refers to falling back to DWConv mixer instead of FFN. Without our proposed stride attention (Sec.3.4), the model encounters memory issues and cannot run on mobile. Note that Sec.3.2 is not included as 5 stage network is not a common practice in detection tasks.

Configuration	Latency (ms)	AP^{box}	AP^{mask}
EfficientFormerV2-S2	187.9	33.5	31.2
Sec.3.1	181.1	31.6	29.5
Sec.3.3	187.2	33.4	31.2
Sec.3.4	Failed	33.9	31.6
Sec.3.5	187.7	32.7	30.6

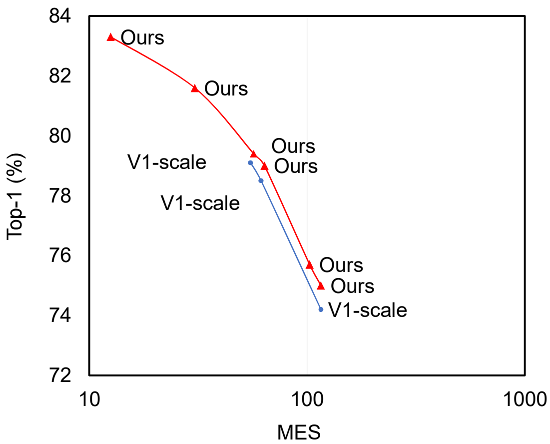

Figure 1. Comparisons between our search method (Ours) and the search pipeline from EfficientFormer [5] (denoted as V1-scale), starting from the same supernet trained on ImageNet-1K.

Design Choice Ablation. We ablate our network design choices on detection/instance segmentation task, and prove that the conclusions from ImageNet-1K classification task can *transfer*. We train EfficientFormerV2-S2 on *MS-COCO* dataset from **scratch** for 12 epochs *without* ImageNet pretraining. The results are included in the Tab. 7. For instance, Sec.3.1 refers to falling back to DWConv mixer instead of FFN. Our design holds clear advantages. In addition, without our proposed stride attention (Sec.3.4), the model encounters memory issues and can not run on mobile. Note that Sec.3.2 is not included as 5 stage network is not a common practice in detection tasks.

More random models and the cost analysis of searching via supernet *vs.* random. We sample more random models (10) with a more extensive latency range to compare against our searched models. As seen in Fig. 2, searching mod-

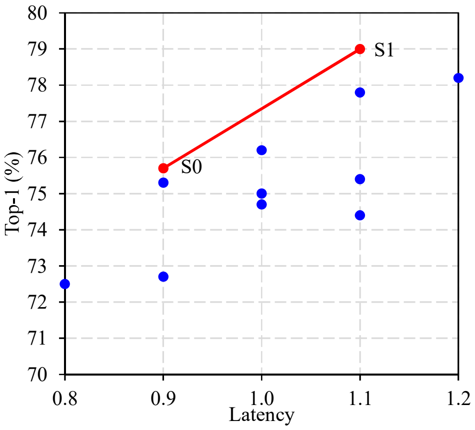

Figure 2. Comparisons with more random sampled models. We take 0.1ms as the significant digit based on mobile measurement precision.

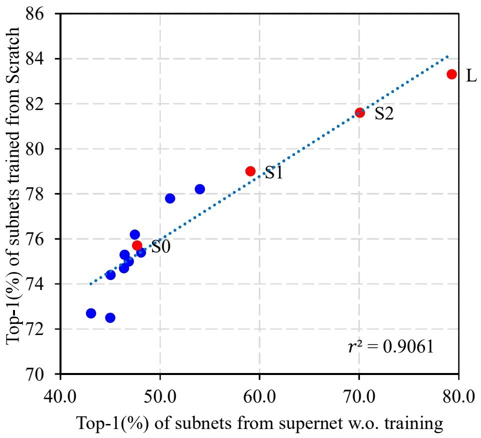

Figure 3. Subnet evaluation vs. training from scratch. We report our searched models (red) along with random ones (blue).

els by our approach (red line) gets better performance than random search (blue dots). Our supernet training takes 37 GPU days (A100), which is 4.6× the training time of the L model (8 GPU days). However, assuming at least 10 random subnets are needed to search each candidate, the cost of random search for L-level model itself accumulates to 80 GPU days (2× longer than supernet). Also, the cost of random search further scales up for multiple networks (four in our work). Thus, our search method is more efficient than random search.

Accuracy of subnets from supernet and their correlation to final accuracy. In the Figure. 3, we show the accuracy of multiple subnets obtained from the supernet and their correlation to final accuracy (training from scratch). We refer EagleEye [4] for comparison. Through effectively-trained supernet, we obtain higher subnet evaluation accuracy ($> 40\%$, *v.s.* $< 10\%$ in EagleEye), as well as better correlations to final accuracy ($r^2 = 0.91$, *v.s* 0.63 in EagleEye) measured by Pearson correlation coefficient.

3. Network Configurations

The detailed network architectures for EfficientFormerV2-S0, S1, S2, and L are provided in Tab. 8. We report the stage resolution, width, depth, and per-block expansion ratios.

References

[1] Yinpeng Chen, Xiyang Dai, Dongdong Chen, Mengchen Liu, Xiaoyi Dong, Lu Yuan, and Zicheng Liu. Mobile-former: Bridging mobilenet and transformer. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5270–5279, 2022. 1

[2] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee, 2009. 1

[3] Benjamin Graham, Alaaeldin El-Nouby, Hugo Touvron, Pierre Stock, Armand Joulin, Herve Jegou, and Matthijs Douze. Levit: A vision transformer in convnet's clothing for faster inference. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 12259–12269, October 2021. 1

[4] Bailin Li, Bowen Wu, Jiang Su, and Guangrun Wang. Eagleeye: Fast sub-net evaluation for efficient neural network pruning. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part II 16*, pages 639–654. Springer, 2020. 3

[5] Yanyu Li, Geng Yuan, Yang Wen, Ju Hu, Georgios Evangelidis, Sergey Tulyakov, Yanzhi Wang, and Jian Ren. Efficientformer: Vision transformers at mobilenet speed. In *NeurIPS*, 2022. 1, 2

[6] Junting Pan, Adrian Bulat, Fuwen Tan, Xiatian Zhu, Lukasz Dudziak, Hongsheng Li, Georgios Tzimiropoulos, and Brais Martinez. Edgevits: Competing light-weight cnns on mobile devices with vision transformers. In *European Conference on Computer Vision*, 2022. 1

[7] Mark Sandler, Andrew Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen. Mobilenetv2: Inverted residuals and linear bottlenecks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4510–4520, 2018. 1

[8] Hugo Touvron, Matthieu Cord, and Hervé Jégou. Deit iii: Revenge of the vit. *arXiv preprint arXiv:2204.07118*, 2022. 1

[9] Weihao Yu, Mi Luo, Pan Zhou, Chenyang Si, Yichen Zhou, Xinchao Wang, Jiashi Feng, and Shuicheng Yan. Metaformer is actually what you need for vision. *arXiv preprint arXiv:2111.11418*, 2021. 1

Table 8. Architecture details of EfficientFormerV2.

Stage	Resolution	Type	Config	EfficientFormerV2			
				S0	S1	S2	L
stem	$\frac{H}{2} \times \frac{W}{2}$	Conv	Kernel, Stride	$3 \times 3, 2$	$3 \times 3, 2$	$3 \times 3, 2$	$3 \times 3, 2$
			N, C	$1, 16$	$1, 16$	$1, 16$	$1, 20$
	$\frac{H}{4} \times \frac{W}{4}$	Conv	Kernel, Stride	$3 \times 3, 2$	$3 \times 3, 2$	$3 \times 3, 2$	$3 \times 3, 2$
			N, C	$1, 32$	$1, 32$	$1, 32$	$1, 40$
1	$\frac{H}{4} \times \frac{W}{4}$	FFN	N, C	$2, 32$	$3, 32$	$4, 32$	$5, 40$
			E	$[4, 4]$	$[4, 4, 4]$	$[4, 4, 4, 4]$	$[4, 4, 4, 4, 4]$
2	$\frac{H}{8} \times \frac{W}{8}$	FFN	N, C	$2, 48$	$3, 48$	$4, 64$	$5, 80$
			E	$[4, 4]$	$[4, 4, 4]$	$[4, 4, 4, 4]$	$[4, 4, 4, 4, 4]$
3	$\frac{H}{16} \times \frac{W}{16}$	FFN	N, C	$6, 96$	$9, 120$	$12, 144$	$15, 192$
			E	$[4, 3, 3, 3, 4, 4]$	$[4(\times 5), 3(\times 4)]$	$[4(\times 6), 3(\times 6)]$	$[4(\times 8), 3(\times 7)]$
		MHSA	N	2	2	4	6
4	$\frac{H}{32} \times \frac{W}{32}$	FFN	N, C	$4, 176$	$6, 224$	$8, 288$	$10, 384$
			E	$[4, 3, 3, 4]$	$[4, 4, 3, 3, 4, 4]$	$[4(\times 4), 3(\times 4)]$	$[4(\times 6), 3(\times 4)]$
		MHSA	N	2	2	4	6