FP-NAS: Fast Probabilistic Neural Architecture Search (Supplementary Material)

A. Index

The supplementary materials are organized as follows.

- We present the details of the searched FP-NAS models and visualize them in Section B.
- Details of our training recipe, and comparisons with those used by other methods are presented in Section C.
- Details of FP-NAS search spaces are presented in Section D.
- A study on the supernet warmup is presented in Section E.
- The derivation of gradients for updating architecture parameters is shown in Section F.

B. Understanding FP-NAS Architectures

We use the proposed FP-NAS method to search for a family of models in different sizes. The complete results are shown in Table B.1.

We also visualize two representative FP-NAS models, including FP-NAS-S1++ and FP-NAS-L2 model, in Fig B.1. Compared with hand-crafted models, the searched architectures select more non-uniform choices along kernel size, non-linearity, feature channel and number of splits over MB-Conv blocks.

For example, small kernel size 3 is more favored in the early blocks while large kernel size 5 is more often chosen in the later blocks. This is likely because large kernel size is more computationally expensive and we can only afford to use it in the later blocks where the spatial size of feature map is small (e.g. 14^2, 7^2). Also in the later blocks, convolution with large kernel size can more effectively capture the global context.

We also find large choices of the number of splits in SA block, such as 2 and 4, are more often used in the later blocks. This is likely because high-level semantic features only emerge in the later blocks, and attention with multiple splits is more needed to attend to certain semantic features relevant to the image content, compared with low-level features in the early blocks where attention is less useful.

Architecture	Input size	FLOPS(M)	Params(M)	Distill	Top-1 acc (%)
S1++	128^2	66	5.9	×	70.0
S2++	160^2	98	5.8	×	72.2
S3++	192^2	147	5.8	×	74.2
S4++	224^2	268	6.4	×	76.6
L0	224^2	399	11.3	×	78.0
L1	240^2	728	15.8	×	79.8
				✓	80.9
L2	256^2	997	20.7	×	80.7
				✓	81.6

Table B.1: **The family of FP-NAS models.** All results are obtained using one model and a single crop on ImageNet-1K. Column *Distill* denotes whether model distillation [2] is used to train the model.

Model	LS	AA	EMA	SD	Top-1 acc(%)
FP-NAS-S1++	✓	×	✓	×	69.8
	✓	✓	×	×	70.0
	✓	✓	✓	×	70.0
	✓	✓	✓	✓	69.5
FP-NAS-S4++	✓	×	✓	×	76.4
	✓	✓	×	×	76.4
	✓	✓	✓	×	76.6
	✓	✓	✓	✓	76.4
FP-NAS-L0	✓	×	✓	×	77.6
	✓	✓	×	×	77.5
	✓	✓	✓	×	77.9
	✓	✓	✓	✓	78.0
FP-NAS-L2	✓	×	✓	×	80.2
	✓	✓	×	×	79.2
	✓	✓	✓	×	80.6
	✓	✓	✓	✓	80.7

Table C.2: **ImageNet top-1 accuracy (%) of FP-NAS models trained with different training recipes**. We start with using LS only, and sequentially add AA, EMA and SD.

C. Training Recipes

When training FP-NAS models, we adopt label smoothing [4] (LS), Auto-Augment [1] (AA), and Exponential Model Averaging (EMA). We study the impact of the training recipe on the testing accuracy by training FP-NAS models under different training recipes. The results are shown in Table C.2.

For EMA, it does not improve our smallest S1++ model,

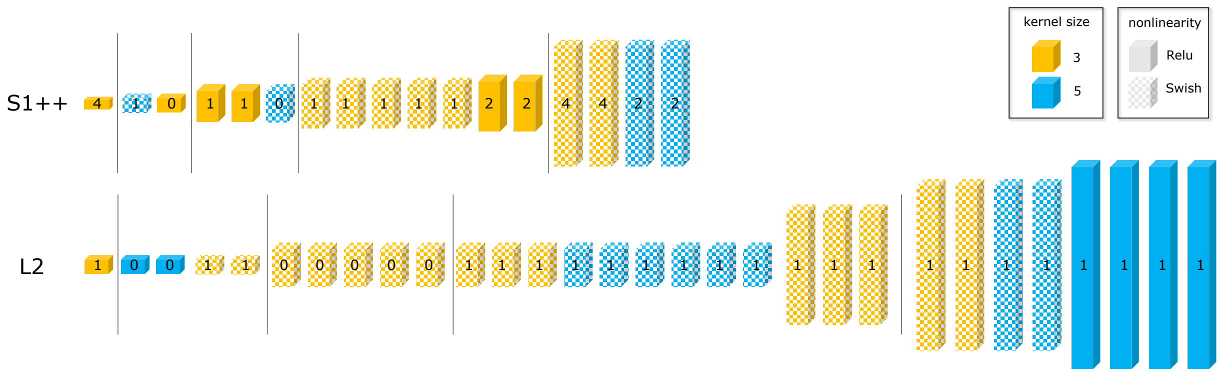

Figure B.1: **The architecture of FP-NAS-S1++ and FP-NAS-L2 model.** MBConv blocks from different stage are separated by a vertical bar, and each MBConv block is shown as a box. Box color encodes kernel size. Box height is proportional to the feature channel. The solid and hatched box denotes ReLU and Swish nonlinearity, respectively. The number of splits $\in \{0, 1, 2, 4\}$ in Split-Attention (SA) block is overlaid on top of each box. Choice 0 means SA block is not used, choice 1 means SE block is used, while choices 2 or 4 means SA block with 2 or 4 splits is used. For clarity, searched expansion rate is not shown.

Model Size	Method	LS	AA	EMA	SD
Small	MnasNet	✓	×	✓	×
	FBNetV2	✓	✓	×	×
	FP-NAS (ours)	✓	✓	✓	×
Large	AtomNAS	✓	×	✓	×
	EfficientNet	✓	✓	✓	✓
	FP-NAS (ours)	✓	✓	✓	✓

Table C.3: **Comparing training recipe used by different methods.** We use the following short-hand notations. *LS*: label smoothing [4], *AA*: auto-augmentation [1], *EMA*: exponential model averaging [6], and *SD*: stochastic depth [3].

Max Input ($S^2 \times C$)	Operator	Expansion	Channel	Repeat	Stride
$224^2 \times 3$	conv 3×3	-	16	1	2
$112^2 \times 16$	MBConv	(1.5, 6.0, 0.75)	(8, 32, 8)	1	1
$112^2 \times 32$	MBConv	(1.5, 6.0, 0.75)	(16, 32, 8)	2	2
$56^2 \times 32$	MBConv	(1.5, 6.0, 0.75)	(16, 32, 8)	2	1
$56^2 \times 32$	MBConv	(1.5, 6.0, 0.75)	(16, 56, 8)	2	2
$28^2 \times 56$	MBConv	(1.5, 6.0, 0.75)	(16, 56, 8)	2	1
$28^2 \times 56$	MBConv	(1.5, 6.0, 0.75)	(48, 96, 16)	4	2
$14^2 \times 96$	MBConv	(1.5, 6.0, 0.75)	(48, 96, 16)	3	1
$14^2 \times 96$	MBConv	(1.5, 6.0, 0.75)	(72, 136, 16)	3	1
$14^2 \times 136$	MBConv	(1.5, 6.0, 0.75)	(112, 224, 16)	2	2
$7^2 \times 224$	MBConv	(1.5, 6.0, 0.75)	(112, 224, 16)	2	1
$7^2 \times 224$	MBConv	(1.5, 6.0, 0.75)	(168, 280, 16)	2	1
$7^2 \times 280$	conv 1×1	-	1984	1	1
$7^2 \times 1984$	avgpool	-	-	1	1
1984	fc	-	1000	1	-

Table D.4: **The macro-architecture of FP-NAS-L0 search space**. It is used to search for FP-NAS-L0 architecture.

but can improve S4++ model by 0.2%. EMA is more effective on large models, including L0 and L2 models, and improve their accuracy by 0.3% and 1.4%, respectively.

For SD, it actually reduces the accuracy of our small models such as S1++ and S4++. Therefore, we do not use SD to train our small models. On the other side, on our large models, such as L0 and L2, SD slightly improves the accuracy by 0.1%.

We also compare our training recipe with that from other methods, and the results are shown in Table C.3. For small models under consideration, including MnasNet [5] and FBNetV2 [7], they are different in whether AA and EMA are used. Although FP-NAS-S++ models are trained with both AA and EMA, the improvement by using either one is much less significant compared with the improvement of FP-NAS-S++ models over other models (See Table 8 in the paper). For training large FP-NAS-L models, we use LS, AA, EMA and SD, which are also used by EfficientNet.

D. FP-NAS Search Spaces

In the Table 3, we introduce three FP-NAS search spaces from L0 to L2. They share the same FP-NAS micro-architecture, but have different macro-architectures, which are shown in Table D.4, D.5, and D.6, respectively.

E. SuperNet Warmup

In our experiments, we fix the architecture hyper-parameters while only updating the model weights at the beginning of search for a number of epochs. This is to warm-up the supernet by uniformly sampling architectures from the initial distribution, and update model weights associated with them. Compared with randomly initialized model weights, the updated weights lead to a better estimation of the data likelihood $P(\mathbf{y}|\mathbf{X}, \omega, \mathbf{A}_k)$ of the sampled architectures $\{\mathbf{A}_k\}$ and therefore a better estimation of the gradients for updating architecture parameters.

Max Input ($S^2 \times C$)	Operator	Expansion	Channel	Repeat	Stride
$240^2 \times 3$	conv 3×3	-	24	1	2
$120^2 \times 24$	MBConv	(1.5, 6.0, 0.75)	(16, 40, 8)	1	1
$120^2 \times 40$	MBConv	(1.5, 6.0, 0.75)	(24, 40, 8)	2	2
$60^2 \times 40$	MBConv	(1.5, 6.0, 0.75)	(24, 40, 8)	2	1
$60^2 \times 40$	MBConv	(1.5, 6.0, 0.75)	(24, 72, 8)	3	2
$30^2 \times 72$	MBConv	(1.5, 6.0, 0.75)	(24, 72, 8)	2	1
$30^2 \times 72$	MBConv	(1.5, 6.0, 0.75)	(64, 120, 16)	3	2
$15^2 \times 120$	MBConv	(1.5, 6.0, 0.75)	(64, 120, 16)	3	1
$15^2 \times 120$	MBConv	(1.5, 6.0, 0.75)	(88, 168, 16)	3	1
$15^2 \times 168$	MBConv	(1.5, 6.0, 0.75)	(88, 168, 16)	3	1
$15^2 \times 168$	MBConv	(1.5, 6.0, 0.75)	(136, 272, 16)	2	2
$8^2 \times 272$	MBConv	(1.5, 6.0, 0.75)	(136, 272, 16)	2	1
$8^2 \times 272$	MBConv	(1.5, 6.0, 0.75)	(208, 336, 16)	2	1
$8^2 \times 336$	MBConv	(1.5, 6.0, 0.75)	(208, 336, 16)	2	1
$8^2 \times 336$	conv 1×1	-	1984	1	1
$8^2 \times 1984$	avgpool	-	-	1	1
1984	fc	-	1000	1	-

Table D.5: **The macro-architecture of FP-NAS-L1 search space**. It is used to search for FP-NAS-L1 architecture.

Max Input ($S^2 \times C$)	Operator	Expansion	Channel	Repeat	Stride
$256^2 \times 3$	conv 3×3	-	24	1	2
$128^2 \times 24$	MBConv	(1.5, 6.0, 0.75)	(16, 40, 8)	1	1
$128^2 \times 40$	MBConv	(1.5, 6.0, 0.75)	(24, 40, 8)	2	2
$64^2 \times 40$	MBConv	(1.5, 6.0, 0.75)	(24, 40, 8)	2	1
$64^2 \times 40$	MBConv	(1.5, 6.0, 0.75)	(24, 72, 8)	3	2
$32^2 \times 72$	MBConv	(1.5, 6.0, 0.75)	(24, 72, 8)	2	1
$32^2 \times 72$	MBConv	(1.5, 6.0, 0.75)	(64, 120, 16)	3	2
$16^2 \times 120$	MBConv	(1.5, 6.0, 0.75)	(64, 120, 16)	3	1
$16^2 \times 120$	MBConv	(1.5, 6.0, 0.75)	(88, 168, 16)	3	1
$16^2 \times 168$	MBConv	(1.5, 6.0, 0.75)	(88, 168, 16)	3	1
$16^2 \times 168$	MBConv	(1.5, 6.0, 0.75)	(136, 272, 16)	2	2
$8^2 \times 272$	MBConv	(1.5, 6.0, 0.75)	(136, 272, 16)	2	1
$8^2 \times 272$	MBConv	(1.5, 6.0, 0.75)	(208, 336, 16)	2	1
$8^2 \times 336$	MBConv	(1.5, 6.0, 0.75)	(208, 336, 16)	2	1
$8^2 \times 336$	conv 1×1	-	1984	1	1
$8^2 \times 1984$	avgpool	-	-	1	1
1984	fc	-	1000	1	-

Table D.6: **The macro-architecture of FP-NAS-L2 search space**. It is used to search for FP-NAS-L2 architecture.

SuperNet Warmup	FLOPS (M)	Top-1 Accuracy (%)
×	59	67.8
✓	58	**68.6**

Table E.7: **Comparing searched architectures when supernet warmup is used or not.**

In a study where architecture is searched in FBNetV2-F space, we use 315 epochs and compare the searched model when the architecture parameters are freezed in the beginning 45 epochs or not. The results are shown in Table E.7. The architecture searched without supernet warmup has inferior accuracy, and thus we always warmup supernet to search FP-NAS models.

F. Derivation of Gradients for Updating Architecture Parameters

In Eqn (7) of the paper, we presented how to compute the gradient w.r.t architecture hyper-parameters α. The full derivation is shown below. First, the cost-aware loss function is defined as follows.

$$\mathcal{L}(\omega, \alpha) = -\log\, P(\mathbf{y}|\mathbf{X}, \omega, \alpha) + \beta\, \log\, C(\alpha) \quad (1)$$

Then, the gradient w.r.t architecture parameters can be derived as follows.

$$\begin{aligned}
\nabla_\alpha\, \mathcal{L}(\omega, \alpha) &= \frac{1}{P(\mathbf{y}|\mathbf{X}, \omega, \alpha)} \int P(\mathbf{y}|\mathbf{X}, \omega, \mathbf{A})\, \nabla_\alpha - P(\mathbf{A}|\alpha)\, \mathrm{d}\mathbf{A} \\
&+ \beta \frac{1}{C(\alpha)} \int C(\mathbf{A})\, \nabla_\alpha P(\mathbf{A}|\alpha)\, \mathrm{d}\mathbf{A} \\
&= \int P(\mathbf{A}|\alpha) (\frac{P(\mathbf{y}|\mathbf{X}, \omega, \mathbf{A})}{P(\mathbf{y}|\mathbf{X}, \omega, \alpha)} - \beta \frac{C(\mathbf{A})}{C(\alpha)})\, \nabla_\alpha - \log\, P(\mathbf{A}|\alpha)\, \mathrm{d}\mathbf{A} \\
&\approx \frac{1}{K} \sum_{k=1}^{K} (\frac{P(\mathbf{y}|\mathbf{X}, \omega, \mathbf{A}_k)}{P(\mathbf{y}|\mathbf{X}, \omega, \alpha)} - \beta \frac{C(\mathbf{A}_k)}{C(\alpha)})\, \nabla_\alpha\, - \log P(\mathbf{A}_k|\alpha) \\
&\approx \sum_k m_k^\alpha\, \nabla_\alpha - \log P(\mathbf{A}_k|\alpha)
\end{aligned} \quad (2)$$

where $m_k^\alpha = \frac{P(\mathbf{y}_{val}|\mathbf{X}_{val}, \omega, \mathbf{A}_k)}{\sum_{k'} P(\mathbf{y}_{val}|\mathbf{X}_{val}, \omega, \mathbf{A}_{k'})} - \beta \frac{C(\mathbf{A}_k)}{\sum_{k'} C(\mathbf{A}_{k'})}$.

References

[1] Ekin D Cubuk, Barret Zoph, Dandelion Mane, Vijay Vasudevan, and Quoc V Le. Autoaugment: Learning augmentation strategies from data. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 113–123, 2019. 1, 2

[2] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2015. 1

[3] Gao Huang, Yu Sun, Zhuang Liu, Daniel Sedra, and Kilian Q Weinberger. Deep networks with stochastic depth. In *European conference on computer vision*, pages 646–661. Springer, 2016. 2

[4] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna. Rethinking the inception architecture for computer vision. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2818–2826, 2016. 1, 2

[5] 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 *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2820–2828, 2019. 2

[6] Mingxing Tan and Quoc V Le. Efficientnet: Rethinking model scaling for convolutional neural networks. *arXiv preprint arXiv:1905.11946*, 2019. 2

[7] Alvin Wan, Xiaoliang Dai, Peizhao Zhang, Zijian He, Yuandong Tian, Saining Xie, Bichen Wu, Matthew Yu, Tao Xu, Kan Chen, et al. Fbnetv2: Differentiable neural architecture search for spatial and channel dimensions. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12965–12974, 2020. 2