Supplementary Materials for ElasticViT

A. Search Space Design

A.1. Mobile-friendly Transformer Block

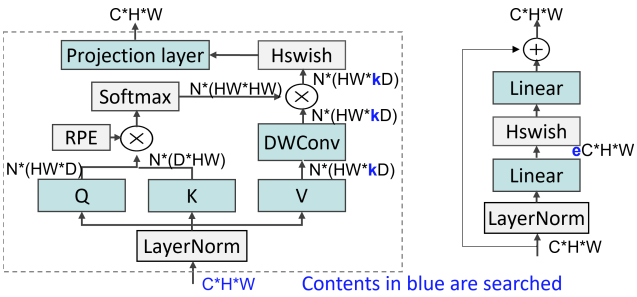

Figure 1. Mobile-friendly transformer block in our search space. Left: multi-head self-attention; for downsampling layer, we add a MobileNetv3 block with kernel size of 3×3 before LayerNorm; Right: MLP layer. During the supernet training, the input channel number C, H, W, V scale ratio k and MLP ratio e are elastic.

To develop an attention layer that is not only mobile-friendly (latency efficient) but also effective, we redesign the attention layer based on LeViT and NASViT attentions. Fig. 1 illustrates the architecture of our proposed mobile-friendly attention blocks. We remove the latency-expensive talking head, and increase the default QKV dimension to 16. We use latency-friendly Hswish as the activation function. Table 1 compares the latency on real-world mobile phones. Our optimized attention can accelerate the inference latency by $> 2\times$, which greatly improves the transformer block efficiency.

Table 1. Latency comparison of two Attention structures. Our optimized attention can achieves $> 2\times$ latency reduction than NASViT attention. HW=14×14, V scale=4, QK dimension=8.

Channel	Pixel 6 NASViT (ms)	Pixel 6 Our Attention (ms) ↓	Pixel 4 NASViT (ms)	Pixel 4 Our Attention (ms) ↓
64	3.66	**1.75** (-52.2%)	5.89	**2.89** (-50.9%)
96	6.52	**2.98** (-54.3%)	10.76	**5.18** (-51.8%)
112	8.14	**3.58** (-56.1%)	13.42	**6.22** (-53.7%)

A.2. Macro Space Design

Our search space includes 6 searchable stages, starting with a stem Conv layer for initial processing of input data, and ending with a classification head for final prediction. For the 6 searchable stages, we adopt a Convolution-Transformer (C-T) setting, where the first 2 stages are Conv stages and the later 4 stages are the Transformer stages.

Table 2. Results for 3 C-T ratio settings. Our paper adopts the 2:4 settings, which use 2 Conv stages and 4 Transformer stages.

C-T Stage Ratio	Top-1 Accuracy	
	500 MFLOPS	600 MFLOPS
2:4	**69.9%**	**70.4%**
3:3	69.1%	69.5%
4:2	67.7%	67.9%

In our experiments, we observe that the allocation of 2:4 Conv Transformer stages is the most effective in leveraging the benefits of the transformer while ensuring efficiency. Placing too many Transformers in the early stages (e.g., C-T ratio = 1:5) results in unacceptable model complexity due to the quadratic complexity of the self-attention module. On the other hand, incorporating too many Convolution stages (e.g., C-T ratio = 5:1) fails to leverage the advantages of the Transformer architecture, thereby hindering performance gains. Therefore, the three feasible solutions are 2:4, 3:3 and 4:2.

Table 2 compares the search space quality under the above three C-T ratio settings. Specifically, each supernet is trained for 100 epochs to reduce training cost. We conduct a FLOPs-constrained evolutionary search to find the best subnets under 500 and 600 MFLOPs. The results in Table 2 indicate that C-T ratio = 2:4 consistently outperforms other settings under the same FLOPs constraints, making it the preferred option for our large C-T ratio setting.

A.3. Overall FLOPs-Latency Relationship

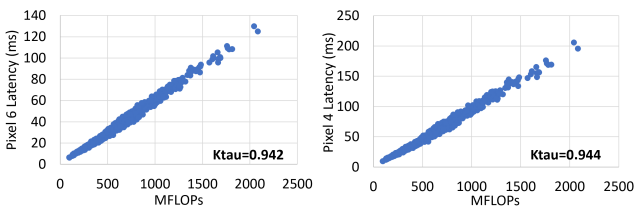

Figure 2. FLOPs-Latency relationship in the very large space.

Figure 2 depicts the FLOPs-Latency relationship in our very large space. We randomly sample 1000 models from the search space and calculate Kendall's tau coefficient. The results show a strong positive correlation between FLOPs and latency in our very large space, which serves as a basis for using FLOPs as a complexity metric in the main paper.

B. Transformer Stages Preference

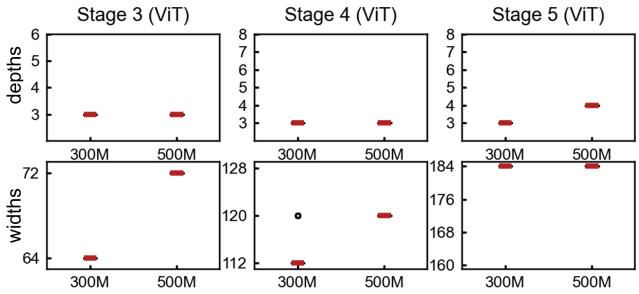

Figure 3. Boxplot visualization of best-searched 50 models from NASViT search space. The y-axis lists out the available choices for the corresponding search dimension. For small FLOPs regime, the top models choose the minimal depth of 3 in Stage 3, 4 and 5; top models choose the maximum width of 184 in Stage 5.

In Section 4, we introduce our proposed path preference rule, which is informed by the insights from recent studies [2, 3]. These studies have shown that in pure ViT models, the later Transformer stages tend to prioritize wider channels over more layers, as compared to CNN models.

To investigate whether this principle holds for hybrid models as well, we analyze the architectures of the best models generated by NASViT and make the following observation: *The ViT stage in hybrid models prefers wider widths and shallower depths.*

Fig. 3 visualizes the depths and channels of top 50 models. Specifically, we perform evolutionary search to explore 5k models and select top 50 with the highest accuracy for analysis. Surprisingly, even allowed to set a deeper depths, our results suggest that top models at 300M FLOPs choose the minimal depth choice of 3 in the 3 ViT stages. We also observe an interesting phenomenon in Stage 5. Both 300M and 500M top models choose the largest width choice in Stage 5. In summary, the top-performing hybrid models prefer wider widths and shallower depths in ViT stages.

These observations also suggest that the existing search space design for ViT stages are suboptimal. In our search space, we enlarge the maximum choice of widths and add smaller depths choices for ViT stages. To include tiny ViTs for weak devices, we also add many small choices for each search dimensions. As a result, our search space contains a huge number of 1.09×10^{17} candidate models, which is $10^7\times$ of previous search space.

C. Search Space Size Comparison

Table 3. Search space size comparison. *: we adopt stage-wise kernel sizes and expand ratios to compute search space size.

Search Space	type	Size*	Min model FLOPs	Max model FLOPs
OFA	CNN	3.6×10^9	88M	1005M
BigNAS	CNN	5.8×10^8	219M	1900M
NASViT	ViT	3.09×10^{10}	190M	1881M
Ours	ViT	1.09×10^{17}	37M	3191M

Search space size comparison. To accommodate both weak and strong mobile devices, we design a very large ViT search space that covers a wide range of models, from 37 to 3191 MFLOPs. Table 3 provides a quantitative comparison of search space sizes, showing that our search space is $10^7\times$ larger than typical search spaces used in two-stage NAS. This exponential increase in size presents significant optimization challenges, as discussed in the main paper.

FLOPs difference. Next, we discuss the impact of FLOPs difference in the very large space to the sandwich rule.

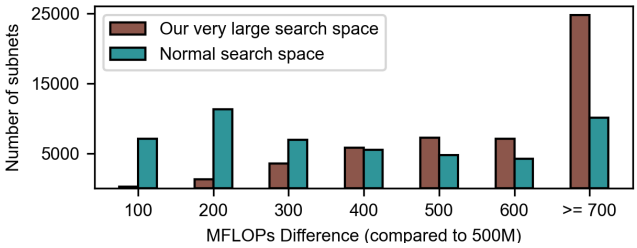

Figure 4. The sizes of two subnets sampled from a very large search space can vary greatly. The FLOPs differences are computed by randomly sampling 50k subnets from two search spaces.

As introduced in the main paper, the sandwich rule samples a minimum and maximum subnet, as well as two randomly selected subnets for supernet training at each step. While this technique is generally effective in a typical search space, it can lead to the sampling of subnets with significantly different FLOPs, which can cause gradient conflicts. As shown in Fig. 4, randomly sampling two subnets from our search space can easily lead to a FLOPs difference of more than 700M, where the gradient similarity is often close to 0.

D. Memory Bank Update Strategy.

We adopt the *Worst-Performing* strategy to replace the subnet in the memory bank. Namely, only the least accurate subnet in the memory bank will be replaced, if a subnet s_t^i at step t satisfies

$$\mathrm{ACC}(s_t^i) \geq \arg\min_{\alpha}\{\alpha | \alpha = \mathrm{ACC}(s), s \in \mathrm{B_i}\}, \quad (1)$$

the architecture α will be replaced, otherwise the memory bank will not be updated. And the accuracy $\mathrm{ACC}(\cdot)$ is ap-

proximated by the cross-entropy loss on a mini-batch.

E. Details of the Hierarchical Smallest Subnets (HSS) Set

To obtain the HSS set without prior knowledge, we first train a meta super-network without proposed method, using the very large search space. To save training costs, we only train the meta supernet 100 epochs.

We use this meta supernet as the accuracy indicators and conduct architecture search on it with two pre-defined complexity levels: 150 MFLOPs (referred as $\min_2$) and 300 MFLOPs (referred as $\min_3$). Accordingly, the space from the original smallest sub-network in the very large space (referred as $\min_1$) to $\min_2$ is used to search for weak mobile devices (e.g., Pixel1), the space from $\min_2$ to $\min_3$ is used to search for neutral mobile devices (e.g., Pixel4), and the remaining space ($\geq \min_3$) is used to search for strong mobile devices (e.g., Pixel6). Hence, the HSS set $\hat{\mathbf{S}} = \{\min_1, \min_2, \min_3\}$. We provide the configurations of $\min_2$ and $\min_2$ in Table 5.

F. Training Settings

We use AdamW optimizer with a cosine learning rate scheduler, the initial learning rate is set to $5e$-4. The total training epochs are set to 600, and the first 5 epochs are used to warm-up with a minimum $5e$-6 learning rate. We set the mixup alpha to 0.01 (rather than the default value 0.8), as well as the cutmix alpha to 0.01, and we do not adopt the autoaugment. We only adopt heavy regularization (e.g., weight decay of 0.05) for Transformer stages [4, 1]. All experiments use 16 NVIDIA V100 GPUs (32G GPU memory) and the PyTorch training system, the batch size is 96 per GPU.

For super-network hyper-parameters, the number of stochastic sub-networks M at every iteration is set to 3. We use the complexity levels = $\{50, 100, 200, 300, 400, 500, 700, 900, 1200\}$ MFLOPs. We set the size of each memory bank $\mathrm{B_i}$ to 150. The memory bank sampling probability is set to 0.2 initially and increases 0.15 every 100 epochs.

FLOPs table. In the supernet training process, our complexity-aware and performance-aware sampling methods involve selecting a subnet with a specific FLOPs level. However, due to the large search space, the cost of sampling a subnet under a FLOPs constraint can become expensive and slow down the training process. To reduce this cost, we construct a FLOPs table. Specifically, we randomly sample 150k subnets for each FLOPs complexity level, which enables us to quickly sample subnets during training.

Table 4. Comparison of subnets with inherited weights and fine-tuned (30 epochs) under different hyperparameters.

Model	Inherit supernet weights	Finetune lr=5e-5	Finetune lr=5e-6	Finetune lr=5e-7
ElasticViT-T2	73.8	72.4	73.6	73.4
ElasticViT-S2	78.6	76.4	77.8	77.9
ElasticViT-L3	80.0	78.2	79.5	79.5

G. Finetuning Experiments

We select three different-sized models and fine-tune them for an additional 30 epochs using different learning rates. We follow BigNAS to set learning rates 10x, 100x, and 1000x smaller than our supernet training rate of 5e-4. Table 4 shows that further fine-tuning does not improve accuracy. This suggests that our supernet training allows subnets to be well-trained within supernets, meaning that searched models do not require fine-tuning.

H. Visualization of Searched Models

The architectures of our searched model family are presented in Table 5. Notably, these top-performing models exhibit a characteristic of wider widths and shallower depths for ViT stages. This finding demonstrates the effectiveness of our proposed path preference rule.

Table 5. Architecture details. “C”, “D”, “K”, “E”, “V” and “M” represent channels, depths, kernel size (CNN stages), expansion ratio (CNN stages), V scale (Transformer stages) and MLP ratio (Transformer stages), respectively.

Model	$\min_2$	$\min_3$	T1	T2	S1	S2	M	L1	L2	L3
FLOPs	159	288	50	100	200	300	400	500	700	800
Resolution	176	224	128	160	192	224	224	256	256	256
MBv2	C=16 D=1 K=3 E=1	C=16 D=1 K=3 E=1	C=16 D=1 K=3 E=1	C=16 D=1 K=3 E=1	C=16 D=1 K=3 E=1	C=16 D=1 K=3 E=1	C=16 D=1 K=5 E=1	C=16 D=1 K=5 E=1	C=16 D=1 K=3 E=1	C=16 D=1 K=5 E=1
MBv2	C=24 D=3 K=3 E=3	C=24 D=3 K=3 E=3	C=24 D=2 K=3 E=3	C=24 D=2 K=3 E=3	C=24 D=3 K=3 E=3	C=24 D=3 K=3 E=3	C=32 D=3 K=3 E=3	C=24 D=3 K=5 E=3	C=24 D=3 K=3 E=6	C=32 D=3 K=3 E=5
MBv3	C=32 D=3 K=3 E=3	C=32 D=3 K=3 E=3	C=32 D=3 K=3 E=3	C=40 D=3 K=3 E=3	C=40 D=3 K=3 E=3	C=48 D=3 K=3 E=3	C=48 D=5 K=3 E=3	C=40 D=3 K=5 E=6	C=40 D=4 K=3 E=6	C=48 D=4 K=5 E=5
Transformer	C=48 D=2 V=2 M=2	C=64 D=2 V=2 M=2	C=64 D=1 V=2 M=2	C=48 D=1 V=2 M=3	C=64 D=2 V=2 M=2	C=64 D=2 V=2 M=2	C=64 D=2 V=2 M=2	C=64 D=2 V=2 M=4	C=80 D=2 V=2 M=3	C=80 D=2 V=2 M=4
Transformer	C=96 D=2 V=2 M=3	C=96 D=2 V=2 M=2	C=96 D=1 V=2 M=2	C=96 D=1 V=2 M=4	C=96 D=2 V=2 M=2	C=96 D=2 V=2 M=2	C=96 D=2 V=2 M=4	C=96 D=2 V=2 M=3	C=128 D=2 V=2 M=2	C=128 D=2 V=2 M=4
Transformer	C=176 D=2 V=2 M=2	C=192 D=2 V=2 M=2	C=176 D=1 V=2 M=3	C=176 D=2 V=2 M=3	C=192 D=2 V=2 M=3	C=192 D=2 V=2 M=2	C=192 D=2 V=2 M=4	C=192 D=2 V=2 M=4	C=256 D=2 V=2 M=3	C=256 D=2 V=2 M=3
Transformer	C=224 D=2 V=2 M=2	C=256 D=2 V=2 M=2	C=272 D=1 V=2 M=4	C=256 D=2 V=2 M=2	C=272 D=2 V=2 M=4	C=256 D=2 V=2 M=4	C=288 D=2 V=2 M=5	C=320 D=2 V=2 M=4	C=304 D=2 V=3 M=5	C=320 D=2 V=2 M=5

References

[1] 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 *Proc. of ICCV*, 2021. 3

[2] Namuk Park and Songkuk Kim. How do vision transformers work? In *Proc. of ICLR*, 2022. 2

[3] Maithra Raghu, Thomas Unterthiner, Simon Kornblith, Chiyuan Zhang, and Alexey Dosovitskiy. Do vision transformers see like convolutional neural networks? *Proc. of NeurIPS*, 2021. 2

[4] Hugo Touvron, Matthieu Cord, Matthijs Douze, Francisco Massa, Alexandre Sablayrolles, and Herve Jegou. Training data-efficient image transformers and distillation through attention. In *Proc. of ICML*, 2021. 3