Supplementary Material for "Network Space Search for Pareto-Efficient Spaces"

Min-Fong Hong, Hao-Yun Chen, Min-Hung Chen, Yu-Syuan Xu,
Hsien-Kai Kuo, Yi-Min Tsai, Hung-Jen Chen, and Kevin Jou
MediaTek Inc.
{romulus.hong, hao-yun.chen, mh.chen, Yu-Syuan.Xu}@mediatek.com

A. Details of Expanded Search Space

We visualize the detailed structure of *Expanded Search Space* in Figure S1. In addition to the stem with input channel $w_{in} = 3$ and the prediction network with c output classes, the body network consists of 3 stages and each stage is built by stacking a sequence of identical blocks. We select the basic residual block [2] as our building block, which is composed of two consecutive 3×3 convolutions along with a residual connection, illustrated in Figure S2. For each stage i, the degrees of freedom include the number of blocks d_i and block width w_i. We consider the settings of $d_i \leq 16$ and $w_i \leq 512$, resulting in $(16 \times 512)^3 \approx 10^{12}$ possible networks in *Expanded Search Space*.

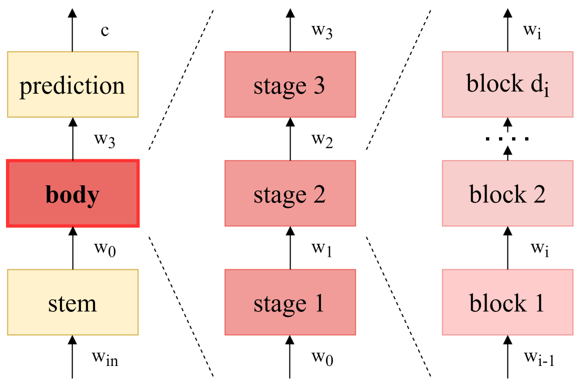


Figure S1: Network structure in *Expanded Search Space*. Each network consists of a stem (3×3 convolution with $w_0 = 16$ output channels), the network body, and a prediction network (global average pooling followed by a fully connected layer) predicting output classes. The network body is composed of 3 stages where each stage is comprised of a sequence of identical blocks. The block parameter, depth d_i, will be discovered by our proposed NSS framework.

B. Derivation of NSS Evaluation

As we assume architectures are uniformly sampled from the network space, which is sampled by Gumbel-Softmax [4], during the searching process of Network Space Search (NSS), we here provide the theoretical proof for this assumption. The proof is derived as follows:

$$\begin{aligned}\mathbf{E}_{\mathcal{A}\sim P_\Theta}[\mathcal{L}(\mathcal{A})] &= \sum_{\mathcal{A}_i \in \mathbb{A}} \mathbf{E}[\mathcal{L}(\mathcal{A})|\mathcal{A}_i] P_\Theta(\mathcal{A}_i) \\ &= \sum_{\mathcal{A}_i \in \mathbb{A}} \frac{\sum_{\alpha\in\mathcal{A}_i}\mathcal{L}(\alpha)}{|\mathcal{A}_i|} P_\Theta(\mathcal{A}_i) \\ &= \sum_{\mathcal{A}_i \in \mathbb{A}} \mathbf{E}_{\alpha\sim U,\alpha\in\mathcal{A}_i}[\mathcal{L}(\alpha)] P_\Theta(\mathcal{A}_i) \\ &= \mathbf{E}_{\mathcal{A}\sim P_\Theta}[\mathbf{E}_{\alpha\sim U,\alpha\in\mathcal{A}_i}[\mathcal{L}(\alpha)]]\end{aligned} \tag{S1}$$

where U denotes the uniform distribution. Since the set of $\mathcal{A}$ is finite, the original expectation $\mathbf{E}[\mathcal{L}(\mathcal{A})]$ can be expanded into the summation for all possible $\mathcal{A}_i$. The expected loss $\mathbf{E}[\mathcal{L}(\mathcal{A})|\mathcal{A}_i]$ conditioned on $\mathcal{A}_i$ can be further rewritten based on the conditional expectation by evaluating the loss of each α in $\mathcal{A}_i$. Moreover, the division of the cardinality of $\mathcal{A}_i$ can be viewed as each $\mathcal{L}(\alpha)$ multiplying the same probability of $\frac{1}{|\mathcal{A}_i|}$. Therefore, the conditional expectation is equal to evaluating the expected loss of each α uniformly sampled from $\mathcal{A}_i$, and our assumption is proved.

C. Efficiency Improvement Techniques

We adopt several techniques to improve the efficiency of NSS, which can be divided into two aspects, *weight sharing techniques* and *improving super network weights*. We here provide more implementation details of these techniques.

Weight Sharing Techniques. As *Expanded Search Space* includes a wide range of possible network depths and widths, simply enumerating each candidate is memory prohibited for either the kernels with various channel sizes or the stages with various block sizes. We first adopt the *chan-*

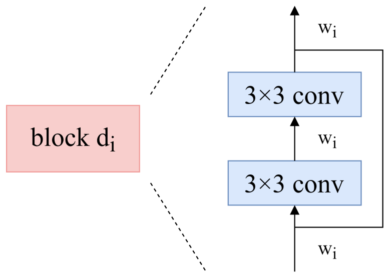


Figure S2: Basic residual block. We select the basic residual block as our building block. Each residual block consists of two 3×3 convolutions, and each convolution is followed by BatchNorm [3] and ReLU. The block parameter, width w_i, will be discovered by our proposed NSS framework.

nel masking technique [7] to efficiently search for channel sizes. With only constructing a single super kernel with the largest possible number of channels, smaller channel sizes $w \leq w_{max}$ can be simulated by retaining the first w channels and zeroing out the remaining ones. The masking procedure achieves the lower bound of memory consumption and more importantly, is differential-friendly. Moreover, the same idea of sharing a portion of the super component among various sizes can be applied to the search for block sizes. A single deepest stage with the largest possible number of blocks is constructed, and shallower block sizes $d \leq d_{max}$ are simulated by taking the output of d^{th} block as the output of the corresponding stage.

Improving Super Network Weights. Ensuring super network weights being sufficiently well-trained provides reliable performance estimation of each candidate and leads the searching process to discover promising results. Therefore, we adopt several *warmup* techniques [1] to improve the quality of super network weights. We update network weights only and disable the searching in the first 25% of epochs since network weights are not able to provide stable signals to appropriately guide the searching process in the early period. In the warmup phase, despite the selected channel size in each sampling, *all* the channels of super kernels are randomly enabled with a probability that is linearly annealed down to 0 over the phase. This filter warmup technique counteracts the side effects of weight sharing that the forepart of the super kernel is always trained across different sampled channel sizes while right-most channels are less updated. For the same reason, we introduce warmup to block search where all the blocks are as well randomly enabled in the warmup phase to guarantee deeper blocks are equally trained with shallower blocks.

D. Additional Experimental Results

D.1. Hyperparameter Settings

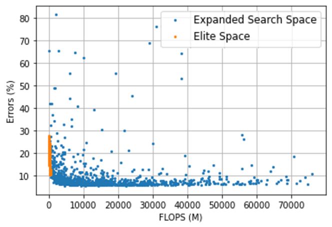
We mostly follow the hyperparameter settings in DARTS [5], and therefore we only list the adjustments made for our experiments here. The searching process lasts for 50 epochs where the first 15 ones are reserved for warmup. The temperature for Gumbel-Softmax is initialed to 5 and linearly annealed down to 0.001 throughout the searching process. The batch size is set to 64 to fit in 4 1080Ti GPUs. The search cost for a single run of the NSS process is roughly 0.5 days under the above settings, and the subsequent NAS performed on *Expanded Search Space* and *Elite Spaces* requires 0.5 days and merely several hours to complete a searching process, respectively.

D.2. Elite Spaces under More FLOPs Regimes

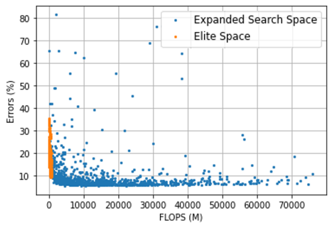
In addition to the FLOPs constraints adopted in the main context, we here provide experimental results under more FLOPs regimes that are completely aligned with the settings in [6]. Following the same evaluation procedure, *Elite Spaces* are illustrated in Figures S3 and S4. Our NSS method is demonstrated to sustainably deliver superior *Elite Spaces* aligned with the Pareto front even for the most rigorous constraint (i.e. 200MF) or the largest complexity (i.e. 32GF).

D.3. Elite Spaces Served as NAS Search Spaces

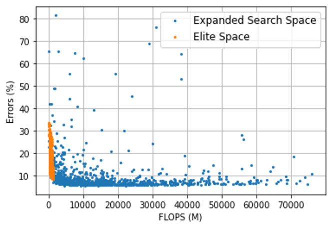
Next, we perform NAS on the discovered *Elite Spaces* from Section D.2 and directly on *Expanded Search Space*, targeting several FLOPs constraints mentioned above. The results are listed in Table 1. It is observed that our approach outperforms the baseline in obtaining superior networks and fulfilling the constraints more rigorously across all FLOPs regimes. Averagely, *Elite Space* achieves a lower error rate (4.13% vs. 4.76%), lower deviation (2.8% vs. 5.6%), and 97.7% fewer samples required to find a satisfactory network (5 vs. 215.6) than the baseline in CIFAR-10. The same trend can also be observed in CIFAR-100 (2.98% lower error rate, 4.3% lower deviation, and 96.0% fewer required samples). It is worth noting that the improvement is more obvious under extremely strict constraints. For example, under the constraint of 400MF constraint in the CIFAR-100 dataset, the architecture obtained from corresponding *Elite Space* achieves a 22.94% error rate and 0.5% deviation from the constraint within merely 5 samples. On the contrary, the baseline requires 661 samples to reach the constraint with a higher deviation (5.0%) while still delivering worse performance (29.86% error rate). Therefore, our NSS framework is demonstrated to benefit the performance of NAS by delivering Pareto-efficient *Elite Spaces*.



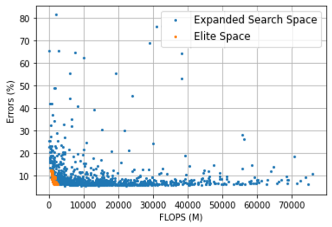
(a) Complexity 200MF



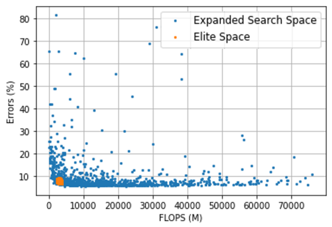
(b) Complexity 400MF



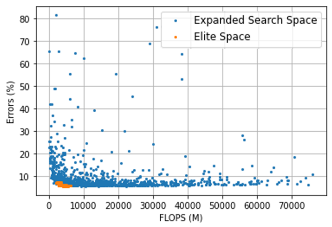
(c) Complexity 600MF



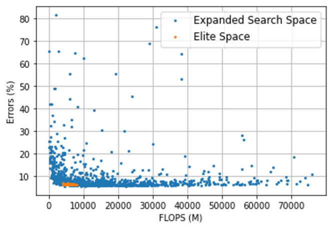
(d) Complexity 1.6GF



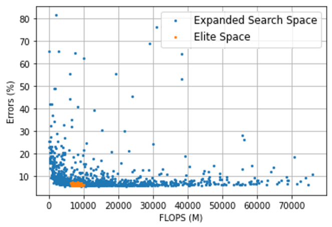
(e) Complexity 3.2GF



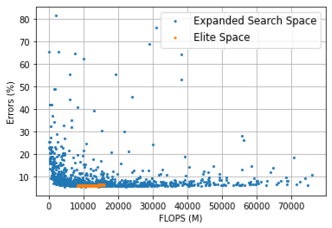
(f) Complexity 4GF



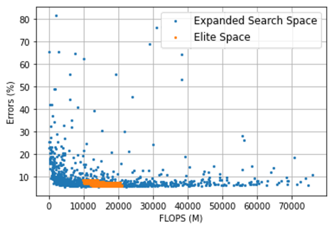
(g) Complexity 6.4GF



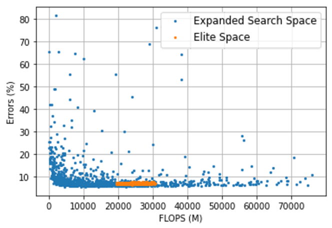
(h) Complexity 8GF



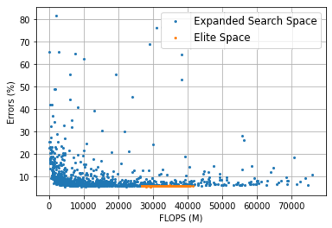
(i) Complexity 12GF



(j) Complexity 16GF



(k) Complexity 24GF

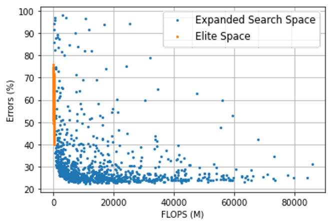


(l) Complexity 32GF

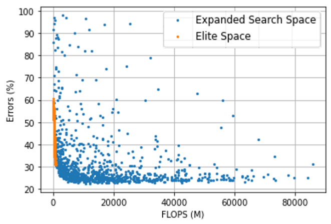
Figure S3: *Elite Spaces* evaluation on CIFAR-10.

	CIFAR-10						CIFAR-100					
	Elite Space			*Expanded Search Space*			*Elite Space*			*Expanded Search Space*		
Complexity	FLOPs (|Δ%|)	#samples	Error	FLOPs (|Δ%|)	#samples	Error	FLOPs (|Δ%|)	#samples	Error	FLOPs (|Δ%|)	#samples	Error
CX 200 MF	183 MF (8.5%)	**5**	**4.98**	213 MF (**6.5%**)	1228	6.83	198 MF (**1.0%**)	**5**	**27.32**	209 MF (4.5%)	470	33.21
CX 400 MF	385 MF (3.8%)	**5**	**5.02**	405 MF (**1.5%**)	976	6.55	398 MF (**0.5%**)	**5**	**22.94**	420 MF (5.0%)	661	29.86
CX 600 MF	571 MF (**4.8%**)	**5**	**4.57**	658 MF (9.7%)	117	6.08	568 MF (5.3%)	**5**	**22.22**	628 MF (**4.7%**)	147	23.48
CX 1.6 GF	1.6 GF (2.3%)	**5**	**3.99**	1.6 GF (**1.1%**)	45	4.33	1.6 GF (**0%**)	**5**	**20.67**	1.6 GF (1.9%)	84	24.45
CX 3.2 GF	3.1 GF (**3.7%**)	**5**	**3.94**	3.4 GF (8.1%)	86	4.51	3.3 GF (**2.7%**)	**5**	**20.14**	2.9 GF (9.6%)	31	25.06
CX 4 GF	3.9 GF (**3%**)	**5**	**3.85**	3.8 GF (4.8%)	57	4.16	4.2 GF (**6%**)	**5**	**20.21**	4.3 GF (7.1%)	9	22.39
CX 6.4 GF	6.4 GF (**1.4%**)	**5**	**3.92**	5.9 GF (7.6%)	18	4.08	6.3 GF (**1.8%**)	**5**	**19.07**	5.8 GF (8.8%)	17	20.74
CX 8 GF	8 GF (**0.4%**)	**5**	**4.13**	7.3 GF (8.7%)	13	4.09	8 GF (**0.5%**)	**5**	**18.77**	7.5 GF (5%)	18	21.41
CX 12 GF	11.8 GF (**1.4%**)	**5**	**3.98**	8.7 GF (6.5%)	14	4.23	12.4 GF (**3.3%**)	**5**	**19.21**	9.3 GF (7.4%)	15	21.47
CX 16 GF	15.8 GF (**1.6%**)	**5**	**3.82**	14.7 GF (8.1%)	12	3.93	16.2 GF (**1.3%**)	**5**	**19.16**	14.5 GF (9.4%)	12	21.21
CX 24 GF	23.8 GF (**0.7%**)	**5**	**3.65**	23.8 GF (0.8%)	15	4.53	23.9 GF (**0.3%**)	**5**	**19.09**	22.2 GF (7.5%)	11	20.71
CX 32 GF	31.3 GF (**2.2%**)	**5**	**3.78**	33.4 GF (4.1%)	6	3.86	32.5 GF (**1.6%**)	**5**	**19.11**	30.6 GF (4.3%)	9	19.65
Average	N/A (**2.8%**)	**5**	**4.13**	N/A (5.6%)	215.6	4.76	N/A (**2.0%**)	**5**	**20.66**	N/A (6.3%)	123.7	23.64

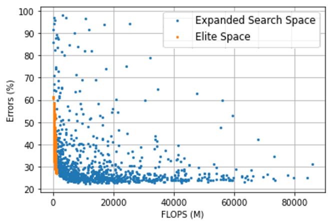
Table 1: The comparison of NAS results performed on *Elite Spaces* and *Expanded Search Space*.



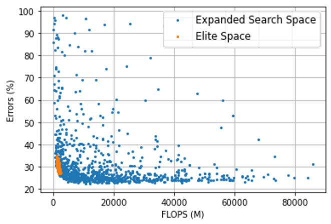
(a) Complexity 200MF



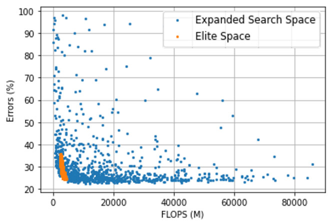
(b) Complexity 400MF



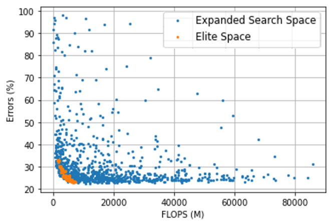
(c) Complexity 600MF



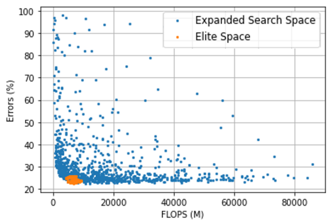
(d) Complexity 1.6GF



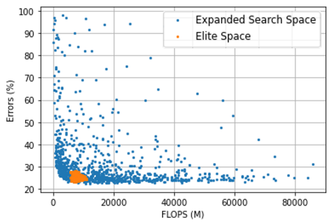
(e) Complexity 3.2GF



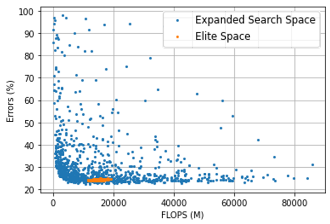
(f) Complexity 4GF



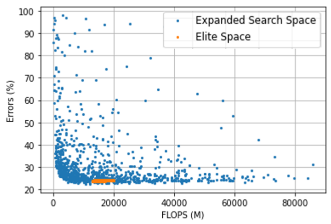
(g) Complexity 6.4GF



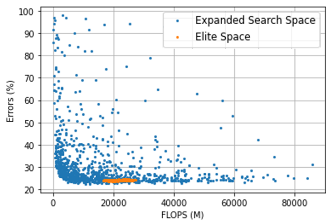
(h) Complexity 8GF



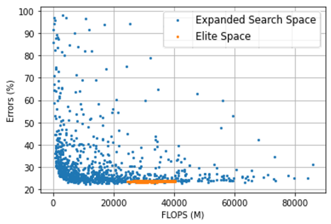
(i) Complexity 12GF



(j) Complexity 16GF



(k) Complexity 24GF



(l) Complexity 32GF

Figure S4: *Elite Spaces* evaluation on CIFAR-100.

References

[1] Gabriel Bender, Hanxiao Liu, Bo Chen, Grace Chu, Shuyang Cheng, Pieter-Jan Kindermans, and Quoc V. Le. Can weight sharing outperform random architecture search? an investigation with tunas. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pages 14311–14320, 2020. 2

[2] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pages 770–778, 2016. 1

[3] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In Proceedings of the International Conference on Machine Learning (ICML), pages 448–456, 2015. 2

[4] Eric Jang, Shixiang Gu, and Ben Poole. Categorical reparameterization with gumbel-softmax. In International Conference on Learning Representations (ICLR), 2017. 1

[5] Hanxiao Liu, Karen Simonyan, and Yiming Yang. Darts: Differentiable architecture search. In International Conference on Learning Representations (ICLR), 2019. 2

[6] Ilija Radosavovic, Raj Prateek Kosaraju, Ross B. Girshick, Kaiming He, and Piotr Dollár. Designing network design spaces. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pages 10425–10433, 2020. 2

[7] Alvin Wan, Xiaoliang Dai, Peizhao Zhang, Zijian He, Yuandong Tian, Saining Xie, Bichen Wu, Matthew Yu, Tao Xu, Kan Chen, Peter Vajda, and Joseph E. Gonzalez. Fbnetv2: Differentiable neural architecture search for spatial and channel dimensions. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pages 12962–12971, 2020. 2