A. Training settings

We use the sandwich sampling rule and always train the smallest and biggest sub-networks in the search space as regularization (see Eqn. (3)). We set $n = 2$ in Eqn. (7). This way, at each iteration, a total of 4 sub-networks are evaluated. We use in-place knowledge distillation, i.e., all smaller sub-networks are supervised by the largest sub-network. To handle different input resolutions, we always fetch training patches of a fixed size (e.g., 224x224 on ImageNet) and then rescale them to our target resolution with bicubic interpolation.

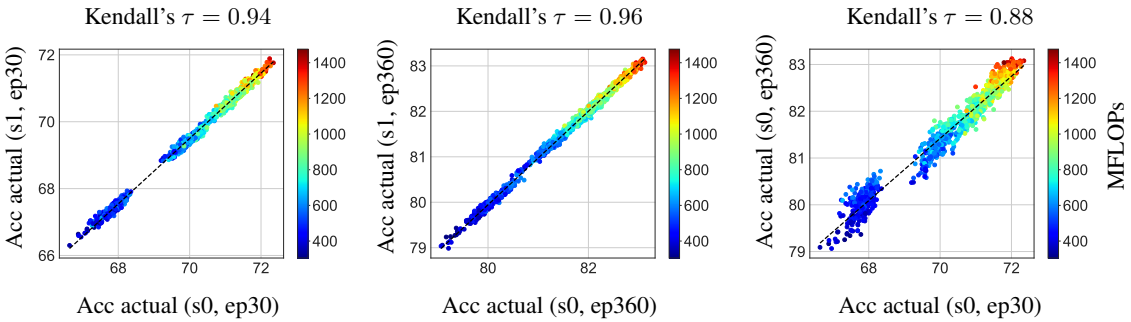
We use SGD with a cosine learning rate decay. All the training runs are conducted with 64 GPUs and the mini-batch size is 32 per GPU. The base learning rate is set as 0.1 and is linearly scaled up for every 256 training samples. We use AutoAugment [9] for data augmentation and set label smoothing coefficient to 0.1. Unless specified, we train the models for 360 epochs. We use momentum of 0.9, weight decay of 10^{-5}, dropout of 0.2 after the global average pooling layer, and stochastic layer dropout of 0.2. We don't use synchronized batch-normalization. Following [43], we only enable weight decay and dropout for training the largest DNN model. All other smaller sub-networks are trained without regularization.

B. Robustness of two-stage NAS

We also study the robustness and stability of stage 1 constraint-free NAS pre-training w.r.t. different data partitions, initializations and training epochs.

We follow the experimental setting in settings 4.3. Specifically, 1) we randomly partitioned the original ImageNet training set into 90% for training and 10% for testing. We then train on the subsampled training set. 2) After training, we randomly sample 1024 sub-networks and evaluate their performance on their corresponding testing data partition.

In Figure 7, we show that our two-stage NAS training is quite robust, achieving reproducible results across a variety of training settings. Specifically, in Figure 7 (a), we terminate early at epoch 30, the Kendall's tau value is 0.94 between two different runs. We further train for 360 epochs, in Figure 7 (b), we observe a high rank correlation of 0.96 between different trials. Furthermore, in Figure 7 (c), we show the performance measured at epoch 30 also correlates well with the performance measured at the end of training. The rank correlation is 0.88. Our results are in alignment with the findings in FairNAS [8].



(a) Rank correlation at *ep30* (b) Rank correlation at *ep360* (c) Rank correlation wrt training epochs

Figure 7. An illustration of robustness of stage 1 training. S0 and s1 denote random data partition with seed 0 and seed 1, respectively. Ep30 and ep360 denote 30 training epochs and 360 training epochs, respectively.

C. Sampling efficiency

Our attentive sampling requires to sample architectures under different FLOPs constraints. Given a randomly drawn FLOPs constraint, naive uniformly sampling requires of an average of 50,878 trials to sample an architecture that satisfies the constraint due to the enormous size of the search space. In section 4.2, we construct a proposal distribution $\hat{\pi}(\alpha \mid \tau)$ in an offline mode to accelerate this sampling process. In Figure 8, we show the average sampling trials for sampling targeted architectures under constraints is about 12 by sampling from $\hat{\pi}$, hence computationally extremely efficient.

D. Comparisons of search space

Our search space is defined in Table 2. Note that our search space is adapted from FBNetV3 [10]. Compared to the search space used in BigNAS [43], our search space contains more deeper and narrower sub-networks.

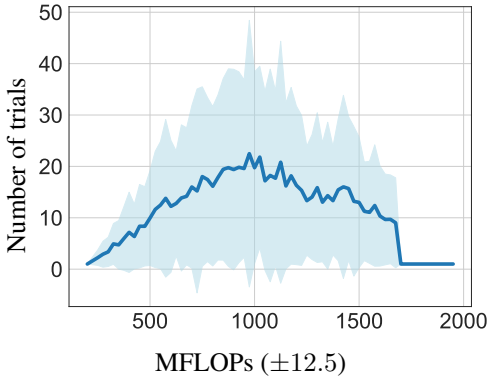


Figure 8. An illustration of mean of the number of trials to sample architectures under constraints along with its standard derivation.

We compare the uniform sampling strategy performance on both search spaces. Specifically, we follow the evaluation flow described in section 4.2. The search space proposed in [3] is not evaluated here as its training pipeline requires complicated progressive network shrinking and carefully tuned hyper-parameters for each training stage.

Block	*Width*	*Depth*	*Kernel size*	*Expansion ratio*	*SE*
Conv	{16, 24}	-	3	-	-
MBConv-1	{16, 24}	{1,2}	{3, 5}	1	N
MBConv-2	{24, 32}	{3, 4, 5}	{3, 5}	{4, 5, 6}	N
MBConv-3	{32, 40}	{3, 4, 5, 6}	{3, 5}	{4, 5, 6}	Y
MBConv-4	{64, 72}	{3, 4, 5, 6}	{3, 5}	{4, 5, 6}	N
MBConv-5	{112, 120, 128}	{3, 4, 5, 6, 7, 8}	{3, 5}	{4, 5, 6}	Y
MBConv-6	{192, 200, 208, 216}	{3, 4, 5, 6, 7, 8}	{3, 5}	6	Y
MBConv-7	{216, 224}	{1, 2}	{3, 5}	6	Y
MBPool	{1792, 1984}	-	1	6	-
Input resolution	{192, 224, 256, 288}				

Table 2. An illustration of our search space. MBConv refers to inverted residual block [28]. MBPool denotes the efficient last stage [17]. SE represents the squeeze and excite layer [19]. *Width* represents the channel width per layer. *Depth* denotes the number of repeated MBConv blocks. *Kernel size* and *expansion ratio* is the filter size and expansion ratio for the depth-wise convolution layer used in each MBConv block. We use swish activation.

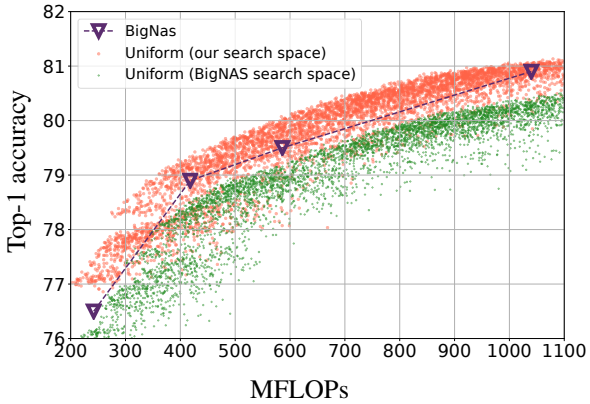


Figure 9. Comparison of the effectiveness of search space.

E. Comparisons of training and search time

Overall, our method yields a computationally efficient NAS framework: 1) compared with RL-based solutions [e.g., 34, 45], our method builds on weight-sharing [35], alleviating the burden of training thousands of sub-networks from scratch or

on proxy tasks; 2) when comparing with conventional differentiable NAS approaches, e.g., DARTS [23], ProxylessNAS [4], etc, our method simultaneously finds a set of Pareto optimal networks for various deployment scenarios, e.g., N different FLOPs requirements, with just one single training. While typical differentiable NAS solutions need to repeat the NAS procedure for each deployment consideration; 3) no fine-tuning or re-training is needed for our method. The networks can be directly sampled from the weight-sharing graph in contrast to Once-for-All (OFA) etc, which usually requires to fine-tune the sub-networks.

As different methods use different hardware for training, it makes wall-clock time comparison challenging. We report the total number of training epochs performed on ImageNet by each method in Table 3. For our method, we train for 360 epochs and our training strategy requires back-propagating through 4 sub-networks at each iteration, which is roughly about $4\times$ slower in per batch time. As we can see from Table 3, our method yields the lowest training cost.

Model	Total training epochs on ImageNet (N=40)
MnasNet [34]	40,000N = 1,600k
ProxylessNAS [4]	200N (weight-sharing graph training) + 300N (retraining) = 20k
OFA [3]	590 (weight-sharing graph training) + 75N (finetuning) = 3.59k
AttentiveNAS (ours)	360 $\times$4 (weight-sharing graph training) = 1.44k

Table 3. Overview of training cost. Here N denotes the number of deployment cases. Following OFA, we consider $N = 40$. Similar to OFA, our method also includes an additional stage of evolutionary search (evaluation with fixed weights, no back-propagation), which amounts to less than 10% of the total training time.

F. Additional results on inference latency

Our attentive sampling could be naturally adapted for other metrics, e.g., latency. In this work, we closely follow the conventional NAS evaluation protocols in the literature and report the accuracy vs. FLOPs Pareto as examples to demonstrate the effectiveness of our method. In table 4, we use GPU latency as an example and provide additional latency comparisons on both 2080 Ti and V100 GPUs. Compared with EfficientNet models [35], our model yield better latency vs. ImageNet validation accuracy trade-offs.

Model	Batch-size	2080 Ti (ms)	V100 (ms)	Top-1
Efficientnet (B0)	128	21.51$\pm$ 0.27	13.13$\pm$ 0.30	77.3
AttentiveNAS-A1	128	**19.13$\pm$ 0.26**	**12.32$\pm$ 0.26**	**78.4**
Efficientnet (B1)	128	28.87$\pm$ 0.45	19.71 $\pm$ 0.40	80.3
AttentiveNAS-A6	128	**23.43$\pm$ 0.37**	**15.99$\pm$ 0.33**	**80.7**

Table 4. Inference latency comparison.

G. Transfer learning results

We evaluate the transfer learning performance of our AttentiveNAS-A1 and AttentiveNAS-A4 model on standard benchmarks, including Oxford Flowers [24], Stanford Cars [22] and Food-101 [2].

Specifically, we closely follow the training settings and strategies in [20], where the best learning rate and the weight decay are searched on a hold-out subset (20%) of the training data. All models are fine-tuned for 150 epochs with a batch size of 64. We use SGD with momentum of 0.9, label smoothing of 0.1 and dropout of 0.5. All images are resized to the size used on ImageNet. As we can see from Table 5, our models yield the best transfer learning accuracy.

Model	MFLOPs	Oxford Flowers	Stanford Cars	Food-101
EfficientNet-B0	390	96.9	90.8	87.6
AttentiveNAS-A1	**279**	**97.4**	**91.3**	**88.1**
EfficientNet-B1	1050	97.6	92.1	88.9
AttentiveNAS-A6	**709**	**98.6**	**92.5**	**89.6**

Table 5. Results (%) on transfer learning.