A. Additional Notations

We use two additional notations for pseudocode description: *(i)* $\mathcal{S}_{id}$ denotes a set of sub-supernets that is split by id numbers of edges. *(ii)* $\mathcal{S}_{id}^n$ denotes the n^{th} sub-supernet in $\mathcal{S}_{id}$.

B. End-to-end Pipeline Pseudocode

Below we list the pseudocode for the end-to-end split and training pipeline in Algorithm 1, the pseudocode for random split the one-shot model into sub-supernets in Algorithm 2, and the pseudocode for training (sub-)supernets in Algorithm 3.

Algorithm 1 (Sub)-supernets split and training

```
1: $\mathcal{S}_0 = \{\mathcal{S}\}$
2: define global $T \leftarrow TIME\_BUDGET$
3: Train($\mathcal{S}$, NONE)
4: $\mathcal{S}_0 \leftarrow \mathcal{S}$
5: $id \leftarrow 0$
6: while total time $< T$ do
7:     $j \leftarrow random(0, \#N)$
8:     $i \leftarrow random(0, j)$
9:     $\mathcal{S}_{id+1} \leftarrow$ RandomSplit($\mathcal{S}_{id}, E_{ij}$)
10:    for $n = 1 \rightarrow sizeof(\mathcal{S}_{id+1})$ do
11:        Train($\mathcal{S}_{id+1}^n, \mathcal{S}_{id}^{'}$)
12:    end for
13:    $id \leftarrow id + 1$
14: end while
```

Algorithm 2 RandomSplit($\mathcal{S}_{id}, E_{ij}$)

```
1:     $\mathcal{S}_{new} \leftarrow$ split $\mathcal{S}_{id}$ to $m$ sub-supernets given $m$ operations
2:     return $\mathcal{S}_{new}$
```

Algorithm 3 Train(s, $parent$)

```
1: if $parent$ IS NOT NONE then
2:     $W_s \leftarrow W_{parent}$
3: end if
4: While $s$ NOT CONVERGE do
5:     forward($s$)
6:     backward($s$)
7: end While
```

C. Experiment Setup for Section 2

Each architecture was trained for 150 epochs with a batch size of 128. The initial channel is 16. We used the SGD optimizer with an initial learning rate of 0.025, followed by a cosine learning rate schedule through the training. We set the momentum rate to 0.9 and a weight decay of 3×10^{-4}. The training setup of supernet and sub-supernets is consistent with architecture candidates. These experiments ran on 50 P100 GPUs.

D. Experiment Setup for Section 4

(Sub-)supernet Training Setup for NasBench-201. Each architecture was trained for 200 epochs with 256 batch size. The initial channel is 16. We used the SGD optimizer with an initial learning rate of 0.1, followed by a cosine learning rate schedule through the training. The momentum rate was set to 0.9. We used a weight decay of 5×10^{-4} and a norm gradient clipped at 5. The cutout technique was not used in training. The supernet training setup is consistent with architecture candidates. For supernet training, we changed the initial learning rate to 0.025 and total epochs to 300. The batch size is 128, and the weight decay was set to 1×10^{-4}. Each sub-supernet approximately took 40-50 epochs to converge after transfer learning. For each NAS algorithm, we used the same setup as described in the NasBench-201 (Dong & Yang, 2020). We used 6 P100 GPUs to train the supernet and 5 sub-supernets.

Search Setup for DARTS on CIFAR-10. We used the same search space and training setup as described in the original DARTS paper (Liu et al., 2019b). Specifically, the available operations in the search space include 3 x 3 and 5 x 5 separable convolutions, 3 x 3 and 5 x 5 dilated separable convolutions, 3 x 3 max pooling, 3 x 3 average pooling, identity, and zero. We trained 8 cells using DARTS for 50 epochs, with batch size 64 (for both the training and validation sets). The initial number of channels was set to 16. Each sub-supernet took 5-20 epochs to converge. We used the momentum SGD optimizer with an initial learning rate of 0.025, followed by a cosine learning rate schedule through the training. We used a momentum rate of 0.9 and a weight decay of 3×10^{-4}. This experiment ran on 10 P100 GPUs for training both supernet and sub-supernets.

We trained the network for 1500 epochs using a batch size of 128 and use a momentum SGD optimizer with an initial learning rate of 0.025, followed by a cosine learning rate schedule through the training. We use weight decay as the regularization.

Search Setup for DARTs on PTB. The search space and the training setup of (sub)supernets are identical to DARTS (Liu et al., 2019b). Concretely, both the embedding and the hidden sizes were set to 300. We used 6 P100 GPUs to train both the supernet and 5 (sub)supernets. Each (sub)supernet was trained for 50 epochs using SGD without momentum, with a learning rate of 20. The batch size was set to 256, and the weight decay was set to 3×5^{-7}. We applied a variational dropout of 0.2 to word embeddings, 0.75 to the cell input, and 0.25 to all the hidden nodes. We also applied a dropout rate of 0.75 to the output layer.

Search Setup for ImageNet (Gong et al., 2019). For proxylessNas, we exactly keep the same search pipeline with original paper (Cai et al., 2019). We randomly sample 50,000 images from the training set as a validation set during the architecture search. For our few-shot NAS, we split 3 sub-supernets. The (sub)supernets parameters are updated using the Adam optimizer with an initial learning rate of 0.001. The (sub)supernets are trained on the remaining training images with batch size 256. For once for all NAS, the search setup is also consistent with the original OFA (Cai et al., 2020). In specific, we use the same architecture space as MobileNetV3 (Howard et al., 2019); for supernet training, we use the standard SGD optimizer with Nesterov momentum 0.9, and weight decay is set to 3×10^{-5}. The initial learning rate is 2.6, and we use the cosine schedule for learning rate decay. We split 5 sub-supernets. The (sub)supernets are trained for 180 epochs with batch size 2048 on 64 32G V100 GPUs.

Search Setup for AutoGAN (Gong et al., 2019). Our search and training settings were identical to AutoGAN (Gong et al., 2019), which followed spectral normalization GAN (Miyato et al., 2018a) when training the (sub-)-supernets. We split the supernet (shared GAN in (Gong et al., 2019)) into 3 sub-supernets. The learning rate of both generator and discriminator was set to $2e^{-4}$. We used the hinge loss and an Adam optimizer. The batch size of the discriminator was 64, and the generator was 128. The initial learning rate was set to $3.5e^{-4}$. The AutoGAN searched for 90 iterations for one supernet. For each iteration, the shared GAN (supernet) was trained for 15 epochs, and the controller was trained for 30 steps. After the shared GAN (supernet) was trained, we transferred the weight to each sub-supernets and trained them for 12 epochs. We trained the controller with 30 steps. The discovered architectures were trained for 50,000 generator iterations. We used 4 P100 GPUs in this experiment.

E. One-shot NAS v.s. Few-shot NAS by Robust DARTS (Zela et al., 2020a)

Table 7. Few-shot Robust DARTS vs. One-shot Robust DARTS over 4 Search Space

Method	Space	Top 1 Acc(%)
one-shot	s1	96.49
few-shot	s1	**96.81**
one-shot	s2	96.22
few-shot	s2	**96.55**
one-shot	s3	97.19
few-shot	s3	**97.28**
one-shot	s4	95.60
few-shot	s4	**96.30**

We use our few-shot NAS with Robust DARTS searching architectures over 4 different search spaces defined by the original paper (Zela et al., 2020a). For table 7, we can see that the accuracy of architectures searched by our few-shot is significantly better than one-shot over all 4 search spaces. Our training setup is strictly the same as its original paper.