Supplementary Materials for
GreedyNAS: Towards Fast One-Shot NAS with Greedy Supernet

A. Details of evolutionary searching in Section 4.2

We present the details of our adopted NSGA-II [1] evolutionary algorithm in the following Algorithm 3. In our experiment, population size $N_{pop} = 50$ and number of generations $T = 20$.

Algorithm 3 Evolutionary Architecture Search

Input: supernet $\mathcal{N}$, candidate pool $\mathcal{P}$, population size N_{pop}, number of generations T, validation data $\mathcal{D}_{val}$, constraints $\mathcal{C}$.
Output: architecture with highest validation accuracy under constraints.

Initialize populations P_0 with $\mathcal{P}$ so that $|P_0| = N_{pop}$ and P_0 satisfies constraints $\mathcal{C}$.
$E = \emptyset$; # evaluation set E which stores all evaluated architectures with accuracy
for $i = 0, 1, ..., T-1$ **do**
  $Q_i = make\text{–}new\text{–}pop(P_i)$;
    # generate offspring population Q_i using binary tournament selection, recombination, and mutation operators
  $R_i = P_i \cup Q_i$;
  $F_i = fast\text{–}non\text{–}dominated\text{–}sort(R_i)$; # generate all nondominated fronts of R_i
  $P_{i+1} = \emptyset$ and $j = 0$;
  while $|P_{i+1}| + |F_i| \leq N_{pop}$ **do**
    $crowding\text{–}distance\text{–}assignment(F_i)$; # calculate crowding-distance in F_j
    $P_{i+1} = P_{i+1} \cup F_j$;
    $j = j + 1$;
  end while
  $E_i = evaluation\text{–}architecture(F_j, D_{val}, \mathcal{C})$; # evaluate architecture with constraints and validation data
  $E = E \cup E_i$ # extend E_i to E
  $Sort(F_j, E_i)$; # sort in descending order using E_i
  $P_{i+1} = P_{i+1} \cup F_j[1 : (N_{pop} - |P_{i+1}|)]$; # choose the first $(N_{pop} - |P_{i+1}|)$ elements of F_j
  $Q_{i+1} = make\text{–}new\text{–}pop(P_{i+1})$; # make new population with constraints
end for
return architecture with highest accuracy in E

B. More Experimental Results

B.1. Details of (augmented) search space

The macro-structure of supernet is presented in Table 5, where each operation choice for Choice Block is attached in Table 6.

Table 5: Macro-structure of supernet. "input" indicates the size of feature maps for each layer, and "channels" means for the number of output channels. "repeat" is for the number of stacking same blocks, and "stride" is for the stride of first block when stacked for multiple times. "MB1_K3" refers to Table 6.

input	block	channels	repeat	stride
$224^2 \times 3$	3×3 conv	32	1	2
$112^2 \times 32$	MB1_K3	16	1	1
$112^2 \times 16$	Choice Block	32	4	2
$56^2 \times 32$	Choice Block	40	4	2
$28^2 \times 40$	Choice Block	80	4	2
$14^2 \times 80$	Choice Block	96	4	1
$14^2 \times 96$	Choice Block	192	4	2
$7^2 \times 192$	Choice Block	320	1	1
$7^2 \times 320$	1×1 conv	1280	1	1
$7^2 \times 1280$	global avgpool	-	1	-
1280	FC	1000	1	-

Table 6: Operation choices for each MobileNetV2-based Choice Block in Table 5, where ID means for an identity mapping.

block type	expansion ratio	kernel	SE
MB1_K3	1	3	no
ID	-	-	-
MB3_K3	3	3	no
MB3_K5	3	5	no
MB3_K7	3	7	no
MB6_K3	6	3	no
MB6_K5	6	5	no
MB6_K7	6	7	no
MB3_K3_SE	3	3	yes
MB3_K5_SE	3	5	yes
MB3_K7_SE	3	7	yes
MB6_K3_SE	6	3	yes
MB6_K5_SE	6	5	yes
MB6_K7_SE	6	7	yes

B.2. Calculating corrected #optimization in Table 1

In our GreedyNAS, when equipped with the stopping principle of candidate pool, the supernet training is stopped at approximately 46-th epoch. Thus the accumulated number of examples calculated for a whole optimization step is equal to

$$\text{\#optimization=1.23M} \times 46,$$

where 1.23M refers to the quantity of training dataset. As for the path filtering, we evaluate 10 paths based on 1000 validation images, and select 5 paths for training, whose batch size is 1024. In this way, the number of images for evaluation amounts to

$$\begin{aligned}\text{\#evaluation} &= 1.23\text{M} \times \frac{1000}{1024} \times \frac{10}{5} \times 46, \\ &= 2.40\text{M} \times 46.\end{aligned}$$

Given our empirical findings that the cost of a whole optimization step is approximately 3.33 times larger than that of a forward evaluation, the corrected #optimization is thus

$$\begin{aligned}\text{corrected \#optimization} &= \text{\#optimization} + \text{\#evaluation}/3.33, \\ &= 1.23\text{M} \times 46 + 2.40\text{M} \times 46/3.33, \\ &= 89.7\text{M}.\end{aligned}$$

B.3. Details of rank correlation coefficient

In ablation study 5.3.1, we use two Spearman rho [3] and Kendall tau [2] rank correlation coefficient to evaluate the correlation of two path orderings, which are generated by ranking the evaluation results using 1000 and 50K validation images, denoted as $\boldsymbol{r}$ and $\boldsymbol{s}$, respectively. Basically, we aim to calculate the correlation of $\boldsymbol{r}$ and $\boldsymbol{s}$.

For Spearman rho rank correlation coefficient, it is simply the Pearson correlation coefficient between random variable r and s, if we regard $\boldsymbol{r}$ and $\boldsymbol{s}$ as two observation vectors of random variable r and s, *i.e.*,

$$\rho_S = \frac{cov(r, s)}{\sigma_r \sigma_s},$$

where $cov(\cdot, \cdot)$ is the covariance of two variables, and σ_r (σ_s) is the standard deviations of r (s). Based on observation vectors, it can be more efficiently calculated as

$$\rho_S = 1 - \frac{6 \sum_{i=1}^{n} (\boldsymbol{r}_i - \boldsymbol{s}_i)^2}{n(n^2 - 1)},$$

where $n = 1000$ in our experiment.

For Kendall tau rank correlation coefficient, it focuses on the pairwise ranking performance. For any pair $(\boldsymbol{r}_i, \boldsymbol{r}_j)$ and $(\boldsymbol{s}_i, \boldsymbol{s}_j)$, it is said to be *concordant* if $\boldsymbol{r}_i > \boldsymbol{r}_j$ and $\boldsymbol{s}_i > \boldsymbol{s}_j$ hold simultaneously, or also for $\boldsymbol{r}_i < \boldsymbol{r}_j$ and $\boldsymbol{s}_i < \boldsymbol{s}_j$. Otherwise, it will be called *disconcordant*. Then the coefficient is calculated as

$$\rho_K = \frac{\text{\#concordant pairs - \#disconcordant pairs}}{\text{\#all pairs}},$$

where \#all pairs $= \mathbb{C}_n^2$ refers to the total number of pairs. In this way, if two rankings $\boldsymbol{r}$ and $\boldsymbol{s}$ are exactly the same, ρ_K will be 1 while if the two are completely different (*i.e.*, one

ranking is the reverse of the other), ρ_K will be -1. According to the definition, it can also be calculated in a closed-form as

$$\rho_K = \frac{2}{n(n-1)} \sum_{i<j} \text{sign}(\boldsymbol{r}_i - \boldsymbol{r}_j)\text{sign}(\boldsymbol{s}_i - \boldsymbol{s}_j),$$

where $\text{sign}(\cdot)$ is the sign function.

B.4. More ablation studies

B.4.1 Performance of trained supernet

To further investigate the performance of the trained supernet, we implement two different searching methods (random search and evolutionary search) on various trained supernet, *i.e.*, greedy supernet, uniform supernet (full training) and uniform supernet-E (same training cost with GreedyNAS). The results can be regarded as supplementary for Table 1.

Table 7: Comparison of performance on ImageNet dataset of searched architectures w.r.t. different supernets under same search space.

supernet	searcher	Top-1 (%)	FLOPs
uniform	random	74.07	321M
uniform-E	random	73.88	320M
greedy	random	74.22	321M
uniform	evolutionary	74.50	326M
uniform-E	evolutionary	74.17	320M
greedy	evolutionary	**74.85**	320M

From Table 7, we can see that a greedy supernet improves consistently the classification accuracy in terms of different searchers. This validates the superiority of our greedy supernet since it helps searchers to probe better architectures. Moreover, to comprehensively investigate the effect of supernets, we implement systematic sampling [1] to sample 30 paths from $50 \times 20 = 1000$ paths, which are discovered by the evolutionary algorithms and ranked according to the accuracy on supernet. Then we retrain these 30 paths from scratch, and report their distribution histogram in Figure 5.

As shown in Figure 5, we can see that on average, paths searched with our greedy supernet have higher retraining Top-1 accuracy than that with uniform supernet. This implies that our greedy supernet serves as a better performance estimator, so that those good paths can be eventually identified and searched.

[1] https://en.wikipedia.org/wiki/Systematic_sampling

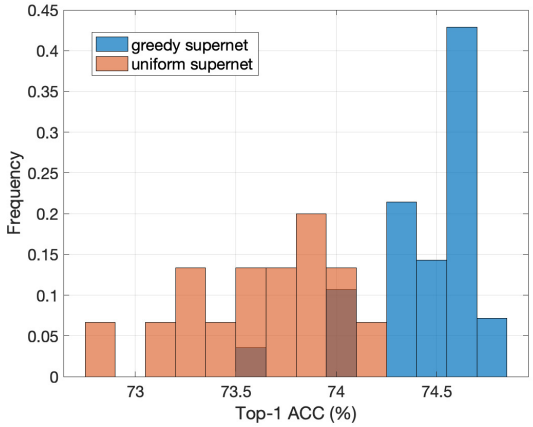


Figure 5: Top-1 accuracy histogram of 30 systematically sampled paths from 1000 paths searched by evolutionary algorithm after trained from scratch.

B.5. Visualization of searched architectures

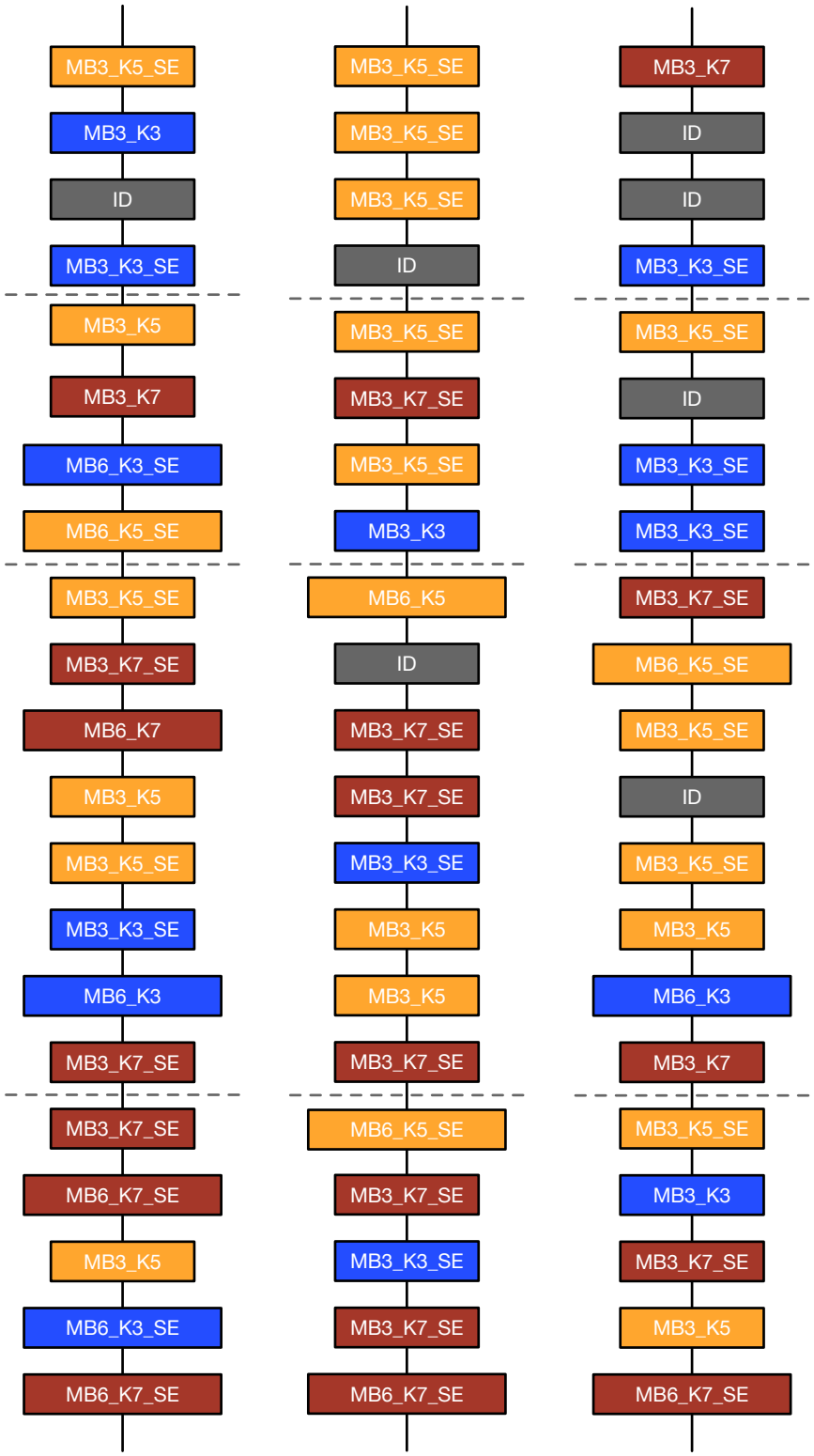
We visualize the searched architectures by our GreedyNAS method in Figure 6.

References

[1] Kalyanmoy Deb, Amrit Pratap, Sameer Agarwal, and TAMT Meyarivan. A fast and elitist multiobjective genetic algorithm: Nsga-ii. *IEEE transactions on evolutionary computation*, 6(2):182–197, 2002.

[2] Maurice G Kendall. A new measure of rank correlation. *Biometrika*, 30(1/2):81–93, 1938.

[3] W Pirie. S pearman rank correlation coefficient. *Encyclopedia of statistical sciences*, 2004.



(a) GreedyNAS-A (b) GreedyNAS-B (c) GreedyNAS-C

Figure 6: Visualization of searched architectures by GreedyNAS in Table 2.