Efficient Few-Shot Neural Architecture Search by Counting the Number of Nonlinear Functions

Youngmin Oh[1]   Hyunju Lee[1]   Bumsub Ham[1,2]

[1]Yonsei University   [2]Korea Institute of Science and Technology (KIST)

AAAI-25 / IAAI-25 / EAAI-25
FEBRUARY 25 – MARCH 4, 2025 | PHILADELPHIA, USA

Summary

Problem statement

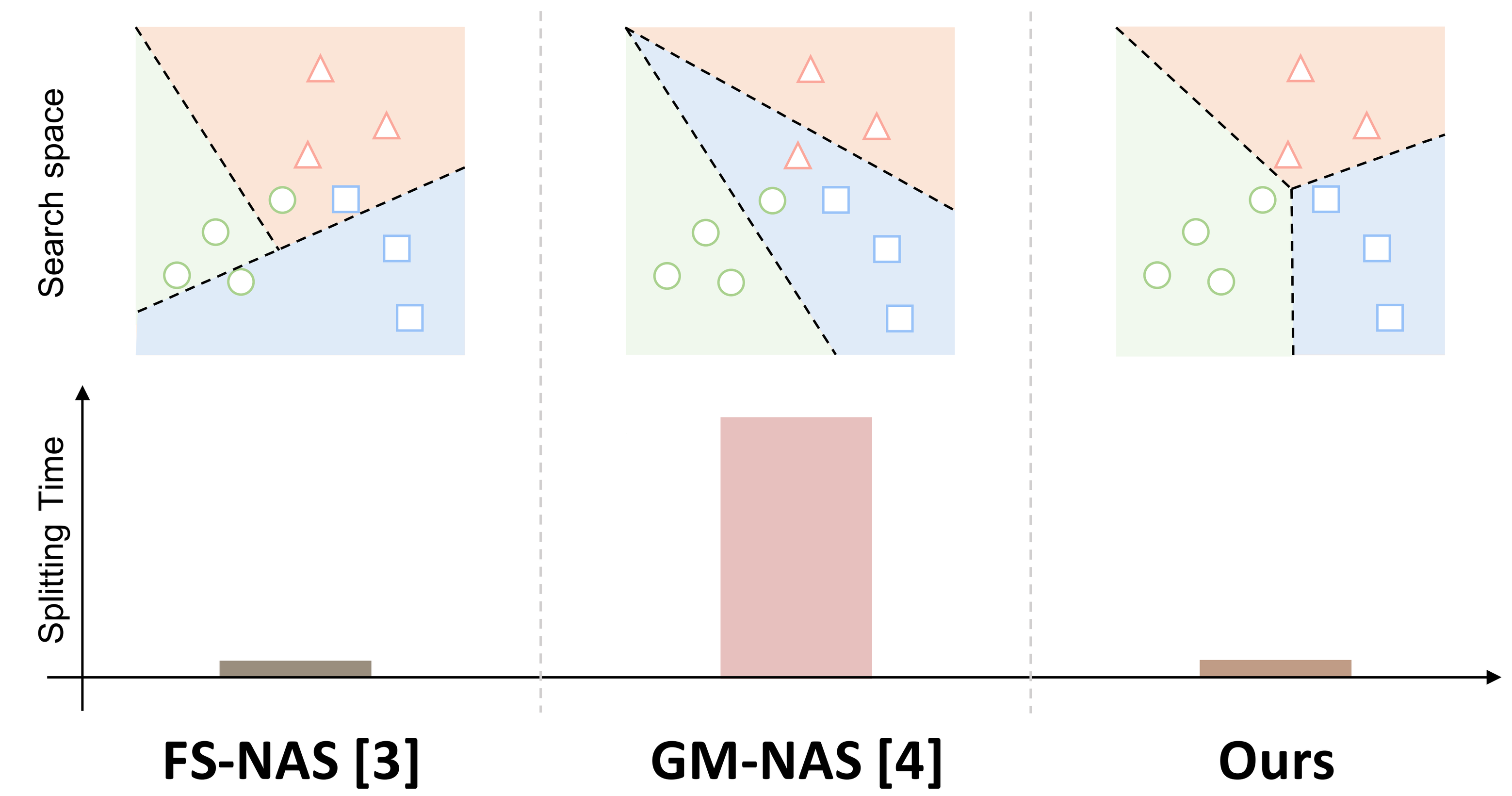
- Neural architecture search (NAS) aims to automatically find high-performing neural networks from a pre-defined search space.
- Early NAS methods adopt reinforcement learning with policy networks. They typically require training many networks from scratch, which takes thousands of GPU hours.
- *One*-Shot NAS [1] adopts a weight-sharing technique to reduce the search time, where they train a single supernet that consists of all possible network architectures (*i.e.*, subnets). The trained supernet can act as a performance estimator, indicating that each subnet does not need to be trained from scratch to predict its performance.
- *Few*-Shot NAS [2,3] proposes to use multiple supernets, as the single supernet is likely to suffer from conflicts between subnets during training. Specifically, they limit the extent of weight sharing by splitting the search space into subspaces and assigning an individual supernet to each subspace.
- *Zero*-Shot NAS [4] aims to avoid training supernets. They rely on training-free measurements (*e.g.*, Neural Tangent Kernels, FLOPs, or feature isotropy), typically referred to as zero-cost proxies, to evaluate the performance of each subnet.

[1] Single path one-shot neural architecture search with uniform sampling, ECCV 2020
[2] Few-shot neural architecture search, ICML 2021
[3] Generalized few-shot nas with gradient matching, ICLR 2022
[4] Neural architecture search without training, ICML 2021

Contributions

- We have introduced a novel few-shot NAS method that counts the number of nonlinear functions within a subnet to divide a search space in an efficient manner.
- We have observed that effectively dividing the search space enables maintaining the performance ranking between subnets even after reducing the number of channels required for supernets.
- Motivated by our finding, we have proposed to adjust the number of channels for each supernet, reducing the computational cost remarkably

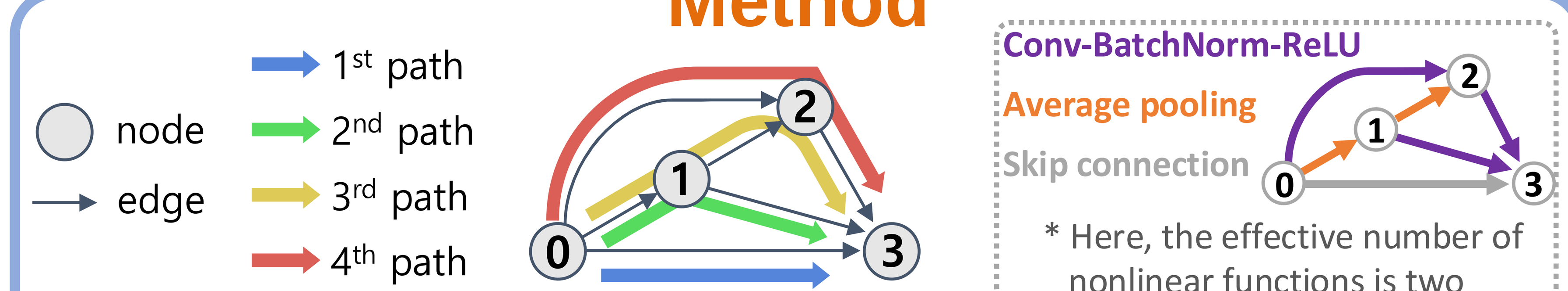
Motivation



* Individual subspaces (*i.e.*, supernets) are highlighted in different colors
** Subnets with similar characteristics are marked by the same shape

- We have found that existing few-shot methods split a search space either randomly [2] or by solving the graph clustering problem [3].
- The random criterion [2] is efficient, but each subspace could have subnets that are likely to conflict with each other. GM-NAS [3] better groups subnets at the cost of increasing the computational cost
- Another way to split a space is to leverage zero-cost proxies. However, they typically require processing forward and/or backward passes for each subnet, which is computationally demanding in that the total number of subnets is extremely large (*e.g.*, $6^6 \times 7^{15}$ subnets)

Method



* Here, the effective number of nonlinear functions is two

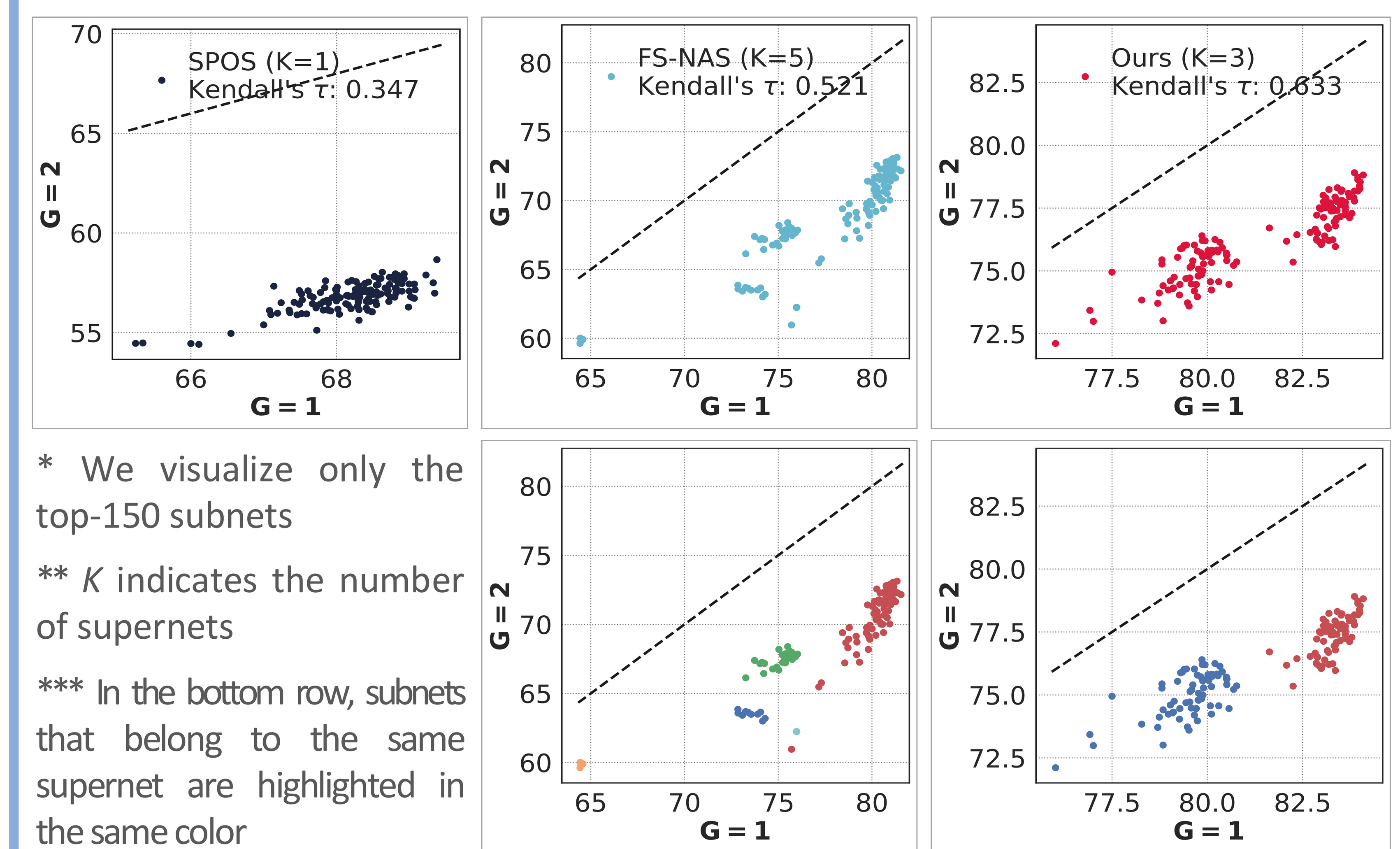
- We introduce an efficient criterion to divide a search space so that each subspace has subnets with the same number of nonlinear functions.
- To count the number of nonlinear functions within a subnet, we define two rules: (1) Accumulate the number of nonlinear functions, if layers are connected in series; (2) Select the path with the maximum number of nonlinear functions, if layers are connected in parallel

Analysis

Adjusting the number of channels for each supernet

We introduce a hyperparameter G to control the number of channels for each supernet
Let us suppose learnable parameters of the i-th operation at the j-th layer as follows:

$$w_k(G,i,j) \in \mathbb{R}^{\frac{C_{\text{out}}(i,j)}{G} \times \frac{C_{\text{in}}(i,j)}{G} \times s(i,j) \times s(i,j)}$$



* We visualize only the top-150 subnets
** K indicates the number of supernets
*** In the bottom row, subnets that belong to the same supernet are highlighted in the same color

Comparison of computational costs

Time (m)	NAS201	MobileNet
Ours	0	0
Ours w/ LR	7	46
Ours w/ ISO	3	29

* Total time for 15K subnets

Method	Cost: splitting	Cost: training	Top-1 Acc. (%)	FLOPs (M)
FS-NAS (Zhao et al. 2021)	-	23.1	75.9	521
GM-NAS (Hu et al. 2022)	17.0	81.0	76.6	530
Ours: K=4 & G=2	-	17.3	76.5	527
Ours: K=6 & G=2	-	23.3	76.7	516
Ours: K=8 & G=2	-	34.1	76.8	522

* In terms of GPU days (with 8 NVIDIA A5000 GPUs)

Results

On NAS-Bench-201

Method	K	Params (M)	Kendall's τ
One-shot NAS			
SPOS (Guo et al. 2020)	1	1.7	0.554
AngleNet (Hu et al. 2020)	1	1.7	0.575
Few-shot NAS			
FS-NAS (Zhao et al. 2021)	5	8.4	0.653
GM-NAS (Hu et al. 2022)	8	13.6	0.656
K-shot NAS (Su et al. 2021)	8	13.6	0.626
Ours			
FLOPs	3	**1.3**	0.711
# of linear regions	3	**1.3**	0.712
Feature isotropy	3	**1.3**	0.693
# of nonlinear functions	3	**1.3**	**0.735**

* Params indicates the total number of parameters required for supernets

On ImageNet

Method	Acc. (%) Top-1	Acc. (%) Top-5	FLOPs (M)	Params (M)
One-shot NAS				
GAEA (Li et al. 2021)	76.0	92.7	-	5.6
SPOS (Guo et al. 2020)	74.7	-	328	3.4
ProxylessNAS (Cai, Zhu, and Han 2019)	75.1	92.5	465	7.1
AngleNet (Hu et al. 2020)	76.1	-	470	-
Shapley-NAS (Xiao et al. 2022)	76.1	-	582	5.4
PC-DARTS (Xu et al. 2020)	75.8	92.7	597	5.3
DrNAS (Chen et al. 2021)	76.3	92.9	604	5.7
ISTA-NAS (Yang et al. 2020)	76.0	92.9	638	5.7
Few-shot NAS				
FS-NAS (Zhao et al. 2021)	75.9	-	521	4.9
GM-NAS (Hu et al. 2022)	76.6	93.0	530	4.9
Ours ($\leq$530M)	76.7	**93.2**	516	4.8
Ours ($\leq$600M)	**76.9**	**93.2**	544	4.9

* Params indicates the number of parameters for the chosen network