AZ-NAS: Assembling Zero-Cost Proxies for Network Architecture Search

Supplementary Material

In this supplement, we provide additional results and in-depth analyses of AZ-NAS on the NDS [18], NAS-Bench-201 [7], and MobileNetV2 [14, 19] search spaces.

Additional results on the NDS benchmark. The Neural Design Space (NDS) [18] benchmark provides the ground-truth accuracies of candidate architectures for several cell-based search spaces, where each space adopts a distinct set of candidate operations. We present in Table A the quantitative comparison of AZ-NAS with the state of the art [1, 13, 15, 21] on the NDS benchmark. For each search space in NDS, we report a correlation coefficient (Kendall's τ) between predicted and ground-truth network rankings, and an average top-1 accuracy of selected networks on CIFAR-10 [11]. We can see that AZ-NAS shows superior results across the search spaces, verifying the generalization ability of AZ-NAS on various search spaces. On the other hand, other methods [1, 13, 15, 21] exploiting a single proxy are either outperformed by or merely comparable to #Params or FLOPs in terms of ranking consistency w.r.t the performance (*i.e.*, Kendall's τ), indicating that they are less effective on the NDS benchmark than NAS-Bench-201 [7] (see Table 1 in the main paper). These results confirm that assembling various proxies could improve the robustness against the variability of search spaces.

Reproducibility of training-free NAS methods. In Table 2 of the main paper, we compare AZ-NAS with the state of the art, mainly focusing on the final performance of selected networks. Considering the randomness during the search phase, *e.g.*, due to a sampling process of candidate architectures from a large-scale search space [14, 19], it is also crucial to achieve consistent NAS results over multiple trials for practical use. We thus evaluate the reproducibility of AZ-NAS and the state-of-the-art training-free NAS methods [13, 14] under a fair experimental setting. Specifically, for each method, we find three network architectures from scratch with different seed numbers on the MobileNetV2 search space [14, 19]. We use the same number of search iterations (100K) and FLOPs constraint (450M) for all the methods for a fair comparison. We train selected networks on ImageNet [5] with a simplified training scheme used in [13] to reduce the training cost. Compared to the training setting for the experiments in Table 2 of the main paper, the simplified one reduces training epochs from 480 to 150, while not incorporating the teacher-student distillation [10] and advanced data augmentation techniques (*e.g.*, AutoAugment [4], MixUp [22], and RandomErase [24]).

We present in Table B the average and standard deviation values of top-1 validation accuracies on ImageNet for selected networks. While the network architectures provided by the authors of ZenNAS [14] and ZiCo [13] show decent performance, we could not reproduce comparable results within three random runs, suggesting that these methods lack reproducibility. On the contrary, AZ-NAS outperforms the others by significant margins in terms of the average accuracy, while showing the lowest deviation, demonstrating its ability to find high-performing networks consistently across multiple runs. We can also see that the relative order of the search costs, measured under a fair search configuration using the same machine, is aligned with the one in Table 1 of the main paper, and AZ-NAS offers a good compromise between the NAS performance and search cost.

Visualization of ranking consistency. We present in Fig. A a visual comparison between the training-free NAS

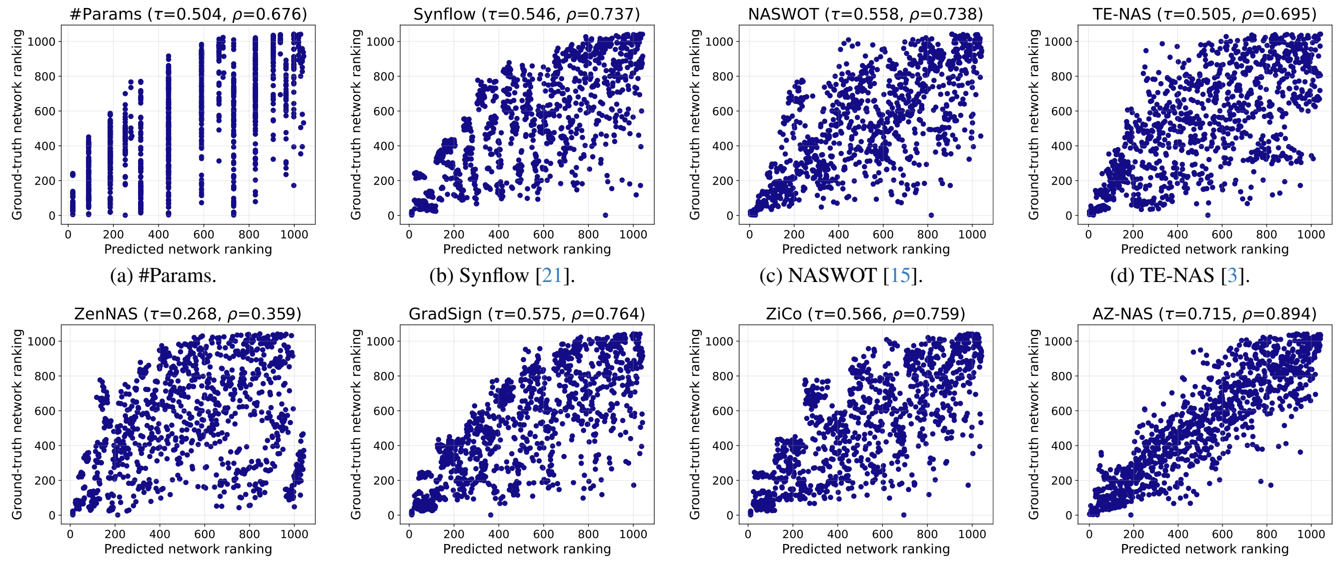
Table A. Quantitative comparison on the NDS [18] benchmark. For each search space, we report Kendall's τ (KT) using all candidate architectures. We also present average and standard deviation of test accuracies (Acc.) for selected networks on CIFAR-10 [11], which are obtained over 5 random runs. To this end, we randomly sample 1000 candidate architectures for each run and select an optimal one among them based on each training-free NAS method.

Method	ENAS		Amoeba		NASNet	
	KT	Acc.	KT	Acc.	KT	Acc.
#Params	0.411	92.94 $\pm$ 1.26	0.241	76.77 $\pm$ 37.3	0.289	92.56 $\pm$ 1.10
FLOPs	0.409	92.94 $\pm$ 1.26	0.239	76.77 $\pm$ 37.3	0.276	92.56 $\pm$ 1.10
Synflow [1, 21]	0.116	67.90 $\pm$ 28.9	-0.076	87.70 $\pm$ 8.29	0.007	79.98 $\pm$ 25.0
NASWOT [15]	0.375	93.56 $\pm$ 1.55	0.192	92.89 $\pm$ 0.29	0.286	93.55 $\pm$ 0.93
ZiCo [13]	0.200	92.19 $\pm$ 1.15	-0.016	92.25 $\pm$ 0.60	0.089	92.72 $\pm$ 0.88
AZ-NAS	**0.495**	**94.41** $\pm$ **0.13**	**0.386**	**93.75** $\pm$ **0.24**	**0.426**	**93.72** $\pm$ **0.50**

Table B. Comparison of the reproducibility for the state-of-the-art training-free NAS methods [13, 14] and ours on the MobileNetV2 [19] search space [14]. For each method, we report the average and standard deviation values of top-1 validation accuracies on ImageNet [5], obtained over three random runs *starting from the search phase*, together with the search costs in terms of GPU hours. The results of [13, 14] are obtained using the official code provided by the authors.

[†]Architectures provided by the authors.
[*]An architecture found with a different FLOPs budget (*i.e.*, 400M).

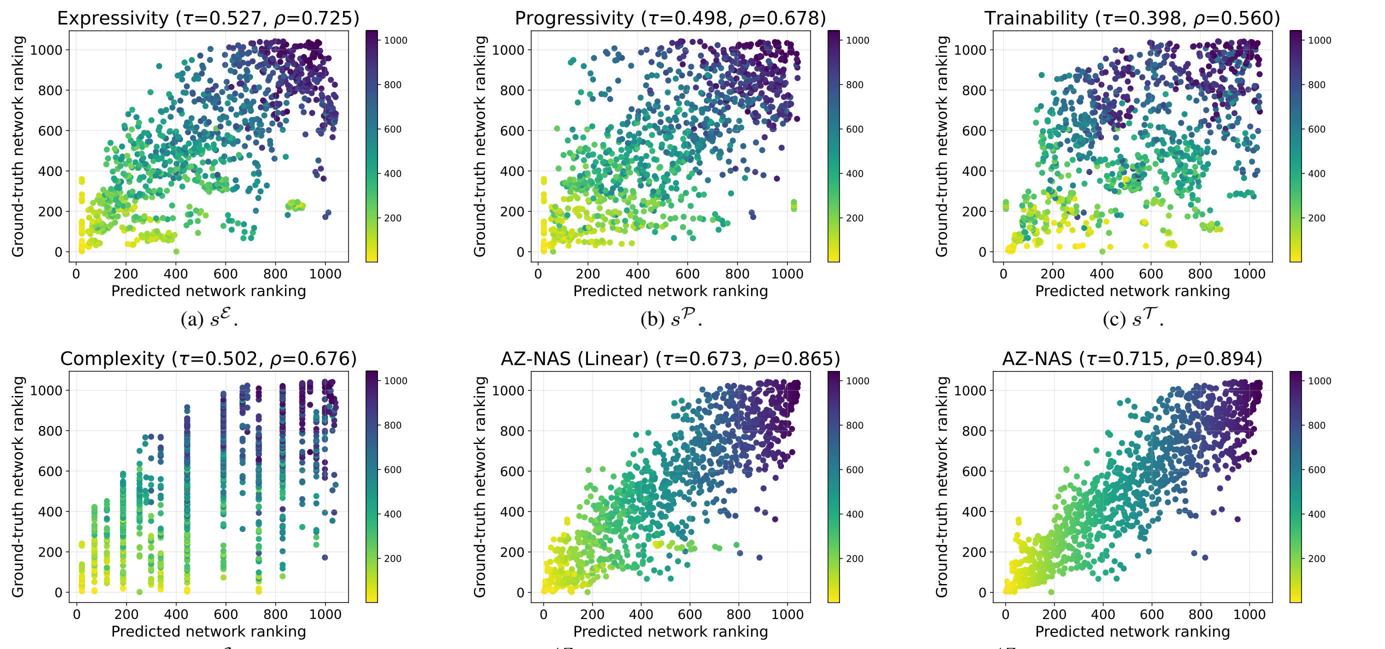
Method	FLOPs	Top-1 acc.	Search cost (GPU hours)
ZenNAS[†] [14]	410M[*]	75.87	-
ZiCo[†] [13]	448M	76.07	-
ZenNAS [14]	458M $\pm$ 1.6M	73.76 $\pm$ 1.32	4.9
ZiCo [13]	450M $\pm$ 5.0M	72.46 $\pm$ 0.84	15.7
AZ-NAS (Ours)	462M $\pm$ 1.5M	**76.46** $\pm$ **0.06**	10.0



(a) #Params. (b) Synflow [21]. (c) NASWOT [15]. (d) TE-NAS [3].

(e) ZenNAS [14]. (f) GradSign [23]. (g) ZiCo [13]. (h) AZ-NAS (Ours).

Figure A. Visual comparison of training-free NAS methods in terms of predicted network ranking (x-axis) *vs*. ground truth (y-axis) on ImageNet16-120 of NAS-Bench-201 [7]. We report the correlation coefficients between them in terms of Kendall's τ and Spearman's ρ, denoted by τ and ρ, respectively.



(a) $s^{\mathcal{E}}$. (b) $s^{\mathcal{P}}$. (c) $s^{\mathcal{T}}$.

(d) $s^{\mathcal{C}}$. (e) s^{AZ} (Linear ranking aggregation). (f) s^{AZ} (Non-linear ranking aggregation).

Figure B. Visual comparison of the zero-cost proxies of AZ-NAS ((a)-(d)), and the linear and non-linear ranking aggregation methods ((e) and (f)), in terms of predicted network ranking (x-axis) *vs*. ground truth (y-axis) on ImageNet16-120 of NAS-Bench-201 [7]. The colors of the points, ranging from light-yellow to dark-blue, correspond to the network ranking in (f) predicted by the AZ-NAS score. (Best viewed in color.)

methods [3, 13–15, 21, 23] and ours, where each plot shows the predicted network ranking (x-axis) *vs*. the ground-truth network ranking (y-axis) on ImageNet16-120 of NAS-Bench-201 [7]. For visualization, we exploit the same 1042 candidate architectures for all the methods, evenly sampled according to the test accuracy on ImageNet16-120. From

Table C. Comparison of AZ-NAS using different network initialization methods. We report correlation coefficients (Kendall's τ) between predicted and ground-truth network rankings on NAS-Bench-201 [7]. For other experiments using convolutional neural networks, we adopt the Kaiming normal initialization with a fan-in mode by default (highlighted by a gray color).

Initialization method	CIFAR-10	CIFAR-100	IN16-120
Kaiming normal (fan-in)	0.741	0.723	0.710
Kaiming normal (fan-out)	0.754	0.733	0.724
Xavier normal	0.754	0.732	0.724
Normal (std=0.1)	0.740	0.715	0.709
Uniform ([-0.1, 0.1])	0.730	0.703	0.702

Table D. Comparison of AZ-NAS using different batch sizes in the search phase. We report the correlation coefficients (Kendall's τ) between predicted and ground-truth network rankings on NAS-Bench-201 [7]. For other experiments, we adopt the batch size of 64 by default (highlighted by a gray color).

Batch size	CIFAR-10	CIFAR-100	IN16-120	Runtime (ms/arch)
8	0.730	0.709	0.687	37.33
16	0.736	0.717	0.701	39.39
32	0.740	0.721	0.708	39.78
64	0.741	0.723	0.710	42.71
128	0.741	0.724	0.710	54.32

Figs. A(a)-(g), we can see that previous training-free NAS methods produce incorrect predictions frequently, such as highly-ranked networks with low ground-truth performance that can be found in the bottom-right regions of the plots, making it difficult to discover a high-performing network accurately. On the other hand, the network ranking predicted by AZ-NAS in Fig. A(h) shows the strongest correlation with the ground truth, with the points concentrated closely around the $y = x$ line in the plot, demonstrating the superiority of our method.

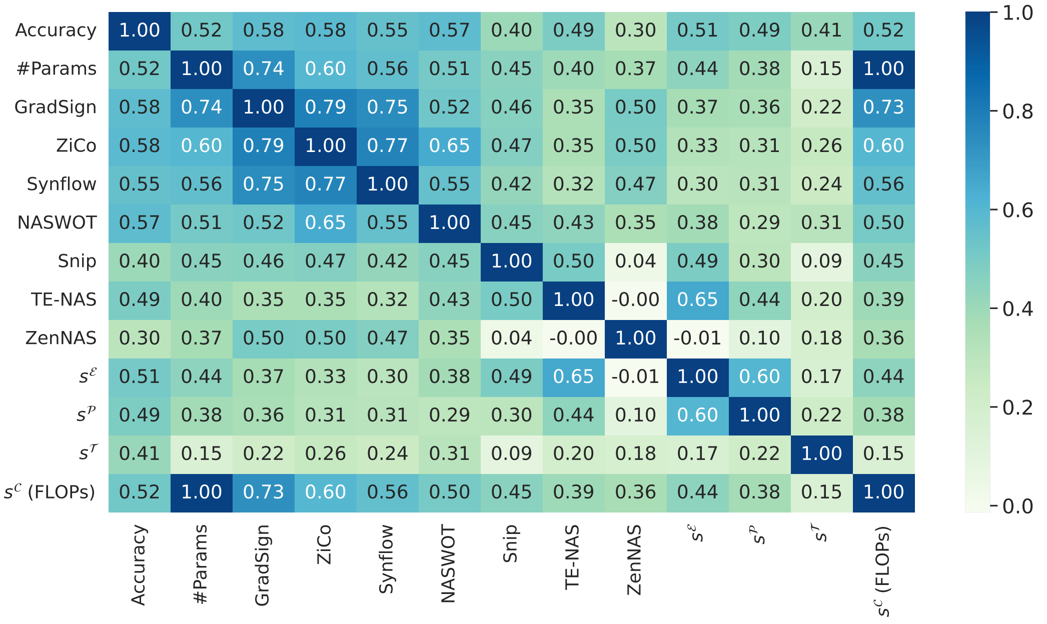
We also compare in Fig. B the network rankings obtained with each of the zero-cost proxies of AZ-NAS, and the aggregated ones using either linear or non-linear methods on ImageNet16-120 of NAS-Bench-201. We can observe in Figs. B(a)-(d) that the network rankings estimated by a single proxy alone show weak consistency w.r.t the ground truth. For example, the proxies in Figs. B(a)-(d) often assign high scores for low-performing networks (*e.g.*, bottom-right regions of Figs. B(c) and (d)) or low scores for high-performing networks (*e.g.*, top-left regions of Figs. B(b) and (c)). By combining the proxies into the AZ-NAS score in Fig. B(f), we can improve the ranking consistency substantially. We can find that the yellow-green and green-blue points near the bottom-right and top-left regions in Figs. B(a)-(d) are located closely to the $y = x$ line in Fig. B(f), suggesting that the misaligned rankings estimated by individual proxies are corrected by assembling. By comparing the results obtained with the linear and non-linear ranking aggregation methods in Figs. B(e) and (f), respectively, we can also see that the non-linear one makes the ranking consistency much stronger, while correcting erroneous predictions (see yellow-green points in the bottom-right region of Fig. B(e)), clearly demonstrating the effectiveness of our non-linear ranking aggregation method.

Robustness to various initialization methods. To verify the robustness of AZ-NAS against network initialization methods, we compare the NAS performance by initializing weight parameters from Kaiming normal [9], Xavier normal [8], normal, or uniform distributions. We provide the results in Table C in terms of the ranking consistency (Kendall's τ) w.r.t the ground-truth performance on NAS-Bench-201 [7]. We can see that AZ-NAS shows satisfactory NAS results consistently, regardless of the types of initialization methods. Note that we simply adopt Kaiming normal with a fan-in mode implemented in PyTorch [16] as our default initialization method for the experiments using convolutional neural networks. We can further improve the NAS performance when exploiting other initialization methods, such as Kaiming normal with a fan-out mode or Xavier normal.

Comparison using various batch sizes for search. We present in Table D the ranking consistency (Kendall's τ) of AZ-NAS w.r.t the performance on NAS-Bench-201 [7] using different batch sizes in the search phase, in order to understand how AZ-NAS is affected by a batch size. We can see that AZ-NAS achieves good NAS performance even when a small batch size (*i.e.*, 8 in the first row) is used for search. As we increase the batch size, the ranking consistency gradually improves at the expense of additional runtime. We can observe that the batch size of 64 provides the best trade-off between runtime and ranking consistency, and we adopt this as a default setting for other experiments.

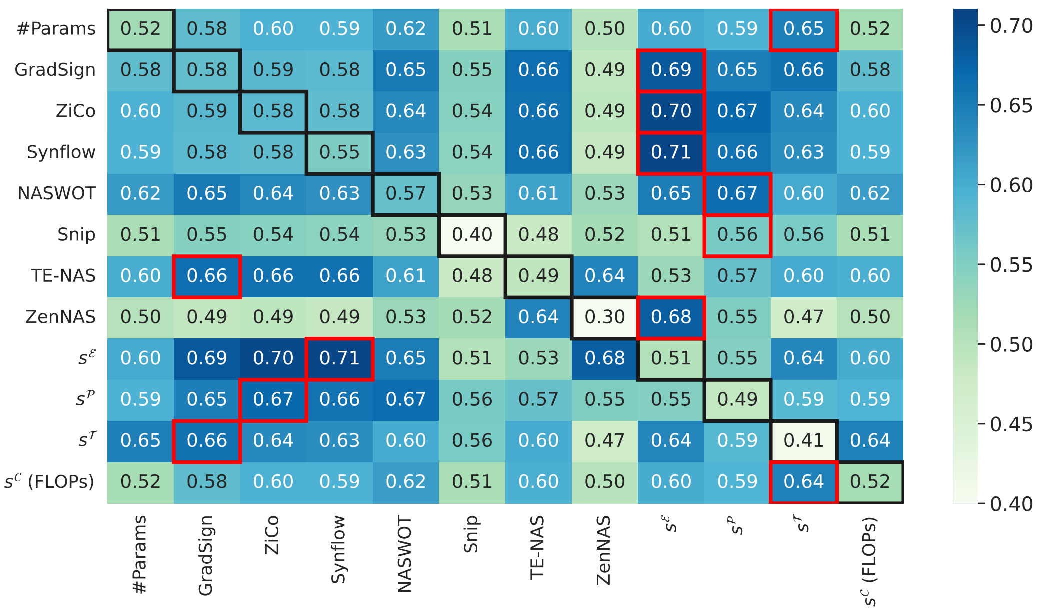
Intercorrelation and complementary features. AZ-NAS is built upon the idea that assembling multiple zero-cost proxies can significantly boost the NAS performance. However, naively assembling previous zero-cost proxies might be less effective, mainly due to a lack of complementary features. To investigate this, we analyze in Fig. C intercorrelation between network rankings estimated by zero-cost proxies on ImageNet16-120 of NAS-Bench-201 [7]. We also compare in Fig. D the ranking consistency w.r.t the performance on ImageNet16-120 of NAS-Bench-201, where network rankings predicted by two of the proxies in the x and y-axes are integrated.

From these results, we have the following observations: (1) We can see in Fig. C that many existing zero-cost proxies, especially the ones exploiting gradients (*e.g.*, GradSign [23], ZiCo [13], and Synflow [21]), predict similar network rankings that are strongly correlated to each other.



	Accuracy	#Params	GradSign	ZiCo	Synflow	NASWOT	Snip	TE-NAS	ZenNAS	$s^{\mathcal{E}}$	$s^{\mathcal{P}}$	$s^{\mathcal{T}}$	$s^{\mathcal{C}}$ (FLOPs)
Accuracy	1.00	0.52	0.58	0.58	0.55	0.57	0.40	0.49	0.30	0.51	0.49	0.41	0.52
#Params	0.52	1.00	0.74	0.60	0.56	0.51	0.45	0.40	0.37	0.44	0.38	0.15	1.00
GradSign	0.58	0.74	1.00	0.79	0.75	0.52	0.46	0.35	0.50	0.37	0.36	0.22	0.73
ZiCo	0.58	0.60	0.79	1.00	0.77	0.65	0.47	0.35	0.50	0.33	0.31	0.26	0.60
Synflow	0.55	0.56	0.75	0.77	1.00	0.55	0.42	0.32	0.47	0.30	0.31	0.24	0.56
NASWOT	0.57	0.51	0.52	0.65	0.55	1.00	0.45	0.43	0.35	0.38	0.29	0.31	0.50
Snip	0.40	0.45	0.46	0.47	0.42	0.45	1.00	0.50	0.04	0.49	0.30	0.09	0.45
TE-NAS	0.49	0.40	0.35	0.35	0.32	0.43	0.50	1.00	-0.00	0.65	0.44	0.20	0.39
ZenNAS	0.30	0.37	0.50	0.50	0.47	0.35	0.04	-0.00	1.00	-0.01	0.10	0.18	0.36
$s^{\mathcal{E}}$	0.51	0.44	0.37	0.33	0.30	0.38	0.49	0.65	-0.01	1.00	0.60	0.17	0.44
$s^{\mathcal{P}}$	0.49	0.38	0.36	0.31	0.31	0.29	0.30	0.44	0.10	0.60	1.00	0.22	0.38
$s^{\mathcal{T}}$	0.41	0.15	0.22	0.26	0.24	0.31	0.09	0.20	0.18	0.17	0.22	1.00	0.15
$s^{\mathcal{C}}$ (FLOPs)	0.52	1.00	0.73	0.60	0.56	0.50	0.45	0.39	0.36	0.44	0.38	0.15	1.00

Figure C. Correlation analysis of various zero-cost proxies and ours on ImageNet16-120 of NAS-Bench-201 [7]. We report correlation coefficients (Kendall's τ) for the pairs of two network rankings estimated by the zero-cost proxies in the x- and y-axes, respectively.



	#Params	GradSign	ZiCo	Synflow	NASWOT	Snip	TE-NAS	ZenNAS	$s^{\mathcal{E}}$	$s^{\mathcal{P}}$	$s^{\mathcal{T}}$	$s^{\mathcal{C}}$ (FLOPs)
#Params	0.52	0.58	0.60	0.59	0.62	0.51	0.60	0.50	0.60	0.59	0.65	0.52
GradSign	0.58	0.58	0.59	0.58	0.65	0.55	0.66	0.49	0.69	0.65	0.66	0.58
ZiCo	0.60	0.59	0.58	0.58	0.64	0.54	0.66	0.49	0.70	0.67	0.64	0.60
Synflow	0.59	0.58	0.58	0.55	0.63	0.54	0.66	0.49	0.71	0.66	0.63	0.59
NASWOT	0.62	0.65	0.64	0.63	0.57	0.53	0.61	0.53	0.65	0.67	0.60	0.62
Snip	0.51	0.55	0.54	0.54	0.53	0.40	0.48	0.52	0.51	0.56	0.56	0.51
TE-NAS	0.60	0.66	0.66	0.66	0.61	0.48	0.49	0.64	0.53	0.57	0.60	0.60
ZenNAS	0.50	0.49	0.49	0.49	0.53	0.52	0.64	0.30	0.68	0.55	0.47	0.50
$s^{\mathcal{E}}$	0.60	0.69	0.70	0.71	0.65	0.51	0.53	0.68	0.51	0.55	0.64	0.60
$s^{\mathcal{P}}$	0.59	0.65	0.67	0.66	0.67	0.56	0.57	0.55	0.55	0.49	0.59	0.59
$s^{\mathcal{T}}$	0.65	0.66	0.64	0.63	0.60	0.56	0.60	0.47	0.64	0.59	0.41	0.64
$s^{\mathcal{C}}$ (FLOPs)	0.52	0.58	0.60	0.59	0.62	0.51	0.60	0.50	0.60	0.59	0.64	0.52

Figure D. Ranking consistency (Kendall's τ) w.r.t the performance on ImageNet16-120 of NAS-Bench-201 [7] obtained with various combinations of zero-cost proxies. We combine two network rankings, predicted by zero-cost proxies in the x- and y-axes, using the non-linear ranking aggregation method. The result in the diagonal entry (represented by a black box) is obtained with the single proxy for the corresponding entry. For each proxy in the y-axis, we specify the best combination providing the highest Kendall's τ by a red box in the corresponding row.

This coincides with the finding that several gradient-based zero-cost proxies are related theoretically [20], providing analogous NAS results. This suggests that these proxies lack complementary features, and thus combining them hardly improves the NAS performance as shown in Fig. D.

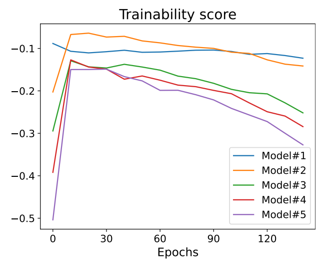
(2) A few zero-cost proxies (*e.g.*, GradSign and ZiCo) show strong correlations with #Params. In their implementation, they compute a proxy score for each weight parameter (or trainable layer), and then obtain the final score of a network by adding all the scores over the weight parameters (or

the trainable layers). This implies that they inherently prefer networks with large #Params, since the final scores are likely to become higher when a larger #Params are used, resulting in a limited synergy effect among these proxies and #Params as shown in Fig. D. (3) In terms of the NAS performance using a single proxy only, ZenNAS [14] and Snip [12] show weak ranking consistency w.r.t the performance, *i.e.*, Kendall's τ of 0.30 and 0.40, respectively. We can see in Fig. D that they rather degrade the NAS performance of several zero-cost proxies (*e.g.*, #Params, GradSign, ZiCo, Synflow, or NASWOT [15]) after integrating the network rankings, suggesting that they are less suitable for an ensemble. On the contrary, while our trainability proxy ($s^{\mathcal{T}}$) also exhibits a weak correlation with the performance, it can help other proxies to improve the ranking consistency by large margins. This clearly demonstrates that the trainability proxy captures a useful network trait for NAS, which is often overlooked by other zero-cost proxies, despite its weak ranking consistency w.r.t the performance. (4) We can see in Fig. C that our zero-cost proxies are less correlated with most of the existing ones. We can also observe in Fig. D that incorporating our proxies into others boosts the ranking consistency drastically. This suggests that our proxies offer distinct and useful cues for training-free NAS that could not be identified by other proxies, highlighting their complementary features. In particular, the proxies leveraging gradients (*e.g.*, GradSign, ZiCo, Synflow, and Snip) typically show the best ensemble results when they are coupled with either the expressivity ($s^{\mathcal{E}}$) or progressivity ($s^{\mathcal{P}}$) proxies that analyze activations. This implies that activations and gradients provide different network characteristics useful for training-free NAS, and it is difficult to achieve good NAS performance when relying solely on one of them. These results confirm once more the importance of the comprehensive evaluation of a network from various and complementary perspectives for effective training-free NAS. (5) In the case of combining two zero-cost proxies in Fig. D, NASWOT and TE-NAS [3] could be good alternatives for improving the NAS performance of other zero-cost proxies. However, as mentioned in the main paper, NASWOT is only applicable for networks adopting ReLU non-linearities, and TE-NAS is computationally expensive. We can also see that coupling one of the zero-cost proxies of AZ-NAS with *e.g.*, GradSign, ZiCo, or Synflow provides better ranking consistency w.r.t the performance, compared to the combinations among our proxies. Nevertheless, they are outperformed by our AZ-NAS method that assembles all of our zero-cost proxies, which offers a good balance between efficiency and the NAS performance without additional complexities (see Sec. 4.2 in the main paper for details).

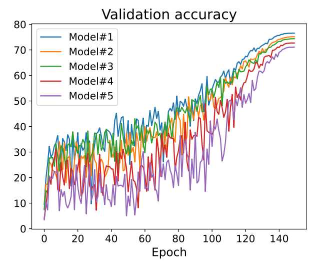
Analysis on the trainability proxy. The zero-cost proxies of AZ-NAS assess a network with randomly initialized weight parameters. The expressivity and progressivity proxies could benefit from this setting, since they can evaluate the feature space of a network by analyzing diverse features. That is, the features extracted by a randomly initialized network could occupy the space to its maximum capacity as they are randomly distributed along various ori-

Table E. Search objectives with varying target values of the trainability score. Note that the trainability score ranges from $-\infty$ to 0. We also report the top-1 validation accuracies on ImageNet [5] for the networks found with corresponding configurations.

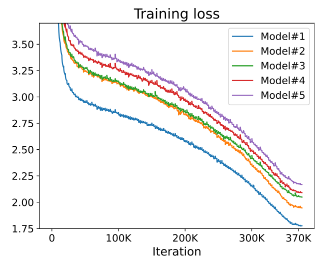
Name	$s^{\mathcal{E}}$	$s^{\mathcal{P}}$	$s^{\mathcal{T}}$	$s^{\mathcal{C}}$	Top-1 acc.
Model#1	Maximize	Maximize	Maximize (Close to 0)	Maximize	76.55
Model#2	Maximize	Maximize	Close to -0.2	Maximize	75.12
Model#3	Maximize	Maximize	Close to -0.3	Maximize	74.37
Model#4	Maximize	Maximize	Close to -0.4	Maximize	72.68
Model#5	Maximize	Maximize	Close to -0.5	Maximize	71.10



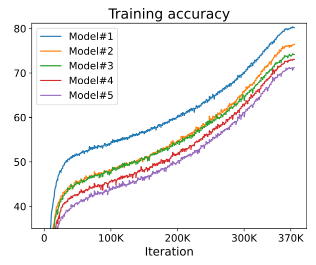
(a) Trainability scores $s^{\mathcal{T}}$ for every 10 epochs.



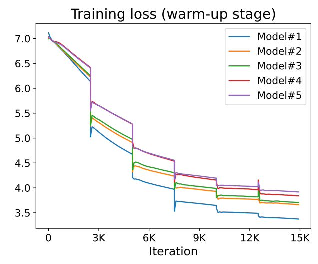
(b) Top-1 validation accuracy.



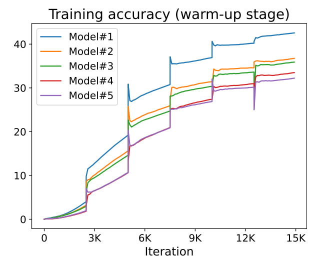
(c) Training loss.



(d) Training accuracy.

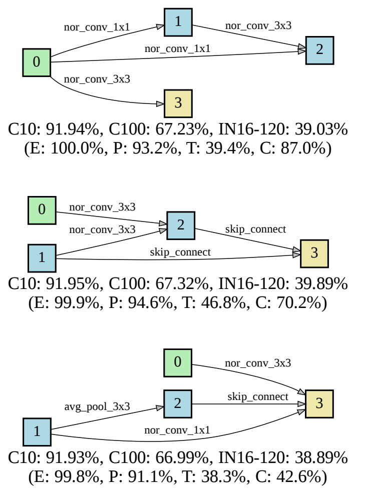


(e) Training loss (warm-up stage).



(f) Training accuracy (warm-up stage).

Figure E. Comparison of the networks with different trainability scores, where they are found with the search objectives specified in Table E. We show in (a) and (b) the trainability scores of the networks and validation accuracy on ImageNet [5] along epochs, respectively. We also compare in (c)-(f) the training curves of the networks. (Best viewed in color.)

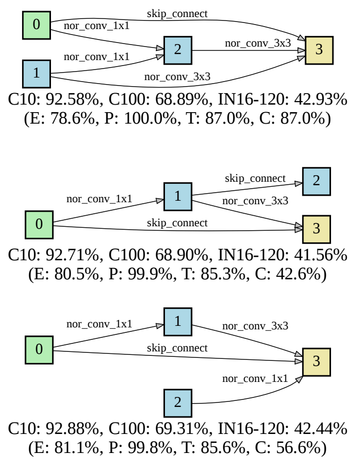


C10: 91.94%, C100: 67.23%, IN16-120: 39.03%
(E: 100.0%, P: 93.2%, T: 39.4%, C: 87.0%)

C10: 91.95%, C100: 67.32%, IN16-120: 39.89%
(E: 99.9%, P: 94.6%, T: 46.8%, C: 70.2%)

C10: 91.93%, C100: 66.99%, IN16-120: 38.89%
(E: 99.8%, P: 91.1%, T: 38.3%, C: 42.6%)

(a) Top-3 architectures selected with $s^{\mathcal{E}}$.

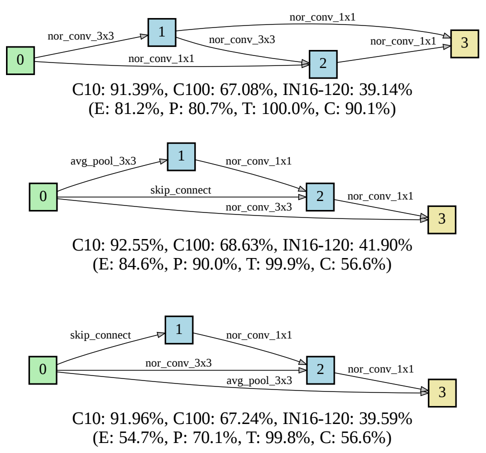


C10: 92.58%, C100: 68.89%, IN16-120: 42.93%
(E: 78.6%, P: 100.0%, T: 87.0%, C: 87.0%)

C10: 92.71%, C100: 68.90%, IN16-120: 41.56%
(E: 80.5%, P: 99.9%, T: 85.3%, C: 42.6%)

C10: 92.88%, C100: 69.31%, IN16-120: 42.44%
(E: 81.1%, P: 99.8%, T: 85.6%, C: 56.6%)

(b) Top-3 architectures selected with $s^{\mathcal{P}}$.

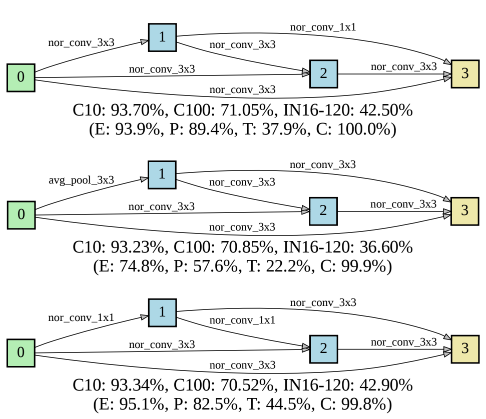


C10: 91.39%, C100: 67.08%, IN16-120: 39.14%
(E: 81.2%, P: 80.7%, T: 100.0%, C: 90.1%)

C10: 92.55%, C100: 68.63%, IN16-120: 41.90%
(E: 84.6%, P: 90.0%, T: 99.9%, C: 56.6%)

C10: 91.96%, C100: 67.24%, IN16-120: 39.59%
(E: 54.7%, P: 70.1%, T: 99.8%, C: 56.6%)

(c) Top-3 architectures selected with $s^{\mathcal{T}}$.

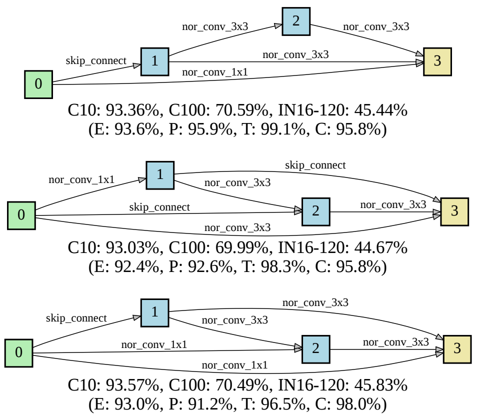


C10: 93.70%, C100: 71.05%, IN16-120: 42.50%
(E: 93.9%, P: 89.4%, T: 37.9%, C: 100.0%)

C10: 93.23%, C100: 70.85%, IN16-120: 36.60%
(E: 74.8%, P: 57.6%, T: 22.2%, C: 99.9%)

C10: 93.34%, C100: 70.52%, IN16-120: 42.90%
(E: 95.1%, P: 82.5%, T: 44.5%, C: 99.8%)

(d) Top-3 architectures selected with $s^{\mathcal{C}}$.



C10: 93.36%, C100: 70.59%, IN16-120: 45.44%
(E: 93.6%, P: 95.9%, T: 99.1%, C: 95.8%)

C10: 93.03%, C100: 69.99%, IN16-120: 44.67%
(E: 92.4%, P: 92.6%, T: 98.3%, C: 95.8%)

C10: 93.57%, C100: 70.49%, IN16-120: 45.83%
(E: 93.0%, P: 91.2%, T: 96.5%, C: 98.0%)

(e) Top-3 architectures selected with s^{AZ}.

Figure F. Visualization of the top-3 network architectures found with each zero-cost proxy score of AZ-NAS and the final AZ-NAS score on NAS-Bench-201 [7]. The green and yellow squares indicate the input and output nodes of a cell, respectively, and the blue ones represent intermediate nodes. For each architecture, we report the top-1 test accuracies on the CIFAR-10/100 (C10/100) and ImageNet16-120 (IN16-120) datasets, together with the percentiles for the proxy scores in parenthesis, where 100% indicates the highest ranking. We denote by E, P, T, and C the expressivity, progressivity, trainability, and complexity proxies, respectively, for brevity.

entations. The complexity proxy measures FLOPs of a network architecture, which remains fixed regardless of training. Similarly, the trainability proxy evaluates a network at initialization in terms of stable gradient propagation, focusing on the spectral norm of the Jacobian matrix for a primary block. Note that the stable gradient propagation of a network at the initial state has been proven to be crucial for high performance [8, 9, 17], supporting our idea of scoring the trainability proxy without training.

To further demonstrate the validity of our approach to measuring the trainability score at initialization, we perform an in-depth analysis of the trainability proxy on the MobileNetV2 search space [14, 19]. Specifically, we select five distinct architectures by setting different search objectives of the trainability proxy for the evolutionary algorithm, and train them with the same training scheme used for the experiments in Table B. We summarize the search configurations and results in Table E. From the table, we can observe that the final performance of a network is largely affected by the trainability score, where a network with a higher trainability score shows better performance. We also present in Fig. E how the trainability score changes during training, and its impact on training and performance, based on the networks chosen in Table E. We can see from Fig. E(a) that the trainability scores are not maintained during training, possibly because they are affected by weight parameters that keep changing during training. Nevertheless, the relative ranking between them is roughly preserved along training epochs.

This suggests that a high trainability score at initialization is important for a network to achieve better performance, as evidenced by Fig. E(b). We can also see from Figs. E(c) and (d) that the better trainability score consistently results in better training losses and accuracies throughout training iterations. In particular, we can find in Figs. E(e) and (f) that the rankings of training losses and accuracies are aligned with the ranking of the trainability scores even at the very beginning of training (*i.e.*, the warm-up stage), highlighting the importance of the trainability proxy at the initial state of a network. This finding also coincides with the concept of learning curve extrapolation [6] or an early stopping technique for NAS [2]. These results confirm that considering our trainability proxy at initialization is effective for training-free NAS, and it plays a significant role in predicting the ranking of candidate networks in terms of the final performance.

Visualization of selected architectures. We visualize in Fig. F network architectures chosen by individual zero-cost proxies of AZ-NAS or the final AZ-NAS score on NAS-Bench-201 [7]. We select top-3 network architectures for each proxy among 1042 candidate networks, where the candidates are sampled evenly according to the test accuracy on ImageNet16-120 of NAS-Bench-201. For each architecture, we report the test accuracies on CIFAR-10/100 and ImageNet16-120, as well as the percentiles of our zero-cost proxy scores (*i.e.*, 100% indicates that a network exhibits the highest score). We can see from Figs. F(a)-(d) that relying solely on one of the zero-cost proxies causes structural biases in the selected architectures. For example, the expressivity proxy in Fig. F(a) consistently assigns high scores to networks with a cell structure consisting of a single convolution layer with a kernel size of 3×3, whereas the progressivity proxy in Fig. F(b) favors networks stacking convolutional layers with kernel sizes of 1×1 and 3×3 with an additional skip connection. These proxies also introduce unnecessary intermediate nodes within a cell structure, whose input or output edges are disconnected. The complexity proxy in Fig. F(d) tends to prefer networks in which all the edges of a cell structure are defined as parametric operations. Such biases prevent us from finding high-performing networks, degrading the NAS performance. We can also observe that the networks chosen by a single zero-cost proxy exhibit low scores for the other proxies frequently, leading to unsatisfactory test accuracies, especially on ImageNet16-120 that includes images depicting more complex scenes and objects. This highlights that exploiting a single proxy solely is insufficient for evaluating a network without training. In contrast, assembling the zero-cost proxies in Fig. F(e) allows us to choose networks highly-ranked for all the proxies, which show high performance across the datasets, without suffering from a specific structural bias.

References

[1] Mohamed S. Abdelfattah, Abhinav Mehrotra, Łukasz Dudziak, and Nicholas D. Lane. Zero-cost proxies for lightweight NAS. In *Int. Conf. Learn. Represent.*, 2021. 1

[2] Bowen Baker, Otkrist Gupta, Ramesh Raskar, and Nikhil Naik. Accelerating neural architecture search using performance prediction. In *Int. Conf. Learn. Represent. Workshop*, 2018. 7

[3] Wuyang Chen, Xinyu Gong, and Zhangyang Wang. Neural architecture search on ImageNet in four GPU hours: A theoretically inspired perspective. In *Int. Conf. Learn. Represent.*, 2021. 2, 5

[4] Ekin D Cubuk, Barret Zoph, Dandelion Mane, Vijay Vasudevan, and Quoc V Le. AutoAugment: Learning augmentation strategies from data. In *IEEE Conf. Comput. Vis. Pattern Recog.*, pages 113–123, 2019. 1

[5] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. ImageNet: A large-scale hierarchical image database. In *IEEE Conf. Comput. Vis. Pattern Recog.*, pages 248–255, 2009. 1, 5

[6] Tobias Domhan, Jost Tobias Springenberg, and Frank Hutter. Speeding up automatic hyperparameter optimization of deep neural networks by extrapolation of learning curves. In *Int. Joint Conf. on Artificial Intell.*, 2015. 7

[7] Xuanyi Dong and Yi Yang. NAS-Bench-201: Extending the scope of reproducible neural architecture search. In *Int. Conf. Learn. Represent.*, 2020. 1, 2, 3, 4, 6, 7

[8] Xavier Glorot and Yoshua Bengio. Understanding the difficulty of training deep feedforward neural networks. In *AISTATS*, pages 249–256, 2010. 3, 6

[9] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Delving deep into rectifiers: Surpassing human-level performance on ImageNet classification. In *Int. Conf. Comput. Vis.*, pages 1026–1034, 2015. 3, 6

[10] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network. In *Adv. Neural Inform. Process. Syst. Workshop*, 2015. 1

[11] Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. Technical report, 2009. 1

[12] Namhoon Lee, Thalaiyasingam Ajanthan, and Philip Torr. SNIP: Single-shot network pruning based on connection sensitivity. In *Int. Conf. Learn. Represent.*, 2019. 5

[13] Guihong Li, Yuedong Yang, Kartikeya Bhardwaj, and Radu Marculescu. ZiCo: Zero-shot NAS via inverse coefficient of variation on gradients. In *Int. Conf. Learn. Represent.*, 2023. 1, 2, 3

[14] Ming Lin, Pichao Wang, Zhenhong Sun, Hesen Chen, Xiuyu Sun, Qi Qian, Hao Li, and Rong Jin. Zen-NAS: A zero-shot nas for high-performance image recognition. In *Int. Conf. Comput. Vis.*, pages 347–356, 2021. 1, 2, 5, 6

[15] Joe Mellor, Jack Turner, Amos Storkey, and Elliot J Crowley. Neural architecture search without training. In *Int. Conf. Mach. Learn.*, pages 7588–7598, 2021. 1, 2, 5

[16] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming

Lin, Natalia Gimelshein, Luca Antiga, et al. PyTorch: An imperative style, high-performance deep learning library. In *Adv. Neural Inform. Process. Syst.*, 2019. 3

[17] Xianbiao Qi, Jianan Wang, Yihao Chen, Yukai Shi, and Lei Zhang. LipsFormer: Introducing lipschitz continuity to vision transformers. In *Int. Conf. Learn. Represent.*, 2023. 6

[18] Ilija Radosavovic, Justin Johnson, Saining Xie, Wan-Yen Lo, and Piotr Dollár. On network design spaces for visual recognition. In *Int. Conf. Comput. Vis.*, pages 1882–1890, 2019. 1

[19] Mark Sandler, Andrew Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen. MobileNetV2: Inverted residuals and linear bottlenecks. In *IEEE Conf. Comput. Vis. Pattern Recog.*, pages 4510–4520, 2018. 1, 6

[20] Yao Shu, Zhongxiang Dai, Zhaoxuan Wu, and Bryan Kian Hsiang Low. Unifying and boosting gradient-based training-free neural architecture search. In *Adv. Neural Inform. Process. Syst.*, pages 33001–33015, 2022. 4

[21] Hidenori Tanaka, Daniel Kunin, Daniel L Yamins, and Surya Ganguli. Pruning neural networks without any data by iteratively conserving synaptic flow. In *Adv. Neural Inform. Process. Syst.*, pages 6377–6389, 2020. 1, 2, 3

[22] Hongyi Zhang, Moustapha Cisse, Yann N Dauphin, and David Lopez-Paz. mixup: Beyond empirical risk minimization. In *Int. Conf. Learn. Represent.*, 2018. 1

[23] Zhihao Zhang and Zhihao Jia. GradSign: Model performance inference with theoretical insights. In *Int. Conf. Learn. Represent.*, 2022. 2, 3

[24] Zhun Zhong, Liang Zheng, Guoliang Kang, Shaozi Li, and Yi Yang. Random erasing data augmentation. In *AAAI*, pages 13001–13008, 2020. 1