NAS-BENCH-ZERO: A LARGE-SCALE DATASET FOR UNDERSTANDING ZERO-SHOT NEURAL ARCHITECTURE SEARCH

Anonymous authors
Paper under double-blind review

ABSTRACT

Zero-shot Neural Architecture Search (ZS-NAS) is a recently developed low-cost NAS framework which identifies top-performer neural architectures from a large candidate pool without training their parameters. Despite its popularity in recent NAS literatures, the effectiveness of ZS-NAS has not been comprehensively understood. Previous works analyze ZS-NAS methods on NAS benchmark datasets such as NAS-Bench-101/201/301 which are initially designed for learning network topology with irregular connections. However, most modern state-of-the-art networks as well as popular classical ones are designed in more conventional, well-established search spaces such as ResNet (RS) and MobileNet (MB) search space. This imposes a significant gap between the benchmark dataset and real-world practice, hindering a deeper understanding of ZS-NAS. In this work, we aim to bridge the gap systematically. First, we collect a novel large-scale dataset termed NAS-Bench-Zero for benchmarking and understanding popular ZS-NAS methods in the conventional RS/MB search spaces. Then the characteristics of these ZS-NAS methods are extensively examined from various aspects. Notably, we find that: 1) the performance of ZS-NAS on NAS-Bench-101/201/301 cannot transfer to RS/MB search spaces; 2) A proxy with higher ranking correlation score may actually perform worse in constrained NAS; 3) existing zero-shot proxies cannot outperform naive proxies such as FLOPs/params in RS/MB search spaces; 4) Top best zero-shot proxies as well as FLOPs/params compensate each other. Based on these new discoveries, we propose i) a novel hybrid zero-shot proxy which outperforms existing ones by a large margin and is transferable among popular search spaces; ii) a new index for better measuring the true performance of ZS-NAS proxies in constrained NAS. Source code and the NAS-Bench-Zero dataset will be released after publication.

1 INTRODUCTION

The success of deep learning models (Bengio et al., 2017) has been demonstrated in various computer vision tasks such as image classification (He et al., 2016), instance segmentation (Long et al., 2015) and object detection (Szegedy et al., 2015). As deep models are widely applied in various applications nowadays, lots of time and money have been spent on adjusting network architectures to find the optimal speed/accuracy trade-off for different hardware platforms. In order to reduce this cost, Neural Architecture Search (NAS) (Zoph & Le, 2016) is invented to automatically discover and design novel architectures, demanding fewer expert efforts and human interactions.

Nevertheless, one of the challenging problems of NAS is its expensive searching cost due to the huge search space and slow network training. A number of methods, including weight sharing (one-shot NAS) (Pham et al., 2018), differentiable search (Liu et al., 2018; Xu et al., 2019), search space reduction (Chen et al., 2020; Chen et al.) and zero-shot NAS (ZS-NAS) (Mellor et al., 2021; Abdelfattah et al., 2021; Chen et al., 2021; Lin et al., 2021), have emerged to conquer this problem. Among these methods, ZS-NAS reduces the searching cost to nearly zero by estimating model performance without training parameters. This makes it training-free which distinguishes itself from previous training-based NAS methods.

As a relatively new research direction, there are many mysteries in ZS-NAS. It is unclear why ZS-NAS can work at all in a training-free way which takes nearly zero knowledge about the data distribution. In (Mellor et al., 2021; Abdelfattah et al., 2021; Chen et al., 2021), ZS-NAS is able to deliver competitive architecture ranking accuracies as good as one-shot NAS. Lin et al. (2021) even show that ZS-NAS, under some mild conditions and constraints, can outperform state-of-the-art (SOTA) training-based NAS methods on the target datasets. These findings are counter-intuitive and require an in-depth investigation.

Just as ImageNet ushered in an era of deep learning, a comprehensive understanding of ZS-NAS is inseparable from large-scale NAS datasets. Previous ZS-NAS works perform ablation studies on existing NAS datasets such as NAS-Bench-101/201/301 (NB101/201/301) (Ying et al., 2019; Zela et al., 2020; Dong & Yang, 2020; Siems et al., 2020). This raises several critical issues. First, NB101/201/301 are originally designed for cell topology search. This means that the networks in these benchmark datasets consist of irregularly connected operations which are rarely used in manually designed networks. In real-world practice, many SOTA networks are built upon conventional well-established designed spaces, such as ResNet (RS) and MobileNet (MB) search space. This imposes a significant gap between the NAS benchmark datasets and real-world best practice. There is no guarantee that a ZS-NAS method working well on existing NAS datasets will also work well in the conventional RS/MB search spaces while the latter ones are more popular in computer vision.

Second, the irregular connection patterns of search spaces defined by existing NAS datasets prevent a deeper understanding of the ablation results. These connection patterns are designed to be computer-friendly rather than human-friendly. Human-brain is good at capturing high-level concepts such as the depth and the width of the networks rather than randomly connected graphs of atomic operations. Therefore, although previous works have conducted considerable amounts of studies on NAS datasets, there is still little structure knowledge we can distill from these experiments.

Third, since the aforementioned NAS datasets share similar design principles, exclusively relying on them might lead to incomprehensive evaluation. For example, we found that NB101/201/301 concentrate on small models in low FLOPs regime. (Ning et al., 2021) find that the model accuracies have a strong correlation with the model FLOPs or number of parameters (params). Such patterns may no longer hold in real-world practice and the transferability of conclusions derived from these datasets remains in doubt.

To this end, this work aims to address all of the above problems systematically. The cornerstone of this work is the collection of a novel large-scale NAS dataset named NAS-Bench-Zero (NBZero). NBZero collects a full coverage of $50,894$ networks in three regularized RS/MB search spaces. Each network is trained on CIFAR10/CIFAR100/ImageNet-16-120, following the best practice of modern training recipes. All networks are endowed with pre-computed popular zero-shot proxy scores. In this way, NBZero approximates the true performance of ZS-NAS in real-world applications as closely as possible.

In NBZero, the building blocks are the well-established ResNet blocks and MobileNetV2 blocks. The collection of networks vary by network depth, width and number of stages. This simulates the practical scenario where the building blocks are manually designed and the macro-shape of the network needs to be searched (Cai et al., 2020; Tan & Le, 2019; 2021; Howard et al., 2019). The search spaces of NBZero are more interpretable than those in the previous NAS datasets. It is much more easier to understand the successes or failures of ZS-Shot NAS in the language of network depth and width, which shreds light on possible future improvements.

Although NBZero is built on regularized search spaces, it is counter-intuitively more challenging than existing NAS datasets. Most zero-shot proxies working well on NB101/201/301 might not achieve the same competitive performance in NBZero. The FLOPs/params/accuracy distribution are more diverse in NBZero. While previous NAS datasets mostly consist of small lightweight models, NBZero additionally includes large models in high accuracy regime. Therefore, NBZero is perpendicular to previous NAS datasets. By studying ZS-NAS on NBZero as well as previous NAS datasets, we will get a more comprehensive understanding of ZS-NAS methods.

After collecting the architecture-accuracy pairs in NBZero, we re-evaluate popular ZS-NAS proxies on NBZero. We find that naive proxies such as FLOPs and params provides a strong baseline in both NBZero and previous NAS datasets. When combining zero-shot proxies with FLOPs/params, we obtain a hybrid zero-shot proxy that outperforms single zero-shot proxies and is comparable to oracle proxy (the ground-true accuracy). We also show that not all combinations of proxies lead to improvements. Therefore, finding the best hybrid proxy is non-trivial.

Finally, as a by-product, we find that conventional indices such as Kendall's τ or Pearson's ρ cannot precisely reflect the true performance of a zero-shot proxy in constrained NAS. We then propose a novel index termed mean best-ranking under constraints (mBR). The definition of mBR is given in Section 3.4.

We summarize our major contributions in this work as follows:

- We publish a large-scale NAS benchmark dataset NBZero for understanding zero-shot NAS in the conventional search spaces.
- We show that the benchmark results on previous NAS datasets might not transfer well to the conventional search spaces. The NBZero together with previous NAS datasets constitute a more comprehensive, less biased evaluation for zero-shot NAS methods.
- Popular zero-shot proxies are extensively analyzed in NBZero from various aspects. Many characteristics of zero-shot proxies are revealed for the first time. Particularly, existing zero-shot proxies share a similar pattern in weighting depth/width importance but this pattern is very different from the oracle proxy (ground-truth accuracy).
- A non-trivial hybrid zero-shot proxy is proposed to achieve the SOTA performance on NBZero as well as previous NAS datasets.
- A novel index mBR is proposed to better measure the true performance of zero-shot proxy in constrained NAS.

2 Related Work

2.1 NAS benchmarks

To boost researchers to verify the effectiveness of NAS methods efficiently, a series of NAS benchmarks, such as NB101 (Ying et al., 2019), NAS-Bench-1shot1 (NB1shot1) (Zela et al., 2020), NB201 (Dong & Yang, 2020), NB301 (Siems et al., 2020), etc, are established. NB101 (Ying et al., 2019) provides the performances of the $423k$ valid architectures in a cell-based search space. To apply NB101 to one-shot NAS and reuse NB101, NB1shot1 (Zela et al., 2020) picks out three sub-spaces of NB101, and a supernet can be easily constructed for these sub-spaces, overcoming NB101's specific channel number rule. Another benchmark, NB201 (Dong & Yang, 2020) offers the performances of all the 15625 architectures in a single-cell search space. But there is a problem that previous tabular benchmarks exhaustively training all architectures in a search space are much smaller than commonly-used ones (e.g. DARTS (Liu et al., 2018) with size over 10^{18}). Hence, NB301 (Siems et al., 2020) is introduced based on the cell-based search space, which adopts a surrogate-based methodology that predicts architecture performances with the performances of about $60k$ anchor architectures. Different from these existed celled-based benchmarks, our NSZero is designed based on chain-structured search space. Also, NSZero is more challenging than existed benchmarks because of their limited FLOPs, parameters and accuracies.

2.2 ZS-NAS

Recently, a new NAS framework called ZS-NAS (Mellor et al., 2021; Abdelfattah et al., 2021) is proposed to further reduce the architecture evaluation cost by training-free architecture searching. Mellor et al. (2021) introduce an zero-shot indicator based on input jacobian correlation. Abdelfattah et al. (2021) combine several pruning proxies, including snip (Lee et al., 2018), grasp (Wang et al., 2020), synflow (Tanaka et al., 2020) and fisher (Theis et al., 2018), to form a new zeros-shot NAS proxy. TE-NAS (Chen et al., 2021) proposes a combination of NTK-score and number of active region as zero-shot proxy. Recently, Zen-NAS (Lin et al., 2021) proposes a re-scaled version of gradient norm of the network named Zen-score as zero-shot proxy and achieves SOTA performance in ImageNet.

3 NAS-BENCH-ZERO

We describe our design of the NBZero in this section. Then we compare NBZero to previous NAS datasets, showing that they have different distribution patterns and the benchmark results on previous ones cannot transfer well to NBZero. Finally we propose the novel mean Best Ranking (mBR) for constrained NAS.

3.1 SEARCH SPACE

The search space of NBZero follows common practice in popular SOTA computer vision models, such as ResNet (He et al., 2016; Bello et al., 2021) and MobileNetV2/EfficientNet (Sandler et al.; Tan & Le, 2019). Moreover, our search spaces consists of 3 or 4 stages and 3 down-sampling parts, and the positions of down-sampling is randomly selected from 3 or 4 stages. Also, the choices of depth or channels are sampled from a series of predefined sets. Particularly, we consider three search spaces:

Search Space SS-R1 SS-R1 uses the same building block as ResNet-18/34, which consist of two 3x3 convolutional layers wrapped inside residual link;

Search Space SS-R2 SS-R2 uses the same building block as ResNet-50, which consists of three convolutional layers of kernel size 1x1+3x3+1x1 respectively;

Search Space SS-MB SS-MB uses the same block as MobileNetV2, which consists of two $1x1$ convolutional layers and one 3×3 depth-wise convolution.

The total number of networks on R1/R2/MB is $14,964$, $10,187$ and $9,743$ respectively. More details can be found in our source code.

3.2 DATASETS AND TRAINING SETTINGS

We train each network in NBZero on three popular image recognition datasets: CIFAR-10, CIFAR-100 Krizhevsky et al. (2009) and ImageNet-16-120. Detail information of these datasets can be found in. For all datasets, we use the official training/testing splitting. For hyper-parameter tuning, we further split the training set into 10% NAS validation set and 90% NAS training set. The validation is used for other NAS methods, like one-shot NAS, not for ZS-NAS methods here. Note that on CIFAR-100 we train all the three search spaces proposed above, but on CIFAR-10 and ImageNet-16-120, we just train SS-R1.

We use SGD to train all models. Following best practice suggested in (He et al., 2018), we use initial learning rate 0.1, cosine learning rate decay, 360 training epochs, weight decay 5e-4, auto-augmentation (Cubuk et al., 2019), mixup (Zhang et al., 2018), label smoothing (Szegedy et al., 2016).

3.3 ZERO-SHOT PROXIES AND META DATA

For easy and fast evaluation of zero-shot proxies, we record several meta data in addition to the accuracy for each network in the dataset. We pre-computed all zero-shot proxies used in this work: grad_norm, plain (Mozer & Smolensky, 1989), snip (Mozer & Smolensky, 1989), grasp (Wang et al., 2020), fisher (Turner et al., 2019; Theis et al., 2018), synflow (Tanaka et al., 2020), jacob_cov (Mellor et al., 2021), ntk (Chen et al., 2021), and zen (Lin et al., 2021). The FLOPs and the number of parameters (params) are saved for budget constrained NAS. In some experiments, FLOPs and params are used as naive zero-shot proxies.

3.4 MEAN BEST RANK (MBR) INDEX FOR CONSTRAINED NAS

We mainly focus on a practical scenario of NAS where we are given inference budget constraints, such as FLOPs and params, and aim to search for architectures with the highest accuracy. Previous works rely on conventional indices such as Kendall's τ or Pearson's ρ to compare the performance of different zero-shot proxies. We find that these conventional indices cannot precisely reflect the true performance of ZS-NAS in constrained NAS. To this end, we propose a novel index termed mean Best Rank (mBR) for this purpose. Suppose we have N architectures in our candidate pool. The set $\mathcal{S}$ $\mathcal{S} = s_1, s_2, \cdots, s_M$ specifies all possible budget constraints we want during the search. The rank r_i is the true rank of architecture found by a ZS-NAS method under budget s_i. The mBR is then defined as:

$$\mathrm{mBR}_{\mathcal{S}} \triangleq \sum_{s \in \mathcal{S}} r_i/(MN) \tag{1}$$

Clearly, **mBR is the smaller the better**. A small mBR means that the corresponding NAS method is able to find top-ranked architectures under budget constraints with high probability.

4 Understanding Zero-Shot NAS

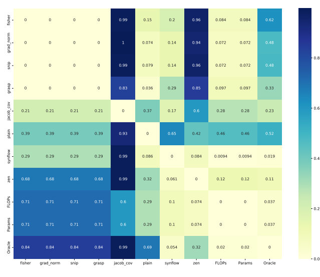
In this section, we prove the validness of mBR. Then the necessity of NSZero is domesticated by comparing it with NB201. As NB101/201/301 follow similar design principles, we focus on NB201 in the following experiments. On top of that, we show the transferability of ZS-NAS on different datasets and search spaces. Finally, improvements of ZS-NAS are presented.

4.1 mBR v.s. Conventional Performance Index in Constrained NAS

Bench	Proxies	Multi-Column				Accuracy (%)				
		LC	KD	SR	mBR	20%	40%	60%	80%	100%
NBZero	Oracle	1.0	1.0	1.0	0.0	75.3	77.1	77.8	78.6	79.1
	FLOPs	**0.79**	**0.704**	**0.881**	0.157	72.2	74.6	76.1	76.5	76.8
	Params	0.626	0.592	0.77	0.213	71.7	72.9	76.6	76.6	76.6
	depth	0.269	0.195	0.271	0.498	68.5	73.2	**76.7**	77.3	**77.3**
	fisher	0.739	0.599	0.795	0.365	72.2	74.5	74.5	75.3	73.4
	grad_norm	0.703	0.498	0.689	0.282	72.2	74.1	76.6	75.0	75.0
	snip	0.789	0.641	0.83	0.356	72.2	73.9	75.5	75.0	73.4
	grasp	-0.17	-0.138	-0.207	0.683	71.5	71.5	71.5	71.5	71.5
	jacob_cov	0.261	0.152	0.228	0.409	65.2	75.1	75.1	75.1	75.1
	plain	-0.034	-0.019	-0.029	0.675	62.2	62.2	74.3	74.3	74.3
	synflow	0.507	0.356	0.514	**0.102**	**73.3**	**75.9**	75.5	**77.6**	76.3
	ntk	-0.162	-0.12	-0.187	0.796	63.4	71.5	71.5	71.5	71.1
	zen	0.596	0.42	0.598	0.108	73.1	75.9	75.5	77.6	76.3

Table 1: Comparison with the state-of-the-art ZS-NAS methods under different assessment criteria.

In Tab. 1, the results of all kinds of ZS-NAS scores under different criterion are reported and the accuracy is under different limitations of FLOPs. It shows that under conventional indices, FLOPs and Params seem to perform better than ZS-NAS scores. For example, while snip achieves almost the best result among ZS-NAS scores (0.789 LC, 0.641 KD τ, 0.83 SpearmanR, 0.025 BR@top0.1% and 0.239% P@top5%), the FLOPs is better than snip (0.79 LC,0.704 KD τ, 0.881 SpearmanR (SR)). However, under the more precise criteria of mBR, FLOPs and Params are worse than some ZS-NAS scores, like zen and synflow, which is also demonstrated by accuracies. We can see from the table, the performance of synflow is better than FLOPs and Params, which corresponds to mBR. That's because BR identities to rank and mBR is the mean of BRs on constrained NAS. Thus, mBR can reflect the performance of constrained NAS better. Also, we find that there is a contradiction between conventional indices and our mBR, which means scores with a high evaluation may not be recognized well under the mBR criteria. It is reasonable because a good performance under conventional indices means good correlation between estimated scores and true accuracies, but good correlation does not mean a great performance of NAS. In Fig. 3, good correlation of fisher, grad_norm, snip, FLOPs and Params can be observed, but they will not obtain great true accuracies showed in Fig. 4(b). Nevertheless, the consistency between mBR and true accuracies can be proved shown in Tab. 1 and Fig. 4(b). In addition, although BR can sometimes represent true accuracies, there are so many K to be selected and which one should we use is very tricky. In sum, the effectiveness of the proposed criteria mBR is demonstrated.


(a) NB201

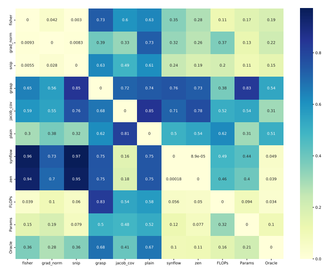

(b) NBZero

Figure 1: mBR between GT (Oracle), FLOPs/Params and ZS-NAS.

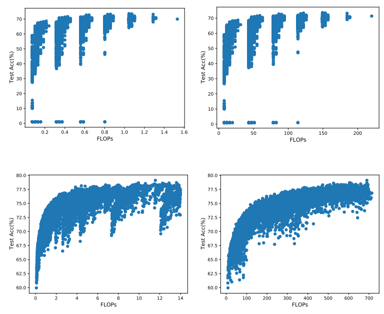
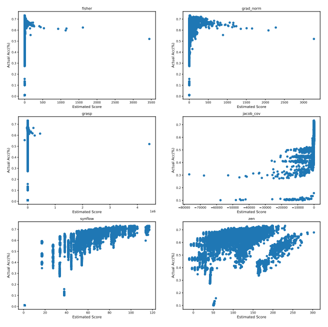


Figure 2: The distribution between GT (Oracle) and FLOPs/Params.

4.2 Transfer from NAS-Bench-201 to NAS-Bench-Zero

From Fig 1 showing the mBR between true accuracies (Oracle), FLOPs, Params and ZS-NAS on both NB201 and NBZero, different phenomenon can be found: 1) both of the best NS-Zero methods is synflow, but other methods, like fisher, grad_norm grasp and jacob_cov, which perform pretty poor on NS201 are not so bad on NSZero; 2) FLOPs and Params perform better than synflow, but on NBZero, the conclusion is reverse, which both demonstrate the necessity of NBZero. The reasons lead to these phenomenon are the different FLOPs, Params and model accuracy, which are represented in Fig. 2. On the Top of Fig. 2 is NB201 and the bottom one is NBZero. From it, we can observe that the FLOPs and Params on NB201 are limited, only from 0 to 1.6 and 0 to 200 respectively, but on NBZero they are not so limited, from 0 to 14 and 0 to 700 respectively, which are commonly used; also, maximum accuracy on NBZero is higher than NB201 (79.1 vs 73.3). Both of them show NSZero is more challenging than NB201.


(a) NB201

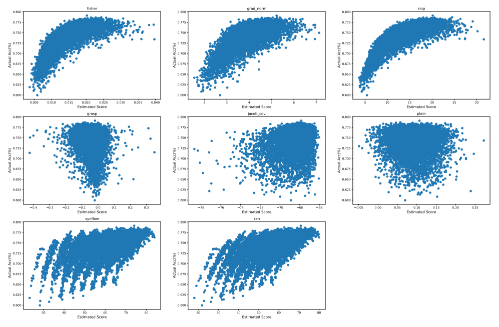

(b) NBZero

Figure 3: The distribution between estimated scores and true accuracies.

In Fig. 3, it is apparent that the distribution of scores on both benchmarks are different. The distribution on NB201 is a little bit strange, but on NBZero that is more regular. On NBZero, there are four groups which have the same shape: the first is fisher, grad_norm and snip; the second is grasp and plain; the third is jacob_cov and ntk; and the forth is synflow and zen.

4.3 The performance of Zero-Shot Proxy in Constrained NAS

In order to observe and analyze constrained NAS, evaluation will be done from three aspects: FLOPs, Params and depth.

FLOPs: Aimed at assessing the whole effectiveness of NAS searching under the different limitations of FLOPs, 5 search subspaces are created based on the original search space under limitations of FLOPs. The limitations are set evenly between minimum and maximum of FLOPs of the original search space. In Fig. 4, both BRs and the best accuracies of different scores on five search subspaces are given. We find that there's an apparent gap between architectures searched by scores and the

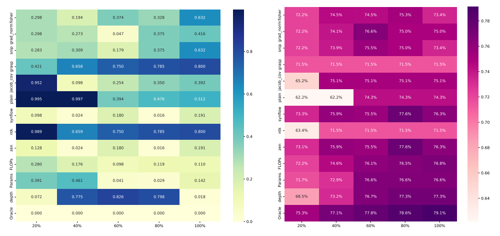

(a) BR (b) Oracle

Figure 4: The accuracy and mBR under different limitations of FLOPs.

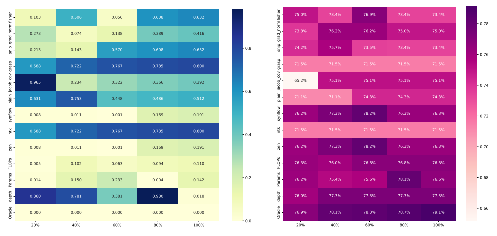

(a) BR (b) Oracle Accuracy

Figure 5: The accuracy and mBR under different limitations of Params.

true best architectures in each search subspace. Also, zen and synflow perform better than FLOPs, Params and depth, especially in the search subspace where FLOPs is $< 20\%$.

Params: Attempting to evaluate the whole effectiveness of NAS searching under the different limitations of Params, 5 search subspaces are created based on the original search space under limitations of Params. The limitations are set evenly between minimum and maximum of Params of the original search space. In Fig. 5, both BRs and the best accuracies of different scores on five search subspaces are given. We also find that there's an apparent gap between architectures searched by scores and the true best architectures in each search subspace. And accuracies between different search subspaces are not obviously different. Here, zen and synflow perform the similar as FLOPs, Params and depth.

Depth: In order to measure the whole effectiveness of NAS searching under the different limitations of depths, 5 search subspaces are created based on the original search space under limitations of depths. The limitations are set evenly between minimum and maximum of depths of the original search space. In Fig. 6, both BRs and the best accuracies of different scores on five bsearch subspaces are given. We find that there's an apparent gap between architectures searched by scores and the true best architectures in each search subspace. Additionally, zen and synflow perform a little worse than FLOPs, but perform better than Params and Depths.

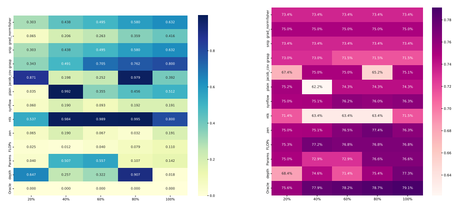

(a) BR (b) Oracle Accuracy

Figure 6: The accuracy and mBR under different limitations of Depths.

4.4 Evaluation of Transferability of ZS-NAS

In order to evaluate the transferability of ZS-NAS, two aspects, including different datasets and search spaces, are considered.

Dataset	Criteria	fisher	grad_norm	snip	grasp	jacob_cov	plain	synflow	ntk	zen	FLOPs	Params	Oracle
CIFAR-10	mBR	0.338	0.488	0.459	0.893	0.895	0.249	0.212	0.482	**0.133**	0.142	0.779	0.0
	Acc (%)	94.9	94.2	94.8	92.6	92.6	95.1	94.7	94.9	95.3	**95.5**	93.4	95.7
CIFAR-100	mBR	0.365	0.282	0.356	0.683	0.409	0.675	**0.102**	0.796	0.108	0.157	0.213	0.0
	Acc (%)	73.4	75.0	73.4	71.5	75.1	74.3	76.3	71.5	76.3	**76.8**	76.6	79.1
ImageNet -16-120	mBR	0.312	0.182	0.175	0.766	0.705	0.345	**0.093**	0.672	**0.093**	0.286	0.277	0.0
	Acc (%)	50.2	51.9	51.0	48.2	45.7	49.3	**51.9**	46.0	**51.9**	50.0	51.3	53.0

Table 2: Comparison with ZS-NAS methods on different datasets.

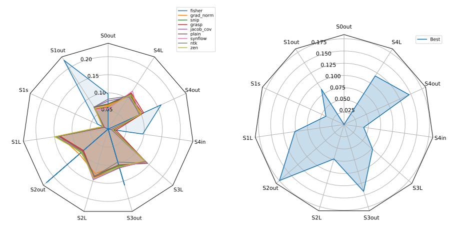
Performances on Various Datasets: Here, three datasets, including CIFAR-10, CIFAR100 and ImageNet-16-120, are applied to evaluate. From the Tab. 2, we can observe that on various datasets, ZS-NAS performs pretty well and the best scores are robust. On the three datasets, synflow and zen performs the best, which indicates the stability of ZS-NAS scores on different datasets and their transference. However, FLOPs and Params do not perform stably: on CIFAR-10, FLOPs performs the similar as zen (0.142 vs 0.133 on mBR), and also on CIFAR-100, FLOPs performs comparably with synflow (0.157 vs 0.102 on mBR), but on ImageNet-16-120, they perform worse than synflow and zen (0.286 vs 0.093 on mBR).

Search Space	Criteria	fisher	grad_norm	snip	grasp	jacob_cov	plain	synflow	ntk	zen	FLOPs	Params	Oracle
Bench-Zero	mBR	0.365	0.282	0.356	0.683	0.409	0.675	**0.102**	0.796	0.108	0.157	0.213	0.0
-R1	Acc (%)	73.4	75.0	73.4	71.5	75.1	74.3	76.3	71.5	76.3	**76.8**	76.6	79.1
Bench-Zero	mBR	0.222	0.269	0.154	0.475	0.342	0.226	0.057	0.934	**0.045**	0.132	0.253	0.0
-R2	Acc (%)	77.7	78.0	78.0	78.9	78.7	78.5	**79.3**	71.8	**79.3**	78.0	79.1	80.0
Bench-Zero	mBR	0.14	0.1	0.142	0.389	0.11	0.409	0.125	0.982	0.125	**0.109**	0.251	0.0
-MB	Acc (%)	73.4	74.0	73.9	72.9	**74.6**	71.9	74.3	58.4	74.3	74.0	74.3	75.7

Table 3: Comparison with ZS-NAS methods on different search space.

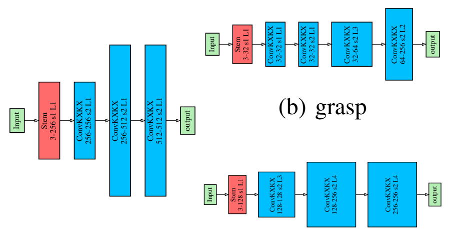
Performances on Assorted Search Space: In order to analyze the performance of ZS-NAS on different structures, three search space, including Bench-Zero-R1, Bench-Zero-R2 and Bench-Zero-MB, are applied to assess. From the Tab. 3, we can observe that on assorted search space, the best scores are almost the same. On the three search space, synflow and zen have the better performance than other scores, except for Bench-Zero-MB where jacob_cov performs the best, but synflow and zen have the similar performance with it. However, FLOPs and Params still have unstably performance: on Bench-Zero-MB, FLOPs performs better than scores, but on Bench-Zero-R2, they perform pretty worse than zen (0.132 vs 0.045 on mBR), which indicates the stability of ZS-NAS scores on different search space and their transference.

4.5 Factor Analysis



(a) Reduction Cell (b) Oracle

Figure 7: The importance of the various attributes of architectures.



(a) snip (b) grasp (c) zen

Figure 8: Best architectures searched by ZS-NAS.

4.5.1 Which elements affect ZS-NAS

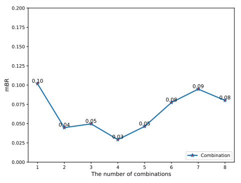
As we all know, depth and width affect the performance of networks, but which one will affect the estimation of ZS-NAS over networks? In order to figure out this question, XGBoost (Chen & Guestrin, 2016) is applied to analyze the importance of the inputs, including input channels S$i$in, output channels S$i$out, stride s and depth L of the ith stage Si. As we can see from Fig. 7(a), all the ZS-NAS scores, except fisher, are affected by depth greatly and they almost have the same shape in radar chart. The most important factors include S1L, S2L, S3L and S4L, which are the depth of each stage. In contrast, fisher is influenced by channels largely and its important factors mainly contain S1out, S4out, S2out and S3out, which are the width of each stage.

In order to improve ZS-NAS scores, it is necessary to find out which factors mainly contribute to true performance. So XGBoost (Chen & Guestrin, 2016) is applied to analyze the importance of the inputs too. As shown in Fig. 7(b), the true accuracy is primarily affected by width, which is different from most of scores shown in Fig. 7(a). The most important factors include S2out, S3out, S4out and S4L, which are almost the width of each stage, which indicates there are still lots of potential to improve ZS-NAS scores. If scores can computed mainly by these factors, they will be improved.

4.5.2 Bias of ZS-NAS Scores

By observing the best architectures searched by ZS-NAS scores, characteristics are of architectures indicated by scores are presented in Fig. 8, where the width of each stage represents the width of its out channels and the length of each stage shows the depth of it. Fig. 8 shows that snip has an excessive preference for wide channels and grasp prefers to narrow architectures, perhaps which is why they perform badly. Moreover, the architecture search by zen is neither too wide nor too deep and its depth is progressive, which matches the common experience of networks designing.

4.6 Hybrid Zero-Shot Proxy


(a) R1

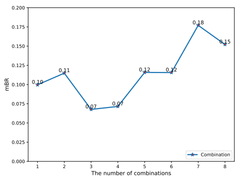
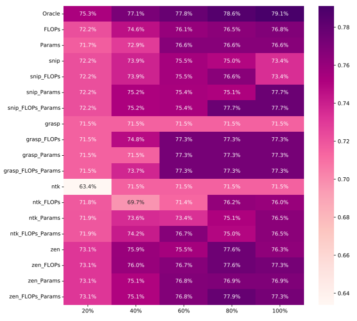

(b) MB

Figure 9: The best result in each number of the combination of scores.


(a)

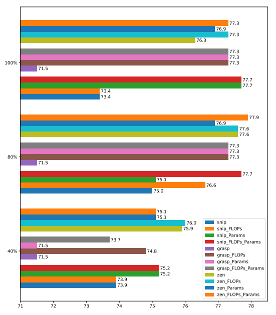

(b)

Figure 10: The results of architectures searched by original scores and combined scores under different limitations of FLOPs.

Combination with Other Scores: Fig 9(a) shows the mBR of traversing the combination of multi scores on NBZero-R1 and only the best result in each number of the combination of scores is presented. It can get the same conclusion as Abdelfattah et al. (2021) that voting (associating with scores) can improve performance: the mBR of combination of 8 scores is better than only using original best scores, but combining all scores is not optimal. Namely, not more is better. Just the combination of 4 kinds of scores achieve the best mBR 0.08%. The four types of scores are jacob_cov, zen, snip and synflow, which are different in the distribution between estimated scores and true accuracies shown in Fig 3: snip is monotonically increasing and the other are like rhombus. It is meaningful because intuitively, mixing different useful characteristics can boost performance. Also, other new scores combined by 2 kinds of scores which have the similar mBR all contain snip and synflow, proving the conclusion. As we all know, ZS-NAS does not use the true performance of networks, but we use performance to find the best combination here. So in order to domesticate the validness of combination, we apply the combinations found on NBZero-R1 to NBRzero-MB directly, and the results show in Fig 9(b). From it, we find that the combinations of 4 types of scores is better than the original score, even if it is not the best one.

Combination with Computation Metrics: Besides combination with other scores, combining with computation metrics, like FLOPs and Params, can also enhance the performance. However, when combine scores with FLOPs and Params, there will be a great improvement in performance. From the Fig. 10(b), we can see that performances of nearly every scores in every subspace are boosted after combinations. For example, zen_FLOPs, the combination of zen and FLOPs, is improved from the original results of zen $73.1\%, 75.9\%, 75.5\%, 77.6\%$ and 76.3% to the combined results $73.1\%, 76.0\%, 76.7\%, 77.6\%$ and 77.3%. Also, even if scores, like grasp, which are worse than FLOPs, they will perform better than FLOPs on the whole after combining with FLOPs, demonstrating roles of ZS-NAS score.

5 Conclusions

In the paper, a new NAS benchmark called NAS-Bench-Zero based on layer-by-layer search space is created to analyze zero-shot better. Also, a new assessment criterion, which can better measure the performance than the existed criteria, is proposed to assess constrained NAS. Within the analysis, ZS-NAS is proved promising, but it still have a long way to go to approach true performance. To help enhance ZS-NAS, several suggestions and directions are proposed to develop ZS-NAS.

REFERENCES

Mohamed S Abdelfattah, Abhinav Mehrotra, Łukasz Dudziak, and Nicholas D Lane. Zero-cost proxies for lightweight nas. In *ICLR*, 2021.

Irwan Bello, William Fedus, Xianzhi Du, Ekin D. Cubuk, Aravind Srinivas, Tsung-Yi Lin, Jonathon Shlens, and Barret Zoph. Revisiting ResNets: Improved Training and Scaling Strategies. *arXiv:2103.07579 [cs]*, 2021.

Y. Bengio, I. Goodfellow, and A. Courville. *Deep learning*. Citeseer, 2017.

Han Cai, Chuang Gan, Tianzhe Wang, Zhekai Zhang, and Song Han. Once-for-All: Train One Network and Specialize it for Efficient Deployment on Diverse Hardware Platforms. In *ICLR*, 2020.

Hanlin Chen, Baochang Zhang, Song Xue, Xuan Gong, Hong Liu, Rongrong Ji, and David Doermann. Anti-bandit neural architecture search for model defense. In *ECCV*.

Hanlin Chen, Baochang Zhang, Xiawu Zheng, Jianzhuang Liu, David Doermann, Rongrong Ji, et al. Binarized neural architecture search. In *AAAI*, 2020.

Tianqi Chen and Carlos Guestrin. Xgboost: A scalable tree boosting system. In *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining*, 2016.

Wuyang Chen, Xinyu Gong, and Zhangyang Wang. Neural architecture search on imagenet in four gpu hours: A theoretically inspired perspective. *ICLR*, 2021.

Ekin D. Cubuk, Barret Zoph, Dandelion Mane, Vijay Vasudevan, and Quoc V. Le. AutoAugment: Learning Augmentation Policies from Data. In *CVPR*, pp. 113–123, 2019.

Xuanyi Dong and Yi Yang. Nas-bench-201: Extending the scope of reproducible neural architecture search. In *International Conference on Learning Representations (ICLR)*, 2020. URL `https://openreview.net/forum?id=HJxyZkBKDr`.

K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *CVPR*, 2016.

Tong He, Zhi Zhang, Hang Zhang, Zhongyue Zhang, Junyuan Xie, and Mu Li. Bag of Tricks for Image Classification with Convolutional Neural Networks. *arXiv:1812.01187 [cs]*, 2018.

Andrew Howard, Mark Sandler, Grace Chu, Liang-Chieh Chen, Bo Chen, Mingxing Tan, Weijun Wang, Yukun Zhu, Ruoming Pang, Vijay Vasudevan, Quoc V. Le, and Hartwig Adam. Searching for MobileNetV3. In *ICCV*, 2019.

Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. Technical report, Citeseer, 2009.

Namhoon Lee, Thalaiyasingam Ajanthan, and Philip HS Torr. Snip: Single-shot network pruning based on connection sensitivity. In *ICLR*, 2018.

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 deep image recognition. In *2021 IEEE/CVF International Conference on Computer Vision, ICCV 2021*, 2021.

H. Liu, K. Simonyan, and Y. Yang. Darts: Differentiable architecture search. In *ICLR*, 2018.

J. Long, E. Shelhamer, and T. Darrell. Fully convolutional networks for semantic segmentation. In *CVPR*, 2015.

Joe Mellor, Jack Turner, Amos Storkey, and Elliot J Crowley. Neural architecture search without training. In *ICML*, 2021.

Michael C Mozer and Paul Smolensky. Skeletonization: A technique for trimming the fat from a network via relevance assessment. In *Advances in neural information processing systems*, 1989.

Xuefei Ning, Changcheng Tang, Wenshuo Li, Zixuan Zhou, Shuang Liang, Huazhong Yang, and Yu Wang. Evaluating efficient performance estimators of neural architectures. In *NIPS*, 2021.

H. Pham, M. Y. Guan, B. Zoph, Q. V. Le, and J. Dean. Efficient neural architecture search via parameter sharing. In *ICML*, 2018.

Mark Sandler, Andrew Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen. Mobilenetv2: Inverted residuals and linear bottlenecks. In *CVPR*.

Julien Siems, Lucas Zimmer, Arber Zela, Jovita Lukasik, Margret Keuper, and Frank Hutter. Nas-bench-301 and the case for surrogate benchmarks for neural architecture search. *arXiv preprint arXiv:2008.09777*, 2020.

C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich. Going deeper with convolutions. In *CVPR*, 2015.

Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jonathon Shlens, and Zbigniew Wojna. Rethinking the Inception Architecture for Computer Vision. In *CVPR*, pp. 2818–2826, 2016.

Mingxing Tan and Quoc Le. EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks. In *ICML*, pp. 6105–6114, 2019.

Mingxing Tan and Quoc V. Le. EfficientNetV2: Smaller Models and Faster Training. In *ICML*, 2021.

Hidenori Tanaka, Daniel Kunin, Daniel LK Yamins, and Surya Ganguli. Pruning neural networks without any data by iteratively conserving synaptic flow. *arXiv preprint arXiv:2006.05467*, 2020.

Lucas Theis, Iryna Korshunova, Alykhan Tejani, and Ferenc Huszár. Faster gaze prediction with dense networks and fisher pruning. 2018.

Jack Turner, Elliot J Crowley, Michael O'Boyle, Amos Storkey, and Gavin Gray. Blockswap: Fisher-guided block substitution for network compression on a budget. 2019.

Chaoqi Wang, Guodong Zhang, and Roger Grosse. Picking winning tickets before training by preserving gradient flow. 2020.

Y. Xu, L. Xie, X. Zhang, X. Chen, G. Qi, Q. Tian, and H. Xiong. Pc-darts: Partial channel connections for memory-efficient differentiable architecture search. In *ICLR*, 2019.

Chris Ying, Aaron Klein, Eric Christiansen, Esteban Real, Kevin Murphy, and Frank Hutter. Nas-bench-101: Towards reproducible neural architecture search. In *International Conference on Machine Learning*, pp. 7105–7114. PMLR, 2019.

Arber Zela, Julien Siems, and Frank Hutter. Nas-bench-1shot1: Benchmarking and dissecting one-shot neural architecture search. In *ICLR*, 2020.

Hongyi Zhang, Moustapha Cisse, Yann N. Dauphin, and David Lopez-Paz. Mixup: Beyond Empirical Risk Minimization. In *ICLR*, 2018.

B. Zoph and Q. V. Le. Neural architecture search with reinforcement learning. In *ICLR*, 2016.