Unleashing the Power of Gradient Signal-to-Noise Ratio for Zero-Shot NAS

Zihao Sun[1,3†], Yu Sun[2,3†], Longxing Yang[1,3], Shun Lu[1,3], Jilin Mei[1], Wenxiao Zhao[2,3*], Yu Hu[1,3*]
[1]Research Center for Intelligent Computing Systems,
Institute of Computing Technology, Chinese Academy of Sciences
[2]Key Laboratory of Systems and Control,
Academy of Mathematics and Systems Science, Chinese Academy of Sciences
[3]University of Chinese Academy of Sciences
{sunzihao18z, yanglongxing20b, lushun19s, meijilin, huyu}@ict.ac.cn, {sunyu211}@mails.ucas.ac.cn, {wxzhao}@amss.ac.cn

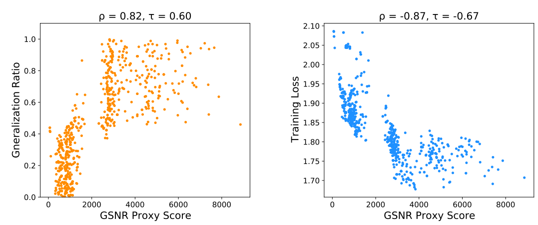
Abstract

Neural Architecture Search (NAS) aims to automatically find optimal neural network architectures in an efficient way. Zero-Shot NAS is a promising technique that leverages proxies to predict the accuracy of candidate architectures without any training. However, we have observed that most existing proxies do not consistently perform well across different search spaces, and are less concerned with generalization. Recently, the gradient signal-to-noise ratio (GSNR) was shown to be correlated with neural network generalization performance. In this paper, we not only explicitly give the probability that larger GSNR at network initialization can ensure better generalization, but also theoretically prove that GSNR can ensure better convergence. Then we design the ξ-based gradient signal-to-noise ratio (ξ-GSNR) as a Zero-Shot NAS proxy to predict the network accuracy at initialization. Extensive experiments in different search spaces demonstrate that ξ-GSNR provides superior ranking consistency compared to previous proxies. Moreover, ξ-GSNR-based Zero-Shot NAS also achieves outstanding performance when directly searching for the optimal architecture in various search spaces and datasets. The source code is available at https://github.com/Sunzh1996/Xi-GSNR.

1. Introduction

Neural architecture search (NAS) [16] is a technique that automates the design of neural network architectures, easing the burden of human trial and error [19, 44]. The main challenge of NAS is to evaluate the performance of each candidate architecture in a given search space. Early approaches that trained each architecture separately until convergence were time-consuming and resource-intensive

† Equal contribution. * Corresponding authors.



(a) *GSNR* vs. Generalization (b) *GSNR* vs. Convergence

Figure 1. (a) Networks with larger *GSNR* tend to have a higher generalization ratio, indicating better generalization. Generalization Ratio: the ratio between one-step validation loss and training loss reduction. (b) Networks with larger *GSNR* have lower training loss, thus better convergence.

[3, 68, 42, 41]. To improve the search efficiency, ENAS [38] proposed to share the same operation weights in a super-network. This reduces the training cost to only one super-network, namely One-Shot NAS [4]. Gradient-based NAS [32, 64, 8, 57, 14, 11, 9, 58] and Sampling-based NAS [17, 10, 61, 65, 62] are two mainstream paradigms for training super-networks. However, they still require training before evaluating each candidate architecture, suffering from extra search overhead.

In order to eliminate the need for training in NAS, Zero-Shot NAS [35] was developed to evaluate network performance without any training, thus significantly promoting search speed. To achieve this, a series of proxies are designed to predict the performance of a given candidate architecture. Some of these proxies are based on empirical inspiration from the pruning-at-initialization literature, using saliency metrics such as SNIP [23], GraSP [54], SynFlow [51] and so on. Whereas, recent studies [1, 36] have shown that these proxies perform poorly in correlating with network performance. Other Zero-Shot proxies are theoretically designed from the deep learning theory of neural networks [35, 30, 46, 7, 66, 28]. For example, some proxies [35, 30] analyzed the relationship between the linear region [40] and network performance, some leveraged Neural Tangent Kernel (NTK) [21] to assess the predictive performance of a network, while others [66, 28] analyzed the gradients of the sample-wise optimization [2]. Nevertheless, most existing proxies cannot consistently perform well across different search spaces, and are less concerned with generalization, motivating us to address this issue.

As revealed in recent studies [47, 53], architectures with faster convergence are preferred by NAS algorithms, but may not guarantee good generalization performance. This insight motivates us to consider both generalization and convergence when designing proxy indicators. The gradient signal-to-noise ratio (*GSNR*) was proposed in [33], and is defined as the ratio between the squared mean and variance of the parameter gradient. In this paper, we clearly give the probability guarantee of good generalization is positively correlated with higher *GSNR*. Moreover, we have further provided theoretical evidence of a strict correlation between larger *GSNR* and better network convergence. To verify our theory, we conduct a toy experiment in NAS-Bench-201 on CIFAR-10. We randomly sample 800 architectures and train them for one epoch. As illustrated in Fig.1, *GSNR* is positively correlated with the generalization ratio and negatively correlated with the training loss. This result suggests that *GSNR* can effectively capture the generalization and convergence at network initialization.

In this paper, we further develop a novel Zero-Shot NAS proxy called ξ-based gradient signal-to-noise ratio (ξ-*GSNR*), which introduces a fairly small ξ term to smooth the variance of the parameter gradient. The network performance is then predicted by ξ-*GSNR* proxy at network initialization without any training. A larger ξ-*GSNR* value indicates better network performance. We conduct experiments in various NAS-Benches and observe that the ξ-*GSNR* proxy exhibits significant ranking consistency with network performance, surpassing existing Zero-Shot NAS proxies. In addition, when directly searching for architectures in different search spaces, the accuracy has also been boosted due to the effectiveness of ξ-*GSNR* proxy.

To this end, we summarize our contributions as follows:

- We provide theoretical proof that gradient signal-to-noise ratio (*GSNR*), which is the ratio between the squared mean and variance of the parameter gradient, is positively correlated with the generalization and convergence of the network at initialization.

- We develop a novel Zero-Shot NAS proxy called ξ-based gradient signal-to-noise ratio (ξ-*GSNR*) by introducing a fairly small ξ term to smooth the variance of the parameter gradient. ξ-*GSNR* is more accurate than vanilla *GSNR* in predicting network performance.

- Our experimental results indicate that ξ-*GSNR* proxy achieves significant ranking consistency with network accuracy in various NAS-Benches. In addition, the performance of the searched architecture has also been improved when involving ξ-*GSNR* proxy during the searching procedure of NAS.

2. Related Works

We reviewed related works of One-Shot NAS and Zero-Shot NAS from the perspective of performance evaluation.

One-Shot NAS. To address the high search overhead of traditional heuristic methods [3, 68, 42, 41], which typically require training each architecture to convergence, One-Shot NAS [4] presents a simpler alternative. This approach involves training only one super-network, allowing all sub-networks can be evaluated through weight-sharing [38]. DARTS [32] and SPOS [17] thereafter became two popular paradigms for super-network training. Specifically, DARTS and its variants [64, 8, 57, 14, 11, 9, 58] alternately optimize architecture parameters and network weights. The final architecture is derived from the optimal architecture parameters on each edge in a cell. Path sampling-based approaches [10, 61, 65, 62] focus on ensuring the fairness and robustness of super-network training. In this way, the performance of sub-networks is evaluated by inheriting the super-network weights. Despite alleviating the cost of evaluating candidate architectures, One-Shot NAS still bears the burden of super-network training.

Zero-Shot NAS. To completely liberate NAS from training, Zero-Shot NAS [35] was proposed to evaluate the network performance without any training. By designing a series of Zero-Shot NAS proxies to estimate the performance of candidate architectures at initialization, the search speed has been significantly promoted. We broadly categorize existing proxies as empirically designed or theoretically designed ones. On the one hand, some Zero-Shot NAS proxies are designed to identify the saliency metric from the perspective of pruning-at-initialization, including SNIP [23], GraSP [54], SynFlow [51] and so on. However, some studies [1, 36] have shown that these training-free proxies are not robust in ranking candidate architectures across different search spaces. Moreover, the simplest metric, such as the number of parameters or FLOPs, can outperform empirically inspired proxies [1, 36].

On the other hand, some Zero-Shot NAS proxies are inspired by the deep learning theory of neural networks. For instance, linear region analysis [40] helps to assess network predictive performance. Different from NASWOT [35], Zen-NAS [30] defined the expected Gaussian complexity to measure network expressivity. As another example, Neural Tangent Kernel (NTK) [21] discusses the training dynamics of infinite-width neural network. NASI [46] leveraged the capability of Neural Tangent Kernel (NTK) to characterize network performance at initialization. TE-NAS [7] traded off the theory of Neural Tangent Kernel (NTK) and the linear regions of networks to evaluate its trainability and expressivity. NNPG [37] approximated network performance by fitting the kernel regression parameters. GradSign [66] and ZiCo [28] analyzed the sample-wise gradient optimization of a network to quantify its performance. However, as illustrated in [47, 53] that the architecture with faster convergence does not necessarily generalize better, indicating that both generalization and convergence are crucial in characterizing network performance. This motivates us to design a novel Zero-Shot NAS proxy, taking into account of generalization and convergence simultaneously.

3. Methodology

We first give the preliminary of gradient signal-to-noise ratio (*GSNR*) illustrated in [33] (Sec.3.1). Then we analyze the property of *GSNR* (Sec.3.2). We further provide theoretical evidence that *GSNR* has a strong correlation with network generalization (Sec.3.3) and convergence (Sec.3.4). Finally, to enhance the performance, we develop a novel Zero-Shot NAS proxy named ξ-*GSNR* (Sec.3.5).

3.1. Preliminary

Let $\theta \in \mathbb{R}^P$ represent the parameters of the neural network $f(\boldsymbol{x}, \theta)$ function, θ_j denotes the j-th parameter θ. Dataset $\mathcal{D} = \{(x_i, y_i), i = 1, ..., n\}$, $(x_i, y_i) \sim \mathcal{Z}$. θ is optimized with loss function $\mathcal{L}$ via gradient descent:

$$\mathrm{g}_{\mathcal{D}}(\theta_j) = \frac{1}{n}\sum_{i=1}^{n} \mathrm{g}(x_i, y_i, \theta_j) := \frac{1}{n}\sum_{i=1}^{n} \frac{\partial \mathcal{L}(y_i, f(x_i, \theta))}{\partial \theta_j} \tag{1}$$

The gradient signal-to-noise ratio (*GSNR*) of one parameter θ_j is defined as:

$$GSNR(\theta_j) := \frac{(\mathbb{E}_{(x,y)\sim\mathcal{Z}}\mathrm{g}(x, y, \theta_j))^2}{Var_{(x,y)\sim\mathcal{Z}}\mathrm{g}(x, y, \theta_j)} \tag{2}$$

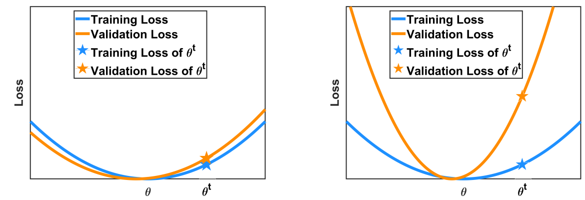
GSNR(θ_j) measures the similarity of parameter gradients across different training samples. Large *GSNR*(θ_j) implies that the parameter gradients tend to be in the similar direction, leading to better generalization and convergence.

3.2. Property of GSNR

To facilitate the proof, we introduce the gradient signal-to-noise ratio of parameter θ_j via the output of network:

$$gsnr(\theta_j) := \frac{(\mathbb{E}_{(x,y)\sim\mathcal{Z}}\mathrm{g}'(x, \theta_j))^2}{Var_{(x,y)\sim\mathcal{Z}}\mathrm{g}'(x, \theta_j)} \tag{3}$$

where $\mathrm{g}'(x, \theta_j) = \frac{1}{n}\sum_{i=1}^{n}\frac{\partial f(x_i, \theta)}{\partial \theta_j}$. Next, we deduce the correlation between *GSNR*(θ_j) and *gsnr*(θ_j) is positive, and thus can associate network generalization and convergence with *gsnr*(θ_j) at initialization.



(a) The same flatness of losses (b) The different flatness of losses

Figure 2. (a) For cases where the loss function flatness of the training set and validation set is similar, the difference between their loss values (red star and blue star) corresponding to the fixed parameter θ^t is small. (b) For cases where the loss function flatness of the training set and validation set is not similar, their loss values (red star and blue star) differ significantly for fixed parameter θ^t.

Theorem 1 *$\forall 1 > \epsilon > 0$, $j = 1, ..., P$, $\exists M_1$ such that with probability at least $(1 - \epsilon)$ over randomly initialized parameters θ^0,*

$$\frac{M_1}{(16M^2 - M_1) + \frac{16M^2}{GSNR(\theta_j^0)}} \leq gsnr(\theta_j^0) \tag{4}$$

where M is the upper bound of the output.

The proof details are provided in Appendix A.1. Theorem 1 shows that with probability arbitrarily close to 1 over random network initialization, the larger *GSNR*(θ_j^0), the larger *gsnr*(θ_j^0) is. In the following sections, we will prove the generalization and convergence effect brought by the increase of *gsnr*(θ_j^0), and thus associate it with *GSNR*(θ_j^0).

3.3. Generalization Analysis

As illustrated in [33] that at each step of gradient descent, the larger *GSNR* implies a greater proximity between the expected reductions in validation loss and training loss. Unfortunately, this does not indicate the probability that the loss reduction of any given validation set will be similar to that of the training set. After several steps of gradient descent, the accumulation of bias can lead to substantial discrepancies between the training loss and validation loss. As a result, we provide a novel and comprehensive proof of the generalization guarantee.

The maximum eigenvalue of the training loss Hessian matrix is a measurement of the flatness of the loss landscape, which is highly correlated with the generalization ability of a neural network [43, 24, 22, 12]. When the minimum point of validation loss is similar to that of training loss and both losses are flat, it can ensure that the difference between the two losses is minimal (see Fig.2(a)). On the other hand, if the validation loss is sharp, the difference between the two losses will be significant, resulting in poor generalization (see Fig.2(b)). Next, we will prove that when *GSNR* is larger, it can ensure with a higher probability that the Hessian maximum eigenvalue of training loss is upper bounded by that of validation loss.

Assumption 1 *$gsnr(\theta_j)$ and $\mathbb{E}_{\mathcal{D}\sim\mathcal{Z}^n}\nabla_\theta f_{\mathcal{D}}(\theta)$ are continuous functions and robust, which means that there is a neighborhood around fixed initialization parameters θ^0 such that for any two parameters α and β, $|gsnr(\alpha_j)-gsnr(\beta_j)|, j = 1, 2...P$ and $||\mathbb{E}_{\mathcal{D}\sim\mathcal{Z}^n}\nabla_\theta f_{\mathcal{D}}(\alpha) - \mathbb{E}_{\mathcal{D}\sim\mathcal{Z}^n}\nabla_\theta f_{\mathcal{D}}(\beta)||$ are small.*

This assumption implies that after gradient descent, $gsnr(\theta_j^0)$ and $gsnr(\theta_j^t)$ are close. Therefore, we can analyze the generalization and convergence using $gsnr(\theta_j^t)$ to approximate that of $gsnr(\theta_j^0)$.

Theorem 2 *Under Assumption 1, for fixed initialization parameters θ^0, if $\nabla^2_\theta\mathcal{L}_{\mathcal{D}}(\theta^t)$ is semi-positive definite matrix, $\mathbb{E}_{(x,y)\sim\mathcal{Z}}|(f(x,\theta^t)-y)|$ is small enough, $\forall t = 1, 2...$, there exist $0 < \alpha_t < 1$ and $\frac{1}{\sqrt{n\alpha_t gsnr(\theta_j^0)}} < r < 1, j = 1, 2...P$, such that ,*

$$\lambda_{max}(\nabla^2_\theta\mathcal{L}_{\mathcal{D}'}(\theta^t)) \leq \frac{n(1+r)^2}{(1-r)^2}\lambda_{max}(\nabla^2_\theta\mathcal{L}_{\mathcal{D}}(\theta^t)) \quad (5)$$

with probability at least

$$1 - \sum_{j=1}^{P}\frac{2n}{r^2\alpha_t gsnr(\theta_j^0)} \quad (6)$$

over randomly chosen possible distributions for all training datasets $\mathcal{D}$ and validation datasets $\mathcal{D}'$ which have the same number of data. $\lambda_{max}(\cdot)$ means the maximum eigenvalue.

The proof details are provided in Appendix A.2. Theorem 2 states that when $gsnr(\theta_j^0)$ is larger, the probability that the Hessian maximum eigenvalue of validation loss bounded by that of training loss will close to 1. This implies that both the training loss and validation loss are flat, just like Fig.2(a), indicating a better generalization. Combining Theorem 1, we can conclude that larger values of *GSNR*(θ_j^0) will result in better generalization performance.

3.4. Convergence Analysis

We prove the convergence guarantee from the perspective of training loss reduction, i.e., the greater the loss reduction between each two steps, the faster the convergence. We give the following theorem:

Theorem 3 *Under Assumption 1, for fixed initialization parameters θ^0, if the learning rate η is small enough, $\forall t = 1, 2...$, there exist $0 < \alpha_t < 1$ and $\frac{1}{\sqrt{n\alpha_t gsnr(\theta_j^0)}} < r < 1, j = 1, 2...P$, such that,*

$$\begin{aligned}&\mathcal{L}_{\mathcal{D}}(\theta^{t+1}) - \mathcal{L}_{\mathcal{D}}(\theta^t) \quad (7)\\ &< -\eta\alpha_t(1-r)^2(\frac{\partial\mathcal{L}_{\mathcal{D}}}{\partial f_{\mathcal{D}}}(\theta^t))^2\mathbb{E}_{\mathcal{D}\sim\mathcal{Z}^n}(\sum_{j=1}^{P}(g'_{\mathcal{D}}(\theta_j^0))^2)\end{aligned}$$

with probability at least

$$1 - \sum_{j=1}^{P}\frac{1}{nr^2\alpha_t gsnr(\theta_j^0)} \quad (8)$$

over randomly chosen possible distributions for all training datasets $\mathcal{D}$.

The proof details are provided in Appendix A.3. Theorem 3 shows that larger $gsnr(\theta_j^0)$ significantly increases the likelihood of Eq.(7) approaching 1. Besides, $\mathbb{E}_{\mathcal{D}\sim\mathcal{Z}^n}(\sum_{j=1}^{P}(g'_{\mathcal{D}}(\theta_j^0))^2$ is positively correlated with $gsnr(\theta_j^0)$, which indicates that larger $gsnr(\theta_j^0)$ leads to faster convergence rate. Combining Theorem 1, we can conclude that larger *GSNR*(θ_j^0) is advantageous in accelerating the convergence speed of neural networks.

3.5. Zero-Shot NAS Proxy

In the vanilla definition of *GSNR* in Eq.(2), the extremely small gradient variance of a parameter θ_j may dominate the summation of all parameters gradient signal-to-noise ratio, resulting in an improper proxy score. Therefore, we enhance it as ξ-*GSNR*$'$ by introducing a small ξ term to smooth the gradient variance:

$$\xi\text{-}GSNR' := \sum_{j=1}^{P}\frac{(\mathbb{E}_{(x,y)\sim\mathcal{Z}}(g(x,y,\theta_j)))^2}{Var_{(x,y)\sim\mathcal{Z}}(g(x,y,\theta_j)) + \xi} \quad (9)$$

In Eq.(9), we introduce ξ which is set to be quite small for stabilizing the computation and producing a smooth proxy score. The ablation study further demonstrates the effectiveness of ξ in improving the ranking consistency as shown in Tab.9.

In our practical experiment, we average a large number of samples (set to M) gradients to obtain an approximation of gradient expectations. To facilitate the calculation, we divide M data samples into N batches $\{\mathcal{B}_i\}_{i=1}^N$. Thus, our practical Zero-Shot NAS proxy is denoted as:

$$\begin{aligned}&\xi\text{-}GSNR := \quad (10)\\ &\sum_{j=1}^{P}\frac{(\frac{1}{N}\sum_{i=1}^{N}g_{\mathcal{B}_i}(\theta_j))^2}{(\frac{1}{N}\sum_{i=1}^{N}g^2_{\mathcal{B}_i}(\theta_j)) - (\frac{1}{N}\sum_{i=1}^{N}g_{\mathcal{B}_i}(\theta_j))^2 + \xi}\end{aligned}$$

where $g_{\mathcal{B}}(\theta_j) := \frac{1}{k}\sum_{i=1}^{k}\frac{\partial\mathcal{L}(y_i,f(x_i,\theta))}{\partial\theta_j}$ denotes the gradient of batch data with k sample data in the batch $\mathcal{B}$.

Method	CIFAR-10		CIFAR-100		ImageNet-16-120	
	Spearman's ρ	**Kendall's τ**	**Spearman's ρ**	**Kendall's τ**	**Spearman's ρ**	**Kendall's τ**
SNIP [23]	0.638	0.472	0.637	0.474	0.578	0.433
GraSP [54]	0.551	0.385	0.549	0.388	0.553	0.395
GradNorm [1]	0.637	0.469	0.637	0.472	0.578	0.430
SynFlow [51]	0.777	0.581	0.763	0.568	0.751	0.561
TE-NAS [7]	0.376	0.257	0.350	0.239	0.335	0.228
Zen-Score [30]	0.251	0.236	0.260	0.236	0.319	0.277
NASWOT [35]	0.691	0.522	0.704	0.535	0.700	0.527
GradSign [66]	0.808	0.619	0.792	0.600	0.783	0.593
ZiCo [28]	0.800	0.610	0.810	0.610	0.790	0.600
FLOPs	0.733	0.541	0.708	0.517	0.673	0.487
Parameters	0.751	0.576	0.727	0.552	0.690	0.519
ξ-*GSNR*	**0.845**	**0.661**	**0.840**	**0.658**	**0.793**	**0.608**

Table 1. Comparison Ranking Consistency of Zero-Shot proxies in NAS-Bench-201 search space.

4. Experiments

We first conduct the ranking consistency experiments (Sec.4.1) in various NAS-Benches to verify the effectiveness of ξ-*GSNR* proxy. Then, we introduce the ξ-*GSNR* proxy into Zero-Shot NAS (Sec.4.2) to directly search for the optimal architecture in different search spaces and datasets. Finally, the ablation studies (Sec.4.3) are analyzed in detail of ξ-*GSNR* proxy.

4.1. Ranking Consistency

Baselines. We compare our ξ-*GSNR* with existing popular Zero-Shot NAS proxies, including both empirically inspired and theoretically designed ones. Specifically, the former ones are SNIP [23], GraSP [54], SynFlow [51] and GradNorm [1]; while the latter ones consist of NASWOT [35], Zen-Score [30], TE-NAS [7], GradSign [66], and ZiCo [28]. We also evaluate the number of parameters and FLOPs, as they are strong baselines highlighted in recent work [36]. To ensure fairness, we keep the parameters initialization and batch size consistent across all proxies.

Metrics. Spearman's ρ [18] and Kendall's τ are two widely used metrics for characterizing the correlation between two sequences. We leverage them to quantify the relationship between Zero-Shot NAS proxies and architecture performance. The range of both ρ and τ is [-1, 1]. When the metric value approaches 1, it implies that the proxy is more reliable in predicting network performance.

4.1.1 Results in NAS-Bench-201

NAS-Bench-201 [15] is a comprehensive cell-based search space that includes almost up-to-date NAS algorithms. The Benchmark comprises 15,625 candidate architectures and provides access to the diagnostic information on accuracy, loss, and parameters across three datasets: CIFAR-10, CIFAR-100, and ImageNet-16-120.

Method	Spearman's ρ	Kendall's τ
SNIP [23]	0.191	0.131
GraSP [54]	0.329	0.223
GradNorm [1]	0.265	0.182
SynFlow [51]	0.360	0.246
TE-NAS [7]	0.084	0.056
Zen-Score [30]	0.261	0.187
NASWOT [35]	0.327	0.220
GradSign [66]	0.422	0.296
FLOPs	0.422	0.297
Parameters	0.423	0.298
ξ-*GSNR*	**0.615**	**0.434**

Table 2. Comparison Ranking Consistency of Zero-Shot proxies in NAS-Bench-101 search space on CIFAR-10.

We evaluate the ranking consistency of all Zero-Shot proxies directly on three datasets, with a batch size of 64 to ensure a fair comparison. In addition, the number of batches is set to 8 with ξ=1e-8 to compute ξ-*GSNR* score on all three datasets. Tab.1 shows that our Zero-Shot proxy ξ-*GSNR* consistently outperforms other proxies across different datasets, demonstrating the robustness of ξ-*GSNR* to different data samples. Obviously, our method significantly improves the ranking consistency by a large margin compared to most other approaches, indicating the effectiveness of the generalization and convergence theory guarantee.

4.1.2 Results in NAS-Bench-101

NAS-Bench-101 [60] is the first large-scale Benchmark for evaluating various NAS algorithms. It contains 423K

Method	DARTS		ENAS		PNAS		NASNet		Amoeba	
	Spe's ρ	Ken's τ	Spe's ρ	Ken's τ	Spe's ρ	Ken's τ	Spe's ρ	Ken's τ	Spe's ρ	Ken's τ
GradNorm [1]	0.325	0.227	0.073	0.055	0.157	0.109	-0.126	0.080	-0.181	-0.116
SynFlow [51]	-0.002	-0.001	-0.137	-0.092	-0.134	-0.090	-0.277	-0.191	-0.001	-0.001
NASWOT [35]	0.652	0.480	0.534	0.387	0.501	0.363	0.424	0.299	0.291	0.208
GradSign [66]	0.715	0.537	0.579	0.424	0.552	0.396	0.413	0.290	0.355	0.250
FLOPs	0.676	0.500	0.563	0.413	0.550	0.395	0.410	0.288	0.338	0.238
Parameters	0.668	0.493	0.561	0.411	0.541	0.387	0.411	0.289	0.344	0.241
ξ-*GSNR*	**0.728**	**0.547**	**0.597**	**0.440**	**0.562**	**0.403**	**0.445**	**0.313**	**0.379**	**0.226**

Table 3. Comparison Ranking Consistency of Zero-Shot proxies in NDS search space. Spe's ρ: Spearman's ρ, Ken's τ: Kendall's τ.

Method	CIFAR-10		CIFAR-100		ImageNet-16-120		Search
	Validation	Test	Validation	Test	Validation	Test	Algorithm
Optimal	91.61	94.37	73.49	73.51	46.77	47.31	N/A
ENAS [38]	37.51±3.19	53.89±0.58	13.37±2.35	13.96±2.33	15.06±1.95	14.84±2.10	RL
RSPS [29]	80.42±3.58	84.07±3.61	52.12±5.55	52.31±5.77	27.22±3.24	26.28±3.09	Random
DARTS [32]	39.77±0.00	54.30±0.00	15.03±0.00	15.61±0.00	16.43±0.00	16.32±0.00	Gradient
GDAS [14]	89.89±0.08	93.61±0.09	71.34±0.04	70.70±0.30	41.59±1.33	41.71±0.98	Gradient
SETN [13]	84.04±0.28	87.64±0.00	58.86±0.06	59.05±0.24	33.06±0.02	32.52±0.21	Gradient
SynFlow‡ [51]	89.83±0.75	93.12±0.52	69.89±1.87	69.94±1.88	41.94±4.13	42.26±4.26	Training-Free
NASWOT‡ [35]	89.55±0.89	92.91±0.99	69.35±1.70	69.48±1.70	42.81±3.05	43.10±3.16	Training-Free
GradSign‡ [66]	89.84±0.61	93.31±0.47	70.22±1.32	70.33±1.28	42.07±2.78	42.42±2.81	Training-Free
ZiCo [28]	N/A	94.00±0.40	N/A	71.10±0.30	N/A	41.80±0.30	Training-Free
TE-NAS [7]	N/A	93.90±0.47	N/A	71.24±0.56	N/A	42.38±0.46	Training-Free
ξ-*GSNR*	**91.20±0.43**	**94.05±0.18**	**71.82±0.49**	**72.18±0.74**	**45.91±0.31**	**46.24±0.06**	Training-Free

Table 4. Search results in NAS-bench-201. We report the average performance with four independent runs. "Optimal" represents the highest accuracy for each dataset. ‡: We report the results with N=100 that denotes the number of networks sampled in each run.

unique architectures trained on the CIFAR-10 dataset. This benchmark provides the test accuracy of each architecture, making it possible to determine the ranking consistency between the proxy score and the corresponding accuracy by using Spearman's ρ and Kendall's τ, respectively.

Following the experimental settings of GradSign [66], we randomly sample 4500 architectures from the Benchmark to compare ranking consistency. As shown in Tab.2, our ξ-*GSNR* proxy performs better than either existing empirical or theoretical proxies, as well as two strong baselines (i.e., Parameters and FLOPs), demonstrating the superiority of our ξ-*GSNR* proxy in predicting network accuracy.

4.1.3 Results in NAS Design Space (NDS)

NAS Design Space (NDS) [39] summarized and analyzed various network design spaces: DARTS [32], ENAS [38], PNAS [31], NASNet [68], AmoebaNet [41]. Each network design space consists of numerous individual architectures trained on CIFAR-10 dataset. We keep the same batch size as 64 when computing the proxy score for each space. The results are presented in Tab.3. Though some proxies fail to rank specific architectures positively, our method still achieves consistently optimal ranking consistency across different design spaces, showing the advantages of ξ-*GSNR* proxy in evaluating different candidate architectures.

4.2. Zero-Shot Search

To search for the optimal architecture without training, we incorporate ξ-*GSNR* into Zero-Shot search algorithm to verify the effectiveness of our proxy. Specifically, we adopt the pruning-based search algorithm [7] in the cell-based search space, including NAS-Bench-201 [15] and DARTS [32] search space, and adopt the evolution algorithm in chain-style ProxylessNAS [5] and ViT-Like BurgerFormer [59] search space. The details of the search algorithm are provided in Appendix B.

4.2.1 Search results in NAS-Bench-201 space

In NAS-Bench-201, the super-network is stacked by cells, with only normal cells being searched and reduction cells

Method	CIFAR-10			ImageNet			Search
	Test Acc.(%)	Params(M)	GPU-days	Top-1(%)	Params(M)	GPU-days	Algorithm
ENAS [38]	97.11	4.6	0.5	N/A	N/A	N/A	RL
NASNet-A [68]	97.35	3.3	1800	74.0	5.3	1800	RL
AmoebaNet-A [41]	96.66±0.06	3.2	3150	75.7	6.4	3150	EA
PNAS [31]	96.59±0.09	3.2	225	74.2	5.1	225	SMBO
DARTS [32]	97.00±0.14	3.3	0.4	73.3	4.7	0.4	Gradient
SNAS [57]	97.15±0.02	**2.8**	1.5	72.7	**4.3**	1.5	Gradient
GDAS [14]	97.07	3.4	0.21	74.0	5.3	0.21	Gradient
BayesNAS [67]	97.19±0.04	3.4	0.2	N/A	N/A	N/A	Gradient
Robust-DARTS [64]	97.05±0.21	N/A	1.6	N/A	N/A	N/A	Gradient
PC-DARTS [58]	97.43±0.07	3.6	0.1	74.9	5.3	0.1	Gradient
DATA [6]	97.41	3.4	1	75.1	5.0	1	Gradient
SGAS [27]	97.34±0.24	3.7	0.24	N/A	N/A	N/A	Gradient
SDARTS-ADV [8]	97.39±0.02	3.3	1.3	N/A	N/A	N/A	Gradient
DARTS+PT [55]	97.39±0.08	3.0	0.8	N/A	N/A	N/A	Gradient
TE-NAS [7]	97.37±0.06	3.8	0.05	75.5	5.4	$0.17^{\ddagger}$	Training-Free
NASI-ADA [46]	97.10±0.13	3.7	0.01	75.2	5.2	$\mathbf{0.01}^{\ddagger}$	Training-Free
***ξ-GSNR* (avg.)**	**97.48±0.04**	3.45±0.28	0.01	N/A	N/A	N/A	Training-Free
***ξ-GSNR* (best)**	**97.53**	3.66	**0.01**	**75.5**	5.5	$0.017^{\ddagger}$	Training-Free

Table 5. Search results in DARTS space and comparison with other state-of-the-art methods. We report the average results on CIFAR-10 with three independent runs of searching. ‡: directly search on ImageNet.

being maintained as residual blocks. Each normal cell consists of six edges and five candidate operations associated with each edge. Similar to TE-NAS [7], we set the pruning number of operations associated with each edge to 1 for each step, and directly search for the optimal architecture on three different datasets. We conduct four independent search runs with different random seeds and report the mean and standard deviation in Tab.4. We achieve the best performance when compared with either training-based or other training-free methods. In particular, *ξ-GSNR* obtains higher test accuracy than TE-NAS [7], demonstrating that our proxy is superior to Neural Tangent Kernel (NTK) or linear region theory in predicting architecture performance.

4.2.2 Search results in DARTS space

We leverage *ξ-GSNR* proxy to search for optimal architectures in DARTS search space directly on CIFAR-10 and ImageNet, respectively. To conduct the search, we adopt the algorithm proposed in TE-NAS [7]. We gradually prune low-importance operations predicted by the Zero-Shot proxy until the single-path cell is reached. The searched architecture is then evaluated following the standard experimental settings [32] to ensure a fair comparison.

Tab.5 summarizes the performance of searched architectures on CIFAR-10 and ImageNet datasets. In general, Zero-Shot NAS has a faster search speed than One-Shot NAS, indicating the efficiency of training-free search. More specifically, *ξ-GSNR* achieves the highest average test accuracy among all methods, demonstrating the effectiveness of our proxy in evaluating architecture performance. When searching directly on the ImageNet dataset, the performance and search speed have also been improved, benefiting from the novel Zero-Shot NAS proxy.

Method	Test Acc. (%)		Params	FLOPs
	Top-1	Top-5	(M)	(M)
NASNet-A [68]	74.0	91.3	5.3	564
AmoebaNet-A [41]	74.5	92.0	5.1	555
MnasNet-A3 [49]	76.7	93.3	5.2	403
PNAS [31]	74.2	91.9	5.1	588
FBNet-C [56]	74.9	N/A	5.5	375
ProxylessNAS [5]	75.1	92.5	7.1	465
SPOS [17]	74.8	N/A	5.4	472
FairNAS-A [10]	77.5	N/A	5.9	392
RLNAS [65]	75.6	92.6	5.3	473
MobileNetV3 [20]	75.2	N/A	5.4	**219**
EfficientNet-B0 [50]	77.1	93.3	5.3	390
DNA-b [25]	77.5	93.3	**4.9**	406
BossNet-M2 [26]	77.4	93.6	N/A	403
ξ-GSNR	**78.2**	**94.0**	5.4	409

Table 6. Search results on ImageNet in ProxylessNAS space and comparison with other state-of-the-art methods.

Model	Params(M)	FLOPs(G)	Top-1(%)	Type
DeiT-S [52]	22	4.7	79.9	Manual
Swin-T [34]	29	4.5	81.3	Manual
Swin-S [34]	50	8.7	83.0	Manual
ViTAS-DeiT [48]	23	4.9	80.2	NAS
DeepMAD [45]	29	4.5	82.5	NAS
BurgerFormer [59]	26	3.9	82.7	NAS
ξ-GSNR	29	4.5	**83.1**	NAS

Table 7. Search results on ImageNet in BurgerFormer space and comparison with other state-of-the-art methods.

	Backbone	Params(M)	AP^b	AP^b_{50}	AP^b_{75}
Object Detection	ResNet-50 [19]	44.2	38.0	58.6	41.4
	PoolFormer [63]	41.0	40.1	62.2	43.4
	Swin-T [34]	48.0	43.7	66.6	47.7
	ξ-GSNR	48.9	**45.0**	**67.1**	**49.1**
	Backbone	**Params(M)**	AP^M	AP^M_{50}	AP^M_{75}
Instance Segmentation	ResNet-50 [19]	44.2	34.4	55.1	36.7
	PoolFormer [63]	41.0	37.0	59.1	36.9
	Swin-T [34]	48.0	39.8	63.3	42.7
	ξ-GSNR	48.9	**40.7**	**63.8**	**43.7**

Table 8. Comparison with state-of-the-art models on COCO.

4.2.3 Search results in ProxylessNAS space

The ProxylessNAS [5] space is the popular chain-style search space with a total of 19 searchable layers. We search for the kernel size $\{3, 5, 7\}$ and expansion ratio $\{3, 6\}$ for the bottleneck blocks on each layer. We use the evolutionary algorithm to search for architectures with around 400M FLOPs constraints. The search iteration elapses 2000 steps with a population of 128. Finally, the optimal architecture is retrained following DNA [25] settings.

As shown in Tab.6, our searched architecture yielded the highest top-1 accuracy 78.2% with only 409M FLOPs. Impressively, our search cost only requires 0.3 GPU-days, surpassing all the other training-based methods. This highlights the effectiveness and efficiency of our Zero-Shot NAS proxy ξ-*GSNR* in discovering excellent architectures.

4.2.4 Search results in BurgerFormer space

We conduct the the experiments on ImageNet using BurgerFormer [59] search space, which is a ViT-like structure. We use the evolutionary algorithm to search for architectures within 30M Params and 5.0G FLOPs. The results in Tab.7 demonstrate that our method not only significantly outperforms other NAS methods, but also outperforms the manually designed networks under similar resource constraints. Even when compared to Swin-S with 50M Params, we still obtain a higher accuracy.

Moreover, we evaluate the performance of our searched architecture in BurgerFormer space for object detection and instance segmentation on COCO datasets. The pre-trained structure is employed as the backbone for the Mask R-CNN detector. The results in Tab.8 show that we achieve 45.0 AP^b and 40.7 AP^M on object detection and instance segmentation, clearly surpassing that of ResNet-50 and the other handcrafted ViT-like structures.

	Spe's ρ	Ken's τ
(1) $\sum_{j=1}^{P}(\mathbb{E}(\mathrm{g}(x,y,\theta_j)))^2$	0.556	0.407
(2) $\sum_{j=1}^{P}\frac{1}{Var(\mathrm{g}(x,y,\theta_j))}$	0.200	0.188
(3) $\sum_{j=1}^{P}\frac{1}{Var(\mathrm{g}(x,y,\theta_j))+\xi}$	0.205	0.198
(4) $\sum_{j=1}^{P}\frac{(\mathbb{E}(\mathrm{g}(x,y,\theta_j)))^2}{Var(\mathrm{g}(x,y,\theta_j))}$	0.764	0.589
(5) $\sum_{j=1}^{P}\frac{(\mathbb{E}(\mathrm{g}(x,y,\theta_j)))^2}{Var(\mathrm{g}(x,y,\theta_j))+\xi}$	**0.828**	**0.656**

Table 9. Comparison Ranking Consistency of different GSNR proxy variants on ImageNet-16-120 in NAS-Bench-201 space.

4.3. Ablation Study

4.3.1 The hyper-parameters of GSNR

There are three hyper-parameters that may fluctuate the ranking consistency when calculating ξ-*GSNR* proxy in Eq.(10). We randomly sample 200 architectures from NAS-Bench-201 and compute their ξ-*GSNR* proxy scores by varying batch sizes $|\mathcal{B}|$, batch numbers N, and random ξ.

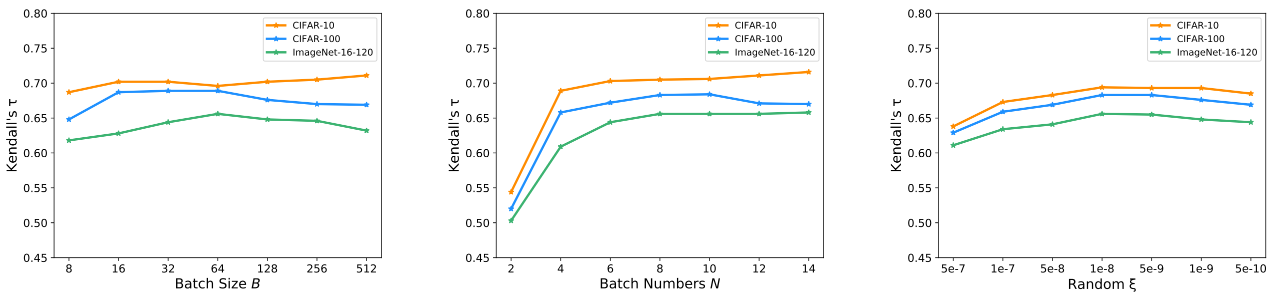
Batch Sizes $|\mathcal{B}|$. The batch sizes $|\mathcal{B}|$ varies from $\{8, 16, 32, 64, 128, 256, 512\}$ while keeping other parameters constant. As shown in Fig. 3(a), a batch size of 64 produces the highest ranking consistency on most datasets. Though the value on CIFAR-10 may not reach the highest, we still keep the batch size to 64 on all experiments for a fair comparison.

Batch Numbers N. The Batch numbers N varies from $\{2, 4, 6, 8, 10, 12, 14\}$ to compute ξ-*GSNR* proxy. Fig.3(b) indicates that N=8 is sufficient to obtain the best correlation. Hence, we adopt 8 batches in other search spaces and datasets as well, considering the trade-off between efficiency and accuracy.

Random ξ. The random ξ is set to an extremely small value for stabilizing the computation. Fig.3(c) demonstrates that ξ=1e-8 is stable enough to improve the ranking. So we keep ξ=1e-8 for all experiments.

4.3.2 The variants of GSNR

Here we investigate the effect of *GSNR* variants, including: (1) gradient's squared mean; (2) inverse of gradient's variance; (3) inverse of gradient's variance with ξ; (4) vanilla *GSNR*; and (5) our practical ξ-*GSNR*. We randomly sample 200 architectures from NAS-Bench-201 and evaluate their ranking consistency on ImageNet-16-120. In Tab.9, we ob-



(a) Kendall's τ vs. Batch Size $|\mathcal{B}|$ (b) Kendall's τ vs. Batch Numbers N (c) Kendall's τ vs. Random ξ

Figure 3. The ranking consistency between test accuracy in NAS-Bench-201 and ξ-*GSNR* computed under varying hyper-parameters, including (a) Batch Size $|\mathcal{B}|$, (b) Batch Numbers N, and (c) Random ξ.

Method	CIFAR-10		CIFAR-100		ImageNet-16	
	P@5%	P@10%	P@5%	P@10%	P@5%	P@10%
SNIP	0.064	0.114	0.060	0.101	0.073	0.118
GradNorm	0.086	0.192	0.092	0.229	0.124	0.247
Zen-Score	0.158	0.379	0.167	0.343	0.155	0.302
NASWOT	0.163	0.383	0.170	0.344	0.156	0.304
ξ-*GSNR*	**0.366**	**0.475**	**0.394**	**0.479**	**0.283**	**0.330**

Table 10. Comparison the Top-K correlation in NAS-Bench-201.

	Seed=0	Seed=1	Seed=2	Avg. (Std.)
CIFAR-10	0.660	0.661	0.665	0.662 (0.002)
CIFAR-100	0.657	0.658	0.644	0.653 (0.006)
ImageNet-16	0.616	0.608	0.599	0.608 (0.007)
	Orthogonal	**Xavier**	**Kaiming**	**Avg. (Std.)**
CIFAR-10	0.666	0.668	0.659	0.664 (0.004)
CIFAR-100	0.649	0.639	0.639	0.642 (0.005)
ImageNet-16	0.590	0.599	0.593	0.594 (0.004)

Table 11. Comparison Ranking Consistency with different initializations in NAS-Bench-201 using Kendall's τ rank.

serve that the squared mean or inverse of variance alone is not as effective as the ratio of the two, demonstrating the effectiveness of *GSNR*. Furthermore, adding a small ξ in (3) can enhance the stability of computation and improve the ranking consistency. Considering this, we develop our novel Zero-Shot proxy depicted in (5), which outperforms all the other variants.

4.3.3 The Top-K Correlation

We compute the Top-K correlation on NAS-Bench-201 across three different datasets. Specifically, P@TopK = $\frac{\#\{(r<KM)\wedge(n<KM)\}}{KM}$, which is defined in [33], representing the proportion of predicted top-K% in true top-K% architectures. The results in Tab.10 show that our method achieves the highest P@Top5% and P@Top10%, indicating our ability to identify superb networks.

4.3.4 Different initializations

To demonstrate the robustness of ξ-*GSNR* proxy with different initializations, we conduct the experiments in NAS-Bench-201 on three datasets. Specifically, we use two types of initializations, including random seeds (seed=0, 1, 2) and different initialization ways (orthogonal, xavier, kaiming). The results in Tab.11 show that we achieve the stable Kendall's τ with quite a low standard deviation when using different initializations, demonstrating the robustness of ξ-*GSNR* proxy.

5. Conclusion

Recently, Zero-Shot NAS has been attracting more and more attention due to its search efficiency. However, designing an effective proxy can be cumbersome and challenging. In this study, we theoretically analyzed that *GSNR* is strongly correlated with network generalization and convergence. Further, to enhance the performance and stabilize the computation, we develop ξ-based gradient signal-to-noise (ξ-*GSNR*) to predict architecture accuracy at initialization. Extensive experiments demonstrate the effectiveness and efficiency of the ξ-*GSNR* proxy. In the future, we will further explore more advanced Zero-Shot proxies to more accurately predict architecture performance.

Acknowledgements

The research of Yu Hu, Zihao Sun, Longxing Yang, Shun Lu, and Jilin Mei is supported by the National Natural Science Foundation of China under Grant No. 62176250 and No. 62203424. The research of Wenxiao Zhao and Yu Sun is supported by National Key Research and Development Program of China (2018YFA0703800).

References

[1] Mohamed S Abdelfattah, Abhinav Mehrotra, Łukasz Dudziak, and Nicholas Donald Lane. Zero-cost proxies for lightweight nas. In *ICLR*, 2020. 1, 2, 5, 6

[2] Zeyuan Allen-Zhu, Yuanzhi Li, and Zhao Song. A convergence theory for deep learning via over-parameterization. In *ICML*, pages 242–252. PMLR, 2019. 2

[3] Bowen Baker, Otkrist Gupta, Nikhil Naik, and Ramesh Raskar. Designing neural network architectures using reinforcement learning. *arXiv preprint arXiv:1611.02167*, 2016. 1, 2

[4] Andrew Brock, Theodore Lim, James M Ritchie, and Nick Weston. Smash: one-shot model architecture search through hypernetworks. In *ICLR*, 2018. 1, 2

[5] Han Cai, Ligeng Zhu, and Song Han. Proxylessnas: Direct neural architecture search on target task and hardware. In *ICLR*, 2018. 6, 7, 8

[6] Jianlong Chang, Xinbang Zhang, Yiwen Guo, Gaofeng Meng, Shiming Xiang, and Chunhong Pan. Data: Differentiable architecture approximation. In *NeurIPS*, 2019. 7

[7] Wuyang Chen, Xinyu Gong, and Zhangyang Wang. Neural architecture search on imagenet in four gpu hours: A theoretically inspired perspective. In *ICLR*, 2021. 2, 3, 5, 6, 7

[8] Xiangning Chen and Cho-Jui Hsieh. Stabilizing differentiable architecture search via perturbation-based regularization. In *ICML*, 2020. 1, 2, 7

[9] Xin Chen, Lingxi Xie, Jun Wu, and Qi Tian. Progressive differentiable architecture search: Bridging the depth gap between search and evaluation. In *ICCV*, 2019. 1, 2

[10] Xiangxiang Chu, Bo Zhang, and Ruijun Xu. Fairnas: Rethinking evaluation fairness of weight sharing neural architecture search. In *ICCV*, 2021. 1, 2, 7

[11] Xiangxiang Chu, Tianbao Zhou, Bo Zhang, and Jixiang Li. Fair darts: Eliminating unfair advantages in differentiable architecture search. In *ECCV*, 2020. 1, 2

[12] Laurent Dinh, Razvan Pascanu, Samy Bengio, and Yoshua Bengio. Sharp minima can generalize for deep nets. In *ICML*, 2017. 3

[13] Xuanyi Dong and Yi Yang. One-shot neural architecture search via self-evaluated template network. In *ICCV*, 2019. 6

[14] Xuanyi Dong and Yi Yang. Searching for a robust neural architecture in four gpu hours. In *CVPR*, 2019. 1, 2, 6, 7

[15] Xuanyi Dong and Yi Yang. Nas-bench-201: Extending the scope of reproducible neural architecture search. In *ICLR*, 2020. 5, 6

[16] Thomas Elsken, Jan Hendrik Metzen, and Frank Hutter. Neural architecture search: A survey. In *The Journal of Machine Learning Research*, pages 1997–2017, 2019. 1

[17] Zichao Guo, Xiangyu Zhang, Haoyuan Mu, Wen Heng, Zechun Liu, Yichen Wei, and Jian Sun. Single path one-shot neural architecture search with uniform sampling. In *ECCV*, 2020. 1, 2, 7

[18] Wolfgang Härdle and Oliver Linton. Applied nonparametric methods. *Handbook of econometrics*, 4:2295–2339, 1994. 5

[19] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, 2016. 1, 8

[20] Andrew Howard, Mark Sandler, Grace Chu, Liang-Chieh Chen, Bo Chen, Mingxing Tan, Weijun Wang, Yukun Zhu, Ruoming Pang, Vijay Vasudevan, et al. Searching for mobilenetv3. In *ICCV*, 2019. 7

[21] Arthur Jacot, Franck Gabriel, and Clément Hongler. Neural tangent kernel: Convergence and generalization in neural networks. In *NeurIPS*, 2018. 2

[22] Nitish Shirish Keskar, Dheevatsa Mudigere, Jorge Nocedal, Mikhail Smelyanskiy, and Ping Tak Peter Tang. On large-batch training for deep learning: Generalization gap and sharp minima. In *ICLR*, 2017. 3

[23] Namhoon Lee, Thalaiyasingam Ajanthan, and Philip HS Torr. Snip: Single-shot network pruning based on connection sensitivity. In *ICLR*, 2019. 1, 2, 5

[24] Aitor Lewkowycz, Yasaman Bahri, Ethan Dyer, Jascha Sohl-Dickstein, and Guy Gur-Ari. The large learning rate phase of deep learning: the catapult mechanism. *arXiv preprint arXiv:2003.02218*, 2020. 3

[25] Changlin Li, Jiefeng Peng, Liuchun Yuan, Guangrun Wang, Xiaodan Liang, Liang Lin, and Xiaojun Chang. Block-wisely supervised neural architecture search with knowledge distillation. In *CVPR*, 2020. 7, 8

[26] Changlin Li, Tao Tang, Guangrun Wang, Jiefeng Peng, Bing Wang, Xiaodan Liang, and Xiaojun Chang. Bossnas: Exploring hybrid cnn-transformers with block-wisely self-supervised neural architecture search. In *ICCV*, 2021. 7

[27] Guohao Li, Guocheng Qian, Itzel C Delgadillo, Matthias Muller, Ali Thabet, and Bernard Ghanem. Sgas: Sequential greedy architecture search. In *CVPR*, 2020. 7

[28] Guihong Li, Yuedong Yang, Kartikeya Bhardwaj, and Radu Marculescu. Zico: Zero-shot nas via inverse coefficient of variation on gradients. In *ICLR*, 2023. 2, 3, 5, 6

[29] Liam Li and Ameet Talwalkar. Random search and reproducibility for neural architecture search. In *Uncertainty in artificial intelligence*, 2020. 6

[30] 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 *ICCV*, 2021. 2, 5

[31] Chenxi Liu, Barret Zoph, Maxim Neumann, Jonathon Shlens, Wei Hua, Li-Jia Li, Li Fei-Fei, Alan Yuille, Jonathan Huang, and Kevin Murphy. Progressive neural architecture search. In *ECCV*, 2018. 6, 7

[32] Hanxiao Liu, Karen Simonyan, and Yiming Yang. Darts: Differentiable architecture search. In *ICLR*, 2018. 1, 2, 6, 7

[33] Jinlong Liu, Guoqing Jiang, Yunzhi Bai, Ting Chen, and Huayan Wang. Understanding why neural networks generalize well through gsnr of parameters. In *ICLR*, 2020. 2, 3, 9

[34] Ze Liu, Yutong Lin, Yue Cao, Han Hu, Yixuan Wei, Zheng Zhang, Stephen Lin, and Baining Guo. Swin transformer: Hierarchical vision transformer using shifted windows. In *ICCV*, 2021. 8

[35] Joe Mellor, Jack Turner, Amos Storkey, and Elliot J Crowley. Neural architecture search without training. In *ICML*, 2021. 1, 2, 5, 6

[36] Xuefei Ning, Changcheng Tang, Wenshuo Li, Zixuan Zhou, Shuang Liang, Huazhong Yang, and Yu Wang. Evaluating efficient performance estimators of neural architectures. In *NeurIPS*, 2021. 1, 2, 5

[37] Daniel S Park, Jaehoon Lee, Daiyi Peng, Yuan Cao, and Jascha Sohl-Dickstein. Towards nngp-guided neural architecture search. *arXiv preprint arXiv:2011.06006*, 2020. 3

[38] Hieu Pham, Melody Guan, Barret Zoph, Quoc Le, and Jeff Dean. Efficient neural architecture search via parameters sharing. In *ICML*, 2018. 1, 2, 6, 7

[39] Ilija Radosavovic, Justin Johnson, Saining Xie, Wan-Yen Lo, and Piotr Dollár. On network design spaces for visual recognition. In *ICCV*, 2019. 6

[40] Maithra Raghu, Ben Poole, Jon Kleinberg, Surya Ganguli, and Jascha Sohl-Dickstein. On the expressive power of deep neural networks. In *ICML*, 2017. 2

[41] Esteban Real, Alok Aggarwal, Yanping Huang, and Quoc V Le. Regularized evolution for image classifier architecture search. In *AAAI*, 2019. 1, 2, 6, 7

[42] Esteban Real, Sherry Moore, Andrew Selle, Saurabh Saxena, Yutaka Leon Suematsu, Jie Tan, Quoc V Le, and Alexey Kurakin. Large-scale evolution of image classifiers. In *ICML*, 2017. 1, 2

[43] Levent Sagun, Léon Bottou, and Yann LeCun. Singularity of the hessian in deep learning. *CoRR*, abs/1611.07476, 2016. 3

[44] Mark Sandler, Andrew Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen. Mobilenetv2: Inverted residuals and linear bottlenecks. In *CVPR*, 2018. 1

[45] Xuan Shen, Yaohua Wang, Ming Lin, Yilun Huang, Hao Tang, Xiuyu Sun, and Yanzhi Wang. Deepmad: Mathematical architecture design for deep convolutional neural network. In *CVPR*, 2023. 8

[46] Yao Shu, Shaofeng Cai, Zhongxiang Dai, Beng Chin Ooi, and Bryan Kian Hsiang Low. Nasi: Label-and data-agnostic neural architecture search at initialization. In *ICLR*, 2022. 2, 7

[47] Yao Shu, Wei Wang, and Shaofeng Cai. Understanding architectures learnt by cell-based neural architecture search. In *ICLR*, 2020. 2, 3

[48] Xiu Su, Shan You, Jiyang Xie, Mingkai Zheng, Fei Wang, Chen Qian, Changshui Zhang, Xiaogang Wang, and Chang Xu. Vitas: Vision transformer architecture search. In *ECCV*, 2022. 8

[49] Mingxing Tan, Bo Chen, Ruoming Pang, Vijay Vasudevan, Mark Sandler, Andrew Howard, and Quoc V Le. Mnasnet: Platform-aware neural architecture search for mobile. In *CVPR*, 2019. 7

[50] Mingxing Tan and Quoc Le. Efficientnet: Rethinking model scaling for convolutional neural networks. In *ICML*, 2019. 7

[51] Hidenori Tanaka, Daniel Kunin, Daniel L Yamins, and Surya Ganguli. Pruning neural networks without any data by iteratively conserving synaptic flow. In *NeurIPS*, 2020. 1, 2, 5, 6

[52] Hugo Touvron, Matthieu Cord, Matthijs Douze, Francisco Massa, Alexandre Sablayrolles, and Hervé Jégou. Training data-efficient image transformers & distillation through attention. In *ICML*, 2021. 8

[53] Xingchen Wan, Binxin Ru, Pedro M Esperança, and Zhenguo Li. On redundancy and diversity in cell-based neural architecture search. In *ICLR*, 2022. 2, 3

[54] Chaoqi Wang, Guodong Zhang, and Roger Grosse. Picking winning tickets before training by preserving gradient flow. In *ICLR*, 2020. 1, 2, 5

[55] Ruochen Wang, Minhao Cheng, Xiangning Chen, Xiaocheng Tang, and Cho-Jui Hsieh. Rethinking architecture selection in differentiable nas. In *ICLR*, 2021. 7

[56] Bichen Wu, Xiaoliang Dai, Peizhao Zhang, Yanghan Wang, Fei Sun, Yiming Wu, Yuandong Tian, Peter Vajda, Yangqing Jia, and Kurt Keutzer. Fbnet: Hardware-aware efficient convnet design via differentiable neural architecture search. In *CVPR*, 2019. 7

[57] Sirui Xie, Hehui Zheng, Chunxiao Liu, and Liang Lin. Snas: stochastic neural architecture search. In *ICLR*, 2018. 1, 2, 7

[58] Yuhui Xu, Lingxi Xie, Xiaopeng Zhang, Xin Chen, Guo-Jun Qi, Qi Tian, and Hongkai Xiong. Pc-darts: Partial channel connections for memory-efficient architecture search. In *ICLR*, 2019. 1, 2, 7

[59] Longxing Yang, Yu Hu, Shun Lu, Zihao Sun, Jilin Mei, Yinhe Han, and Xiaowei Li. Searching for burgerformer with micro-meso-macro space design. In *ICML*, 2022. 6, 8

[60] Chris Ying, Aaron Klein, Eric Christiansen, Esteban Real, Kevin Murphy, and Frank Hutter. Nas-bench-101: Towards reproducible neural architecture search. In *ICML*, 2019. 5

[61] Shan You, Tao Huang, Mingmin Yang, Fei Wang, Chen Qian, and Changshui Zhang. Greedynas: Towards fast one-shot nas with greedy supernet. In *CVPR*, 2020. 1, 2

[62] Jiahui Yu, Pengchong Jin, Hanxiao Liu, Gabriel Bender, Pieter-Jan Kindermans, Mingxing Tan, Thomas Huang, Xiaodan Song, Ruoming Pang, and Quoc Le. Bignas: Scaling up neural architecture search with big single-stage models. In *ECCV*, 2020. 1, 2

[63] Weihao Yu, Mi Luo, Pan Zhou, Chenyang Si, Yichen Zhou, Xinchao Wang, Jiashi Feng, and Shuicheng Yan. Metaformer is actually what you need for vision. In *CVPR*, 2022. 8

[64] Arber Zela, Thomas Elsken, Tonmoy Saikia, Yassine Marrakchi, Thomas Brox, and Frank Hutter. Understanding and robustifying differentiable architecture search. In *ICLR*, 2020. 1, 2, 7

[65] Xuanyang Zhang, Pengfei Hou, Xiangyu Zhang, and Jian Sun. Neural architecture search with random labels. In *CVPR*, 2021. 1, 2, 7

[66] Zhihao Zhang and Zhihao Jia. Gradsign: Model performance inference with theoretical insights. In *ICLR*, 2022. 2, 3, 5, 6

[67] Hongpeng Zhou, Minghao Yang, Jun Wang, and Wei Pan. Bayesnas: A bayesian approach for neural architecture search. In *ICML*, 2019. 7

[68] Barret Zoph, Vijay Vasudevan, Jonathon Shlens, and Quoc V Le. Learning transferable architectures for scalable image recognition. In *CVPR*, 2018. 1, 2, 6, 7