EENAS: An Efficient Evolutionary Algorithm for Neural Architecture Search

Jian Zheng zhengjian2322@buaa.edu.cn
Wenran Han hanwenran@buaa.edu.cn
Ying Zhang yingzhang1998@buaa.edu.cn
Shufan ji* jishufan@buaa.edu.cn
Beihang University, Beijing, P.R. China, 100191.
** Corresponding Author*

Editors: Emtiyaz Khan and Mehmet Gönen

Abstract

Neural Architecture Search (NAS) has been widely applied to automatic neural architecture design. Traditional NAS methods often evaluate a large number of architectures, leading to expensive computation overhead. To speed-up architecture search, recent NAS methods try to employ network estimation strategies for guidance of promising architecture selection. In this paper, we have proposed an efficient evolutionary algorithm for NAS, which adapts the most advanced proxy of synthetic signal bases for architecture estimation. Extensive experiments show that our method outperforms state-of-the-art NAS methods, on NAS-Bench-101 search space and NAS-Bench-201 search space (CIFAR-10, CIFAR-100 and ImageNet16-120). Compared with existing works, our method could identify better architectures with greatly reduced search time.

Keywords: Neural Architecture Search, Evolutionary Algorithm

1. Introduction

Nowadays neural architecture search has been widely applied to automatic neural architecture design in many domains, including image classification (Zoph et al., 2018; Real et al., 2017), object detection (Wang et al., 2020b; Peng et al., 2019; Ghiasi et al., 2019), semantic segmentation (Chen et al., 2019; Nekrasov et al., 2019; Liu et al., 2019a), speaker recognition (Ding et al., 2020) and etc. NAS aims to search architectures with outstanding performance in given search spaces and datasets. Traditional NAS methods include random search (Bergstra and Bengio, 2012), reinforcement learning (Zoph and Le, 2017; Zoph et al., 2018), evolutionary algorithms (Real et al., 2017; Wei et al., 2020; Lopes et al., 2021), bayesian optimization (Kandasamy et al., 2018; White et al., 2021; Ru et al., 2021), and predictor-based methods (Wen et al., 2020; Lu et al., 2021; Ning et al., 2020; Dudziak et al., 2020; Wu et al., 2021). Although those methods could get promising results, they are computationally expensive to train networks from scratch to convergence for hours and days. To reduce training cost, some weight-sharing methods (Liu et al., 2019b; Pham et al., 2018; Guo et al., 2020) have been proposed to share weights among subnets sampled from a supernet. However, those shared weights are under-trained so as to have limited ability to deliver optimal architectures.

Recently, to speed-up architecture search, some methods try to employ network estimation strategies for guidance of promising architecture selection. NPENAS (Wei et al., 2020) guides the evolutionary search by two predictors: a graph-based uncertainty estimation network and a performance predictor. In addition, some proxy methods have been proposed to estimate network performance, delivering architecture ranking according to proxies. Zero-cost proxies use initial parameters with a single forward or backward propagation pass to rank architectures (Abdelfattah et al., 2021; Mellor et al., 2021). Accordingly, a zero-proxy estimator adopted by G-EA (Lopes et al., 2021) guides evolutionary algorithm(EA) to explore search space, which can efficiently rank offspring on NAS-Bench-201. In addition, ProxyBO (Shen et al., 2021) combines BO with zero-cost proxies to select promising architectures. As there is a trade-off between the performance and time complexity of estimation strategies, proxies delivering better architecture ranking with reasonable training cost would provide better guidance for search. Li et al. (2021) has proposed a proxy based on Synthetic Signal Bases(SSB) to measure the intrinsic capability of networks, which outperforms zero-cost proxy methods in terms of ranking with near-zero training cost. Therefore, we are motivated to try the most advanced estimation proxy SSB on evolutionary algorithm, in order to search for better architectures with reduced search time.

In this paper, we propose an **E**fficient **E**volutionary algorithm for **NAS** (**EENAS**), which adapts the SSB proxy (Li et al., 2021) to EA, leading to improved architecture search with greatly reduced search time. Compared with previous works, our method has three following advantages.

- The adaption of SSB proxy could enhance the exploration and exploitation of EA, and significantly accelerate NAS. Traditional EA methods have no estimation on offspring at each EA iteration, leading to catastrophic training cost for optimal architecture search. At each iteration of our **EENAS**, only the best offspring estimated by SSB proxy is trained. As SSB proxy could provide reasonable estimation on offspring with near-zero processing time (a few seconds for one single batch of data, much less than network training cost), **EENAS** could get much better architecture with greatly reduced search time.

- **EENAS** progressively optimizes SSB proxy with the increase of evaluated architectures, so that the architecture estimation delivered by SSB proxy is continuously improved, which would provide better guidance for architecture search.

- Extensive experiments show that our **EENAS** outperforms state-of-the-art methods on NAS-Bench-101 (Ying et al., 2019) search space and NAS-Bench-201 (Dong and Yang, 2020) search space (CIFAR-10, CIFAR-100 and ImageNet16-120). Especially, **EENAS** successfully gets the optimal architecture on NAS-Bench-201 CIFAR-100 in each repeated experimental trial.

2. Related Work

Neural Architecture Search. Zoph and Le (2017) is the first to bring NAS into research, since then many works have been proposed to explore search spaces for neural architectures with outstanding performance on given datasets. In general, NAS contains three components: search space, search method and evaluation strategy.

Search spaces determine the upper bound of NAS performance, which are usually classified into global search spaces (Zoph and Le, 2017; Xie and Yuille, 2017; Tan et al., 2019) and cell-based search spaces (Zoph et al., 2018; Liu et al., 2019b). Within cell-based search spaces, some benchmarks with fully trained neural networks have been developed, including NAS-Bench-101 (Ying et al., 2019) and NAS-Bench-201 (Dong and Yang, 2020). As NAS experiments usually take high computation resources to train and evaluate networks, those benchmarks offer a convenient way of performance comparison for NAS methods.
Evaluation strategies deliver estimation of network performance. Synaptic saliency metrics (Lee et al., 2019; Wang et al., 2020a; Tanaka et al., 2020) measure the loss change of removing a certain parameter, which are often applied in model compression. Fisher (Theis et al., 2018) estimates the loss change of removing activation channels. NASWOT (Mellor et al., 2021) uses activation patterns in untrained networks to estimate performance of architectures. KNAS (Xu et al., 2021) uses the Gram matrix of gradients to rank architectures. TE-NAS (Chen et al., 2021) and NASI (Shu et al., 2021) have employed the theory of Neural Tangent Kernel (NTK) (Jacot et al., 2018) for architecture estimation. Shu et al. (2022) proposes HNAS which consistently boosts training-free NAS in a principle way. GenNAS (Li et al., 2021) uses the combinations of synthetic signal bases, which can represent a wide range of complicated real-world signals, to rank architectures with near-zero training cost, and gets outstanding ranking performance on NAS-Bench-101 and NAS-Bench-201 search spaces. Evaluation strategies that deliver efficient and reliable architecture estimation could provide better guidance for optimal architecture search.
Search methods provide architecture search strategies for search space exploration. Traditional NAS methods usually get expensive computation in network training, which include random search (Bergstra and Bengio, 2012), reinforcement learning (Zoph and Le, 2017; Zoph et al., 2018), evolutionary algorithms (Real et al., 2017; Wei et al., 2020; Lopes et al., 2021), bayesian optimization (Kandasamy et al., 2018; White et al., 2021; Ru et al., 2021), and predictor-based methods (Wen et al., 2020; Lu et al., 2021; Ning et al., 2020; Dudziak et al., 2020; Wu et al., 2021).

To reduce training cost, weight-sharing among subnets sampled from a supernet is widely adopted for NAS methods (Liu et al., 2019b; Pham et al., 2018; Guo et al., 2020). For example, DARTS (Liu et al., 2019b) and DARTS-PT (Wang et al., 2021) make continuous search spaces so as to optimize the supernet via gradient-based methods. ENAS (Pham et al., 2018) trains a controller to identify architectures through optimal subgraph search within a large computational graph, which uses weight-sharing to accelerate the training of sampled networks. However, those shared weights are under-trained, which have limited ability to deliver optimal architectures.

Recently, NAS methods attempt to employ network estimation strategies for architecture selection. NPENAS (Wei et al., 2020) guides the evolutionary search by two predictors: a graph-based uncertainty estimation network and a performance predictor. G-EA (Lopes et al., 2021) adopts a zero-proxy estimator to guide EA for architecture search. ProxyBO (Shen et al., 2021) combines bayesian optimization(BO) with zero-cost proxies to select promising architectures. Those methods could deliver promising architectures with much efficient search time, casting light on future NAS research.
Evolutionary Algorithm is one of the most popular methods in NAS. Traditional EA selects parents with top performance from a population of architectures, and then generates

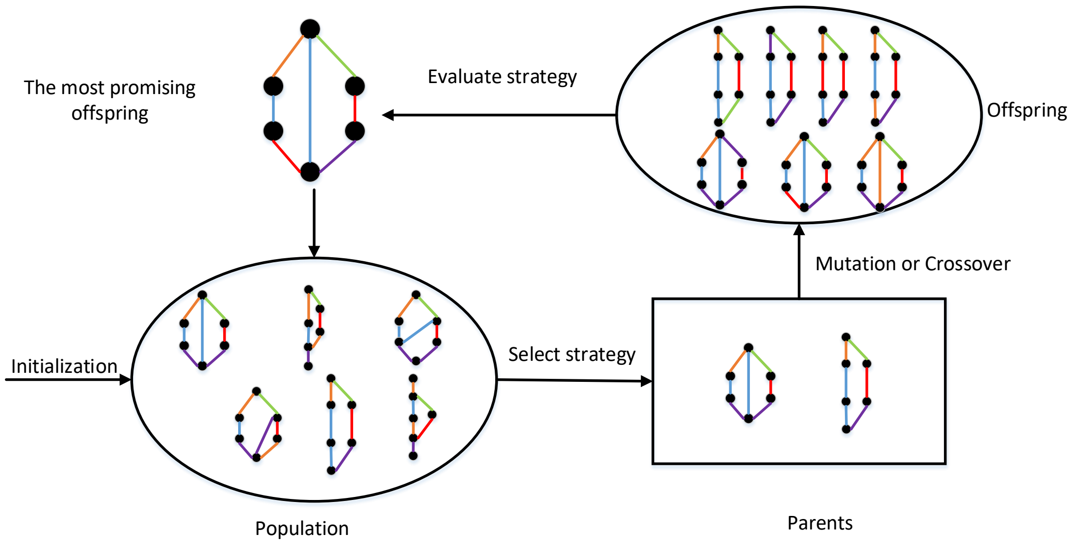


Figure 1: Pipeline of traditional EA

offspring through mutation or crossover (illustrated in Figure 1). At each iteration, the offspring are evaluated and promising ones are added into the population. Xie and Yuille (2017) performs a Russian roulette process to determine which individual will survive or be discarded among the population according to their fitness. REA (Real et al., 2019) proposes an aging evolution which replace the oldest individual in the population with the most promising offspring. However, the time cost is catastrophic to evaluate multiple offspring in order to select the most promising one. Instead, methods (e.g. NPENAS (Wei et al., 2020), and G-EA (Lopes et al., 2021)) employing estimation strategies are proposed for promising architecture selection. Therefore, we are motivated to try the most advanced estimation proxy SSB (Li et al., 2021) on evolutionary algorithm, in order to search for better architectures with reduced search time.

3. EENAS Algorithm

For a certain task, **EENAS** is to search for the optimal architecture in given search space with efficiency. The performance of architectures is evaluated by accuracy, which usually takes expensive training cost for validation. To reduce training cost, **EENAS** adapts the most advanced estimation proxy SSB (Li et al., 2021) to efficiently estimate EA offspring. In addition, the SSB proxy is progressively optimized with the increase of evaluated architectures (with validated accuracy), so as to deliver more reliable estimation.

3.1. EENAS Framework

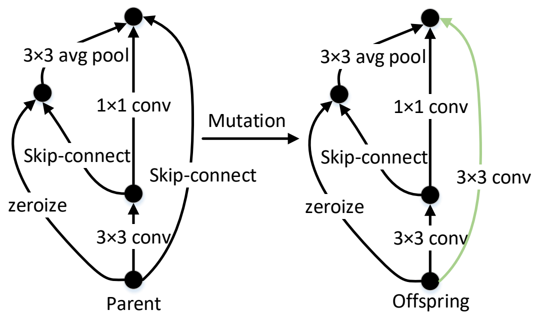
EENAS adopts evolutionary algorithm for search space exploration. As shown in Algorithm 1, **EENAS** is to identify the architecture of highest validated accuracy in search

Algorithm 1 EENAS

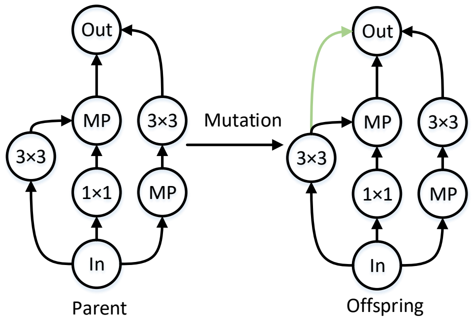
```
Input: Search space $\mathcal{A}$, Max iteration $\mathcal{T}$, Population size $\mathcal{PS}$, Random probability $\mathcal{E}$,
    Random pool size $\mathcal{RPS}$, Parent candidate size $\mathcal{PCS}$, Mutation size $\mathcal{MS}$, Proxy update
    interval $M$
Output: The architecture with highest validated accuracy
 1: $population \leftarrow$ empty queue
 2: $history \leftarrow \emptyset$
 3: while size of $population < \mathcal{PS}$ do
 4:     $arch$ = RandomArchitecture()
 5:     $accuracy$ = ValidatedAccuracy($arch$)
 6:     $history = history \cup (arch, accuracy)$
 7:     $population$.add($arch$)
 8: end while
 9: $proxy \leftarrow$ OptimalSSB($history$)
10: for $iter$ from 1 to $\mathcal{T}$ do
11:     if Random() $>= \mathcal{E}$ then
12:         $candidates \leftarrow$ Random.sample($population$, $\mathcal{PCS}$)
13:         $parent \leftarrow$ architecture of highest validated accuracy in $candidates$
14:         $offspring \leftarrow$ MUTATE($parent$, $MS$)
15:         $removal \leftarrow parent$
16:     else
17:         $offspring \leftarrow$ Random.sample($\mathcal{A} \backslash history$, $\mathcal{RPS}$)
18:         $removal \leftarrow$ architecture of lowest validated accuracy in $population$
19:     end if
20:     $individual \leftarrow$ Estimator($offspring$, $proxy$)
21:     $individual.accuracy$ = ValidatedAccuracy($individual$)
22:     $history = history \cup (individual, individual.accuracy)$
23:     if $individual.accuracy > removal.accuracy$ then
24:         $population$.add($individual$)
25:         $population$.delete($removal$)
26:     end if
27:     if $iter! = 0$ and $iter\%M == 0$ then
28:         $proxy \leftarrow$ OptimalSSB($history$)
29:     end if
30: end for
31: return Architecture with highest validated accuracy in $history$
```

space $\mathcal{A}$ with given iterations $\mathcal{T}$. The population contains architectures with validated accuracy, which are initialized with $\mathcal{PS}$ architectures randomly sampled from search space $\mathcal{A}$ and validated by training. Architectures with validated accuracy in the population are kept in history(Lines 3-8). Then the estimator SSB proxy is initialized with the initial history by function $OptimalSSB()$ (Line 9), which will return the optimal proxy for the given history.

At each iteration, offspring will be generated by either parent mutation (Line 14) or random selection (Line 17). With a probability no less than $\mathcal{E}$, offspring are generated by



(a) NAS-Bench-201 search space



(b) NAS-Bench-101 search space

Figure 2: Examples of offspring produced by mutation on NAS-Bench-201 and NAS-Bench-101 search spaces, with 1×1, 3×3, and MP referring to 3×3 convolution, 1×1 convolution, and 3×3 max-pool, respectively.

parent mutation. And the parent is the architecture with the highest validated accuracy in parent candidate set, randomly sampled from the population. Mutation is conducted by randomly replacing one operation with another (e.g. NAS-Bench-201 search space in Figure 2(a)), or adding/removing an edge between two nodes (e.g. NAS-Bench-101 search space in Figure 2(b)). To avoid local optimization, offspring will also be generated by random sampling from search space outside the history, with a probability less than $\mathcal{E}$. We suggest that the number of sampled offspring ($\mathcal{RPS}$) should be much larger than that of mutated offspring ($\mathcal{MS}$), in that random sampling might have less chance to get outstanding architectures.

Instead of evaluating offspring by expensive training, SSB proxy is employed as an estimator to efficiently return the promising architecture (with lowest proxy score) of all offspring (Line 20). Only the promising architecture is evaluated by training to get its validated accuracy (Line 21), and then kept in the history (Line 22). At the end of each iteration, the population is updated (Lines 23-26) in the way that 1) mutated offspring with higher validated accuracy (better) than its parent will replace its parent; and 2) random offspring better than the worst architecture in the population will replace the worst architecture. As such, the population could be progressively improved and no architecture would dominate the population. In order to get more reliable offspring estimation, the SSB proxy is progressively optimized with the increase of history in every M iterations (Lines 27-29).

Figure 3 illustrates the search procedure for optimal architecture on NAS-Bench-201 CIFAR-100 via t-SNE, where architecture ranking is encoded by different colors. The rankings of all architectures on NAS-Bench-201 search space are shown in Figure 3(a), with the red star representing the optimal architecture. In the first iteration (Figure 3(b)), offspring (square points) of the initial population (round points) are generated by mutation, from which the most promising architecture (magenta triangle point) is figured out by SSB estimator. Later in the third iteration (Figure 3(c)), more offspring are generated by random sampling from the rest of the search space, and another promising architecture is detected.

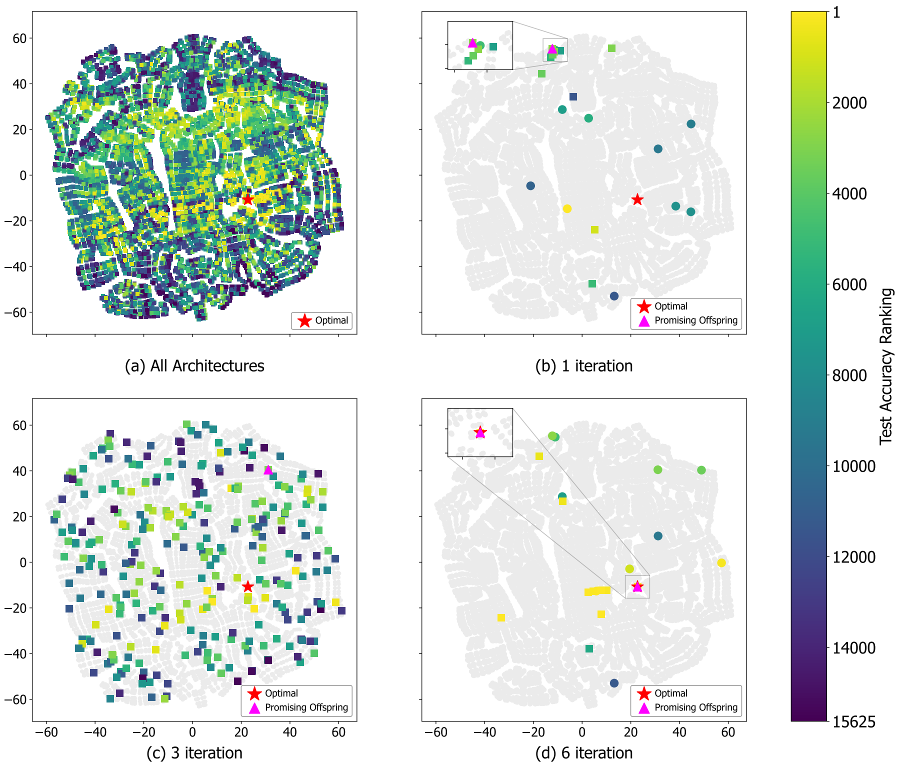


Figure 3: Visualization of optimal architecture search on NAS-Bench-201 CIFAR-100 via t-SNE. (a) The ranking of all architectures on NAS-Bench-201 search space. (b) The first iteration with offspring generated by mutation. (c) The third iteration with offspring generated by random sampling. (d) The sixth and finial iteration with optimal architecture returned from offspring. Note that the round points, square points, magenta triangle point, and red star denote population, offspring, most promising offspring, optimal architecture, respectively.

Finally in the sixth iteration (Figure 3(d)), the optimal architecture is among the mutated offspring and successfully returned by SSB estimator.

3.2. Estimation by SSB Proxy

Efficient architecture estimation is an important guidance for offspring selection. Here, we take the most advanced SSB proxy (Li et al., 2021) for architecture estimation, which contains four types of synthetic signal basis: (1) 1-D frequency basis (Sin1D defined in Equation 1, with x and y representing pixel indices); (2) 2-D frequency basis (Sin2D defined in Equation 2); (3) Spatial basis (Dot and GDot); and (4) Resized input signal (Resize). Dot is defined as randomly setting k % pixels to ± 1 on zeroed feature maps according to biased Rademacher distribution (Montgomery-Smith, 1990), and GDot is generated by applying a Gaussian filter with $\sigma = 1$ on Dot and normalizing between ± 1.

$$\text{Sin1D} : \sin(2\pi f x + \phi) \text{ or } \sin(2\pi f y + \phi), \tag{1}$$

$$\text{Sin2D}: \ \sin(2\pi f_x x + 2\pi f_y y + \phi), \tag{2}$$

For a certain architecture estimation task, function $OptimalSSB()$ (Lines 9 and 28 in Algorithm 1) will search for the optimal synthetic signal basis combination in given SSB search space. Spearman's ρ is taken to evaluate the estimation performance of SSB combination. SSB combination with highest Spearman's ρ will be returned as the best SSB proxy. SSB proxy can efficiently estimate an architecture by calculating the MSE loss between feature map tensor and synthetic signal tensor, detailed in Li et al. (2021). Architecture with lowest proxy score (MSE loss) will be returned by function $Estimator()$ (Line 20 in Algorithm 1) as optimal architecture.

As it is time expensive to get SSB combinations from SSB search space, SSB combinations are searched only once with initial architecture history, and those with higher Spearman's ρ are kept in a proxy pool. In later SSB proxy optimization, Spearman's ρ of each combination in the proxy pool is recalculated according to updated history, based on which the best SSB combination is returned by $OptimalSSB()$.

4. Experimental Results

In this section, we take NAS-Bench-101 search space (Ying et al., 2019) and NAS-Bench-201 search space (Dong and Yang, 2020) as benchmark, and compare the performance of **EENAS** with state-of-the-art NAS methods. Estimation performance of SSB proxy (Li et al., 2021) is also studied.

4.1. Benchmark

NAS-Bench-101 search space is the first benchmark for image classification on CIFAR-10, which contains 423,624 different cell-based architectures. The maximum number of nodes and edges for each cell are 7 and 9, respectively. Each node can be chosen from operators of 1×1 convolution, 3×3 convolution, and 3×3 max-pooling. The edges between two nodes are represented by a 7×7 upper-triangular binary matrix.
NAS-Bench-201 search space contains 15625 different architectures, stacked by repeated cells. For each cell, the number of edges and nodes are fixed at 6 and 4, respectively. Different from NAS-Bench-101, the edge in NAS-Bench-201 represents an operation that can be chosen from a candidate pool with 1×1 convolution, 3×3 convolution, 3×3 avg-pooling, skip-connect, and zeroize. NAS-Bench-201 provides results of each architecture on three datasets: CIFAR-10, CIFAR-100 and ImageNet16-120.

Both search spaces provide validation accuracy and test accuracy in three different runs. We take the average validation accuracy as the return for function $ValidatedAccuracy()$ in Algorithm 1, and employ average test accuracy for experimental evaluation.

4.2. Parameter Setting

In our experiments, the parameters of our algorithms are set as follows: max iteration for architecture search $\mathcal{T} = 90$; population size $\mathcal{PS} = 10$ and parent candidate size $\mathcal{PCS} = 4$; probability to generate random sampled offspring $\mathcal{E} = 0.2$; random pool size for sampled offspring $\mathcal{RPS} = 400$; mutation size for mutated offspring $\mathcal{MS} = 10$; SSB proxy update

Table 1: Performance Evaluation of SSB proxy by Spearman's ρ (@iteX refers to X^{th} iteration).

	@initialization	@ite10	@ite20	@ite30	@ite40	@ite50	@ite60	@ite70	@ite80
Mean	0.912	0.918	0.920	0.928	0.931	0.934	0.931	0.934	0.930
Std	0.041	0.041	0.025	0.027	0.036	0.026	0.030	0.028	0.030
Max	0.962	0.951	0.964	0.957	0.954	0.972	0.966	0.968	0.970
Min	0.783	0.817	0.853	0.841	0.858	0.864	0.849	0.857	0.849

interval $M = 10$. We maintain a SSB proxy pool of size 20. All results are reported based on the average of 50 repeated trials.

4.3. Performance Evaluation on SSB Proxy

As SSB Proxy is employed as important guidance for architecture search in **EENAS**, we firstly evaluate the performance of SSB proxy on architecture estimation. We randomly sample 1000 architectures from NAS Bench-101 search space, to calculate Spearman's ρ (Li et al., 2021) of SSB proxy on different **EENAS** iterations.

As shown in Table 1, the Spearman's ρ is getting better with the increase of iterations, indicating that the estimation accuracy of SSB proxy could be improved with the increase of evaluated architectures. Therefore, progressively updating SSB proxy is necessary in optimal architecture search. As we find that ρ becomes relatively stable after 40 iterations, we stop updating SSB proxy after 40 iterations in the following experiments.

Table 2: Comparison of EENAS vs. state-of-the-art NAS methods on four benchmarks.

Method	Validated No.	NAS-Bench-101	NAS-Bench-201		
			CIFAR-10(VALID)	CIFAR-100	ImageNet16-120
Traditional Methods					
RS	100	6.41 ± 0.14	9.27 ± 0.30	28.41 ± 0.73	54.48 ± 0.55
RL	100	6.36 ± 0.12	9.15 ± 0.25	28.35 ± 0.66	54.21 ± 0.66
REA	100	6.32 ± 0.20	8.82 ± 0.30	27.33 ± 0.78	53.59 ± 0.51
NAS-BOWL	100	5.95 ± 0.14	8.68 ± 0.30	26.70 ± 0.41	53.32 ± 0.34
Weight-sharing Methods					
ENAS	≈7	8.17 ± 0.42	46.11 ± 0.58	86.04 ± 2.33	85.19 ± 2.10
DARTS-PT	≈18	7.79 ± 0.61	15.33 ± 2.23	34.03 ± 2.24	61.36 ± 1.91
Predictor-based Methods					
BRP	100	6.18 ± 0.19	8.77 ± 0.29	26.82 ± 0.50	53.43 ± 0.59
WeakNAS	100	6.01 ± 0.12	8.64 ± 0.27	26.78 ± 0.49	53.51 ± 0.32
Proxy-based Methods					
G-EA	100	6.77 ± 0.21	8.73 ± 0.25	26.82 ± 0.38	53.39 ± 0.23
ProxyBO	100	6.00 ± 0.15	8.58 ± 0.12	26.68 ± 0.18	53.19 ± 0.31
EENAS	100	$\mathbf{5.82 \pm 0.04}$	$\mathbf{8.49 \pm 0.03}$	$\mathbf{26.49 \pm 0.00}$	$\mathbf{52.74 \pm 0.07}$
Benchmark Optimal	/	5.68	8.48	26.49	52.69

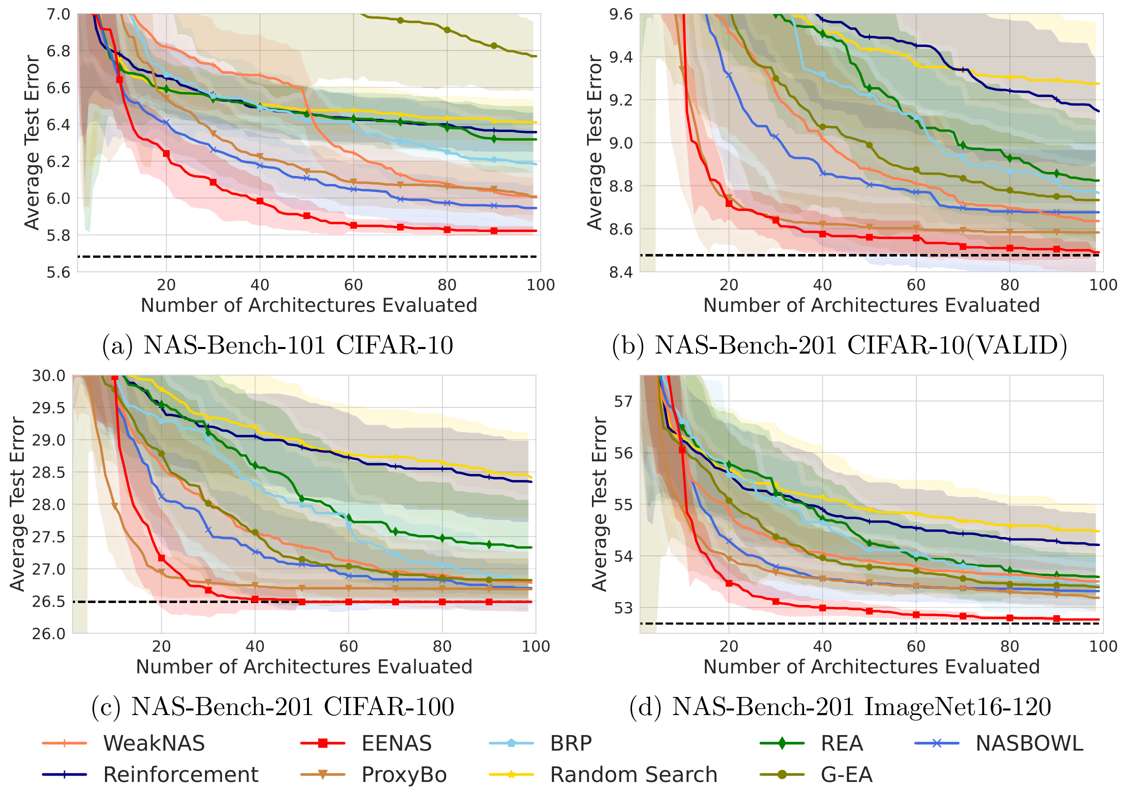


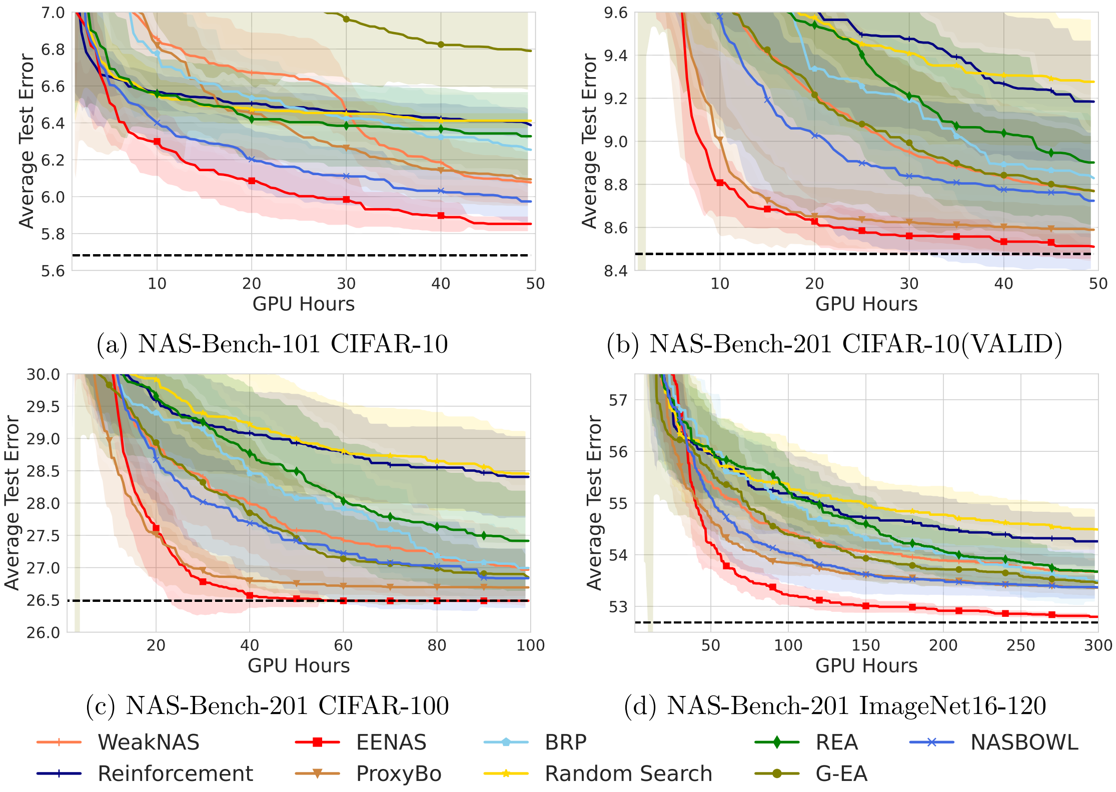
Figure 4: Average test error vs. No. of evaluated architectures. Black dash lines refer to global optima.

4.4. Comparative Studies of EENAS vs. State-of-the-art NAS Methods

Now that we are to compare the performance of **EENAS** with State-of-the-art NAS Methods, including some traditional methods (RS (Bergstra and Bengio, 2012), RL (Zoph and Le, 2017), REA (Real et al., 2019), NAS-BOWL (Ru et al., 2021)), weight-sharing methods (ENAS (Pham et al., 2018), DARTS-PT (Wang et al., 2021)), predictor-based methods (BRP (Dudziak et al., 2020) , WeakNAS (Wu et al., 2021)), and proxy-based methods (G-EA (Lopes et al., 2021), ProxyBO (Shen et al., 2021)). All baselines are implemented according to their original papers.

Table 2 shows the average test accuracy (with mean± std) of optimal architecture identified by **EENAS** against state-of-the-art NAS methods in 50 repeated trials. The second column shows the validated number of architectures, while the last row delivers the average test accuracy of optimal architecture reported by the benchmarks. Compared with other baselines, **EENAS** delivers lowest average test accuracy with smallest standard deviation. That is, **EENAS** could get the best architecture on the benchmarks in average, with the same number of validated architectures. Specially, **EENAS** can successfully identify the optimal architecture on NAS-Bench-201 CIFAR-100 in every trial.

Figure 4 demonstrates the change of average test error in the process of architecture search. We can see that with the increase of evaluated architectures (iterations), the average



(a) NAS-Bench-101 CIFAR-10 (b) NAS-Bench-201 CIFAR-10(VALID)

(c) NAS-Bench-201 CIFAR-100 (d) NAS-Bench-201 ImageNet16-120

Figure 5: Average test error vs. architecture search time. Black dash lines refer to global optima.

test error delivered by architecture identified by **EENAS** drops fastest among all baselines. That is, **EENAS** could identify better architectures with more efficient search time, because architecture evaluation takes the most expensive training cost in search time.

As different baselines have different process time for each search iteration, we have done further experiments to study the change of average test error with the increase of search time. As shown in Figure 5, **EENAS** outperforms other baselines by delivering better architectures that drop the average test error fastest.

5. Conclusion

In this paper, we propose an efficient and effective NAS method for optimal architecture search, by adapting SSB proxy to evolutionary algorithm. Extensive experiments show that our method outperforms state-of-the-art NAS methods on NAS-Bench-101 search space and NAS-Bench-201 search space (CIFAR-10, CIFAR-100 and ImageNet16-120). Compared with other baselines, our method could identify better architectures with greatly reduced search time.

References

Mohamed S Abdelfattah, Abhinav Mehrotra, Łukasz Dudziak, and Nicholas D Lane. Zero-cost proxies for lightweight nas. *arXiv preprint arXiv:2101.08134*, 2021.

James Bergstra and Yoshua Bengio. Random search for hyper-parameter optimization. *Journal of machine learning research*, 13(2), 2012.

Wuyang Chen, Xinyu Gong, Xianming Liu, Qian Zhang, Yuan Li, and Zhangyang Wang. Fasterseg: Searching for faster real-time semantic segmentation. *arXiv preprint arXiv:1912.10917*, 2019.

Wuyang Chen, Xinyu Gong, and Zhangyang Wang. Neural architecture search on imagenet in four gpu hours: A theoretically inspired perspective. *arXiv preprint arXiv:2102.11535*, 2021.

Shaojin Ding, Tianlong Chen, Xinyu Gong, Weiwei Zha, and Zhangyang Wang. Autospeech: Neural architecture search for speaker recognition. *arXiv preprint arXiv:2005.03215*, 2020.

Xuanyi Dong and Yi Yang. Nas-bench-201: Extending the scope of reproducible neural architecture search. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net, 2020. URL https://openreview.net/forum?id=HJxyZkBKDr.

Lukasz Dudziak, Thomas Chau, Mohamed Abdelfattah, Royson Lee, Hyeji Kim, and Nicholas Lane. Brp-nas: Prediction-based nas using gcns. *Advances in Neural Information Processing Systems*, 33:10480–10490, 2020.

Golnaz Ghiasi, Tsung-Yi Lin, and Quoc V Le. Nas-fpn: Learning scalable feature pyramid architecture for object detection. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 7036–7045, 2019.

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 *European conference on computer vision*, pages 544–560. Springer, 2020.

Arthur Jacot, Franck Gabriel, and Clément Hongler. Neural tangent kernel: Convergence and generalization in neural networks. *Advances in neural information processing systems*, 31, 2018.

Kirthevasan Kandasamy, Willie Neiswanger, Jeff Schneider, Barnabas Poczos, and Eric P Xing. Neural architecture search with bayesian optimisation and optimal transport. *Advances in neural information processing systems*, 31, 2018.

Namhoon Lee, Thalaiyasingam Ajanthan, and Philip H. S. Torr. Snip: single-shot network pruning based on connection sensitivity. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net, 2019. URL https://openreview.net/forum?id=B1VZqjAcYX.

Yuhong Li, Cong Hao, Pan Li, Jinjun Xiong, and Deming Chen. Generic neural architecture search via regression. *Advances in Neural Information Processing Systems*, 34, 2021.

Chenxi Liu, Liang-Chieh Chen, Florian Schroff, Hartwig Adam, Wei Hua, Alan L Yuille, and Li Fei-Fei. Auto-deeplab: Hierarchical neural architecture search for semantic image segmentation. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 82–92, 2019a.

Hanxiao Liu, Karen Simonyan, and Yiming Yang. DARTS: differentiable architecture search. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net, 2019b. URL https://openreview.net/forum?id=S1eYHoC5FX.

Vasco Lopes, Miguel Santos, Bruno Degardin, and Luís A. Alexandre. Guided evolution for neural architecture search. *CoRR*, abs/2110.15232, 2021. URL https://arxiv.org/abs/2110.15232.

Shun Lu, Jixiang Li, Jianchao Tan, Sen Yang, and Ji Liu. Tnasp: A transformer-based nas predictor with a self-evolution framework. *Advances in Neural Information Processing Systems*, 34, 2021.

Joe Mellor, Jack Turner, Amos J. Storkey, and Elliot J. Crowley. Neural architecture search without training. In Marina Meila and Tong Zhang, editors, *Proceedings of the 38th International Conference on Machine Learning, ICML 2021, 18-24 July 2021, Virtual Event*, volume 139 of *Proceedings of Machine Learning Research*, pages 7588–7598. PMLR, 2021. URL http://proceedings.mlr.press/v139/mellor21a.html.

Stephen J Montgomery-Smith. The distribution of rademacher sums. *Proceedings of the American Mathematical Society*, 109(2):517–522, 1990.

Vladimir Nekrasov, Hao Chen, Chunhua Shen, and Ian Reid. Fast neural architecture search of compact semantic segmentation models via auxiliary cells. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 9126–9135, 2019.

Xuefei Ning, Yin Zheng, Tianchen Zhao, Yu Wang, and Huazhong Yang. A generic graph-based neural architecture encoding scheme for predictor-based nas. In *European Conference on Computer Vision*, pages 189–204. Springer, 2020.

Junran Peng, Ming Sun, ZHAO-XIANG ZHANG, Tieniu Tan, and Junjie Yan. Efficient neural architecture transformation search in channel-level for object detection. *Advances in Neural Information Processing Systems*, 32, 2019.

Hieu Pham, Melody Y. Guan, Barret Zoph, Quoc V. Le, and Jeff Dean. Efficient neural architecture search via parameter sharing. In Jennifer G. Dy and Andreas Krause, editors, *Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholmsmässan, Stockholm, Sweden, July 10-15, 2018*, volume 80 of *Proceedings of Machine Learning Research*, pages 4092–4101. PMLR, 2018. URL http://proceedings.mlr.press/v80/pham18a.html.

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 *International Conference on Machine Learning*, pages 2902–2911. PMLR, 2017.

Esteban Real, Alok Aggarwal, Yanping Huang, and Quoc V. Le. Regularized evolution for image classifier architecture search. In *The Thirty-Third AAAI Conference on Artificial Intelligence, AAAI 2019, The Thirty-First Innovative Applications of Artificial Intelligence Conference, IAAI 2019, The Ninth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2019, Honolulu, Hawaii, USA, January 27 - February 1, 2019*, pages 4780–4789. AAAI Press, 2019. doi: 10.1609/aaai.v33i01.33014780. URL https://doi.org/10.1609/aaai.v33i01.33014780.

Bin Xin Ru, Xingchen Wan, Xiaowen Dong, and Michael A. Osborne. Interpretable neural architecture search via bayesian optimisation with weisfeiler-lehman kernels. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net, 2021. URL https://openreview.net/forum?id=j9Rv7qdXjd.

Yu Shen, Yang Li, Jian Zheng, Wentao Zhang, Peng Yao, Jixiang Li, Sen Yang, Ji Liu, and Cui Bin. Proxybo: Accelerating neural architecture search via bayesian optimization with zero-cost proxies. *arXiv preprint arXiv:2110.10423*, 2021.

Yao Shu, Shaofeng Cai, Zhongxiang Dai, Beng Chin Ooi, and Bryan Kian Hsiang Low. Nasi: Label-and data-agnostic neural architecture search at initialization. *arXiv preprint arXiv:2109.00817*, 2021.

Yao Shu, Zhongxiang Dai, Zhaoxuan Wu, and Bryan Kian Hsiang Low. Unifying and boosting gradient-based training-free neural architecture search. *arXiv preprint arXiv:2201.09785*, 2022.

Mingxing Tan, Bo Chen, Ruoming Pang, Vijay Vasudevan, M Sandler, A Howard, and Mnasnet Le QV. platform-aware neural architecture search for mobile. 2019 ieee. In *CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2815–2823, 2019.

Hidenori Tanaka, Daniel Kunin, Daniel L. K. Yamins, and Surya Ganguli. Pruning neural networks without any data by iteratively conserving synaptic flow. In Hugo Larochelle, Marc'Aurelio Ranzato, Raia Hadsell, Maria-Florina Balcan, and Hsuan-Tien Lin, editors, *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, 2020. URL https://proceedings.neurips.cc/paper/2020/hash/46a4378f835dc8040c8057beb6a2da52-Abstract.html.

Lucas Theis, Iryna Korshunova, Alykhan Tejani, and Ferenc Huszár. Faster gaze prediction with dense networks and fisher pruning. *CoRR*, abs/1801.05787, 2018. URL http://arxiv.org/abs/1801.05787.

Chaoqi Wang, Guodong Zhang, and Roger B. Grosse. Picking winning tickets before training by preserving gradient flow. In *8th International Conference on Learning Representations,* *ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net, 2020a. URL https://openreview.net/forum?id=SkgsACVKPH.

Ning Wang, Yang Gao, Hao Chen, Peng Wang, Zhi Tian, Chunhua Shen, and Yanning Zhang. Nas-fcos: Fast neural architecture search for object detection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11943–11951, 2020b.

Ruochen Wang, Minhao Cheng, Xiangning Chen, Xiaocheng Tang, and Cho-Jui Hsieh. Rethinking architecture selection in differentiable nas. *arXiv preprint arXiv:2108.04392*, 2021.

Chen Wei, Chuang Niu, Yiping Tang, and Jimin Liang. NPENAS: neural predictor guided evolution for neural architecture search. *CoRR*, abs/2003.12857, 2020. URL https://arxiv.org/abs/2003.12857.

Wei Wen, Hanxiao Liu, Yiran Chen, Hai Li, Gabriel Bender, and Pieter-Jan Kindermans. Neural predictor for neural architecture search. In *European Conference on Computer Vision*, pages 660–676. Springer, 2020.

Colin White, Willie Neiswanger, and Yash Savani. Bananas: Bayesian optimization with neural architectures for neural architecture search. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pages 10293–10301, 2021.

Junru Wu, Xiyang Dai, Dongdong Chen, Yinpeng Chen, Mengchen Liu, Ye Yu, Zhangyang Wang, Zicheng Liu, Mei Chen, and Lu Yuan. Stronger nas with weaker predictors. *Advances in Neural Information Processing Systems*, 34:28904–28918, 2021.

Lingxi Xie and Alan Yuille. Genetic cnn. In *Proceedings of the IEEE international conference on computer vision*, pages 1379–1388, 2017.

Jingjing Xu, Liang Zhao, Junyang Lin, Rundong Gao, Xu Sun, and Hongxia Yang. Knas: green neural architecture search. In *International Conference on Machine Learning*, pages 11613–11625. PMLR, 2021.

Chris Ying, Aaron Klein, Eric Christiansen, Esteban Real, Kevin Murphy, and Frank Hutter. Nas-bench-101: Towards reproducible neural architecture search. In Kamalika Chaudhuri and Ruslan Salakhutdinov, editors, *Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA*, volume 97 of *Proceedings of Machine Learning Research*, pages 7105–7114. PMLR, 2019. URL http://proceedings.mlr.press/v97/ying19a.html.

Barret Zoph and Quoc V. Le. Neural architecture search with reinforcement learning. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net, 2017. URL https://openreview.net/forum?id=r1Ue8Hcxg.

Barret Zoph, Vijay Vasudevan, Jonathon Shlens, and Quoc V. Le. Learning transferable architectures for scalable image recognition. In *2018 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2018, Salt Lake City, UT, USA, June 18-22, 2018*, pages 8697–8710. Computer Vision Foundation / IEEE Computer Society, 2018. doi: 10.1109/CVPR.2018.00907. URL http://openaccess.thecvf.com/content_cvpr_2018/html/Zoph_Learning_Transferable_Architectures_CVPR_2018_paper.html.