

Rank-based Training-Free NAS Algorithm

Hsieh Cheng-Han

emiliastruelove@gmail.com

Department of Computer Science and Engineering,
National Sun Yat-sen University
Kaohsiung, Taiwan

Chun-Wei Tsai

cwtsai@mail.cse.nsysu.edu.tw

Department of Computer Science and Engineering,
National Sun Yat-sen University
Kaohsiung, Taiwan

ABSTRACT

Although the training-free neural architecture search (NAS) are typically faster than training-based NAS methods, however, the correlation between the measurement and the final accuracy of an architecture is not well enough in most cases. To address this problem, we propose a rank-based training-free NAS algorithm, which combines three complement training-free score functions to evaluate an architecture by ranking. More precisely, noise immunity (NI) and the correlation between binary activation patterns, named as NASWOT, are used as measurement of image recognition ability, and, SynFlow are used as measurement of trainability. With that, a modified version of simulated annealing (SA) algorithm can applies on and obtains a better performance on searching high accuracy architectures. To evaluate the performance of the proposed algorithm, this paper compared it with several NAS algorithms, including weight-sharing methods, non-weight-sharing methods, and several state-of-the-art training-free score functions. The final result shows that in relatively larger search space, like NATS-Bench SSS, the proposed algorithm outperforms most of NAS methods.

ACM Reference Format:

Hsieh Cheng-Han and Chun-Wei Tsai. 2023. Rank-based Training-Free NAS Algorithm. In *Proceedings of ACM Conference (Conference'17)*. ACM, New York, NY, USA, 6 pages. <https://doi.org/10.1145/nnnnnnn.nnnnnnn>

1 INTRODUCTION

Neural architecture search (NAS) has recently drawn a big amount of attention, since the ability to automatically design a "good" neural architecture. By leveraging machine learning algorithms [1], NAS algorithms can explore a search space, which is comprised of numerous potential architectures, to find out a good architectures that outperform those designed by human experts. Recently, the use of NAS is widespread, from object detection [2], image recognition [3] and speech recognition [4] [5] to natural language processing. [6] [7] [8]

Despite the promising results of NAS, there are still many challenges to conquer. One major problem is the extremely high computational cost to search for an optimal architecture, which can make NAS impractical for real-world applications, particularly on resource-constrained platforms like embedded system. The reason

Table 1: THE KENDALL CORRELATION BETWEEN TRAINING-FREE SCORE AND TEST ACCURACY, EVALUATED ON THE THREE DATASETS OF NATS-BENCH [17]. EACH METRIC HAS BEEN COMPUTED THREE TIMES WITH DIFFERENT INITIALIATIONS AND THE AVERAGE IS TAKEN AS FINAL SCORE.

Training-free score function	CIFAR-10	CIFAR-100	ImageNet-16-120
NTK	-0.33	-0.30	-0.39
Snip	0.45	0.47	0.41
Fisher	0.39	0.40	0.36
Grasp	0.28	0.35	0.35
NASWOT	0.61	0.62	0.60
SynFlow	0.57	0.56	0.56
LogSynFlow	0.61	0.60	0.59
Rank-Based (ours)	X	X	X

why NAS is costly is that during the searching, a candidate architecture must be trained to evaluate how good of this architecture.

To overcome this challenge, recent works developed and proposed lots of method which is so called training-free NAS. For example, Mellor et al. [3] proposed the measurement of the correlation between the binary activation patterns, induced by the untrained network at two inputs, named as neural architecture search without training (NASWOT). On the other hand, Lee et al. [9] proposed pruning parameters based on a saliency matrix, which then extended further by Wang et al. [10] and Tanaka et al. [11]. In [11], Tanaka et al. proposed Iterative Synaptic Flow Pruning (SynFlow) algorithm which intends to deal with the layer collapse problem when pruning a network. The score function used in the algorithm is so-called *synaptic saliency* score. Later, Abdelfattah et al. [12] extended SynFlow to score a network architecture by summing synaptic saliency score over all parameters in the model. Cavagnero et al. [13] found that SynFlow is likely to suffer from gradient explosion, then proposed the LogSynFlow score function which prevents the problem. For another example, Chen et al. [14] proposed to compute the condition number of neural tangent kernel (NTK) [15] [16], which is used as the score to estimate the trainability of an architecture.

However, Table 1 shows most of score functions suffer from low correlation between score values and the final accuracy of architectures, leading to a predicament that no matter how good the search method is used, we can hardly find an optimal architecture. The major problem causes the low correlation is that a single score function can only evaluate one perspective/characteristic of an architecture. To address the problem, we propose cooperating three complement score functions by ranking, which allows every score functions evaluate an architecture without losing fairness. The first score function

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.

Conference'17, July 2017, Washington, DC, USA

© 2023 Association for Computing Machinery.

ACM ISBN 978-x-xxxx-xxxx-x/YY/MM...\$15.00

<https://doi.org/10.1145/nnnnnnn.nnnnnnn>

is noise immunity (NI) [18], as an indicator of the ability to distinguish two images. The second one is NASWOT, as score to estimate the ability of expression of an architecture and also a complement of NI. The last one is SNIP used as the measurement of trainability and also as a complement to NI and NASWOT. With these three complement score functions, an evaluation of an architecture is not simply from a single aspect but three different aspects, which is like estimate the weight of an object by not only the length but its height and width.

The remainder of this paper is organized as follows: Section 2 provides the detail about the three complement score functions. Section 3 gives a detailed description about the proposed method. Section 4 begins with parameters setting and provide the simulation results following is the experiment results in different search space. The conclusion and further prospect are given in Section 5.

2 RELATED WORKS

2.1 Training-free Score Functions

In [3], Mellor et al. proposed a score function without the requirement for training which is named neural architecture search without training (NASWOT). Figure 1 gives a simple example to illustrate the procedure of NASWOT score function. Consider a mini-batch of data $X = \{x_i\}_{i=1}^N$ passing through a neural network architecture. The activated ReLU units in every layer of the architecture form a binary code c_i that define the linear region. The correlation between binary codes for the whole mini-batch can be examined by computing the kernel matrix

$$K_H = \begin{pmatrix} N_A - d_H(c_1, c_1) & \cdots & N_A - d_H(c_1, c_N) \\ \vdots & \ddots & \vdots \\ N_A - d_H(c_N, c_1) & \cdots & N_A - d_H(c_N, c_N) \end{pmatrix}, \quad (1)$$

where N_A is the number of ReLU units and $d_H(c_i, c_j)$ is the hamming distance between the binary code c_i and c_j . With the kernel matrix, the score of an architecture can be evaluated as follow:

$$s = \log|K_H|, \quad (2)$$

The rationale behind is that, the more different between the binary codes the better the architecture learns. And the determinant of the kernel matrix measures how "different" they are by calculating the volume formed by the row vector of K_H .

Wu et al. [18] found, in some case, an architecture with high NASWOT score may classify the same kind of input data into different classes. To fix this defect, Wu et al. proposed using noise immunity (NI) to evaluate an architecture. Figure 2 gives an example

to illustrate how noise immunity evaluate an architecture. The score function picks a mini-batch of data, denoted X , and then applies Gaussian noise on it. The process can be defined by $X' = X + z$ where z is the Gaussian noise. By passing X and X' through the untrained architecture, then computing the difference of square of each feature maps captured at all pooling layers, denoted η . According to the fact that the input data are the same kind, the small η the better the noise immunity of the architecture is.

Besides from these two image-related aspects, there is another aspect to evaluate a network. *The Lottery Ticket Hypothesis* [21], reveals that a network may have a "core" which decides the final accuracy of the network. How to pruning a network correctly is therefore tempting. Lee et al. [9] proposed to use connection sensitivity as a criterion to prune the network which can be briefly viewed as

$$s_j = |w_j \frac{\partial \mathcal{L}}{\partial w_j}|, \quad (3)$$

where \mathbf{w} is the weight of the network, w_j the j -th element of \mathbf{w} , \mathcal{D} the dataset, \mathbf{e} the connection mask and \odot is Hadamard product. The meaning behind is to approximate the contribution to the change of the loss function from a specific connection. By pruning the connections which has relatively small contribution, the expensive prune, retrain cycles, can be prevented. Later, Wang et al. [10] noticed that SNIP with a high pruning ratio tends to cause layer-collapse, which prune all parameters in a single weight layer. Therefore, they propose to approximate the change in gradient norm which defined as

$$S_p(\theta) = -(H \frac{\partial \mathcal{L}}{\partial \theta}) \odot \theta, \quad (4)$$

where H is the Hessian matrix, θ is the parameters. In [11], Tanaka et al. generalized the synaptic saliency scores as

$$S_p(\theta) = \frac{\partial \mathcal{L}}{\partial \theta} \odot \theta, \quad (5)$$

which is named SynFlow. Abdelfattah et al. [12] extended the work, conclude that summing synaptic saliency score over all parameters in the model can be used to score a network architecture, which is formula as

$$S_n = \sum_i^N S_p(\theta)_i \quad (6)$$

Finally, Cavagnero et al. [13] improved SynFlow, which is likely to fall into gradient explosion problem, and proposed LogSynFlow score function, simply scaling down the gradient. The formula is defined by

$$S(\theta) = \theta \cdot \log\left(\frac{\partial \mathcal{L}}{\partial \theta} + 1\right) \quad (7)$$

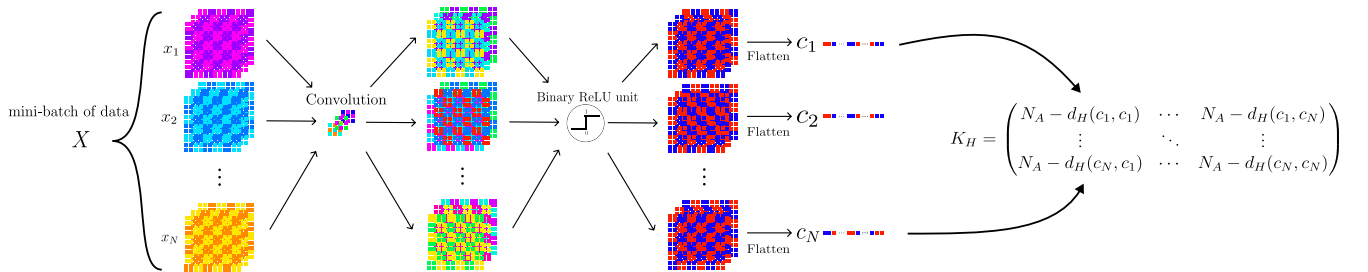


Figure 1: A simple example to illustrate the procedure of NASWOT.

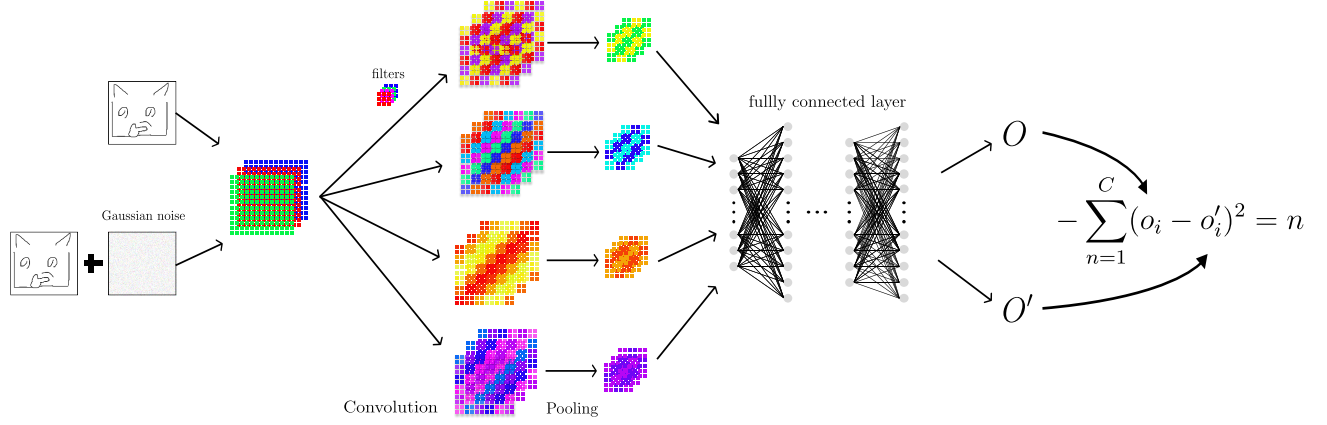


Figure 2: A simple example to illustrate the procedure of noise immunity.

2.2 Search Algorithms

First assume that input data D are separated into two subset, D_r and D_t . Further more, assume \mathcal{F}_A is the accuracy function, \mathcal{F}_S the score function, \mathcal{A}_S the search algorithm of NAS, and \mathcal{A}_L the learning algorithm. Then, NAS problem can be described as an optimization problem as follows:

$$\mathbb{N}^* = \max_{\mathbb{N}^b \in \mathbb{N}} \mathcal{F}_A(\mathcal{A}_L(\mathbb{N}^b, D_r), D_t) \quad (8)$$

where \mathbb{N} is a set of neural architectures, namely, the search space of NAS. For non-training-free NAS

$$\mathbb{N}^b = \mathcal{A}_S(\mathcal{F}_A(\mathcal{A}_L(\mathbb{N}^s, D_r), D_t), \mathbb{N}) \quad (9)$$

and for training-free NAS

$$\mathbb{N}^b = \mathcal{A}_S(\mathcal{F}_S(\mathbb{N}^s, D_r), \mathbb{N}) \quad (10)$$

It can be seen that training-free NAS is comprised of two part, search algorithm \mathcal{A}_S and score function \mathcal{F}_S . There are several search algorithms applied on NAS, including random search, reinforcement learning, and metaheuristic algorithm.

In [3] [22], random search is applied on. The advantage of random search is simple, easy to implement, and the result can be taken as a baseline compared to more comprehensive search algorithm. Generally speaking, when applying the random search on a small search space, e.g., nasbench201 [23], the performance is similar to other search algorithms. But when it comes to a relatively larger search space, e.g., nasbench101 [24] and natsbenchss [17], random search can no longer stand out in other search algorithms.

In [25], reinforcement learning is used to find the maximum accuracy of an architecture generated by a network architecture controller. The actions $a_{1:T}$ of the reinforcement learning is equivalent to updating the parameters θ_c for the controller. More specifically, the goal is to maximize its expected reward, defined by

$$J(\theta_c) = E_{P(a_{1:T}; \theta_c)} [R] \quad (11)$$

and the gradient of $J(\theta_c)$ is defined by

$$\nabla_{\theta_c} J(\theta_c) = \sum_t^T = 1 E_{P(a_{1:T}; \theta_c)} [\nabla_{\theta_c} \log P(a_t | a_{(t-1):1}; \theta_c) R] \quad (12)$$

As for an example of metaheuristic algorithm, in [26], Wu et al. leveraged genetic algorithm (GA) as search strategy. By encoding the network architecture, the architecture can be viewed as gene. Therefore, GA can be easily applied on. Later in [18], Wu et al. used search economic (SE) [27] as search strategy. The basic idea of SEs is to first divide the search space into a set of subspaces and investigate those subspaces based on the expected value of each subspace. The expected value is comprised of

- The number of times the subspace has been investigated.
- The average objective value of the new candidate solutions in the subspace at current iteration.
- The best solution so far in this subspace.

Based on these design, SE can avoid fall into local optimum, while search for high expected value instead.

3 METHODS

3.1 Motivation

The motivation origins from these three questions:

1. *Can we combine multiple score functions to improve the correlation between score values and the final accuracy of architectures?*

The answer is yes. It's known that if the chosen score functions can complement with each others, the new correlation can eventually beyond the original ones. [26] shows that possibility. When an architecture gain a high NASWOT score, it may classify images, which are in the same class originally, into different classes. NI here comes to rescue. By measuring the noise immunity of an architecture, the miss-classifying can be solved. Thus, if mix NASWOT with NI, the correlation increases. An example is shown in Figure 3.

2. *How to combine multiple different kinds of score functions together and prevent one from overwhelming the others numerically?*

One possible way is normalization, but the necessary information is known only after evaluating the whole search space, e.g., range, mean or standard deviation, which is practically impossible. In order to achieve the goal, the proposed score function, named rank-based NAS, leverage multiple score functions by ranking, which provides a solution to have equal contribution from each score function.

3. *How to choose the score functions?*

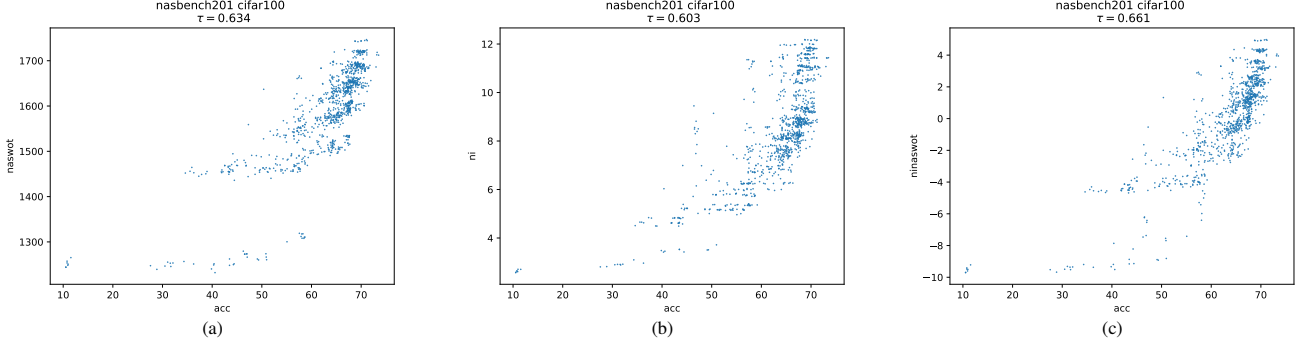


Figure 3: (a) NASWOT score for 1,000 randomly chosen architectures from NAS-Bench-201 in the CIFAR-10 dataset (b) NI score for 1,000 identical architectures from NAS-Bench-201 in the CIFAR-10 dataset. (c) NI score + NASWOT score for 1,000 identical architectures from NAS-Bench-201 in the CIFAR-10 dataset.

As mentioned, the key is to choose the complement of a score function. For example, NASWOT and NI. And for NASWOT and NI, which are both description of the ability of the graphic recognition of an architecture. The complement of them can be a measurement of trainability, e.g., SynFlow, NTK, which means whether this architecture is easy to train or not.

3.2 Rank-Based Score Function

Algorithm 1 The Rank Algorithm

Input: $\mathbb{N}^s \subset \mathbb{N}^*$

```

1 for each  $\mathbb{N}^i$  in  $\mathbb{N}^s$  do
2   for each  $j \in \{1, \dots, M\}$  do
3      $s_{\mathcal{F}_j}(\mathbb{N}^i) = \mathcal{F}_j(\mathbb{N}^i)$ 
4   end for
5 end for
6 for each  $j \in \{1, \dots, M\}$  do
7    $r_{\mathcal{F}_j}(\mathbb{N}^i) = \text{index of } \mathbb{N}^i \text{ in list } [s_{\mathcal{F}_j}(\mathbb{N}^1), \dots, s_{\mathcal{F}_j}(\mathbb{N}^{|\mathbb{N}^s|})]$ 
   sorted by descending order
8 end for
9  $r(\mathbb{N}^i) = \sum_{j=0}^M r_{\mathcal{F}_j}(\mathbb{N}^i)$ 
10  $i^* = \arg \min_i \{r(\mathbb{N}^i)\}$ 
11 Return  $\mathbb{N}^{i^*}$ 

```

The proposed rank-based score function is outlined in Algorithm 1. The first step is to evaluate the network architectures in the subset, denoted \mathbb{N}^s , of search space, denoted \mathbb{N}^* . The evaluation of an architecture determined by j -th score function is denoted $s_{\mathcal{F}_j}(\mathbb{N}^i)$. After evaluating the subset of search space, the rank of each score function is calculated, denoted $r_{\mathcal{F}_j}$. Then the final rank, r , calculated by summing up the ranks. If a network architecture gains higher score in these score functions, the final rank should be higher (smaller index) too. Therefore, the highest (minimum) rank in r shall be the best network architecture. In summary, the rank-based score function makes the contribution of each score function even.

3.3 Simulated Annealing with Rank-Based Score function

Figure 4 illustrates how SA with rank-based score function works in an iteration.

4 EXPERIMENT RESULT

natsbenchss

The result of natsbenchss is shown in Table 2.

nasbench201

The result of nasbench201 is shown in Table 3.

5 CONCLUSION

REFERENCES

- [1] B. Zoph and Q. V. Le, "Neural architecture search with reinforcement learning," 2016. [Online]. Available: <https://arxiv.org/abs/1611.01578>
- [2] Z. Sun, M. Lin, X. Sun, Z. Tan, H. Li, and R. Jin, "Mae-det: Revisiting maximum entropy principle in zero-shot nas for efficient object detection," 2021. [Online]. Available: <https://arxiv.org/abs/2111.13336>
- [3] J. Mellor, J. Turner, A. Storkey, and E. J. Crowley, "Neural architecture search without training," 2020. [Online]. Available: <https://arxiv.org/abs/2006.04647>
- [4] H. Zheng, K. An, and Z. Ou, "Efficient neural architecture search for end-to-end speech recognition via straight-through gradients," 2020. [Online]. Available: <https://arxiv.org/abs/2011.05649>
- [5] A. Mehrotra, A. G. C. P. Ramos, S. Bhattacharya, L. Dudziak, R. Vipperla, T. Chau, M. S. Abdelfattah, S. Ishtiaq, and N. D. Lane, "{NAS}-bench-{asr}: Reproducible neural architecture search for speech recognition," in *International Conference on Learning Representations*, 2021. [Online]. Available: <https://openreview.net/forum?id=CU0APx9LMaL>
- [6] Y. Jiang, C. Hu, T. Xiao, C. Zhang, and J. Zhu, "Improved differentiable architecture search for language modeling and named entity recognition," in *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*. Hong Kong, China: Association for Computational Linguistics, Nov. 2019, pp. 3585–3590. [Online]. Available: <https://aclanthology.org/D19-1367>
- [7] N. Klyuchnikov, I. Trofimov, E. Artemova, M. Salnikov, M. Fedorov, and E. Burnaev, "Nas-bench-nlp: Neural architecture search benchmark for natural language processing," 2020. [Online]. Available: <https://arxiv.org/abs/2006.07116>
- [8] H. Wang, Z. Wu, Z. Liu, H. Cai, L. Zhu, C. Gan, and S. Han, "Hat: Hardware-aware transformers for efficient natural language processing," 2020. [Online]. Available: <https://arxiv.org/abs/2005.14187>
- [9] N. Lee, T. Ajanthan, and P. H. S. Torr, "Snip: Single-shot network pruning based on connection sensitivity," 2019.
- [10] C. Wang, G. Zhang, and R. Grosse, "Picking winning tickets before training by preserving gradient flow," 2020.

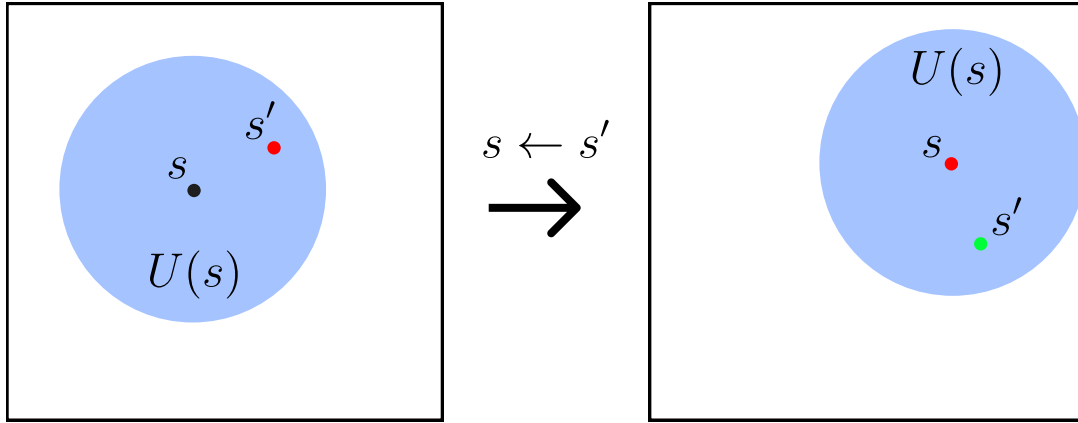


Figure 4: A simple example to illustrate an iteration of SA with Rank-based score function.

Table 2: COMPARISON OF RANK-BASED NAS AND ALL THE OTHER NAS ALGORITHMS.

Method	Search (s)	CIFAR-10		CIFAR-100		ImageNet-16-120	
		validation	test	validation	test	validation	test
Non-weight sharing							
AREA	12,000	84.62 ± 0.41	93.16 ± 0.16	59.24 ± 1.11	69.56 ± 0.96	37.58 ± 1.09	45.30 ± 0.91
REA	12,000	90.26 ± 0.22	93.17 ± 0.21	69.48 ± 0.76	69.49 ± 0.94	44.84 ± 0.72	45.47 ± 0.91
RS	12,000	90.08 ± 0.24	93.01 ± 0.29	69.14 ± 0.94	69.17 ± 1.00	44.66 ± 1.02	44.83 ± 1.05
RL	12,000	90.17 ± 0.23	93.08 ± 0.21	69.23 ± 0.87	69.29 ± 1.08	44.68 ± 0.91	45.05 ± 0.93
BOHB	12,000	90.05 ± 0.30	93.03 ± 0.20	69.04 ± 0.76	69.16 ± 0.90	44.71 ± 0.78	44.91 ± 1.05
Training-free							
NI (N=1,000)	X	89.56 ± 0.147	92.55 ± 0.187	X ± X	X ± X	X ± X	X ± X
NASWOT (N=1,000)	X	89.25 ± 0.412	92.21 ± 0.298	X ± X	X ± X	X ± X	X ± X
logsynflow (N=1,000)	X	89.61 ± 0.101	92.60 ± 0.188	X ± X	X ± X	X ± X	X ± X
rk (N=1000)	X	89.59 ± 0.136	92.51 ± 0.171	X ± X	X ± X	X ± X	X ± X
GA-rk	457.54	90.29 ± 0.149	93.27 ± 0.193	69.88 ± 0.497	70.06 ± 0.481	45.57 ± 0.425	46.19 ± 0.846
SA-rk	682.36	90.37 ± 0.204	93.29 ± 0.182	70.11 ± 0.457	70.32 ± 0.458	45.38 ± 0.499	46.45 ± 0.687

- [11] H. Tanaka, D. Kunin, D. L. K. Yamins, and S. Ganguli, "Pruning neural networks without any data by iteratively conserving synaptic flow," 2020.
- [12] M. S. Abdelfattah, A. Mehrotra, Łukasz Dudziak, and N. D. Lane, "Zero-cost proxies for lightweight nas," 2021.
- [13] N. Cavignero, L. Robbiano, B. Caputo, and G. Averta, "FreeREA: Training-free evolution-based architecture search," in *2023 IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*. IEEE, jan 2023. [Online]. Available: <https://doi.org/10.1109%2Fwacv56688.2023.00154>
- [14] W. Chen, X. Gong, and Z. Wang, "Neural architecture search on imagenet in four gpu hours: A theoretically inspired perspective," 2021. [Online]. Available: <https://arxiv.org/abs/2102.11535>
- [15] H. Wang, Y. Wang, R. Sun, and B. Li, "Global convergence of maml and theory-inspired neural architecture search for few-shot learning," 2022. [Online]. Available: <https://arxiv.org/abs/2203.09137>
- [16] Y. Shu, S. Cai, Z. Dai, B. C. Ooi, and B. K. H. Low, "Nasi: Label-and data-agnostic neural architecture search at initialization," 2021. [Online]. Available: <https://arxiv.org/abs/2109.00817>
- [17] X. Dong, L. Liu, K. Musial, and B. Gabrys, "NATS-bench: Benchmarking NAS algorithms for architecture topology and size," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pp. 1–1, 2021. [Online]. Available: <https://doi.org/10.1109%2Ftpami.2021.3054824>
- [18] M.-T. Wu, H.-I. Lin, and C.-W. Tsai, "A training-free neural architecture search algorithm based on search economics," *IEEE Transactions on Evolutionary Computation*, pp. 1–1, 2023.
- [19] M. Mitchell, *An introduction to genetic algorithms*. MIT press, 1998.
- [20] S. Kirkpatrick, C. D. Gelatt Jr, and M. P. Vecchi, "Optimization by simulated annealing," *science*, vol. 220, no. 4598, pp. 671–680, 1983.
- [21] J. Frankle and M. Carbin, "The lottery ticket hypothesis: Finding sparse, trainable neural networks," 2019.
- [22] V. Lopes, S. Alirezazadeh, and L. A. Alexandre, "EPE-NAS: Efficient performance estimation without training for neural architecture search," in *Lecture Notes in Computer Science*. Springer International Publishing, 2021, pp. 552–563. [Online]. Available: https://doi.org/10.1007%2F978-3-030-86383-8_44
- [23] X. Dong and Y. Yang, "Nas-bench-201: Extending the scope of reproducible neural architecture search," 2020.
- [24] C. Ying, A. Klein, E. Real, E. Christiansen, K. Murphy, and F. Hutter, "Nas-bench-101: Towards reproducible neural architecture search," 2019.
- [25] B. Zoph and Q. V. Le, "Neural architecture search with reinforcement learning," 2017.
- [26] M.-T. Wu, H.-I. Lin, and C.-W. Tsai, "A training-free genetic neural architecture search," in *Proceedings of the 2021 ACM International Conference on Intelligent Computing and Its Emerging Applications*, ser. ACM ICEA '21. New York, NY, USA: Association for Computing Machinery, 2022, p. 65–70. [Online]. Available: <https://doi.org/10.1145/3491396.3506510>
- [27] C.-W. Tsai, "Search economics: A solution space and computing resource aware search method," in *2015 IEEE International Conference on Systems, Man, and Cybernetics*, 2015, pp. 2555–2560.

Table 3: COMPARISON OF RANK-BASED NAS AND ALL THE OTHER NAS ALGORITHMS.

Method	Search (s)	CIFAR-10		CIFAR-100		ImageNet-16-120	
		validation	test	validation	test	validation	test
Non-weight sharing							
AREA	12,000	91.18 ± 0.43	93.95 ± 0.39	71.84 ± 1.21	71.92 ± 1.29	45.04 ± 1.03	45.40 ± 1.14
REA	12,000	91.08 ± 0.54	93.89 ± 0.50	71.69 ± 1.34	71.83 ± 1.33	44.96 ± 1.41	45.30 ± 1.51
RS	12,000	90.91 ± 0.33	93.67 ± 0.33	70.91 ± 1.04	70.99 ± 0.99	44.52 ± 0.99	44.56 ± 1.25
RL	12,000	90.87 ± 0.41	93.63 ± 0.36	70.62 ± 1.08	70.77 ± 1.05	44.20 ± 1.22	44.23 ± 1.37
BOHB	12,000	88.47 ± 1.19	91.79 ± 1.11	67.18 ± 2.05	67.50 ± 2.05	38.94 ± 3.58	39.00 ± 3.73
Weight sharing							
RSWS	4,154	76.95 ± 16.74	82.60 ± 12.10	52.51 ± 18.33	52.93 ± 18.32	29.76 ± 9.50	29.16 ± 9.61
DARTS-V1	5,475	39.77 ± 0.00	54.30 ± 0.00	15.03 ± 0.00	15.61 ± 0.00	16.43 ± 0.00	16.32 ± 0.00
DARTS-V2	16,114	39.77 ± 0.00	54.30 ± 0.00	15.03 ± 0.00	15.61 ± 0.00	16.43 ± 0.00	16.32 ± 0.00
GDAS	11,183	90.05 ± 0.23	93.46 ± 0.13	71.02 ± 0.31	70.56 ± 0.24	41.77 ± 1.24	41.96 ± 0.90
SETN	16,787	84.25 ± 5.05	88.01 ± 4.52	59.72 ± 7.30	59.91 ± 7.51	33.93 ± 3.85	33.48 ± 4.22
ENAS	7,061	40.11 ± 3.28	56.33 ± 3.70	14.09 ± 1.60	14.77 ± 1.45	16.20 ± 0.48	15.93 ± 0.67
Training-free							
NI (N=1,000)	X	X ± X	X ± X	X ± X	X ± X	X ± X	X ± X
NASWOT (N=1,000)	X	X ± X	X ± X	X ± X	X ± X	X ± X	X ± X
logsynflow (N=1,000)	X	X ± X	X ± X	X ± X	X ± X	X ± X	X ± X
rk (N=1000)	X	X ± X	X ± X	X ± X	X ± X	X ± X	X ± X
GA-rk	747.25	89.93 ± 0.196	93.41 ± 0.083	70.70 ± 0.417	70.76 ± 0.378	42.70 ± 1.315	43.10 ± 1.428
SA-rk	X	89.95 ± 0.194	93.37 ± 0.114	70.69 ± 0.391	70.75 ± 0.532	X ± X	43.10 ± 1.506