Free-Training Neural Architecture Search: zero-cost metrics embedded in zero-cost search strategies

Capobianco Francesco
Politecnico di Torino
s281307@studenti.polito.it

Compagnone Pierluigi
Politecnico di Torino
s288301@studenti.polito.it

De Cristofaro Carmine
Politecnico di Torino
s291129@studenti.polito.it

***Abstract*—Neural Architecture Search is the task of automatically design a neural network that yields good performance on a given dataset. This task has seen growing interest since to manually craft an architecture requires substantial effort of human experts, because of the immense variety of possible alternatives faced during the hand process. However the traditional NAS approaches are heavy in terms of time and resources required, due to the need to train and evaluate multiple networks. Thus we perform an empirical study to investigate the capacity of four different training-free metrics to score the performance a network will have on a dataset after training. Furthermore we analyze two search strategies which take advantage of one of these metrics to explore the architectures search space, with the aim to find the best possible network. Eventually we perform multiple experiments using all the possible (metric, search algorithm) combinations. In *Results* section we show that the NAS framework we implement is often able to find networks whose performance are near to the best architecture in the search space we explore, while requiring computation time orders of magnitude lower than traditional NAS.**
The code for the experiments is available at zero_cost_NAS.

I. Introduction

Neural Architecture Search is the field of machine learning that aims to automatically find a well-performing neural network architecture for a given dataset. The interest in this task develops from the need to reduce the time and resources required by the first NAS algorithms [1] [2], whose bottleneck is the training and the evaluation of a big amount of networks. A generic NAS pipeline is mainly described by three elements:

- ***Search Space***: the whole set of candidate architectures;
- ***Search Strategy***: the algorithm used to explore the search space;
- ***Performance evaluation technique***: the metric used to rank the architectures within the search space.

The goal of our work is to alleviate the NAS process adopting a training-free metric as performance estimator instead of the actual networks test accuracy. Furthermore, we propose two different search strategies that incorporate this metric. The analysis is performed over three datasets *Cifar10, Cifar100, ImageNet16-120*, in order to avoid that the good performance of a metric are strictly related to a specific dataset, without generalization capabilities.
As search space we consider the **NATS-Bench topological search space** [3] while as search strategy we implement a simple **Random Search** [4] and a variation of the **Aging Evolutionary Search** [2]. Regarding to performance evaluation technique we examine the following training-free metrics: **hook_logdet score** [4], **synflow** and **snip** [5]. Moreover we construct a new metric as convex combination of the two best performing ones (hook_logdet and synflow), that we will refer as **combined score**. All the experiments are performed also using the **validation accuracy after 12 epochs** as proxy metric, whose results are considered as upper benchmark for the other metrics.

II. Realted Works

Although in recent years several studies on NAS have been carried out by researches, we conduct our analysis starting from those that aim to optimize NAS efficiency. Therefore, we investigate the state-of-art of zero-cost metrics and search strategies.

A. NAS Efficiency

Several approaches have been proposed to speed up NAS frameworks. *DARTS* [6] works both on search space and search strategy, respectively relaxing the search space from a discrete set of architectures to a continuous domain and performing the research through gradient descent. *ENAS* [7] acts on search space, allowing candidate networks to share weights in order to reduce the training time in performance evaluation step. *EcoNAS* [8] focuses on performance evaluation technique, proposing reduced-training proxies to improve time consumption in evaluation phase.

B. Zero-cost metrics

There is another interesting cluster of papers that still operates on performance evaluation step, but differently by *EcoNAS* it focuses on the opportunity of identifying a good architecture completely avoiding training. *NASWOT* [4] and *Zero-Cost Proxies* [5] propose metrics able to score networks at initialization (without training) which are good as well as they preserve ranking between neural network models when compared with the ranking produced by post training test accuracy. To compute these metrics only a forward pass of a single batch of data is required.
Some of the metrics introduced in [5] and the one presented by [4] will be explained in more detail in the next session, because we adopt them in our work.

C. Search Strategies

One of the first papers that focuses on a search strategy specifically designed for the NAS task is *Regularized Evolution for Image Classifier Architecture Search* [2]. It consists in a variation of the traditional evolutionary algorithm, whose main difference is that an individual is removed from the population according to its "age" rather than its fitness. However this framework still requires the training and evaluation of all the architectures that occur within the population.
There are other search strategies which are more interesting for our work, because they integrate zero cost proxies. [4] investigates two simple methods: the first one consists in randomly sampling architectures from a set of candidates and scoring them using their training free metrics; the second is a variation of the aging evolution known as *Assisted REA*, which randomly-samples a larger population and uses their score to select the initial population.
In [5] they use the training free metrics to enhance NAS algorithms in two distinct manners: *the zero cost warmup* which is basically the same innovation made in *AREA* and the *zero-cost move* proposal that improves *REA* by choosing the best candidate child according to the proxy at each iteration.

D. NATS-Bench

NATS-Bench [3] has become increasingly relevant in the state of art for almost any up-to-date NAS algorithms, since it investigates both topology and size of the networks. Indeed the authors define two possible search space whose designs are cell-based : each cell is accounted as a densely connected DAG and we search for cell structure as building block of a bigger architecture.
The *Size Search Space*, Top of Figure 1, is thought to look for the number of channels in each layer, while the *Topological search space*, Bottom of Figure 1, is designed to find the operation assigned to each edge of the DAG.

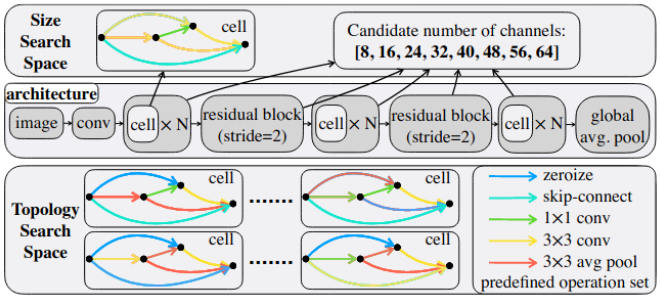

Fig. 1. NATS-Bench structure; **Middle**: the macro skeleton of each architecture candidate. **Top**: The size search space S_s in NATS-Bench. In S_s, each candidate architecture has different configuration for the channel size. **Bottom**: The topology search space S_t in NATS-Bench. In S_t, each candidate architecture has different cell topology.

III. Methods

Our work begins with the implementation of three training-free metrics: the proxy proposed in [4] that we call **hook_logdet score**, **snip** and **synflow** illustrated in [5]. We use these metrics to score the 15.625 architectures contained in *NATS-Bench tss* for each one of the three computer vision datasets, obtaining a scores roster for all possible combinations *metric-dataset*. Afterwards we deploy two search algorithms, **free-training Random Search** and **free-training Aging Evolutionary Search**, that exploit a zero-cost proxy, in order to find a high scored network according to the proxy itself.
The same search algorithms have been also tested using the validation accuracy after 12 epochs as indicator of the final test accuracy of the network, and the results achieved are consider an upper benchmark for the experiments performed with the other metrics. Eventually, trying to get as close as possible to the performance reached considering the 12 epochs validation accuracy we introduce **combined score**, a new metric without training defined as combination of the two most successful ones.
During every search we keep track of the time needed to complete it, because our whole analysis focuses on the trade-off between the found network accuracy and the employed time.

A. Metrics

- **hook_logdet score**: It is computed starting from the difference between the activated rectified linear units by each image of the batch. In details, an image i is associated to a binary code c_i composed by multiple indicator as many as the ReLUs in the network. The elements of c_i are set to one if the corresponding ReLU is activated, zero otherwise.
  The idea behind this procedure is that the more similar the two codes are, the more the network struggles distinguishing the two images. Contrariwise, the more different are the binary codes of the images in the batch, the more the network is able to distinguish among them.
  The dissimilarity between the codes is evaluated using the Hamming distance, therefore likeness between the inputs of the whole batch is expressed through the following matrix.

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

  N_A is the number of rectified linear units

  Eventually the *hook_logdet score* is predicted using the logarithm of the determinant of the matrix, that is higher for matrices closest to diagonal:

$$s = log|\mathbf{K}_H|;$$

- **snip & synflow**: They are two of the multiple metrics shown in [5]. They are both originally thought for pruning, indeed they are referred to a single parameter of the

network and try to estimate the change in loss when that specific parameter is removed.
The first one uses a single batch of data to calculate the loss, while the *synflow* generalizes this concept using the product of all parameters as loss function, therefore no data is needed to compute this metric.

$$\text{snip:} \quad S_p(\theta) = \left| \frac{\partial L}{\partial \theta} \odot \theta \right| \qquad \text{synflow:} \quad S_p(\theta) = \frac{\partial L}{\partial \theta} \odot \theta;$$

where L is the loss function, θ is a network's parameter, S_p is the per-parameter saliency and $\odot$ is the Hadamard product.
The saliency metrics are extended to the entire network by simply summing over its all parameters: $S_n = \sum_i^N S_p(\theta)_i$;

- **combined score**: It is the convex combination of *hook_logdet score* and *synflow*, in details:

$$combined = \alpha * hook\_logdet + (1 - \alpha) * synflow$$

We investigate different value of α in order to find the best trade-off among the two metrics for all the three datasets: $\alpha \in [0.2, 0.4, 0.5, 0.6, 0.8]$.
Then the new scores obtained are evaluated looking at their *Spearman rank correlation* with the networks test accuracy, ending up that $\alpha = 0.6$ is the best configuration in all the cases.

B. Search algorithms

- **Random Search**: It consists in randomly extract *N* architectures in sequence, every time a network is sampled we score it using a specific metric, then if its score is higher than the greatest value seen until that iteration the network becomes the best network and its score becomes the new maximum to beat. At the end we are interested in the best network (the one with the highest score) among the *N* candidates sampled and we check its actual test accuracy to verify how well the combination *(metric,search algorithm)* performed.

Algorithm 1 Random Search

```
generator = RandomGenerator()
best_net, best_score = None, 0
for i = 1 : N do
    net = generator.generate()
    score = net.score()
    if score > best_score then
        best_net, best_score = net, score
chosen_net = best_net
```

- **Aging Evolutionary Search**: Our implementation of this algorithm is inspired to the modified *REA* proposed in [5], with the difference that it is completely zero cost, because we never train and validate the networks. We introduce both the proposal of [5]:
  - **warmup**: It consists in taking advantage of the zero-cost metric to initialize the population. In particular we score a number of network *W*, greater than the population size *P*, then only the *P* individuals with the highest scores are inserted in the initial population.
  - **zero-cost move**: The mutations are not performed at random, but starting from the parent architecture all its candidate children are scored with a specific metric and the one with the highest score is selected. To avoid to get stuck in a local minimum, we extract a random number in $[0, 1]$ range and when it is greater than 0.75 we opt for a random mutation.

Algorithm 2 Aging Evolutionary Search

```
best_net, best_score, visited_models = None, 0, 0
population, warmup_pool = [ ], [ ]
generator = RandomGenerator()
if W > P then
    for _ = 1 : W do
        net = generator.generate()
        score = net.score()
        warmup_pool.append(score, net)
    warmup_pool.sort(key = score)
for i = 1 : P do
    if W > P then
        net_to_insert = warmup_pool[i][1]
    else
        net_to_insert = generator.generate()
    score = net_to_insert.score()
    population.append(score, net_to_insert)
    visited_models += 1
    if score > best_score then
        best_net, best_score = net_to_insert, score
while True do
    sub_population = [ ]
    for _ = 1 : T do
        candidate = population.random_tuple()
        sub_population.append(candidate)
    sub_population.sort(key = score)
    parent = sub_population[0][1]
    child = zero_cost_move(parent)
    child_score = child.score()
    population.append(child_score, child),  population.pop(0)
    if child_score > best_score then
        best_net, best_score = child, child_score
    visited_models += 1
    if visited_models > M then
        break
chosen_net = best_net
```

So we start with a network population *P*, that can be initialized randomly or with *warmup*, whose individuals are scored with the considered proxy. At each iteration a sub-sample *T* of *P* is randomly extracted and the network

with the highest score becomes the parent of the next generation. Finally we derive the child network according to the *zero-cost move* and it is inserted in the population, while the network that has been in the population for the longest time is removed. The algorithm ends when a fixed number of networks M are visited and the returned network is the one with the highest score among all the models seen during the process.

IV. Experiments

We make multiple experiments on each one of the three vision datasets: *Cifar10*, *Cifar100* and *ImageNet16-120*. First of all we compute the *Spearman correlation* between each metric and the test accuracy to make a deeper analysis of the quality of these proxies, without any bias due to the chosen algorithm.
Successively we perform all the searches incorporating the four metrics or the 12 epochs validation accuracy one at a time. In particular we run the random search and two different kind of evolutionary search, with or without warmup. Therefore the total number of configuration is given by *number of datasets* x *number of metrics* x *number of algorithms* = 45.
All the tests are repeated 30 times collecting mean and standard deviation of the found network test accuracy and the mean time to perform them. The size of the batch of data used to calculate the different zero-cost metrics has been fixed at 128. We also fix the hyperparameters of each search algorithm independently by the exploited metric and the considered dataset. In details: we consider N=500 models in the random search; in the evolutionary algorithms we select P=64, T=10, M=500 and in the warmup variant we consider a *warmup pool size*=128. We choose the maximum number of visited models in the evolutionary searches specifically equal to the number of analyzed models in the random search, in order to make a comparison that is as fair as possible between the algorithms.

Spearman correlation metrics-test accuracy				
Dataset	hook_logdet	synflow	snip	combined
Cifar10	0.788637	0.775205	0.637092	0.789481
Cifar100	0.806000	0.760552	0.632603	0.806564
ImageNet16-120	0.784085	0.751185	0.575196	0.784199

TABLE I
SPEARMAN RANK CORRELATION BETWEEN EACH METRIC AND THE NETWORKS TEST ACCURACY FOR ALL DATASETS.

Top 10% Spearman correlation metrics-test accuracy		
Dataset	hook_logdet	synflow
CIFAR-10	0.261112	0.304743
CIFAR-100	0.245638	0.267092

TABLE II
SPEARMAN RANK CORRELATION OF HOOK_LOGDET AND SYNFLOW FOR THE TOP 10% MODELS.

V. Results & Discussion

Spearman rank correlations reported in Table I show that *hook_logdet score* and *synflow* are quite fair estimators of the network test accuracy, while *snip* definitely underperforms on both the three computer vision dataset. In details, *hook_logdet* is the best proxy w.r.t. correlations, exceeding *synflow* of 0.1, 0.4 and 0.3 respectively on *Cifar10, Cifar100 and ImageNet16-120*.

The results of the searches for all possible configurations *(metric, search algorithm)* for *Cifar10, Cifar100 and ImageNet16-120* are individually reported in Tables III, IV, V. As we expected, the *validation accuracy after 12 epochs* is the best proxy to perform the search in all the cases, confirming that a training phase, even if in reduced settings always returns a more accurate estimation of the actual capability of the architecture than a free-training metric. In contrast it is clear that the limitation of this proxy regards the time to compute it. Indeed despite the small number of training epochs, repeating it for N=500 models leads to search times that are 4 orders of magnitude greater than using other metrics. The latters allow to complete the search taking less than one minute in *Random Search* case, and about tens of minutes in the *Aging Evolutionary* ones. Nevertheless with some of these zero-cost proxies we are still able to find networks whose test accuracy is less than 1% lower than those found by means of 12 epochs validation accuracy.
According to the correlations shown in Table I *snip* leads to the worst results for all datasets, independently by the search strategy adopted. Indeed looking the graphs in Figure 2 we can see that the test accuracy of the candidate networks selected according to this proxy during the process has a descending trend, highlighting a total deterioration when this metric is used in the evolutionary search for *ImageNet16-120*. With regards to *hook_logdet score* and *synflow* their performance in combination with the search algorithms are satisfactory, since the test accuracy trends in Figure 2 increase and get close enough to the upper benchmark of *validation accuracy*. The search experiments on *ImageNet16-120* confirm *hook_logdet* as the best zero-cost proxy for this dataset, reaching a mean test accuracy which is only 0.5% lower than the upper benchmark in the case of the evolutionary searches. The results obtained on *Cifar10* and *Cifar100*, however, oppose to the correlations in Table I because the searches with *synflow* outperform the ones using *hook_logdet*. To explain this contradiction we compute *Spearman correlation* for the top-10% of models across these two datasets considering only the two metrics involved. Assuming most of searches probably end up with a network scored among the top-10% of the whole architectures set, Table II justifies the results reported in Tables III, IV, V, since *synflow* correlation get higher than *hook_logdet* one for both *Cifar10* and *Cifar100*.

The following step of our analysis aims to give an overview of the search strategy proposed without taking into account the proxy used. *Random Search* is a very simple algorithm which returns a different best network at every run, how the variance in Figure 2 underlines. In opposition *Aging*

TABLE III
CIFAR10

Metrics	Random Search		Aging Evolutionary		AE with Warmup	
	test	mean time(s)	test	mean time(s)	test	mean time(s)
12val	93.92 ± 0.24	73788.41	94.31 ± 0.20	1612104.53	94.51 ± 0.20	1631786.75
hook_logdet	92.63 ± 1.23	45.54	92.68 ± 0.34	1316.89	92.92 ± 0.21	1314.18
synflow	93.34 ± 0.57	59.82	93.75 ± 0.18	1921.64	$93.74 \pm 4e-14$	1929.51
snip	91.41 ± 0.57	57.74	86.43 ± 2.97	1079.19	85.54 ± 3.74	1089.26
combined	93.30 ± 0.64	109.43	93.76 ± 0.16	3211.57	$93.74 \pm 4e-14$	3231.32

TABLE IV
CIFAR100

Metrics	Random Search		Aging Evolutionary		AE with Warmup	
	test	mean time(s)	test	mean time(s)	test	mean time(s)
12val	72.11 ± 0.83	113433.39	72.41 ± 0.38	2455668.47	72.44 ± 0.44	2454234.67
hook_logdet	69.47 ± 2.06	57.43	69.94 ± 0.79	1614.24	69.19 ± 0.78	1608.10
synflow	70.28 ± 1.86	46.19	71.06 ± 0.81	1185.94	$72.00 \pm 1e-14$	1191.12
snip	65.22 ± 2.49	56.40	49.37 ± 2.14	1080.72	$47.62 \pm 1e-14$	1086.55
combined	70.25 ± 1.97	107.81	70.99 ± 0.81	2864.30	$72.00 \pm 1e-14$	2875.36

TABLE V
IMAGENET16-120

Metrics	Random Search		Aging Evolutionary		AE with Warmup	
	test	mean time(s)	test	mean time(s)	test	mean time(s)
12val	45.71 ± 1.023	334931.77	46.06 ± 0.62	7325113.47	46.40 ± 0.67	7339394.07
hook_logdet	43.50 ± 3.14	50.84	45.51 ± 0.41	1362.46	$45.07 \pm 7e-15$	1366.32
synflow	42.59 ± 3.39	43.49	41.49 ± 0.65	1136.90	$42.37 \pm 1e-14$	1142.67
snip	32.65 ± 6.58	49.77	$0.83 \pm 3e-10$	912.65	$0.83 \pm 3e-16$	919.41
combined	43.01 ± 3.44	98.54	42.10 ± 1.66	2517.49	42.55 ± 0.67	2533.19

Evolutionary Search is a more complex and reliable algorithm. How shown in Figure 2 it gets stable after around 100 visited models because it is able to find the network with the highest metric value within the whole search space. The main factor that leads to this fast convergence is the *zero-cost move* that algorithm adopts in the defining child phase. Unfortunately *Zero-cost move* is a double-edged sword because increases the search times since to pick the best child it has to score all the candidates. Regarding the *warmup*, it has no effect on the found network since the standard evolutionary algorithm is already able to find the highest scored model. Its only improvement is a very slight reduction of number of visited models needed to select the best architecture.

Because of *Aging Evolutionary Search* always finds the best network according to the metric adopted, the bottleneck of our NAS framework are the metrics. For this reason we implement the *combined score*. This proxy has a convincing correlation w.r.t. the actual accuracy of the models, as shown in Table I (slightly higher than *hook_logdet*). Graphs in Figure 2 illustrates how *combined*'s behavior is strictly linked to *synflow*, in fact in the search experiments on *Cifar10* and *Cifar100* it obtains results better than *hook_logdet*. However, on *ImageNet16-120*, where *hook_logdet* outperforms *synflow*, our *combined* metric obtains networks with test accuracy with about 0.5% greater than the ones returned by *synflow* (Table V). Since the computation of *combined* requires scoring the models with both *hook_logdet* and *synflow*, research times are about double but sill remain in the order of minutes. Even if this metric is not capable of beating the one that performs best of the proxies that make it up, it always approaches the better of the two. So, the use of this is metric is strongly encouraged in a real NAS application where the test accuracy of the all explored networks are not available in advance.

VI. CONCLUSION

After having discussed the results, we would like to draw some conclusions. First of all, we show that running a partial training allows to have a reliable estimation of the actual capability of an architecture but as much as it can lighten computational times, it does not completely bypass the time limitations of a standard NAS.
Moving on to *zero-cost metrics*, snip proves to be the worst in model ratings and its use in a NAS framework could be totally avoided. On the other hand, both *hook_logdet score* and *synflow* well-behave as performance evaluation technique in this task. However, it is not possible to uniquely assert which one is the best, since the empirical study proves they are linked to the dataset. Hence, the metric we propose as a convex combination of these two can be considered as a valid

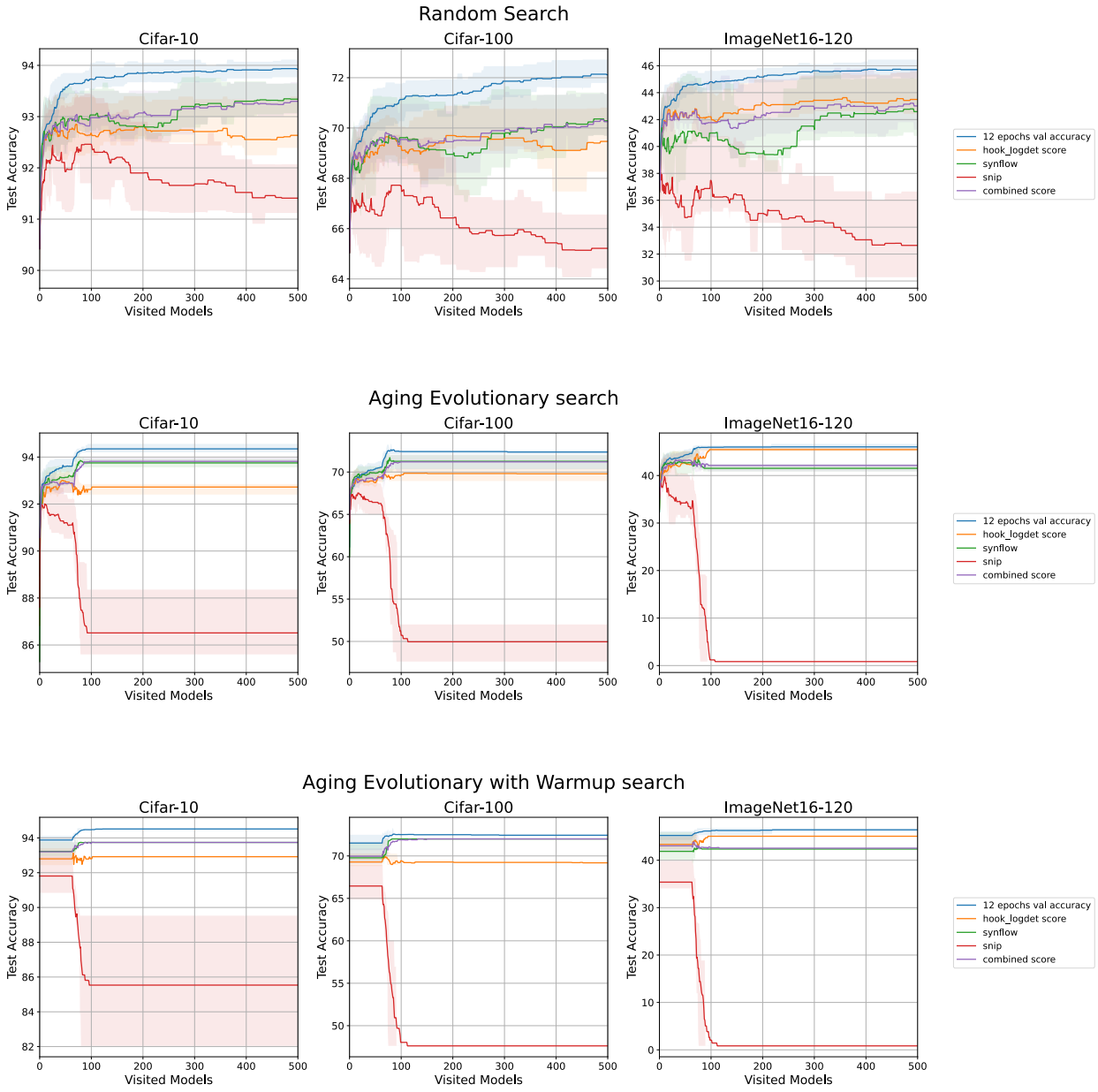

Fig. 2. Test accuracy progress during the searches for all possible combinations (metrics, search algorithms) on each dataset.

free-training proxy.

As search strategy, *Random Search* turns out to be a extremely fast algorithm but affected by a high variance in terms of test accuracy of found model. *Aging Evolutionary Search* ensures selecting the highest scored architecture within the *NATS-Bench tss* according to the chosen metric, however this result should be investigated in case of more variable search spaces. Moreover, even if it requires more time than *Random search* it is still affordable for the majority of its possible application, especially considering that it stabilizes long before the set threshold of 500 models to visit.

Summing up, our work can be placed as a reference point for future project of NAS, particularly for the ones oriented through a complete training-free pipeline.

REFERENCES

[1] B. Zoph, Q. V. Le, Neural architecture search with reinforcement learning, arXiv preprint arXiv:1611.01578 (2016).

[2] E. Real, A. Aggarwal, Y. Huang, Q. V. Le, Regularized evolution for image classifier architecture search, in: Proceedings of the aaai conference on artificial intelligence, Vol. 33, 2019, pp. 4780–4789.

[3] X. Dong, L. Liu, K. Musial, B. Gabrys, Nats-bench: Benchmarking nas algorithms for architecture topology and size, IEEE transactions on pattern analysis and machine intelligence (2021).

[4] J. Mellor, J. Turner, A. Storkey, E. J. Crowley, Neural architecture search without training, in: International Conference on Machine Learning, PMLR, 2021, pp. 7588–7598.

[5] M. S. Abdelfattah, A. Mehrotra, Ł. Dudziak, N. D. Lane, Zero-cost proxies for lightweight nas, arXiv preprint arXiv:2101.08134 (2021).
[6] H. Liu, K. Simonyan, Y. Yang, Darts: Differentiable architecture search, arXiv preprint arXiv:1806.09055 (2018).
[7] H. Pham, M. Guan, B. Zoph, Q. Le, J. Dean, Efficient neural architecture search via parameters sharing, in: International conference on machine learning, PMLR, 2018, pp. 4095–4104.
[8] D. Zhou, X. Zhou, W. Zhang, C. C. Loy, S. Yi, X. Zhang, W. Ouyang, Econas: Finding proxies for economical neural architecture search, in: Proceedings of the IEEE/CVF Conference on computer vision and pattern recognition, 2020, pp. 11396–11404.