Training Free Neural Architecture Search

Francesca Cossu
Politecnico di Torino
Torino, Italia
s305746@studenti.polito.it

Niccolò Dimonte
Politecnico di Torino
Torino, Italia
s303439@studenti.polito.it

Michele Presti
Politecnico di Torino
Torino, Italia
s290453@studenti.polito.it

***Abstract*—One of the most difficult aspects of Neural Networks is developing their structure. Hand-design requires a large number of human resources and for this reason NAS algorithms have been created to automate this process. These methods are extremely slow, requiring a full training for each observed model. To expedite this process, various quick-to-compute measures can be used instead of training to evaluate networks. In this paper, we evaluate two of these metrics (NASWOT and Synflow) and test several search algorithms in order to determine the optimum network configuration for image classification of three distinct datasets using the Nats-Bench in a matter of minutes.**

Our work: here

I. Introduction

Most neural architectures are hand designed by experts with a trial-and-error approach, but manually designing architectures is costly in terms of time and computing resources and overall does not guarantee the optimal solution. Neural Architecture Search (NAS), introduced by Zoph et al. automates this task: an architecture proposal is generated by a controller network, which is then trained to send a signal to the controller using reinforce, which generates another proposal and so on. Though training every single network is time-consuming, Mellor et al. proposed a solution that reduces the computing time, using NASWOT: this approach aims to use a metric that can evaluate a network before training.

A. Search space

In our experiments, we explored the NATS-bench search space (Dong et al.) that comprehends two search spaces: a topology search space and a size search space. The two spaces have the same structure: the skeleton is initiated with one 3x3 convolution with 16 output channels and a batch normalization layer. The main body of the skeleton includes three stacks of cells connected by a residual block. The shortcut path in this residual block consists of a 2x2 average pooling layer with a stride of 2 and a 1x1 convolution. The skeleton ends up with a global average pooling layer. The topology search space is the same as NAS-Bench-201.

B. Datasets

In this project we use CIFAR-10, CIFAR-100 (Krizhevsky et al., 2009) and ImageNet-16-120 (Chrabaszcz et al., 2017).

- CIFAR-10 It is a standard image classification dataset and consists of 60K 32×32 colour images equally divided in 10 classes.

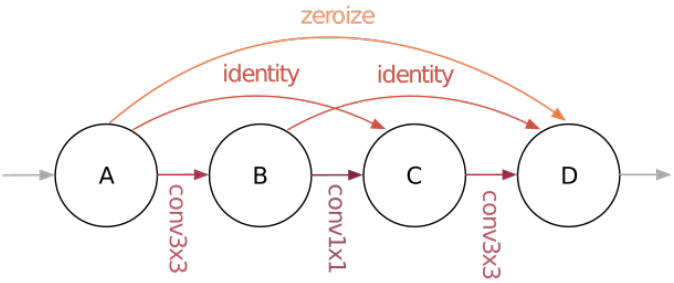

Fig. 1. A NAS-Bench-201 and NATS-Bench TSS cell

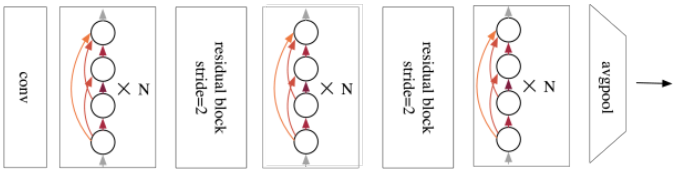

Fig. 2. The skeleton for NAS-Bench-101 (N=3), 201 (N=5), and NATS-Bench TSS (N=5).

- CIFAR-100 This dataset is just like CIFAR-10. It has the same images as CIFAR-10 but categorizes each image into 100 fine-grained classes.
- ImageNet-16-120 is built from the down-sampled variant of ImageNet (ImageNet16×16). ImageNet-16-120 contains 151.7K training images, 3K validation images, and 3K test images with 120 classes

II. NASWOT

The goal of NASWOT is to find a means to score network architecture at initialization so that it is indicative of its final trained accuracy. The metric used is based on linear maps present at initialization, because they can provide information about the flexibility of the architecture with respect to the data: those linear maps can be derived from the overlap of ReLU activations (Rectified Linear Unit activations) between two data points in a mini-batch. The activations can be summarized in a binary code: the more similar the binary codes associated with the two inputs are the less flexible is the architectures. The difference between the activation of two data points can be quantified using the Hamming distance, the minimum number of changes needed to turn one binary code into another.

TABLE I
RANDOM SEARCH WITH NASWOT METRIC

dataset	N	Validation accuracy		Score		Elapsed time	
		mean	std	mean	std	mean	std
CIFAR-10	10	67.387	10.056	1720.614	113.901	1.650	0.285
	100	68.717	12.804	1753.807	118.315	1.603	0.298
	1000	69.244	11.472	1746.939	109.274	1.603	0.551
CIFAR-100	10	39.960	14.548	1735.023	106.904	1.432	0.152
	100	41.785	8.854	1767.798	90.208	1.549	0.231
	1000	41.057	10.032	1752.213	102.623	1.533	0.465
ImageNet	10	23.303	5.460	1633.648	55.645	1.457	0.402
	100	22.682	5.719	1572.067	102.468	1.401	0.287
	1000	22.727	6.112	1564.490	114.384	1.390	0.321

TABLE II
RANDOM SEARCH WITH NASWOT METRIC

Dataset	Score		Accuracy		Elapsed time	
	mean	std	mean	std	mean	std
CIFAR-10	1689.081	353.722	67.638	15.585	1.898	1.728
CIFAR-100	1738.566	204.765	41.407	11.404	1.529	0.248
ImageNet	1559.831	122.048	22.482	7.439	1.388	0.313

Taking into account a mini batch of data $X = \{x_i\}_{I=1}^{N}$ and a network, we can define an indicator variable c_i that forms a binary code corresponding to a linear region.

The Hamming distance $d_H(c_i, c_j)$ between two data points is used to measure how dissimilar the two points are. The correspondence between two binary codes for the whole mini-batch is computed through 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 rectified linear units.

The plots of the normalized kernel are very specific: high performing network have fewer off diagonal elements with high similarity.

The networks are scored using $s = \log(|\mathbf{K}_H|)$; set two kernels with the same trace, s is higher for the kernels closest to the diagonal. A higher score at initialization implies improved final accuracy after training.

Table I shows the results we obtained both in score and accuracy. The results we obtained show that both the accuracy and the score decrease with the increasing of N. The accuracy results was calculated at 12 epochs. Figure 3 shows a clearly positive correlation between the NASWOT score and validation accuracy, in fact the correlations are respectively for CIFAR-10, CIFAR-100 and ImageNet, 0.7357, 0.6398 and 0.6260.

III. METHODS

A. Synflow

Zero-cost proxies were designed by Abdelfattah et al. to have a quick estimate of the performance of architectures with a forward pass of a single minibatch of data.

In our work, we used synflow, a data independent proxy, that was introduced as a pruning-at-initialization technique:

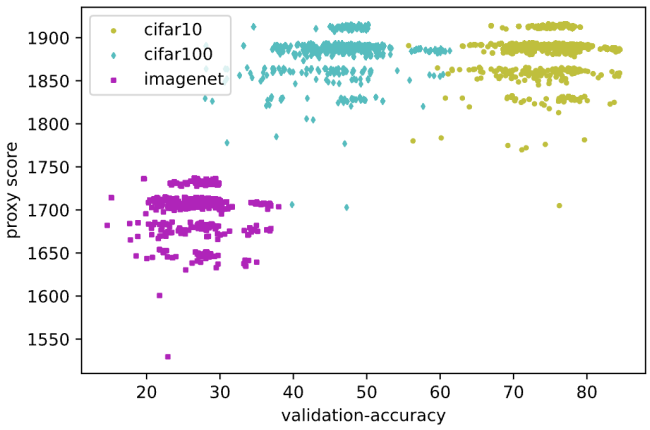

Fig. 3. Correlations

this metric generalize the synaptic saliency and avoid layer collapse when performing parameter pruning; with synflow loss is the product of all parameters in the network so that no data is needed to compute this loss or the synflow metric itself.

$$\mathcal{S}(\theta) = \frac{\partial \mathcal{L}}{\partial \theta} \odot \theta \quad (2)$$

where $\mathcal{L}$ is the loss function of a neural network with parameters θ and $\odot$ is the Hadamard product. We extend this metric to score an entire neural network by summing over all the parameters N in the model. The score present in our tables is normalized with the natural logarithm.

B. REA: Regularized Evolution Algorithm

In our work, we focused on the regularized evolutionary algorithm (introduced by Real et al.) due to the shared structure of the search space.

The algorithm works as an evolutionary algorithm, but, after the tournament selection, the oldest architecture is removed from the population instead of the least fitted. The algorithm keeps a population of P trained models through the experiment, and at each cycle, it samples S random models: the model with the highest validation fitness is selected as a parent of the child model that is obtained by applying a mutation on the parent. Then the child architecture is trained and added to the population, and the earliest model trained is removed. This method favors the newer models in the population and allows to explore search space without focusing on good models too early.

When using the searching algorithm REA there were some parameters fixed: the maximum population size is set to 1000, the pool size is set to 64 and the tournament size is set to 20.

IV. EXPERIMENTS

The two proposed metrics are NASWOT one and synflow. The main difference between them is that the Jacobian covariance metric is data-dependent, while synflow is a data-agnostic metric that uses the dataset only to set dimensions. While in

TABLE III
RANDOM SEARCH WITH SYNFLOW

Dataset	Score		Accuracy	
	mean	std	mean	std
CIFAR-10	48.894	18.761	70.313	9.641
CIFAR-100	51.865	21.209	41.648	9.717
ImageNet	51.249	17.827	23.423	5.402

TABLE IV
REA WITH NASWOT METRIC

Dataset	Score		Accuracy	
	mean	std	mean	std
CIFAR-10	1891.764	19.074	75.374	3.714
CIFAR-100	1890.665	18.458	47.411	4.550
ImageNet	1712.392	18.359	26.543	3.393

NASWOT the score is dependent on the number of ReLU of the network, synflow is directly correlated to the number of parameters of a network. The number of parameters is a more appropriate guideline for the accuracy of the network in the search space that we analyzed since the natural data-independent baseline works - the number of parameters in the network - that simply counts the weight in the network is an already competitive baseline.

Tables II and III summarize our results using the random search algorithm and the two metrics: the score of the synflow are slightly better than those of the NASWOT in terms of accuracy for all the datasets.

Tables IV and V summarize our results using the regularized evolutionary algorithm and the two metrics: it is visible that with REA are obtainable optimal scores both with synflow and NASWOT metric on all three datasets. In addition, it is an ideal solution in terms of time, ensuring valuable results regarding the final metrics while keeping the computing time always under 150 seconds.

A. REA instead of random search algorithm

REA is an evolutionary algorithm and it naturally tend towards the best architectures within the search space. Hence REA is a valuable option for neural architecture search since it gave great results on accuracy and achieving it with great time performance.

On the other hand random search, used in the NASWOT work, chooses architectures in a random way trough he search space and is for this reason less accurate than REA that chooses the best networks intentionally.

V. CONCLUSIONS

Summing up our observations on the obtained results, we can conclude that both REA and utilizing zero-costs proxies like synflow is an optimal baseline for future works in NAS research. In all of our experiments, there are some recurrent patterns in the results with the different datasets: the CIFAR-100 has the best scores (both with synflow and NASWOT metric), while CIFAR-10 obtains the 70 of accuracy most of the time; ImageNet16-120 reaches the networks with good accuracy only with REA, maybe because of the more significant number of classes compared to the other datasets. We can conclude that in a search space like NATS-Bench, a proxy like Synflow work relatively good, and there is a good correlation with the final accuracies, but it cannot be assumed that this also works with more complex neural architectures.

TABLE V
REA WITH SYNFLOW

Dataset	Score		Accuracy	
	mean	std	mean	std
CIFAR-10	96.175	12.069	75.305	7.425
CIFAR-100	98.294	12.298	46.650	9.027
ImageNet	95.642	11.569	24.464	6.405

VI. REFERENCES

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

Mellor, J. Turner, A. Storkey and E. J. Crowley, Neural architecture search without training, 2021.

X. Dong, L. Liu, K. Musial and B. Gabrys, Nats-bench: Benchmarking nas algorithms for architecture topology and size, 2021.

A. Krizhevsky, G. Hinton et al., Learning multiple layers of features from tiny images, 2009.

P. Chrabaszcz, I. Loshchilov and F. Hutter, A downsampled variant of imagenet as an alternative to the cifar datasets, 2017.

M. S. Abdelfattah, A. Mehrotra, Ł. Dudziak and N. D. Lane, Zero-cost proxies for lightweight NAS, 2021.

E. Real, A. Aggarwal, Y. Huang and Q. V. Le, Regularized evolution for image classifier architecture search, 2019.