Improving Neural Architecture Search by Mixing a FireFly algorithm with a Training Free Evaluation

Conference Paper · July 2022

DOI: 10.1109/IJCNN55064.2022.9892861

4 authors, including:

Nassim Mokhtari
École Nationale d'Ingénieurs de Brest
3 PUBLICATIONS **14** CITATIONS
SEE PROFILE

Marlene Gilles
Laboratory of Science and Technology of Information, Communication and Knowled…
15 PUBLICATIONS **34** CITATIONS
SEE PROFILE

Pierre De Loor
Laboratory of Science and Technology of Information, Communication and Knowledge
106 PUBLICATIONS **573** CITATIONS
SEE PROFILE

Improving Neural Architecture Search by Mixing a FireFly algorithm with a Training Free Evaluation

Nassim Mokhtari*, Alexis Nédélec*, Marlène Gilles* and Pierre De Loor*
*Lab-STICC (CNRS UMR 6285) - ENIB
Centre Européen de Réalité Virtuelle
Brest, France
Emails: {nassim.mokhtari; alexis.nedelec; marlene.gilles; pierre.deloor}@enib.fr

***Abstract*—Neural Architecture Search (NAS) algorithms are used to automate the design of deep neural networks. Finding the best architecture for a given dataset can be time consuming since these algorithms have to explore a large number of networks, and score them according to their performances to choose the most appropriate one. In this work, we propose a novel metric that uses the Intra-Cluster Distance (ICD) score to evaluate the ability of an untrained model to distinguish between data in order to approximate its quality. We also use an improved version of the FireFly algorithm, more robust to the local optimums problem than the baseline FireFly algorithm, as a search technique to find the best neural network model adapted to a specific dataset. Experimental results on the different NAS Benchmarks show that our metric is valid for either scoring CNNs and RNNs, and that our proposed FireFly algorithm can improve the result obtained by the state-of-art training-free methods.**

***Index Terms*—Deep Learning, Neural Architecture Search, Training-free Score, Intra Cluster Distance, Machine Learning, Metaheuristics, FireFly algorithm**

I. Introduction

Recent advances in the field of image classification and speech recognition related to deep learning research, particularly convolutional neural networks, have demonstrated their interest in feature extraction and classification and seem to be best suited to many classification problems [1]–[4].

According to the increasing number of hyperparameters (hidden layers, hidden units,etc.) used to define Deep Learning architectures and consequently, the increasing number of possible network organisations (exponential increase) implied by these hyperparameters, a new challenge is to find solutions to design the architecture itself with algorithms rather than manually [5], [6]. To do that, the deep learning community introduced Neural Architecture Search (NAS) algorithms, that are capable of automating the discovery of effective architectures [7]–[12].

In order to compare between NAS algorithms efficiency, the NAS community designed benchmarks specifically for this purpose [13]–[16]. Indeed, the choice of the architecture of a neural network can be seen as a combinatorial problem, where the goal is to find the combination of hyperparameters (number of layers, size of layers, etc.) that offers the best performances. The benchmarks provide a fixed search space of possible architectures that can be used to compare between NAS algorithms. Metaheuristics are often used to solve this kind of problems, and there are several works that exploit these methods [17]–[21]. The use of metaheuristics involves evaluating the solutions (neural network architectures) in order to score their quality.

One of the main problem is that training a model to evaluate its performance is time consuming, resulting in a huge computation time, which can take days even using hundred of GPUs [22]. Therefore methods exists to bypass this learning phase and evaluate an architecture from metrics based on the distribution of cells activation relative to different inputs values gather in a mini-batch, such as Mellor's metric [22] and Lopes et al. [23] proposal.

Our contribution in this work can be summarized as follow:

1) A new **training-free** metric that can approximate the quality of a neural network by evaluating its ability to distinguish between data. This metric can be used to score more kind of neural networks than previous metrics can do.
2) An Improved version of the FireFly Algorithm (IFA) using genetic algorithms operators, allowing our IFA to be more robust to local optimums than the standard version.
3) A Combination of the proposed metric and the improved FireFly algorithm in order to find the most interesting model in a neural architecture search space.
4) An evaluation of our proposition that show that it outperform the state of the art in term of performances to find the better architecture for different training tasks.

The rest of the document is organised as follows: Section 2 introduces a synthesis of the various works carried out in the neural network evaluation in NAS. Section 3 will present the FireFly algorithm. In Section 4 we present our proposed method to evaluate an untrained neural network, and a proposal to improve the FireFly algorithm. Section 5 will present the experimental results obtained on 4 NAS benchmarks, showing

French National Agency for Research (ANR-20-CE26-0014-01)

978-1-7281-8671-9/22/$31.00 ©2022 IEEE

the efficiency of our proposal (valid and improves the state-of-art scores). Finally, we will summarize in Section 6 the results of this work as well as the possible improvements.

II. Related works

The search space (the set of all possible networks) being very large, evaluating the effectiveness of a NAS algorithm can not be done in an exhaustive way. This has led to the creation of several benchmarks [13]–[16], that consist of tractable NAS search spaces, and metadata for the training of networks within that search space [22].

NAS-Bench-101 is composed of 423,624 neural networks (CNN) that have been trained exhaustively, with three different initialisations, on the CIFAR-10 dataset for 4, 12, 36, and 108 epochs (423K * 3 * 4 = 5M total trained models) [13]. NAS-Bench-201 includes 15,625 networks trained multiple times on CIFAR-10, CIFAR-100, and ImageNet-16-120 [14]. The NAS-BENCH-NLP is made of 14K neural networks (RNN) trained on the Penn Tree Bank dataset and the WikiText-2 dataset [16].

NAS algorithms explore the search space in order to find the best one. However, training each neural network architecture to select the most appropriate is a time consuming process. Therefore, being able to evaluate an architecture quality without training it is an alternative to find the most suitable neural network without spending days on calculation.

Mellor et al. [22] proposed a way to evaluate a neural network without previous training by identifying a binary indicator, focusing only on the rectified linear units of the network (0 for inactive unit, and 1 for active unit). The intuition behind their approach is that the more similar the binary codes associated with two inputs are, the more challenging it is for the network to learn how to discriminate these inputs. A mini-batch of data X mapped through a neural network (composed of ReLU and several other units type) as $f(x_i)$ to get binary codes C, where each c_i refers to the binary code of x_i (output of the $i-th$ ReLU units). Mellor et al. construct a matrix K_h, using the Hamming distance, where each component of the matrix is calculated using the Eq. (1). Then, they use the Eq. (2) to evaluate the neural network: the higher the score, the better the network.

$$K_h[i,j] = N_A - Hamming\_distance(c_i, c_j) \quad (1)$$

where N_A is the number of rectified linear units.

$$s = log|K_h| \quad (2)$$

Lopes et al. [23] proposed another way to evaluate a neural network without training. Based on Mellor's et al. work [22], they proposed to use the binary codes to compute a Jacobian matrix (J). The objective is to determine if an untrained network could distinguish local linear operators for each data point, but also have similar results for similar data points (belong to the same class in a supervised approach). To estimate this behavior, they evaluate the correlation of J values with respect to their class by computing a co-variance matrix for each class present in J. The correlations are individually evaluated first, as they may have different sizes due to the number of data points per class. The final score is the sum of the individual correlation matrices' scores.

Exploring all the NAS benchmark in order to find the best model can be a difficult and time consuming task, even if we use this kind of metrics to avoid the training of the network, due to the millions of models included in this search space. To avoid that, there are several works that exploit metaheuristics to find the more suitable network for a given task to avoid performing an exhaustive search. These methods use a representation, like a genome in the case of the Genetic Algorithm (GA), of the solutions (neural networks in this case) that will be transformed according to the metaheuristic's operator (like the crossover operation in GA). For example, Sun et al. [19] used a Genetic Algorithm to automatically design a convolutional neural network, by using the trained network accuracy as a fitness function.

Rere et al. [21] proposed several metaheuristics (SA, DE and HS algorithms). Their fitness score was the standard error on the training set. Carvalho et al. [20] proposed a fitness function based on both train and test error, that where used with VNS, SA, GEO and GA algorithms. Ayumi et al. [18] used a Microcanonical Annealing Algorithm (MAA) to design a neural network in their works, using the training loss as a fitness function. Strumberger et al. [17] preferred to use a FireFly algorithm to design their CNN, where the used fitness function was based on the error on the test set. Mellor et al. [22] proposed to use an Assisted Regularised Evolution Algorithm (AREA), an improved version of REA proposed by Rere et al. [21], combined with their proposed metric to find the best neural network architecture.

In this work, we are interested in the use of an improved version of the FireFly algorithm, combined with a proposed model evaluation metric as a fitness function (like done by Mellor et al. [22] and Rere et al. [21]). Note that this metric allows to address more general networks than previous methods, with any kind of cells, in order to design a neural network suitable for a given task without any train.

Our choice of using the FireFly algorithm comes from the fact that this method includes several metaheuristics such as simulated annealing (SA), or particle swarm optimization (PSO), in addition to its fast convergence towards an optimum [24].

III. Overview of the FireFly Algorithm (FA)

Based on the behavior of fireflies, this algorithm initially was developed by Xin-She Yang in 2018 [24]. The FireFly Algorithm (FA) is based on three rules which are:

1) Fireflies are unisex, so one firefly is attracted to another without considering its gender.
2) Attractiveness is proportional to brightness, and both are inversely proportional to distance (decrease as distance increases). For any pair of fireflies, the less bright will be attracted by the brighter one, and therefore will move

towards it; if there is no firefly brighter the latter will move randomly.

3) The brightness of a firefly (solution) is given by the objective function.

The movement of a firefly i towards a brighter firefly j is defined in Eq. (3)

$$x_i^{t+1} = x_i^t + \beta_0 e^{-\gamma r_{ij}^2}(x_j^t - x_i^t) + \alpha \tau_i^t \tag{3}$$

- β_0: attractively at r = 0.
- γ: absorption coefficient, (input parameter).
- r: Distance between two fireflies
- α: Weight of random move
- x_i^t: Position of firefly i at time t
- τ_i^t: Vector of random numbers given by a Gaussian distribution at time t for the firefly i

The FireFly algorithm is illustrated in algorithm 1.

Algorithm 1 FireFly Algorithm

Randomly generate a population of n fireflies x_i=(i=1,2,...,n).
The light intensity I_i of each firefly x_i is given by f(x_i) where f is the objective function
Define the coefficients α, β and γ.
while not Stopping criteria **do**
  for i=1 to n **do**
    for j=1 to n **do**
      if I_i¡ I_j **then**
        Move firefly i to firefly j
      end if
      Vary the attractiveness according to the distance r.
      Evaluate the new solution and update the light intensity
    end for
  end for
  Determining the best solution
end while

IV. Proposed Method

In this part, we present our approach based on:

- Network quality evaluation with a training-free metric.
- The use of an improved FireFly algorithms to guide the choice among the huge amount of possible architectures.

A. Network quality evaluation

We propose a new way to score a neural network without training it by mapping a mini-batch of data through the network to be evaluated in order to get the binary code (0 for negative value and 1 for positive one) of each input data. We follow Mellor's et al. [22] methodology except that we use all the units of the network, not only the ReLU ones.

Our intuition is that units using an activation function other than ReLU (or not even using one) can also be used to formalize the way the model interprets the data (binary codes). Since we seek to evaluate the ability of a model to distinguish between data, we can avoid checking similarities (by calculating the co-variance as done by Rere el al. [21] for example) and focus more on the degree of difference between representations. To do that, we propose to use the Intra-Cluster Distance (ICD), calculated on the binary codes, as a metric to score the untrained network. Fig. 1 illustrates our network scoring process, for a mini-batch of data containing 4 samples.

ICD is usually used to evaluate the quality of a clustering method where the goal is to find a clustering providing groups that have a small Intra-Cluster Distance value. The Idea is to calculate the mean distance between each data point and the center of the cluster (mean of data).

In our work, we score the network according to the ICD calculated using Eq. (4).

$$ICD = \frac{\sum_{n=1}^{N} d(\bar{c}, c_i)}{N} \tag{4}$$

Where d is the euclidean distance, c_i is the binary code of the input x_i, $\bar{c}$ is the center of the binary codes (mean of binary codes) and N is the total number of binary codes (number of samples in the mini-batch of data).

A small ICD value means that the cluster is compact (composed of similar samples), a large one means that the cluster is stretched (the samples are different). Since we are looking for a network that can provide different binary codes for different samples, the cluster composed of these binary codes should be stretched, giving a high ICD value. The higher the score, the better the network.

B. Improved FireFly Algorithm

Thanks to the attraction mechanism, the FireFly algorithm quickly reaches an optimum, increasing the chances of being stuck in a local optimum, even if the FireFly algorithm includes a diversification (the random walk). In order to improve the diversification, we propose to use the operators of the Genetic Algorithm (selection, crossover and mutation), which could allow a better exploration of the search space, by combining the components of the solutions already explored. More details about how these metaheuristic's operators are implemented can be found in the Section 4.C

We propose to run an iteration of the Genetic Algorithm each time the FireFly algorithm is stuck in a local optimum, to create a new population, completely different from the old one. We consider the FireFly algorithm stuck in a local optimum, if it can not improve the best solution after a given number of iterations (chances). To do that, before each Genetic Algorithm execution, the current optimum is stored in a list (*candidates*). The final result of the exploration (when stopping criteria is reached) is the best solution from the candidates list. The algorithm 2 illustrates the proposed approach.

Using this mechanism give the chance to our proposed algorithm to perform a "minor" diversification (FireFly's random walk), which would allow it to better explore a region of the search space, before exploring another one (obtained by the Genetic Algorithm).

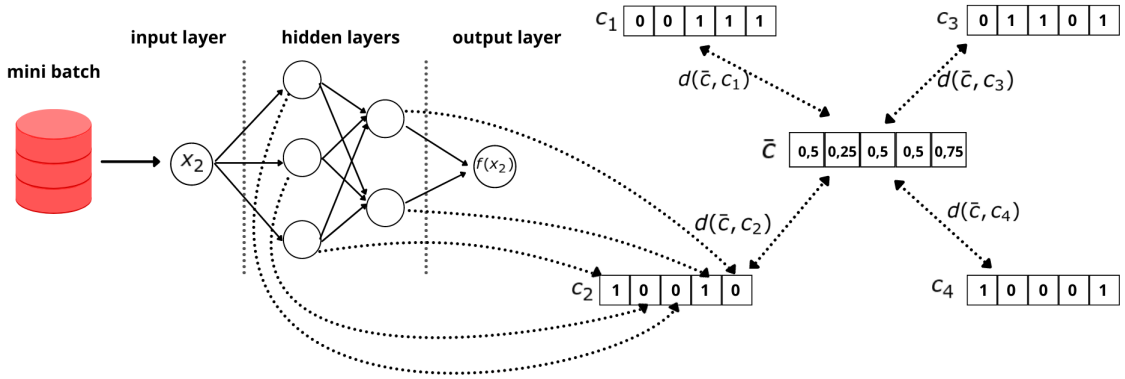


Fig. 1: Our proposed network scoring process: each sample x_i from the mini-batch of data is used to generate a binary code c_i, using the outputs of the hidden units given by x_i as input of the neural network. $\bar{c}$ is the mean of all binary codes, and is used to calculate the ICD value.

Algorithm 2 Improved FireFly Algorithm

```
Randomly generate the population
Define MaxChances
chances = MaxChances
candidates = [ ]
LocalBest = NULL
while not Stopping criteria do
  Running an iteration of FireFly
  Best_t = current population's best solution
  if LocalBest = NULL then
    LocalBest = Best_t
  else
    if fitness(Best_t) ≥ fitness(Best_{t-1}) then
      LocalBest = Best_t
    else
      chances–
    end if
  end if
  if chances=0 then
    add LocalBest to candidates
    LocalBest = NULL
    Perform an iteration of the Genetic Algorithm
    chances = MaxChances
  end if
end while
Determining the best solution from the candidates list
```

Note that the selection of the best solution from the *candidates* list is done according to the models performances after training. Since our metric can only approximate model's quality and cannot measure it exactly, keeping a list of candidates increases the chances of finding a good architecture.

C. Implementation

Each solution (firefly) represents a network architecture contained in the NAS benchmark, since each NAS benchmark has his own representation of an architecture, we have to create a dedicated implementation for each one. In the following, we will describe our choice for representing a solution, and the implementation of the metaheuristic's operators.

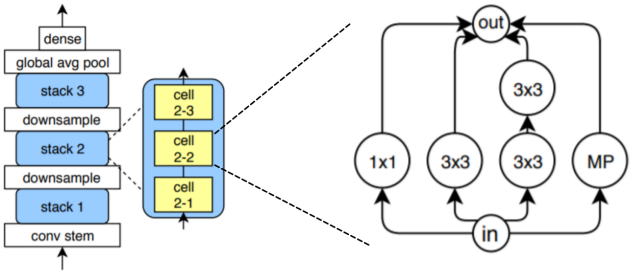


Fig. 2: Network architecture in the NAS-BENCH-101 [13]: Left part is the skeleton shared by all models, middle part is a stack of cells and the right part is an example of a cell (module)

1) NAS-BENCH-101:

In the NAS-BENCH-101, all the networks share the same skeleton, which is composed of 3 stacks, each one includes 3 cells, as shown in the left part of the Fig. 2. The networks are different in the "module" (cell), which is represented by directed acyclic graphs (up to 7 vertices and 9 edges). The valid operations at each vertex are "3x3 convolution", "1x1 convolution", and "3x3 max-pooling" [13]. The right part of the Fig. 2 shows an example of a module in the NAS-BENCH-101.

In the NAS-BENCH-101, a possible architecture is represented by the adjacency matrix of the module, and a list containing the operations at each vertex. The example illustrated in Fig. 2 can be represented using the adjacency matrix shown in Fig. 3 and following operation list : [INPUT, CONV1X1, CONV3X3, CONV3X3, CONV3X3, MAXPOOL3X3, OUT-

$$\begin{bmatrix} 0 & 1 & 1 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

Fig. 3: adjacency matrix for NAS-BENCH-101

PUT].

In order to represent our solution, we choose to keep the adjacency matrix, and encode the operation from string to float numbers (between 0 and 1), in order to make possible the firefly's move. The encoding is done as follow :

- MAXPOOL3X3 = 0
- CONV1X1 = 0.33
- CONV3X3 = 0.66

The metaheuristic's operators are implemented as follow :

- Crossover: after the selection of two solutions (parents), a new solution is generated by:
  - Splitting parents' matrices horizontally in two parts, then joining the top part of the first parent with the bottom part of the second one, in order to create the new adjacency matrix.
  - Splitting parents' encoded operations lists (at vertices) vertically in two parts, then joining the left part of the first parent with the right one of the second parent, to create the new encoded operation list.
- Mutation: randomly select a vertex, and then generate a new float for its operation (random number between 0 and 1).
- Firefly's move: the move is done by calculating Eq.(3) at element wise (for each element of the matrix and each element of the encoded operation list). After each move, the values less than 0 are set to 0 and the ones that are greater than 1 are set to 1.

In order to look for a new architecture (given by the metaheuristic's operators) in the NAS-BENCH-101, the value of vertices (x) operators are decoded as follow:

$$operator(x) = \begin{cases} MAXPOOL3X3, & \text{if } x < 0.33 \\ CONV1X1, & \text{if } 0.33 \leq x < 0.66 \\ CONV3X3, & \text{otherwise} \end{cases}$$

2) NAS-BENCH-201:

The architectures of the NAS-BENCH-201 share the same skeleton, which is initiated with a 3-by-3 convolution and a batch normalization layer. The body includes three stacks of cells, connected by a residual block. The skeleton ends up with a global average pooling layer followed by a classification layer using *softmax* [14]. The NAS-BENCH-201 supports 5 operations that are encoded as follow:

- none = 0

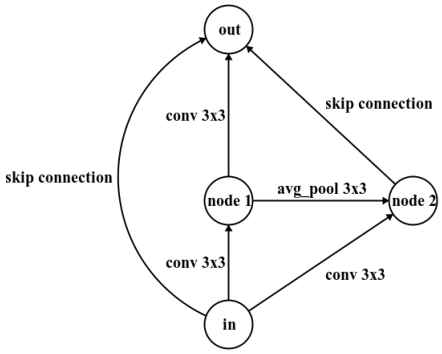


Fig. 4: Example of a module (cell) in NAS-BENCH-201

$$\begin{bmatrix} 0 & 0 & 0 & 0 \\ 3 & 0 & 0 & 0 \\ 3 & 4 & 0 & 0 \\ 1 & 3 & 1 & 0 \end{bmatrix}$$

Fig. 5: adjacency matrix for NAS-BENCH-201

- skip connection = 1
- 1-by-1 convolution = 2
- 3-by-3 convolution = 3
- 3-by-3 average pooling = 4

In the same way as NAS-BENCH-101, the NAS-BENCH-201 cells can be represented as a directed acyclic graph [14], so, they can be represented using an adjacency matrix (M), where $M[i, j]$ defines the operation existing between the nodes i and j. The Fig. 5 illustrates the adjacency matrix used to represent the cell illustrated in Fig. 4.

The metaheuristic's operators are implemented as follow :

- Crossover: done in the same way as the NAS-BENCH-101 adjacency matrices crossover.
- Mutation: randomly select an element from the matrix, and then generate a new integer for its value (random number between 0 and 4).
- Firefly's move: done in the same way as the NAS-BENCH-101 adjacency matrices move. After each move, the values are first rounded to the nearest integer, then, the ones less than 0 are set to 0 and the ones greater than 4 are set to 4.

V. Experimentation

In this section of the paper, we will present the experimental results obtained on different benchmarks/dataset. All the tests were conducted on a Laptop, having an Intel i7-11850H CPU, 32GB of RAM, with a NVIDIA RTX A3000 GPU. Note that in the following **untrained** model evaluation, only 100 samples were used to score the neural networks.

A. Metric validation

In order to validate our proposition (based on ICD) for the evaluation of a neural network without training, we follow the same testing protocol as Mellor et al. in [22].

Benchmark	Dataset	Spearman's τ	p-value
NAS-BENCH-101	CIFAR-10	0.499497	2.212611e-129
NAS-BENCH-201	CIFAR-10	0.798250	0.0
NAS-BENCH-201	CIFAR-100	0.796226	0.0
NDS (DARTS)	CIFAR-10	0.624632	7.627274e-217
NDS (Amoeba)	CIFAR-10	0.256211	2.408139e-31
NDS (ENAS)	CIFAR-10	0.501247	1.031932e-127
NDS (NASNET)	CIFAR-10	0.387336	1.369921e-72
NDS (PNAS)	CIFAR-10	0.490744	1.044203e-121
NAS-BENCH-NLP	Penn TreeBank	-0.420435	4.588134e-73

TABLE I: Spearman's τ and p-value of ICD score on different benchmarks/dataset

We used 2000 architectures from each of NAS-BENCH-101, NAS-BENCH-201, NAS-BENCH-NLP and NDS. All of these benchmarks provide the accuracy and loss for every architecture after its train, this allows us to evaluate and validate our metric on a huge number of data in a short time.

We calculated the Spearman's τ to check if a correlation exists between the network's score (without learning) and its final performance (after training), which is the accuracy in the case of NAS-BENCH-101, NAS-BENCH-201 and NDS, and the loss in the case of NAS-BENCH-NLP. The results obtained on the different benchmark/datasets are illustrated in table I.

According to Spearman's τ presented in table I, we can say that there is a correlation in all the benchmarks/datasets tested, achieving 0.798 and 0.796 in NAS-BENCH-201 for CIFAR-10 and CIFAR-100 meaning that there is a strong correlation between our score before training and the final accuracy after network's training. The p-value is equals to 0 which means that the probability that τ value is due to random is equals to 0. The smallest τ value was 0.256 obtained on the NDS benchmark (Amoeba) using CIFAR-10 dataset, meaning that the correlation is poor.

On the NAS-BENCH-NLP, we can notice that the value of Spearman's τ is negative (-0.42): there is a correlation between the score before training and the network's final result. In this case, the correlation is negative because the network is scored using the loss, so the least the loss is, the better the network is.

Fig. 6 illustrates the correlations between ICD scores (before training) and the final test accuracy (obtained after training) using a scatter plot. We can notice that in most cases, the resulted plot has the shape of a line with a positive slope, showing that the higher the score before training is, the higher the final test accuracy is. This shows that our proposed metric is valid and can be used to score a neural network without training, whatever the network is a CNN or a RNN.

B. Comparison with Mellor et al. [22]

In this part, we compare the results obtained using our metric with Mellor's et al proposition [22]. The comparison is done on the Kendall's τ scores calculated on the different NAS benchmarks. Table II shows the obtained results.

We can notice that the Kendall's τ obtained by Mellor's et al. proposition and the ones obtained by our ICD score are almost similar, with some better results for Mellor's proposition in some cases, and better results for our proposition in other cases, but no significant difference were observed.

Benchmark	Dataset	Our score	Mellor et al. score
NAS-BENCH-101	CIFAR-10	0.353	0.285
NAS-BENCH-201	CIFAR-10	0.595	0.574
NAS-BENCH-201	CIFAR-100	0.594	0.611
NDS (DARTS)	CIFAR-10	0.457	0.467
NDS (Amoeba)	CIFAR-10	0.185	0.223
NDS (ENAS)	CIFAR-10	0.362	0.365
NDS (NASNET)	CIFAR-10	0.271	0.304
NDS (PNAS)	CIFAR-10	0.352	0.382

TABLE II: Kendall's τ for ICD score and Mellor's proposition on different benchmarks/dataset

Parameter	**Value**
# of run	10
Stopping Criteria	100 generations
Population size	20
Max chances (for IFA only)	5
β_0	0.95
γ	0.15
α	0.5

TABLE III: Experimental parameters

From the different results, we can say that Mellor's et al. proposed metric and our ICD proposition are equivalent to evaluate networks on a classification task. However, using all the network's units makes our method more generic, since it supports RNNs models (valid on the NAS-BENCH-NLP benchmark).

C. Comparison with other training-free methods

In this part, we compare our proposition: proposed metric combined with the Improved FireFly Algorithm (IFA), to the state of art training-free methods, and also the baseline FireFly algorithm.

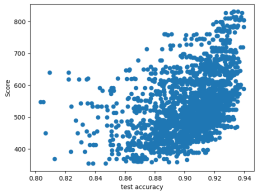
All the experiments where done using the parameters presented in table III.

1) NAS-BENCH-101:

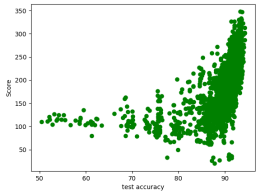
We compared our obtained results (for 10 runs) of the FA, GA and IFA on the NAS-BENCH-101 benchmark, using the CIFAR-10 dataset, with the NASWOT and AREA proposed by Mellor et al. [22]. NASWOT consists of choosing the best network from the N possibilities generated randomly while REA is the method proposed by Rere et al. [21]. The mean search time, mean test accuracy and the std are summarized in table IV.

Method	Search (s)	Test Accuracy
NASWOT (N=100)	23	91.77 ± 0.05
REA	12000	93.87 ± 0.22
AREA	12000	93.91 ± 0.29
FA (our)	3260	93.00 ± 1.44
IFA (our)	3596	$\mathbf{94.03 \pm 0.12}$

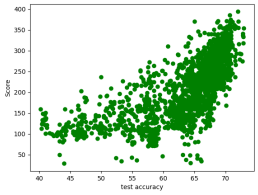
TABLE IV: Comparison on the NAS-BENCH-101 (CIFAR-10) between our proposition and state of the art methods. Performance shown in accuracy with mean±std



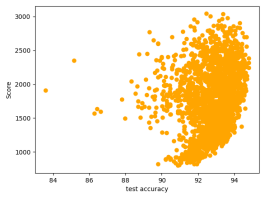
(a) NAS-BENCH-101 (CIFAR-10)



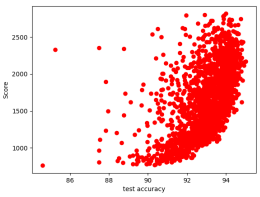
(b) NAS-BENCH-201 (CIFAR-10)



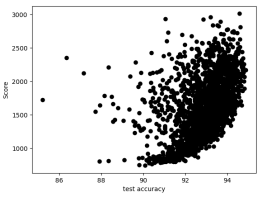
(c) NAS-BENCH-201 (CIFAR-100)



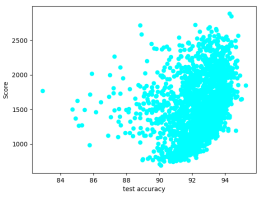
(d) NDS-Amoeba (CIFAR-10)



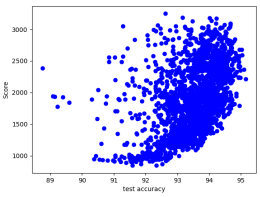
(e) NDS-DARTS (CIFAR-10)



(f) NDS-ENAS (CIFAR-10)



(g) NDS-NASNET (CIFAR-10)



(h) NDS-PNAS (CIFAR-10)

Fig. 6: Plots of our ICD score for **untrained** architectures in NAS-Bench-201, NAS-Bench-101 and NDS against test accuracy when trained. The inputs when computing the score for each plot are from CIFAR-10 except for (c) which use CIFAR-100

First, we can notice that IFA performs better than the baseline FireFly algorithm, giving a better test accuracy with a smallest standard deviation (more stable). Then, we can also notice that our proposal outperforms the other methods, giving 0.12% more accuracy than the AREA method and more stability (0.12 for IFA versus 0.22 for AREA). All this while being more than 3 times faster.

2) NAS-BENCH-201:

We compared our obtained results (for 10 runs) of the FA, GA and IFA on the NAS-BENCH-201 benchmark, using the CIFAR-10, CIFAR-100, ImageNet16-120 datasets, with the NASWOT and EPE-NAS methods. EPE-NAS was introduced by Lopes et al. [23] to be combined to their proposed metric, their search algorithm is similar to NASWOT (based on random selections). As Lopes et al. [23], we run our proposition on the CIFAR-10 dataset, the obtained architectures are then trained on the CIFAR-100 and ImageNet16-120. The mean test accuracy and the std are summarized in table V.

According to the obtained results, we can notice that our proposed IFA outperforms the state-of-art training-free methods, on the three datasets of the NAS-BENCH-201, by achieving better mean accuracies in all cases.

We can also notice that the baseline FireFly (FA) that is using our proposed metric outperforms the state-of-art training-free methods on the CIFAR-10 and ImageNet16-120 datasets,

Method	CIFAR-10	CIFAR-100	ImageNet16-120
NAS-WOT (N=10)	92.47 ± 0.04	69.20 ± 1.05	42.20 ± 1.37
EPE-NAS (N=10)	92.63 ± 0.32	**70.10 ± 1.71**	41.92 ± 4.25
NAS-WOT (N=100)	91.41 ± 2.24	67.18 ± 4.14	41.42 ± 1.53
EPE-NAS (N=100)	91.59 ± 0.87	67.19 ± 3.82	38.80 ± 5.41
NAS-WOT (N=500)	91.71 ± 1.37	67.54 ± 2.23	39.84 ± 3.68
EPE-NAS (N=500)	92.27 ± 1.75	69.33 ± 0.66	42.05 ± 3.09
NAS-WOT (N=1000)	91.20 ± 2.04	68.95 ± 0.72	38.08 ± 1.58
EPE-NAS (N=1000)	91.31 ± 1.69	69.58 ± 0.83	41.84 ± 2.06
FA (our)	**92.90 ± 1.07**	**70.10 ± 1.56**	**43.38 ± 1.80**
IFA (our)	**93.58 ± 0.15**	**70.27 ± 0.75**	**44.53 ± 1.54**

TABLE V: Comparison on the NAS-BENCH-201 between our proposition and state of the art methods. Performance shown in accuracy with mean±std

and gets the same mean accuracy as EPE-NAS proposed by Lopes et al. [23] on CIFAR-100 dataset, but has a smaller std value, meaning that FA is more stable than EPE-NAS.

When comparing FA and IFA, we can notice that IFA achieves better results with an increased accuracy and smaller sdt. As excepted using the genetic operations makes our proposed method more robust to the local optimums problem.

D. Comparison with the top performing state of the art NAS methods

In the following, we will compare between our proposed method and the top performing state of the art NAS methods (based on training), according to the test accuracy.

Benchmark(Dataset)	Method	score	our score
NAS-BENCH-101(CIFAR-10)	GA-NAS [26]	94.23	94.03
NAS-BENCH-201(CIFAR-10)	β-DARTS [25]	94.36	93.58
NAS-BENCH-201(CIFAR-100)	β-DARTS [25]	73.51	70.27
NAS-BENCH-201(ImageNet)	β-DARTS-RS [25]	46.71	44.53

TABLE VI: Comparison on different benchmarks/dataset between our proposition and the top performing state of the art NAS methods according to the test accuracy

Table VI includes the comparison with GA-NAS proposed by Rezaei el al. [26] on the NAS-BENCH-101, and also, the comparison with β-DARTS and β-DARTS-RS proposed by Peng et al. [25]. We can notice that even if our proposed training free method does not reach the same performances as the ones using training, the gap between them is small.

VI. Conclusion

The NAS algorithms are able to automate the discovery of effective architectures for the search space. Finding the most interesting model can be time consuming since NAS algorithms have to score the networks according to their performances to choose the most adapted one.

In order to avoid training models when performing a NAS, we proposed a novel metric, that approximates the quality of a model without any training. The latter is based on the use of the IntraCluster Distance score. The obtained results on different NAS Benchmarks (NAS-BENCH-101, NAS-BENCH-201, NDS and NAS-BENCH-NLP) show that our metric is valid for scoring either CNNs and RNNs.

We proposed to use an Improved Firefly algorithm (IFA), that uses Genetic Algorithm's operators, allowing it to be more robust than the baseline FireFly algorithm to the local optimums problem, as a search technique to find the best architecture for a given dataset. Experimental results on the NAS-BENCH-101 and NAS-BENCH-201 benchmarks show that our IFA combined to our proposed metric outperform the state-of-art training-free methods and the baseline FireFly algorithm.

As a future work, we expect to find a generic way to represent the hyperparameters of a neural network architecture (layer's count, number of unit, cell type, etc.) in order to use our improved FireFly algorithm associated to the proposed metric to find the most suitable model for a given dataset.

ACKNOWLEDGEMENTS

This work has been carried out within the French-Canadian project DOMAID which is funded by the National Agency for Research (ANR-20-CE26-0014-01) and the FRQSC

References

[1] Cao, X., Yao, J., Xu, Z., and Meng, D. : Hyperspec-tral image classification with convolutional neural net-work and active learning.IEEE Transactions on Geo-science and Remote Sensing, 58(7):4604–4616, 2020.

[2] Martins, V., Kaleita, A., Gelder, B., Silveira, H., and Abe,C. : Exploring multiscale object-based convo-lutional neural network (multi-ocnn) for remote sens-ing image classification at high spatial resolution.IS-PRS Journal of Photogrammetry and Remote Sensing,168:56–73,2020.

[3] Mustaqeem and Kwon, S.: Mlt-dnet: Speech emo-tion recognition using 1d dilated cnn based on multi-learning trick approach.Expert Systems with Applica-tions, 167, 2020.

[4] Zhang, N., Wang, J., Wei, W., Qu, X., Cheng, N., and Xiao,J. : Cacnet: Cube attentional cnn for automaticspeech recognition. In2021 International Joint Con-ference on Neural Networks (IJCNN), pages 1–7, 2021.

[5] Elsken, T., Metzen, J. H., and Hutter, F. Neural architecture search: A survey. Journal of Machine Learning Research, 20(55):1–21, 2019.

[6] Wistuba, M., Rawat, A., and Pedapati, T. A survey on neural architecture search. arXiv preprint arXiv:1905.01392, 2019.

[7] Zoph, B. and Le, Q. V. Neural architecture search with reinforcement learning. In International Conference on Learning Representations, 2017.

[8] Zoph, B., Vasudevan, V., Shlens, J., and Le, Q. V. Learning transferable architectures for scalable image recognition. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2018.

[9] Pham, H., Guan, M. Y., Zoph, B., Le, Q. V., and Dean, J. Efficient neural architecture search via parameter sharing. In International Conference on Machine Learning, 2018.

[10] Tan, M., Chen, B., Pang, R., Vasudevan, V., Sandler, M., Howard, A., and Le, Q. V. MnasNet: Platform-aware neural architecture search for mobile. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2019.

[11] Liu, H., Simonyan, K., and Yang, Y. DARTS: Differentiable architecture search. In International Conference on Learning Representations, 2019.

[12] Real, E., Aggarwal, A., Huang, Y., and Le, Q. V. Regularized evolution for image classifier architecture search. In Proceedings of the AAAI Conference on Artificial Intelligence, 2019.

[13] Ying, C., Klein, A., Christiansen, E., Real, E., Murphy, K. & Hutter, F. : NAS-Bench-101: Towards Reproducible Neural Architecture Search. Proceedings of the 36th International Conference on Machine Learning, in Proceedings of Machine Learning Research, 2019.

[14] Dong, X. and Yang, Y., "NAS-Bench-201: Extending the Scope of Reproducible Neural Architecture Search", arXiv e-prints, 2020.

[15] Radosavovic, Ilija & Johnson, Justin & Xie, Saining & Lo, Wan-Yen & Dollár, Piotr : On Network Design Spaces for Visual Recognition, 2019.

[16] Klyuchnikov, Nikita & Trofimov, Ilya & Artemova, Ekaterina & Salnikov, Mikhail & Fedorov, Maxim & Burnaev, Evgeny. NAS-Bench-NLP: Neural Architecture Search Benchmark for Natural Language Processing, 2020.

[17] I. Strumberger, E. Tuba, N. Bacanin,M. Zivkovic, M. Beko, and M. Tuba. Designing convolutional neural network architecture by the firefly algorithm. In2019 International Young Engineers Forum (YEF-ECE), pages 59–65, 2019.

[18] Vina Ayumi, L.M. Rere, Mohamad IvanFanany, and Aniati Arymurthy. Optimization of convolutional neural network using microcanonical annealing algorithm. 102016.

[19] Yanan Sun, Bing Xue, Mengjie Zhang, and Gary Yen. Automatically designing cnn architectures using Genetic Algorithm for image classification, 08 2018.

[20] Adenilson Carvalho, Fernando Ramos, and Antonio Chaves. Metaheuristics for the feed forward artificial neural network (ann)architecture optimization problem.Neural Computing and Applications, 20, 10 2010.

[21] L.M. Rere, Mohamad Ivan Fanany,and Aniati Arymurthy.Metaheuristic algorithms for convolution neural network.Computational Intelligence and Neuroscience, 2016.

[22] Mellor, J., Turner, J., Storkey, A. Crowley, E.J. : Neural Architecture Search without Training. Proceedings of the 38th International Conference on Machine Learning, in Proceedings of Machine Learning Research,2021

[23] Lopes, Vasco et al. "EPE-NAS: Efficient Performance Estimation Without Training for Neural Architecture Search." ICANN (2021).

[24] Jianbo Yang, Minh Nhut Nguyen,Phyo Phyo San, Xiao Li Li, and Shonali Krishnaswamy. Deep convolutional neural networks on multichannel time series for human activity recognition, IJCAI, 2015

[25] Ye, Peng and Li, Baopu and Li, Yikang and Chen, Tao and Fan, Jiayuan and Ouyang, Wanli. β-DARTS: Beta-Decay Regularization for Differentiable Architecture Search,2022

[26] Rezaei, Seyed Saeed Changiz and Han, Fred X and Niu, Di and Salameh, Mohammad and Mills, Keith and Lian, Shuo and Lu, Wei and Jui, Shangling. Generative Adversarial Neural Architecture Search,arXiv,2021