A Study of Fitness Landscapes for Neuroevolution

Nuno M. Rodrigues
LASIGE,
Departamento de Informática,
Faculdade de Ciências,
Universidade de Lisboa,
1749-016 Lisboa, Portugal
nmrodrigues@fc.ul.pt

Sara Silva
LASIGE,
Departamento de Informática,
Faculdade de Ciências,
Universidade de Lisboa,
1749-016 Lisboa, Portugal
sara@fc.ul.pt

Leonardo Vanneschi
NOVA Information Management
School (NOVA IMS),
Universidade Nova de Lisboa,
Campus de Campolide,
1070-312 Lisboa, Portugal
lvanneschi@novaims.unl.pt

***Abstract*—Fitness landscapes are a useful concept to study the dynamics of meta-heuristics. In the last two decades, they have been applied with success to estimate the optimization power of several types of evolutionary algorithms, including genetic algorithms and genetic programming. However, so far they have never been used to study the performance of machine learning algorithms on unseen data, and they have never been applied to neuroevolution. This paper aims at filling both these gaps, applying for the first time fitness landscapes to neuroevolution and using them to infer useful information about the predictive ability of the method. More specifically, we use a grammar-based approach to generate convolutional neural networks, and we study the dynamics of three different mutations to evolve them. To characterize fitness landscapes, we study autocorrelation and entropic measure of ruggedness. The results show that these measures are appropriate for estimating both the optimization power and the generalization ability of the considered neuroevolution configurations.**

***Index Terms*—Fitness Landscapes, Neuroevolution, Convolutional Neural Networks, Autocorrelation, Entropic Measure of Ruggedness.**

I. Introduction

Fitness landscapes (FLs) [1], [2] have been studied for many years to characterize the dynamics of meta-heuristics in optimization. In particular, several measures have been introduced to capture some important characteristics of FLs, that can give useful information about the difficulty of optimization problems. Among those measures, autocorrelation [3] and entropic measure of ruggedness (EMR) [4]–[6] have been intensively studied, and they revealed useful indicators of the ruggedness of the FLs induced by several variants of local search meta-heuristics and evolutionary algorithms (EAs) [7]. However, to the best of our knowledge, despite existing few works on Fls applied to standard neural networks [8], [9], no measure related to FLs has ever been applied so far to study the performance of machine learning (ML) algorithms on unseen data, and none of those measures has ever been used to characterize the functioning of neuroevolution. In this work, for the first time, we adapt the well-known definitions of autocorrelation and EMR to neuroevolution, and we use those measures not only to study the optimization effectiveness of various configurations of the method, but also to characterize their performance on unseen data.

Neuroevolution is a branch of evolutionary computation that has been used for almost three decades, with application in multiple areas such as supervised classification tasks [10] and agent building [11]. In neuroevolution, an EA is used to evolve, for instance, weights, topologies and/or hyper-parameters of artificial neural networks. In this study, we focus on the evolution of convolutional neural networks (CNNs), not only because they are arguably one of the most popular deep neural network architectures, but also because they have a vast amount of tunable parameters, which makes CNNs appropriate to test the capabilities of neuroevolution. We use our own grammar-based neuroevolution approach, inspired by existing systems, also introduced here.

This work is aimed at testing the reliability of autocorrelation and EMR in predicting the performance of neuroevolution of CNNs on training and unseen data. For achieving this task, we consider three different types of mutations and four different multiclass classification problems, with different degrees of difficulty. For each type of mutation, and for each one of the studied problems, we calculate the value of these measures and we compare them to the results obtained by actual simulations of our neuroevolution system, to test the reliability of the measures. We consider this work as the first proof of concept in a wider study, aimed at establishing the use of FL measures as indicators to characterize neuroevolution of CNNs. If successful, this study will be extremely impactful. In fact, CNNs usually have a slow learning phase, which makes neuroevolution a very intensive computational process, since it requires the evaluation of several CNNs in each generation. For this reason, the task of executing simulations to choose among different types of genetic operators, and/or among several possible parameter settings, is generally prohibitively expensive. On the other hand, the calculation of measures such as the autocorrelation and the EMR is much faster. So, these measures could help us find appropriate neuroevolution configurations much more efficiently.

The paper is organized as follows: in Section II, we introduce the concept of FL and the used measures. Section III introduces neuroevolution and presents our grammar-based approach to evolve CNNs. In Section IV, we present our experimental study, first discussing the used test problems and the experimental settings, and then presenting the obtained

978-1-7281-6929-3/20/$31.00 ©2020 IEEE

results. Finally, Section V concludes the paper and suggests ideas for future research.

II. Fitness Landscapes

Using a landscape metaphor to gain insight about the workings of a complex system originates with the work of Wright on genetics [1]. Probably, the simplest definition of FL is the following one: a FL is a plot where the points in the horizontal direction represent the different individual genotypes in a search space and the points in the vertical direction represent the fitness of each one of these individuals [12]. If genotypes can be visualized in two dimensions, the plot can be seen as a three-dimensional "map", which may contain peaks and valleys. The task of finding the best solution to the problem is equivalent to finding the highest peak (for maximization problems) or the lowest valley (for minimization). The problem solver is seen as a short-sighted explorer searching for those optimal spots. Crucial to the concept of FL is that solutions should be arranged in a way that is consistent with a given neighborhood relationship. Indeed, a FL is completely identified by the triple (S, f, N), where S is the set of all admissible solutions (the search space), f is the fitness function, and N is the neighborhood. Generally, the neighborhood N should have a relationship with the transformation (mutation) operator used to explore the search space. A typical example is to consider as neighbors two solutions a and b if and only if b can be obtained by applying mutation to a.

The FL metaphor can be helpful to understand the difficulty of a problem, i.e., the ability of a searcher to find the optimal solution for that problem. However, in practical situations, FLs are impossible to visualize, both because of the vast size of the search space S and because of the multi-dimensionality of the neighborhood N. For this reason, researchers have introduced a set of mathematical measures, able to capture some characteristics of FLs and express them with a single number [13]. Although none of these measures is capable of expressing completely the vast amount of information that characterize a FL, some of them revealed to be reliable indicators of the difficulty of problems, for instance: autocorrelation [3], entropic measure of ruggedness (EMR) [4]–[6], density of states [14], fitness-distance correlation [13], [15], length of adaptive walks [2], basins of attraction size [16], plus various measures based of the concepts of fitness clouds [13] and local optima networks [17]. In this paper, we have decided to investigate autocorrelation and EMR because, among the previously mentioned measures, they are probably the most simple ones, and also the ones that can be calculated more efficiently. For this reason, they seem appropriate for studying a computationally intensive method such as neuroevolution of CNNs. Autocorrelation and EMR are presented next.

A. Autocorrelation

The autocorrelation coefficient is used to measure the ruggedness of a FL [3]. It is applied over a series of fitness values, determined by a *walk* on the landscape. A walk on a FL is a sequence of solutions $(s_0, s_1, ..., s_n)$, such that for each $t = 1, 2, ..., n$, s_t is a neighbor of s_{t-1} or, in other words, s_t is obtained by applying mutation to s_{t-1}. For walks of a finite length n, autocorrelation with a *step* k is defined as:

$$\hat{\rho}(k) = \frac{\sum_{t=1}^{n-k}(f(s_t) - \bar{f})(f(s_{t+k}) - \bar{f})}{\sqrt{\sum_{t=1}^{n}(f(s_t) - \bar{f})^2}\sqrt{\sum_{t=1}^{n}(f(s_{t+k}) - \bar{f})^2}},$$

where $\bar{f} = \frac{1}{n}\sum_{t=1}^{n} f(s_t)$.

Given the huge size of the neuroevolution search space, and in the attempt to generate walks that are, as much as possible, representatives of the portions of the search space actually explored by the evolutionary algorithm, in this work we have decided to calculate autocorrelation using *selective walks* [13]. In selective walks, for each $t = 1, 2, ..., n$, s_t is a selected solution from the neighborhood of s_{t-1}. To apply selection pressure to the neighbors, tournament selection is used; in other words, s_t is the best solution (i.e., the one with the best fitness on the training set) in a sample of m randomly generated neighbors of s_{t-1}. We study the autocorrelation both on the training and on the test set, by using the same selective walk. In both cases selection acts using only training data, but in the former case the individuals are evaluated on the training set, while in the latter case they are evaluated on the test set.

Because of the large complexity of neuroevolution, and given the relatively short length of the walks that we are able to generate with the available computational resources[1] ($n = 30$ in our experiments), we have decided to calculate $\hat{\rho}(k)$ several times (10 in our experiments), using independent selective walks, and to report boxplots of the results obtained over these different walks. In order to broadly classify the ruggedness of the landscape, we have adopted the heuristic threshold suggested by Jones for fitness-distance correlation [15]: $\hat{\rho}(k) > 0.15$ will correspond to a smooth landscape (and thus, in principle, an easy problem), while $\hat{\rho}(k) < 0.15$ will correspond to a hard landscape. To visualize the results, the threshold will be shown as an horizontal line in the same diagram as the boxplots of the autocorrelation, and the position of the box compared to the threshold will allow us to classify problems as easy or hard. The situation in which the boxplot lays across the threshold (i.e., the case in which $\hat{\rho}(k) \approx 0.15$) will be considered as an *uncertain* case, in which predicting the hardness of the problem is difficult. One of the typical situations in which we have an uncertain case is when several different neuroevolution runs give significantly different outcomes (for instance, half of the runs converge towards good quality solutions and the other half stagnate in bad quality ones). Finally, several values of the step k are compared ($k = 1, 2, 3, 4$ in our experiments).

B. Entropic Measure of Ruggedness

The EMR is an indicator of the relationship between ruggedness and neutrality. In the context of Fitness Landscapes, we

[1] Our experiments were performed on a gtx 970 and on a gtx 2070.

define neutrality, or neutral degree, as the capacity of the algorithm to generate solutions with different fitness values. If multiple solutions have different neighbors with the same fitness, the landscape is deemed to have a high degree of neutrality. It was introduced by Vassilev [4]–[6] and is defined as follows: assuming that a walk of length n, performed on a landscape, generates a time series of fitness values $\{f_t\}_{t=0}^{n}$, that time series can be represented as a string $S(\varepsilon) = \{x_1 x_2 ... x_n\}$, where, for each $i = 1, 2, ..., n$, $x_i \in \{\bar{1}, 0, 1\}$. For each $i = 1, 2, ..., n$, $x_i \in S(\varepsilon)$ is obtained using the following function:

$$x_i = \Psi_{f_t}(i, \varepsilon) = \begin{cases} \bar{1}, \text{ if } f_i - f_{i-1} < -\varepsilon \\ 0, \text{ if } |f_i - f_{i-1}| \leq -\varepsilon \\ 1, \text{ if } f_i - f_{i-1} > -\varepsilon \end{cases}$$

where ε is a real number that determines the accuracy of the calculation of $S(\varepsilon)$, and increasing this value results in increasing the neutrality of the landscape. The smallest possible ε for which the landscape becomes flat is called the information stability, and is represented by ε^*. Using $S(\varepsilon)$, the EMR is defined as follows [4]:

$$H(\varepsilon) = -\sum_{p \neq q} P_{[pq]} \log_6 P_{[pq]},$$

where p and q are elements from the set $\{\bar{1}, 0, 1\}$, and $P_{[pq]} = \frac{n_{[pq]}}{n}$, where $n_{[pq]}$ is the number of pq sub-blocks in $S(\varepsilon)$ and n is the total number of sub-blocks. The output of $H(\varepsilon)$ is a value in the $[0, 1]$ range, and it represents an estimate of the variety of fitness values in the walk, with a higher value meaning a larger variety and thus a more rugged landscape. In this definition, for each walk that is performed, $H(\varepsilon)$ is calculated for multiple ε values, usually $\{0, \varepsilon^*/128, \varepsilon^*/64, \varepsilon^*/32, \varepsilon^*/16, \varepsilon^*/8, \varepsilon^*/4, \varepsilon^*/2, \varepsilon^*\}$, and

then the mean of $H(\varepsilon)$ ($\bar{H}(\varepsilon)$) over all performed walks is calculated for each value of ε. In this work, we employ the adaptations suggested by Malan [18], aimed at reducing the characterization of the landscape to a single scalar. To characterise the ruggedness of a function f, the following value is proposed:

$$R_f = \max_{\forall \varepsilon \in [0, \varepsilon^*]} H(\varepsilon)$$

To approximate the theoretical value of R_f, the maximum of $\bar{H}(\varepsilon)$ is calculated for all ε values.

III. Neuroevolution

Neuroevolution is usually employed to evolve the topology, weights, parameters and/or learning strategies of artificial neural networks. Some of the most well known neuroevolution systems include EPNet [19], NEAT [11], EANT [20], and hyperNEAT [21]. Most recently, works have appeared that apply neuroevolution to other types of neural networks, such as CNNs [22]–[24]. In this section we describe how we represent networks using a grammar-based approach, and we discuss the employed genetic operations.

A. Grammar-Based Neuroevolution

We have decided to use a grammar-based approach because of its modularity and flexibility. The employed grammar is reported in Fig. 1. It contains all the possible values for the parameters of each available layer. This way, adding and removing layers or changing their parameters is simple and requires minimal changes. Using this grammar, we are discretizing the range of the possible values that each parameter can take. This greatly reduces the search space, while keeping the quality of the solutions under control, as in most cases, intermediate values can have little to no significant influence on the effectiveness of the solutions, as reported in [10]. In our representation, genotypes are composed by two different sections, S_1 and S_2, that are connected using the so called Flatten gene. The Flatten gene implements the conversion (i.e., the "flattening") of multidimensional features matrixes from the convolutional layers into a one dimensional array that is fed to the following fully connected layer. On S_1 we have genes that encode the layers that deal with feature extraction from the images, convolutional and pooling layers, and on S_2 we have genes that encode the classification and pruning parts of the network, dense and dropout layers. Separating the network into these two segments helps make the implementation more modular and expandable, provided that when adding or removing new layers will only affect interactions within their segment. Besides the flatten layer, the only other layer that is the same for all possible individuals is the output layer, which is a fully connected (i.e., dense) layer with softmax activation and a number of units equal to the number of classes to be predicted. The genetic operators cannot modify this layer, except for the bias parameter.

Conv ::	filters	|32,64,128,256|
	kernel_size	|2,3,4,5|
	stride	|1,2,3|
	activation	|relu, elu, sigmoid|
	use_bias	|true, false|
Pool ::	type	|Max, Avg|
	pool_size	|2,3,4,5|
	stride	|1,2,3|
Dense ::	units	|8,16,32,64,128,256,512|
	activation	|relu, elu, sigmoid|
	use_bias	|true, false|
Dropout ::	rate	[0.0 → 0.7]
Optimizer ::	learning_rate	|0.01, 0.001, 0.0001, 0.00001|
	decay	|0.01, 0.001, 0.0001, 0.00001|
	momentum	|0.99, 0.9, 0.5, 0.1|
	nesterov	|true, false|

Fig. 1: Grammar used to evolve the CNNs.

Before evaluation, a genotype is mapped into a phenotype, that is a neural network itself. Evaluation involves training the network and calculating its performance on the given data. During the evolutionary process, we use the loss value on the training set as a fitness function to evaluate the networks. Regarding the optimizer used for training the networks, we have chosen Stochastic Gradient Descent (SGD) [25]. Also, since we are working with multiclass classification problems that are not one-hot encoded, we used Sparse Categorical Cross-Entropy as a loss function, which motivates the need to have the fixed number of neurons and activation function in the output layer. We also measure the accuracy and loss in a separate test set in order to study the generalization ability.

Due to the difficulty of defining a crossover-based neighborhood for studying FLs [26], we consider only mutation. Having in mind the vast amount of possible mutation choices, we have decided to restrict our study to three different types of operators:

- Topology mutations. Mutations that add or delete a gene, except for the flatten gene, changing the topology of the genotype and, consequently, the one of the phenotype.
- Parameter mutations. Mutations that change the parameters encoded in a gene. They cover all parameters of all gene types, excluding the flatten gene.
- Learning mutations. Mutations that change the parameters related to the learning process, which are encoded in the Optimizer gene.

IV. Experimental Study

A. Datasets and Experimental Settings

Table I describes the main characteristics of the datasets used as test cases in our experiments. The partition into training and test set is made randomly, and it is different at each run. For all datasets, a simple image scale adjustment was done, setting pixel values in the $[0, 1]$ range. No further data pre-processing or image augmentation was applied to the datasets. The MNIST dataset consists in a set of gray scale images of handwritten digits from 0 to 9 [27]. Fashion-MNIST (FMNIST) is similar to MNIST, but with clothing articles instead of digits [28]. CIFAR10 contains RGB pictures of 10 different types of real world objects [29]. Finally, SVHN contains RGB pictures of house numbers, containing digits from 0 to 9 [30]. For each one of these four datasets and for each one of the three studied mutation operators, we perform selective walks (that allow us to have all the needed information to calculate the autocorrelation and the EMR) and we execute the neuroevolution. From now on, we will use the term *configuration* to indicate an experiment in which a particular type of mutation was used on a particular dataset. For each configuration, we perform 10 independent selective walks and 10 independent neuroevolution runs. All neuroevolution runs are performed starting with a randomly initialized population of individuals, and all the selective walks are constructed starting with a randomly generated individual.

To determine the values of the main parameters (e.g., population size and number of generations for neuroevolution, length of the walk and number of neighbors for selective walks) we have performed some benchmark tests with multiple values, and selected ones that allowed us to obtain results in "reasonable" time[2] with our available computational resources. The employed parameter values are reported in Table II. The first two columns contain the parameters used to build the selective walks and the parameters of the neuroevolution, respectively. One should keep in mind that, in order to evaluate all the neural networks in the population, all the networks need to go through a learning phase at each generation of the evolutionary process. So, the third column reports the values used by each one of those networks for learning. When decoding the genotype into the phenotype, the weights of the network are randomly initialized using the Xavier initialization.

TABLE I: Number of training and test observations, and number of classes of each dataset.

	Training set	**Test set**	**Classes**
MNIST	60000	10000	10
FMNIST	60000	10000	10
CIFAR10	50000	10000	10
SVHN	73257	26032	10

TABLE II: Parameter values used in our experiments.

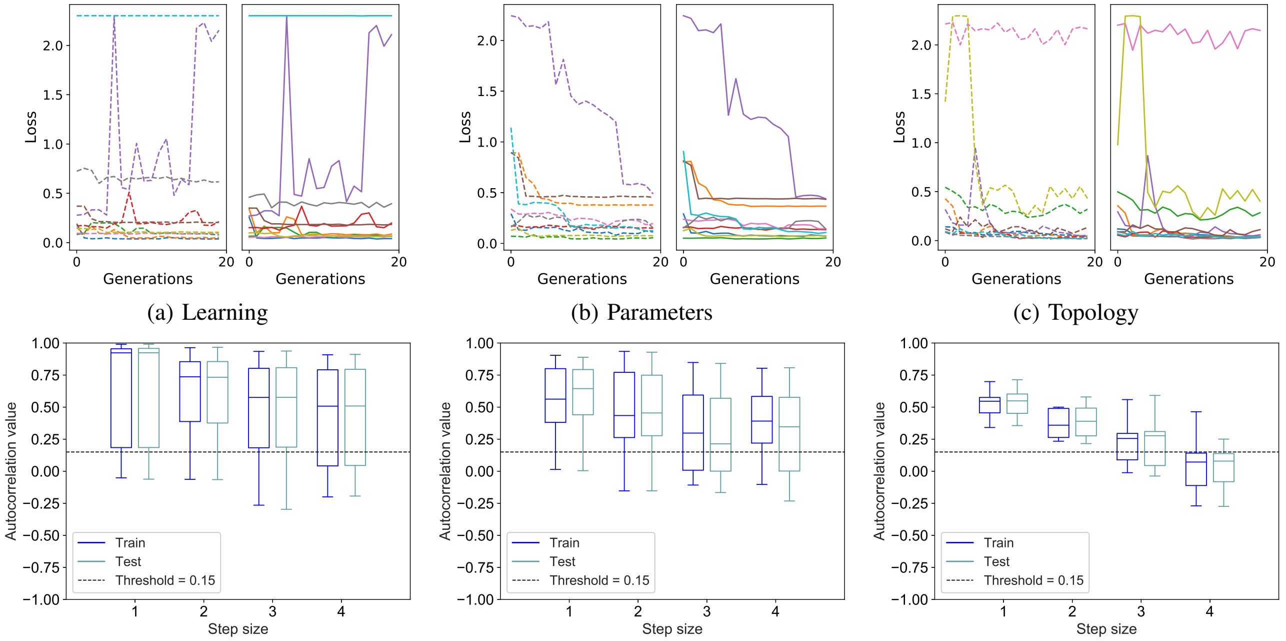
Selective walk		**Neuroevolution**		**Learning**	
Length	30	Population size	10	Epochs	8
# Neighbors	3	# Generations	20	Batch	64
		Tournament size	2		
		Mutation rate	1		
		Crossover rate	0		
		No elitism			

B. Experimental Results

We begin by analyzing the ability of autocorrelation to characterize training and test performance of neuroevolution of CNNs. Fig. 2 reports the evolution of the loss and the autocorrelation for the MNIST problem. The first line of plots reports the evolution of the loss against generations for the three studied mutation operators. Each plot in the first line is partitioned into two halves: the leftmost one reports the evolution of the training loss of the best individual, while the rightmost one reports the loss of the same individual, evaluated on the unseen test set. Each curve in these plots reports the results of one neuroevolution run. The second line contains the boxplots of the autocorrelation values, calculated over 10 independent selective walks, both on the training and on the test set. Each column of plots reports the results for a different type of mutation, allowing us to easily compare the outcome of the neuroevolution and the one of the autocorrelation for the different configurations.

As we can observe from plots (a) and (b) of Fig. 2, when we employ learning mutation and parameters mutation, the MNIST problem is easy to solve, both on training and test set. In fact, except for the outlier runs in plot (a), all the runs either approximate the ideal value of loss equal to zero, or tend towards it. Now, looking at plots (d) and (e), it is possible to observe that the autocorrelation is able to capture the fact that the problem is easy. In fact, in both cases, practically the whole autocorrelation box stands above (and the medians never go below) the 0.15 threshold. When the topology mutation is used, the situation changes: the number of runs in which the

[2] On average, 5 hours per run.



(a) Learning

(b) Parameters

(c) Topology

(d) Learning

(e) Parameters

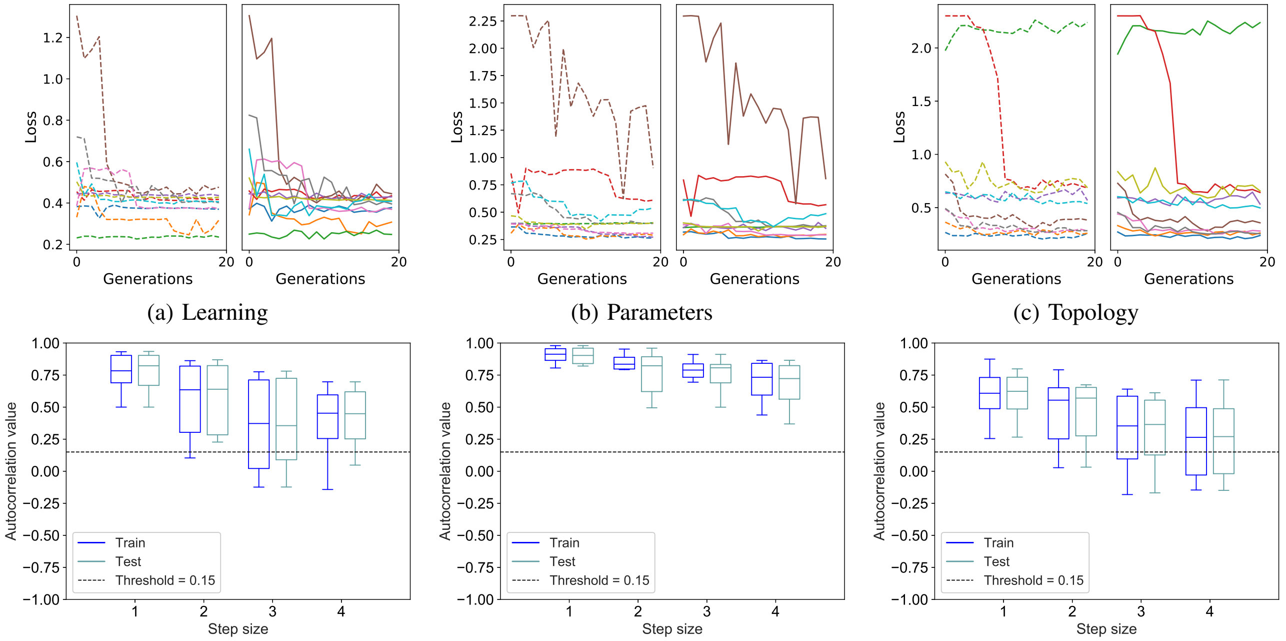
(f) Topology

Fig. 2: *MNIST dataset*. Plots (a), (b) and (c): neuroevolution results; plots (d), (e) and (f): autocorrelation results. Plots (a), (b) and (c) report the evolution of the best fitness in the population at each generation (one curve for each performed neuroevolution run). Each plot is partitioned into two subfigures: training loss on the left and test loss on the right. Plots (d), (e) and (f) report the boxplots of the autocorrelation, calculated over 10 independent selective walks.

evolution does not have a regular trend is larger. This may not be obvious by looking at plot (c) because of the scale of the y axis, but the lines are now much more rugged than they were for the other two cases. The problem is now harder than it was, and as we can see in plot (f), the autocorrelation catches this difficulty. In particular, we can observe that when the step is equal to 4, the whole autocorrelation boxes are below the threshold. The partial conclusion that we can draw for the MNIST dataset is that learning and parameter mutations are more effective operators than topology mutations, and this is correctly predicted by the autocorrelation. Furthermore, we can observe that the neuroevolution results obtained on the test set are very similar to the ones on the training set, practically for all the runs we have performed. Also this feature is captured by the autocorrelation, since the training and test boxes are very similar to each other for practically all the configurations.

Now we consider the results obtained for the FMNIST dataset, reported in Fig. 3. Describing the results for this dataset is straightforward: observing the neuroevolution results, we can see that for the three configurations the problems are easy, both on training and test set. In fact, all the curves are steadily decreasing and/or close to zero. Observing the scale on the left part of the plots, we can also observe that when topology mutation is used (plot (c)), the problem is slightly harder than when the other mutations are used, since the achieved values of the loss are generally higher. All this is correctly predicted by the autocorrelation, given that the boxes are above the threshold for all the configurations, and, in the case of the topology mutation (plot (f)), they are slightly lower than in the other cases. Last but not least, also in this case training and test evolution of loss are very similar between each other, and this fact finds a precise correspondence in the autocorrelation results, given that the training boxes are generally very similar to the test boxes. All in all, we can conclude that also for the FMNIST dataset, autocorrelation is a reliable indicator of problem hardness.

The results for the CIFAR10 dataset are reported in Fig. 4. Observing the neuroevolution results, we can say that when the learning mutation is used, the problem is easy (almost all the loss curves have a smooth decreasing trend, except for some outliers on the test), when the parameters mutation is used the problem is uncertain (in some runs the loss curves have a decreasing trend, while in others they have an increasing trend), and when the topology mutation is used, the problem is hard (almost all the loss curves have an increasing trend). Observing the autocorrelation results, we find a reasonable correspondence. For the learning mutation all the boxes are clearly above the threshold, for the parameters mutation the boxes are not as high as for the learning mutation, beginning to cross the threshold with steps 3 and 4, and finally for the topology mutation the boxes are even lower, with the medians below the threshold for steps 3 and 4, and more than half the height of the boxes also below the threshold for step 4.



(a) Learning

(b) Parameters

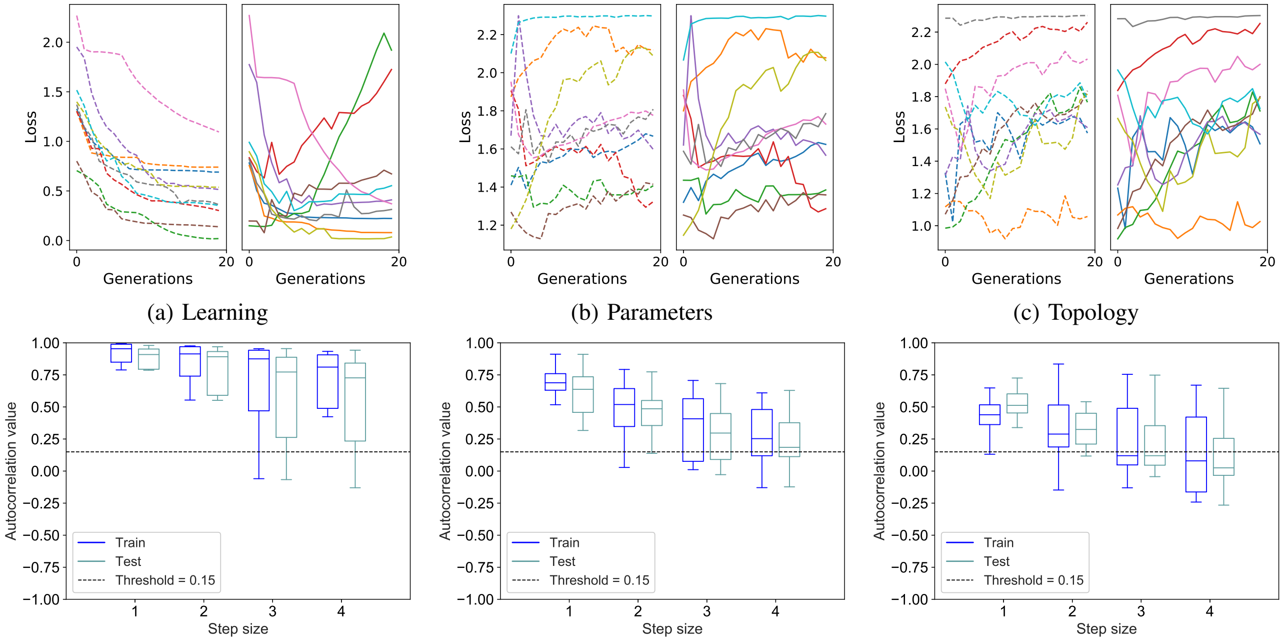
(c) Topology

(d) Learning

(e) Parameters

(f) Topology

Fig. 3: *FMNIST dataset*. The organization of the plots is analogous to Fig. 2.



(a) Learning

(b) Parameters

(c) Topology

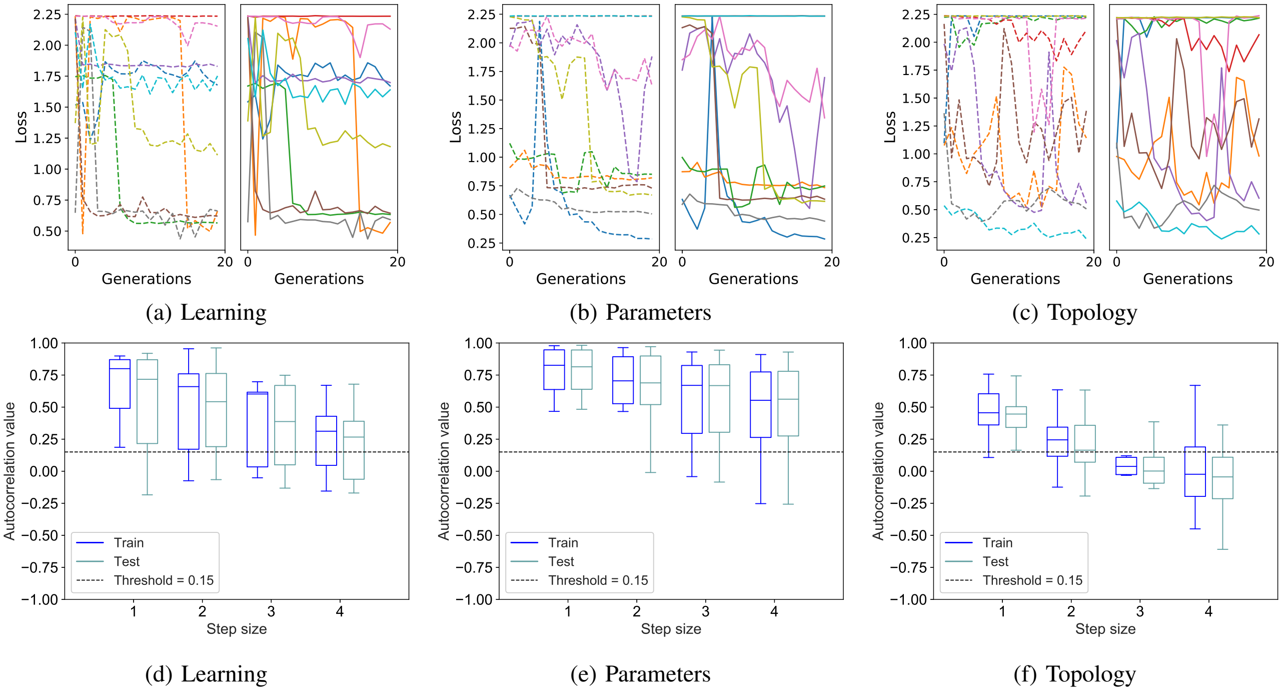
(d) Learning

(e) Parameters

(f) Topology

Fig. 4: *CIFAR10 dataset*. The organization of the plots is analogous to Fig. 2.

As already observed in plot (f) of Fig. 3, longer steps seem to be better indicators when the autocorrelation is applied to hard problems. Another observation for CIFAR10 is that the evolution of the loss curves for the learning mutation (plot (a)) clearly shows that there is a much larger diversity of behaviors on the test set than on the training set. This fact also finds a correspondence in the autocorrelation results (plot (d)), given that the test boxes are taller than the training boxes, in



(a) Learning

(b) Parameters

(c) Topology

(d) Learning

(e) Parameters

(f) Topology

Fig. 5: *SVHN dataset*. The organization of the plots is analogous to Fig. 2.

particular for step 4.

As a last test case for autocorrelation, we now analyse the results obtained on the SVHN dataset, reported in Fig. 5. In this case, the plots of the loss evolution indicate that the problem is uncertain when learning mutation is used (given that approximately half of the curves have a decreasing trend, while the other half have an oscillating trend), easy when parameters mutation is used (with the majority of the curves having a decreasing trend) and hard when topology mutation is used (with most curves exhibiting an oscillatory behaviour, which indicates poor optimization ability). Also in this case, autocorrelation is a reasonable indicator of problem difficulty. For learning mutation the boxes are crossing the threshold for steps 3 and 4, for parameters mutation they are above the threshold, and for topology mutation they are almost completely below the threshold for steps 3 and 4. The medians are lower and the dispersion of values is larger for step 4, which reflects well the neuroevolution behavior observed in plot (c) (unstable and often returning to high values of the loss). The highest step length is once again the most reliable.

Finally, we study the results of the EMR, reported in Fig. 6. Each plot reports the results for one mutation type, showing the values of $H(\varepsilon)$ for multiple ε values (see Sect. II-B) on the four studied datasets. These curves illustrate the trend of how ruggedness changes with respect to neutrality. The results show that, overall, the obtained landscapes have a low degree of neutrality, not maintaining the value of $H(\varepsilon)$ as ε increases. The most neutral landscape is the one produced by topology mutation on the MNIST dataset (plot (c) of Fig. 6). Its highest $H(\varepsilon)$ happens when $\varepsilon = \varepsilon^*/64$, but the value suffers minimal change from $\varepsilon = \varepsilon^*/128$ to $\varepsilon = \varepsilon^*/8$.

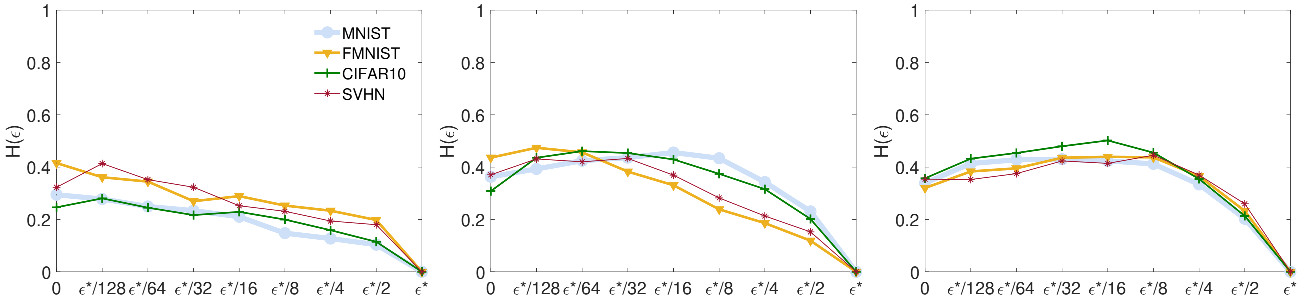
TABLE III: R_f for each mutation on the studied test problems.

	Learning	**Parameters**	**Topology**
MNIST	0.29	0.45	0.43
FMNIST	0.41	0.47	0.43
CIFAR10	0.28	0.46	0.50
SVHN	0.41	0.43	0.44

Table III, which reports the values of R_f for each type of mutation, and for each studied test problem, corroborates the previous discussion: the maximum value for learning mutation is 0.41, while for parameters mutation is 0.47 and for topology mutation is 0.5. Again, we can see that learning mutations induce the smoothest landscapes, while topology mutations induce the most rugged ones. Also in this case, the prediction of the EMR corresponds to what we can observe from the actual neuroevolution runs.

V. Conclusions and Future Work

Two different measures of fitness landscapes, autocorrelation and entropic measure of ruggedness, were used for the first time to characterize the performance of neuroevolution of convolutional neural networks. The results were obtained on four different test problems, and confirm that these measures are reasonable indicators of problem hardness, both on the training set and on the test set. Future work involves the study of other measures of fitness landscapes, on more test problems, with the objective of developing well established, theoretically motivated predictive tools for neuroevolution, that can significantly simplify the



(a) Learning

(b) Parameters

(c) Topology

Fig. 6: Results of the Entropic Measure of Ruggedness $\bar{H}(\varepsilon)$ over different values of ε^* for the the three mutation operators on the four considered test problems.

configuration and tuning phases of the algorithm. Being able to use more powerful computational architectures, so that we are able to calculate the measures on larger and more significant samples of solutions, is crucial for achieving such an ambitious goal.

Acknowledgments

This work was partially supported by FCT through funding of LASIGE Research Unit (UIDB/00408/2020 and UIDP/00408/2020) and projects PTDC/CCI-INF/29168/2017, PTDC/CTA-AMB/30056/2017, PTDC/CCI-CIF/29877/2017, PTDC/ASP-PLA/28726/2017, DSAIPA/DS/0022/2018, DSAIPA/DS/0113/2019.

References

[1] S. Wright, *The roles of mutation, inbreeding, crossbreeding, and selection in evolution*, 1932.

[2] P. F. Stadler, *Fitness landscapes*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2002, pp. 183–204.

[3] E. Weinberger, "Correlated and uncorrelated fitness landscapes and how to tell the difference," *Biological Cybernetics*, vol. 63, no. 5, pp. 325–336, 1990.

[4] V. K. Vassilev, "Fitness landscapes and search in the evolutionary design of digital circuits," Ph.D. dissertation, Napier University, 2000.

[5] V. K. Vassilev, T. C. Fogarty, and J. F. Miller, "Information characteristics and the structure of landscapes," *Evolutionary Computation*, vol. 8, no. 1, pp. 31–60, 2000.

[6] ——, *Smoothness, Ruggedness and Neutrality of Fitness Landscapes: from Theory to Application*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2003.

[7] S. Vérel, "Fitness landscapes and graphs: multimodularity, ruggedness and neutrality," in *Genetic and Evolutionary Computation Conference, GECCO '13, Companion Material Proceedings*. ACM, 2013, pp. 591–616. [Online]. Available: https://doi.org/10.1145/2464576.2480804

[8] A. Rakitianskaia, E. Bekker, K. M. Malan, and A. Engelbrecht, "Analysis of error landscapes in multi-layered neural networks for classification," in *IEEE Congress on Evolutionary Computation*, 2016, pp. 5270–5277.

[9] M. Gallagher, "Fitness distance correlation of neural network error surfaces: A scalable, continuous optimization problem," in *Machine Learning: ECML 2001*. Springer Berlin Heidelberg, pp. 157–166.

[10] A. Baldominos, Y. Saez, and P. Isasi, "Evolutionary convolutional neural networks: An application to handwriting recognition," *Neurocomputing*, vol. 283, pp. 38 – 52, 2018.

[11] K. O. Stanley and R. Miikkulainen, "Evolving neural networks through augmenting topologies," *Evol. Comput.*, vol. 10, no. 2, p. 99–127, 2002.

[12] *Foundations of Genetic Programming*. Berlin, Heidelberg: Springer-Verlag, 2002.

[13] L. Vanneschi, "Theory and practice for efficient genetic programming," Ph.D. dissertation, Faculty of Sciences, University of Lausanne, Switzerland, 2004.

[14] H. Rosé, W. Ebeling, and T. Asselmeyer, "The density of states - a measure of the difficulty of optimisation problems," in *Proceedings of the 4th International Conference on Parallel Problem Solving from Nature, PPSN IV*. Springer-Verlag, 1996, p. 208–217.

[15] T. Jones and S. Forrest, "Fitness distance correlation as a measure of problem difficulty for genetic algorithms." Morgan Kaufmann, 1995, pp. 184–192.

[16] G. Ochoa, S. Verel, and M. Tomassini, "First-improvement vs. best-improvement local optima networks of nk landscapes," in *Parallel Problem Solving from Nature, PPSN XI*. Springer Berlin Heidelberg, 2010, pp. 104–113.

[17] G. Ochoa, M. Tomassini, S. Vérel, and C. Darabos, "A study of nk landscapes' basins and local optima networks," in *Proceedings of the 10th Annual Conference on Genetic and Evolutionary Computation, GECCO '08*, 2008, p. 555–562. [Online]. Available: https://doi.org/10.1145/1389095.1389204

[18] K. M. Malan and A. P. Engelbrecht, "Quantifying ruggedness of continuous landscapes using entropy," in *IEEE Congress on Evolutionary Computation*, 2009, pp. 1440–1447.

[19] X. Yao and Y. Liu, "A new evolutionary system for evolving artificial neural networks," *IEEE Transactions on Neural Networks*, vol. 8, no. 3, pp. 694–713, 1997.

[20] Y. Kassahun and G. Sommer, "Efficient reinforcement learning through evolutionary acquisition of neural topologies," in *In 13th European Symposium on Artificial Neural Networks (ESANN)*, 2005.

[21] K. O. Stanley, D. B. D'Ambrosio, and J. Gauci, "A hypercube-based encoding for evolving large-scale neural networks," *Artif. Life*, vol. 15, no. 2, p. 185–212, 2009.

[22] P. Verbancsics and J. Harguess, "Image classification using generative neuro evolution for deep learning," in *Proceedings of the 2015 IEEE Winter Conference on Applications of Computer Vision*, ser. WACV '15. USA: IEEE Computer Society, 2015, p. 488–493.

[23] C. Fernando, D. Banarse, M. Reynolds, F. Besse, D. Pfau, M. Jaderberg, M. Lanctot, and D. Wierstra, "Convolution by evolution: Differentiable pattern producing networks," in *Proceedings of the Genetic and Evolutionary Computation Conference 2016*, 2016, p. 109–116.

[24] R. Miikkulainen, J. Liang, E. Meyerson, A. Rawal, D. Fink, O. Francon, B. Raju, H. Shahrzad, A. Navruzyan, N. Duffy, and B. Hodjat, "Evolving deep neural networks," in *Artificial Intelligence in the Age of Neural Networks and Brain Computing*, 2018.

[25] A. C. Wilson, R. Roelofs, M. Stern, N. Srebro, and B. Recht, "The marginal value of adaptive gradient methods in machine learning," 2017.

[26] S. Gustafson and L. Vanneschi, "Operator-based distance for genetic programming: Subtree crossover distance," in *Proceedings of the 8th European Conference on Genetic Programming*. Springer-Verlag, 2005.

[27] Y. LeCun and C. Cortes, "MNIST handwritten digit database," 2010. [Online]. Available: http://yann.lecun.com/exdb/mnist/

[28] H. Xiao, K. Rasul, and R. Vollgraf, "Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms," 2017.

[29] A. Krizhevsky, "Learning multiple layers of features from tiny images," *University of Toronto*, 2012.

[30] Y. Netzer, T. Wang, A. Coates, A. Bissacco, B. Wu, and A. Y. Ng, "Reading digits in natural images with unsupervised feature learning," in *NIPS Workshop on Deep Learning and Unsupervised Feature Learning 2011*, 2011.