Automatic Generation of Neural Networks with Structured Grammatical Evolution

Filipe Assunção, Nuno Lourenço, Penousal Machado and Bernardete Ribeiro
CISUC, Department of Informatics Engineering
University of Coimbra, Coimbra, Portugal
{fga, naml, machado, bribeiro}@dei.uc.pt

***Abstract*—The effectiveness of Artificial Neural Networks (ANNs) depends on a non-trivial manual crafting of their topology and parameters. Typically, practitioners resort to a time consuming methodology of trial-and-error to find and/or adjust the models to solve specific tasks. To minimise this burden one might resort to algorithms for the automatic selection of the most appropriate properties of a given ANN. A remarkable example of such methodologies is Grammar-based Genetic Programming. This work analyses and compares the use of two grammar-based methods, Grammatical Evolution (GE) and Structured Grammatical Evolution (SGE), to automatically design and configure ANNs. The evolved networks are used to tackle several classification datasets. Experimental results show that SGE is able to automatically build better models than GE, and that are competitive with the state of the art, outperforming hand-designed ANNs in all the used benchmarks.**

I. Introduction

The process of searching for adequate classification models often follows a difficult and time consuming trial-and-error approach [1]. Moreover, such models tend to behave well on specific problems and fail to generalise to others where they were not trained on, i.e., learning is not incremental or cumulative. That said, for each problem one wants to solve there is the need to repeat the entire trial-and-error search.

To promote learning in an incremental fashion, first there is the need to develop automatic approaches capable of finding accurate models able to learn specific tasks, so that at a later stage those models can be reused to allow modular and cumulative knowledge acquisition. In order to answer this question, we propose Neural Networks Structured Grammatical Evolution (NN-SGE): a novel approach, where we rely on evolution to automatically build Artificial Neural Networks (ANNs). It is a Grammar-based Genetic Programming (GGP) methodology designed not only for the evolution of the topologies of ANNs (number of neurons and their connectivity) but also for the evolution and tuning of the numerical parameters (weights, bias). We rely on the use of grammars to define the problem domain space, in order to allow a plug-and-play approach, easy to adapt to a wide range of problems and ANN types.

The remainder of the paper is organised as follows. Section II details the background concepts and state of the art methodologies focusing on grammar-based methodologies for the evolution of ANN topologies and/or weights optimisation. In Section III we present NN-SGE, followed by an experimental analysis (Section IV). To end, in Section V, conclusions are drawn and future work is addressed.

II. Background and State of the Art

In this section we introduce the background concepts related with GGP: Grammatical Evolution (GE) (Section II-A) and Structured Grammatical Evolution (SGE) (Section II-B). In Sections II-C and II-D we review the state of the art in the field of Evolutionary Artificial Neural Networks (EANNs).

A. Grammatical Evolution

Grammatical Evolution (GE) is a Genetic Programming (GP) algorithm introduced by Ryan et al. [2], where the evolved programs are derivations of an a-priori defined Backus-Naur Form (BNF) grammar: a notation for representing Context-Free Grammars (CFGs). CFGs are rewriting systems that are formally defined by a 4-tuple, $G = (N, T, P, S)$, where: (i) N is the set of non-terminal symbols; (ii) T is the set of terminal symbols; (iii) P is the set of production rules of the form $x ::= y$, $x \in N$ and $y \in N \cup T^*$; and (iv) S is the initial symbol.

In GE individuals are commonly represented as variable-length decimal strings, where each integer represents a grammar derivation step. A decoding procedure is applied to each individual, mapping it into a grammar derivation. The decoding procedure reads each codon from left to right and decides which derivation step is to be applied, according to the following rule:

$$\textit{rule} = \textit{codon MOD non\_terminal\_possibilities},$$

where *non_terminal_possibilities* is the number of possibilities for the expansion of the current non-terminal symbol. If for a given non terminal symbol there is only one possibility for its expansion no integer is consumed. An example of the mapping procedure can be found in Section 4.1 of [2].

When applying the decoding procedure, and due to recursive grammars, it is common for the number of codons (i.e., integers) to be insufficient. To tackle that, it is possible to use wrapping, i.e., the genetic material is reused and codons reread starting from the first one. However, it has been shown in the literature that wrapping is not effective [3].

B. Structured Grammatical Evolution

Motivated by the redundancy and locality issues in GE, Lourenço et al. [4] proposed Structured Grammatical Evolution (SGE): a new genotypic representation for GE. Redundancy is linked with the modulus mathematical operation used

978-1-5090-4601-0/17/$31.00

in GE's decoding operation, that allows multiple codons to generate the same output. Consequently, mutation is affected because it is likely that a mutation in one of the individual's codons generates no change in its phenotype. Additionally, when a mutation does impact the phenotype of the individual it is probable that the mutation changes too much of the individual structure (low locality), since a modification in one of the codons may alter all the derivation procedure from that point onwards.

In SGE the genotype of an individual consists of a list of genes, one for each possible non-terminal symbol. Furthermore, a gene is a list of integers of the size of the maximum number of possible expansions for the non-terminal it encodes; each integer is a value in the interval $[0, \mathit{non\_terminal\_possibilities} - 1]$, where *non_terminal_possibilities* is the number of possibilities for the expansion of the considered non-terminal symbol. As a consequence, it is necessary to pre-process the input grammar so that the maximum number of expansions for each non-terminal is discovered (further detailed in Section 3.1 of [4]). To deal with recursion, a set of intermediate symbols are created, which are no more than the same production rule replicated a pre-defined number of times. In a sense, the creation of intermediate symbols works as the definition of a maximum tree depth, as in standard tree-based GP. The direct result of the SGE representation is that, on the one hand, the decoding of individuals no longer depends on the modulus operation and thus its redundancy is removed; on the other hand, locality is enhanced, as codons are associated with specific non-terminals.

The procedure of mapping an individual from the genotype to the phenotype is similar to the one found in GE but with two main differences: (i) as mentioned above, there is no need to use the modulus operation, and integers are used as in the genotype; (ii) integers are not read sequentially from a single list of integers, but are rather read sequentially from lists associated to each non-terminal symbol. An example of the aforementioned procedure is detailed in Section 3 of [4].

C. Evolution of Artificial Neural Networks

Several works concerning the automatic evolution of ANNs can be found in the literature [5]. Often, they are grouped according to the aspects of the ANN they optimise: (i) evolution of the network parameters; (ii) evolution of the topology; and (iii) evolution of both the topology and parameters.

The evolution of the weights is mainly motivated by the gradient descent nature of some of the learning algorithms (e.g., backpropagation) that makes them likely to get trapped in local optima [6]. When training ANNs using evolutionary approaches the topology of the network has to be provided and the only target of evolution are the weights of the connections between the different neurons, which are normally encoded using binary [7] or real representations (e.g., CoSyNE [8], Gravitational Swarm and Particle Swarm Optimization applied to OCR [9], training of deep neural networks [10]).

When applying evolutionary approaches to evolve the weights, the topology of the network is predefined and kept fixed during the evolutionary process. To overcome this limitation methodologies for automatising the evolution of the structure have also been investigated. When focusing on the evolution of the topology there are two possibilities for finding the weights: (i) use a deterministic learning methodology (e.g., backpropagation) or (ii) simultaneously evolve the topology and weights of the networks.

Regarding the used representation for the evolution of the topology of ANNs it is possible to divide the methodologies into two main types: those that use direct encodings (e.g., Topology-optimization Evolutionary Neural Network [11], NeuroEvolution of Augmenting Topologies [12]) and those that use indirect encodings (e.g., Cellular Encoding [13]). As suggested, in the first two works the genotype is a direct representation of the network, and in the latter there is the need to apply a mapping procedure to transform the genotype into a readable and interpretable network. Focusing on indirect representations, in the next section we detail grammar-based approaches for evolving ANNs.

D. Grammar-based Evolutionary Artificial Neural Networks

Over the last years several approaches applying GE to EANNs have been proposed. Tsoulos et al. [14] describe a methodology that uses a BNF grammar to encode both the network topology and parameters, i.e., input vectors, weights and bias. The evolved networks are simple Feed-Forward Neural Networks (FFNNs) with just one hidden-layer. Later, motivated by the fact that GE is not suited for the evolution and tuning of real values, Soltanian et al. [15] introduced GE-BP: an approach similar to the one proposed by Tsoulos et al., but that uses GE only for the evolution of the topology. The training of the networks relies on the backpropagation algorithm, this way avoiding the optimisation of the weights and bias. The used BNF grammar is a simpler version of the one used in [14], where all the production rules associated with the encoding of real values are removed.

In another approach to try to optimise both the topology and weights of FFNNs with just one hidden-layer, Ahmadizar et al. combine GE with a Genetic Algorithm (GA) [16]. Whilst GE is applied to the evolution of the topology, the GA is used for searching the weights and bias. Furthermore, a novel fitness assigning scheme is detailed, considering not only the quality of the networks based on their classification performance, but also an adaptive penalty term. This penalty term is a function of the number of neurons in the hidden-layer and aims at rewarding simpler structures, increasing the generalisation ability of the evolved ANNs.

Although GE is the most common approach for the evolution of ANNs by means of BNF grammars it is not the only one. In [17], Si et al. present Grammatical Swarm Neural Networks (GSNN): an approach that uses Grammatical Swarm for the evolution of the weights and bias of a fixed ANN architecture. As only the weights are being evolved the used grammar is a simple rewriting system for generating real

numbers in the $[-10, 10]$ interval. In [18], Jung and Reggia detail a method for searching adequate topologies of ANNs based on descriptive encoding languages: a formal way of defining the environmental space and how should individuals be formed. To train the ANNs the Resilient Backpropagation (RPROP) algorithm is used. Finally, important to mention Cellular Encoding [13] which is also based on a rewriting system and grammar-trees to promote the evolution of ANNs.

III. Neural Networks Structured Grammatical Evolution

NN-SGE is our proposal for the evolution of ANNs using SGE. Due to the higher locality and lower redundancy of SGE we hypothesise that NN-SGE is better suited for optimising the topology and parameters of the evolved ANNs. Next, we detail the components of the evolutionary engine that differ from the standard SGE implementation.

A. Mutation

In the standard SGE implementation the probability of mutating each gene, i.e., changing one of the integers from the list of integers associated to a non-terminal symbol is the same, and equal to $1/n$, where n is the number of genes (which equals the number of non-terminal symbols). However, the number of integers in each gene is not the same. To overcome this drawback, we propose a roulette-like approach for selecting the gene to be mutated, being the probability of choosing each gene equal to:

$$p_i = \frac{len(gene_i)}{\sum_{j=1}^{n} len(gene_j)},$$

where i is the i-th gene, n is the total number of genes and $len(gene)$ is the number of integers of that gene that are being used to map the individual from the genotype to the phenotype. Next, we select one of the integers from the chosen gene, and change its value to a new one. Mutations are not applied to non-expressed integers, i.e., those that are not used in the mapping procedure.

B. Crossover

We rely on one-point crossover to combine two parents. We start by choosing a random cutting point in the genotype, and then we swap the genetic material between the two parents. In SGE the genetic material that is swapped corresponds to genes and as such not directly to the integers, i.e., the genetic information that is swapped consists of lists of integers encoding the expansions of a given non-terminal symbols.

C. Fitness Evaluation

The performance of each ANN is measured by the Root Mean Square Error (RMSE) obtained while solving a classification task. To avoid overfitting and to tackle unbalanced datasets we consider the RMSE per class, and the fitness function is the multiplication of the exponential values of the multiple RMSEs per class, as follows:

$$\text{fit} = \prod_{c=1}^{m} exp\left(\sqrt{\frac{\sum_{i=1}^{n_c}(o_i - t_i)^2}{n_c}}\right),$$

where m is the number of classes of the problem, n_c is the number of instances of the problem that belong to class c, o_i is the confidence value predicted by the evolved network, and t_i is the target value. The rationale behind the use of the exponential function is related to the fact that we aim to ensure that higher errors are more penalised than lower ones, to avoid the classification function from being too constrained to the error in one of the classes, failing to learn the overall problem.

Other metrics such as the accuracy or f-measure could have been used. However such type of properties just take into account the networks output, and therefore we would easily end with multiple individuals having the same quality, which would make evolution more difficult.

IV. Experimental Results

To validate our approach, NN-SGE is used to evolve both the topology and parameters of ANNs in 4 classification problems. The results are analysed and compared with others obtained using GE and GE-based approaches designed specifically for the evolution of ANNs [14]–[16]. We also compare NN-SGE with hand-designed ANNs, trained with the backpropagation (BP) learning algorithm.

A. Datasets

We selected 4 binary classification problems from the UCI Machine Learning repository [19]. All problems are binary due to the limitations imposed by the grammatical formulations, which do not allow the generation of dynamic productions, so that neurons can be reused. The problems have an increasing complexity in terms of the classification task that is to be performed. In the next paragraphs we present a brief description of the used datasets.

Flame [20] – This dataset contains artificial generated data for clustering purposes. It has 240 instances with two attributes each that are to be separated into two different classes, the first one containing 87 instances and the second one with 153 instances.

Wisconsin Breast Cancer Detection (WDBC) [21] – The WDBC is comprised of 30 features extracted from digitalised images of breast masses. The dataset has 569 instances, where 212 are malign and 357 are benign.

Ionosphere [22] – This benchmark is used for the classification of ionosphere radar returns, where the returns are classified into two different classes: good (225 instances) if it returns evidences of structure; and bad (126 instances) otherwise. 34 features are provided.

Sonar [23] – The sonar dataset contains 60 properties of sonar signals that allow a classification model to separate between signals that are bounced off a metal cylinder (111 instances) or a rock cylinder (97 instances).

B. Grammar

The grammar used to evolve ANNs is depicted in Figure 1 and is based on the ones used in [14]–[16]. The grammar allows the evolution of both the topology and parameters of one hidden-layer ANNs. In simple words, it is capable of

```
<sigexpr> ::= <node>
            |   <node> + <sigexpr>

<node> ::= <weight> * sig(<sum> + <bias>)

<sum> ::= <weight> * <features>
        |   <sum> + <sum>

<features> ::= x_1
             |   ...
             |   x_n

<weight> ::= <number>

<bias> ::= <number>

<number> ::= <digit>.<digit><digit>
           |   -<digit>.<digit><digit>

<digit> ::= 0 | 1 | 2 | 3 | 4
          |   5 | 6 | 7 | 8 | 9
```

Fig. 1. BNF grammar used for the conducted experiments. n represents the number of features of the problem.

representing a series of neurons (*<sigexpr>*) as well as their connections to the input neurons. Each neuron is represented by *<node>* and is no more than a weight multiplied by the result of the activation function, which takes as input a weighted sum of the multiple input nodes: *<features>*, where n denotes the number of attributes of the problem. All neurons use the sigmoid function (sig) as activation function, and therefore the output of the ANNs is sig(*<sigexpr>*). By encoding connections, the evolved networks are not fully connected and feature selection is performed. An example of an ANN that can be built using the detailed grammar is: $-9.99 * sig(-6.93 * x8 + 9.93 * x1 + 9.40 * x10 + 6.56 * x12 + -1.99) + -6.98 * sig(8.99 * x20 + -4.69 * x32 + -4.68) + 9.99 * sig(1.84 * x16 + -7.60 * x47 + 0.99)$. The former example represents an ANN generated for the sonar dataset and is graphically represented in Figure 4. The network is composed by three hidden-neurons, where each neuron is connected to 4, 2 and 2 features, respectively.

C. Experimental Setup

Table I details the experimental parameters for NN-SGE and GE. To make the comparison as fair as possible, we apply just one mutation to 95% of the population individuals, instead of defining a per-gene mutation probability. By doing this we ensure that only one change occurs in each individual, making the two methodologies similar in terms of the way they explore their search space. Additionally, in both engines the mutation operator is only allowed to change the integers that are used in the genotype to phenotype mapping and populations are initialised at random.

All datasets were partitioned in the same way: 70% of each class instances are used for training and the remaining 30% for testing. Only the train data is used to assess the fitness of the individuals; thus, the test data is kept aside from the evolutionary process and used exclusively for validation purposes, i.e., to evaluate the behaviour of the networks in the classification of instances that have not been seen during the creation of the ANN. No pre-processing or data augmentation methodologies were applied to the datasets, i.e., the datasets are used as they were obtained.

TABLE I
EXPERIMENTAL PARAMETERS.

Parameter	**Value**
Number of runs	30
Population size	100
Number of generations	500
Total number of evaluations	50000
Crossover rate	95%
Mutation rate	1 mutation in 95% of the individuals
Tournament size	3
Elite size	1%
SGE Parameter	**Value**
Recursion level	6
GE Parameter	**Value**
Individual size	200
Wrapping	0
Dataset Parameter	**Value**
Training percentage	70%
Testing percentage	30%

D. Evolving Artificial Neural Networks

To evaluate the ability of NN-SGE to evolve models suited for the classification of the used datasets we focus our attention on the analysis of several network properties, namely: (i) fitness; (ii) RMSE; (iii) accuracy; (iv) Area Under the ROC Curve (AUROC); (v) f-measure (vi) number of neurons; and (vii) number of used features. All properties except the fitness are analysed in the train and test sets. For a theoretical explanation of each of the used metrics refer to [24].

Evolutionary Results

Figure 2 depicts the evolution of the fitness of the best individuals across 50000 evaluations (100 individuals during 500 generations) for all the datasets described in Section IV-A. Each plot shows the comparative evolution of NN-SGE with GE and the results are averages of 30 independent runs. NN-SGE consistently presents mean fitness values below GE for all datasets. Remind that the goal of evolution is to minimise the per class exponential RMSE value and thus lower values mean better ANN performances.

An one-by-one analysis of the plots shows that the differences between NN-SGE and GE are greater in the flame and ionosphere datasets than in the WDBC and sonar ones. In the flame and ionosphere datasets GE converges faster (around 30000 and 20000 evaluations, respectively) and the evolution of new ANNs stagnates. On the contrary, NN-SGE takes around 40000 evaluations to converge in the ionosphere dataset and it seems to still be evolving in the flame one. This is explained by the dimensionality of the search spaces: flame only has 2 attributes and thus the domain is flatter, making it less likely for the algorithms to be trapped in a local optimum; on the other hand, the ionosphere dataset has

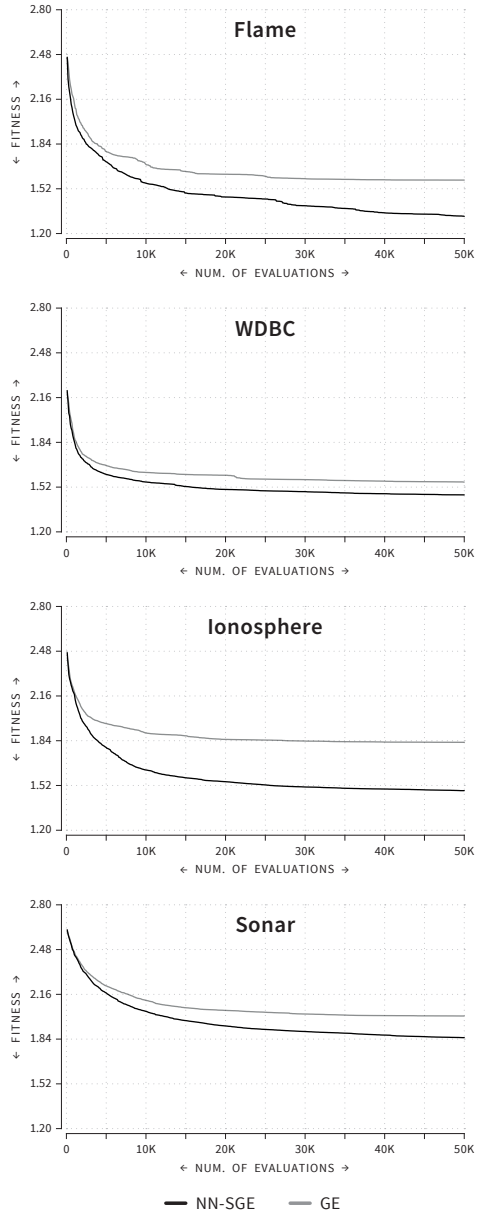

Fig. 2. Evolution of the fitness of the best individuals across 50000 evaluations for the flame, WDBC, ionosphere and sonar datasets. The results are averages of 30 independent runs.

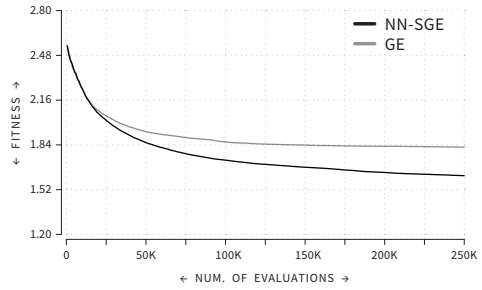

Fig. 3. Evolution of the fitness of the best individuals across 250000 evaluations for the sonar dataset. Results are averages of 30 independent runs.

34 features and thus a larger search space. Regarding the number of evaluations that are needed in each problem for the different methodologies to converge, NN-SGE because of its higher locality and lower redundancy is able to explore the search space more efficiently, and therefore it takes longer to converge due to the number of feasible solutions it encounters.

In WDBC the NN-SGE approach is able to obtain better results in terms of means, but the differences are not statistically significant (see below). In WDBC the average quality of the best individuals of the initial population is close to 2.2, whilst in the remaining problems is around 2.5. As such, the degree for improvement is much smaller, and with the number of evaluations given NN-SGE and GE end having similar results. In the sonar results we have the opposite. The problem is much more difficult (perceptible from the higher fitness of the initial population) because of the greater number of attributes (60), which leads to a larger search space. For that reason, we performed this experiment again, with a larger number of evaluations. In concrete, we increased the maximum number of evaluations by 5, i.e., 250000 (500 individuals over 500 generations). The results with the larger number of evaluations are depicted in Figure 3. The difference becomes larger and the results are better than the ones previously achieved.

So far, the results presented focus only on the analysis of the training set and therefore we have not yet analysed the generalisation ability of NN-SGE, i.e., the ability to evolve models that also perform well on unseen data. Table II reports the RMSE, accuracy, AUROC and f-measure of the best networks for the train and test sets. We complement this information with the fitness of the best individuals (only measured in the training set), number of neurons and number of features that are used. Each cell is formatted as follows: mean $\pm$ standard deviation; bold values indicate the methodology that obtained the best results for that specific dataset. Looking at the results depicted in the table it is possible to see that NN-SGE is consistently superior to GE. The NN-SGE fitness and RMSE values are lower than the ones attained by GE and the accuracy, AUROC and f-measure values are higher in the experiments performed with NN-SGE. This behaviour is consistent between the training and testing sets. Moreover, standard deviation values are lower in NN-SGE which indicates that it consistently finds good results.

Comparing the train and test results in GE the differences between the train and test sets are, on average, 0.06, 0.05, 0.05 and 0.05 for the RMSE, accuracy, AUROC and f-measure, respectively. For NN-SGE the differences are, on average, 0.08, 0.06, 0.05, 0.06 for the RMSE, accuracy, AUROC and f-measure, respectively. Despite the fact that the differences in NN-SGE are slightly superior to the ones found in GE for the analysed performance metrics, it is our perception that this is not an indicator of overfitting, but rather a result of the NN-SGE superior results. This is also supported by the networks complexity. NN-SGE is capable of finding and optimising the weights and bias of topologies with a greater number of neurons and that use a larger number of problem features, proving that NN-SGE performs a better exploration

TABLE II

EXPERIMENTAL RESULTS: FITNESS, RMSE, ACCURACY, AUROC, F-MEASURE, NUMBER OF NEURONS AND NUMBER OF USED FEATURES. RESULTS ARE BASED ON THE 30 BEST NETWORKS IN TERMS OF FITNESS, ONE FROM EACH INDEPENDENT RUN.

		Fitness	RMSE		Accuracy		AUROC		F-measure		Neurons	Features
			Train	Test	Train	Test	Train	Test	Train	Test		
Flame	NN-SGE	**1.32 ± 0.25**	**0.16 ± 0.13**	**0.22 ± 0.13**	**0.96 ± 0.08**	**0.93 ± 0.09**	**0.98 ± 0.04**	**0.96 ± 0.08**	**0.96 ± 0.08**	**0.94 ± 0.09**	4.87 ± 1.83	2.00 ± 0.00
	GE	1.58 ± 0.36	0.28 ± 0.15	0.31 ± 0.15	0.90 ± 0.10	0.88 ± 0.11	0.96 ± 0.05	0.93 ± 0.08	0.91 ± 0.09	0.89 ± 0.09	3.33 ± 1.40	1.97 ± 0.18
WDBC	NN-SGE	**1.46 ± 0.08**	**0.19 ± 0.03**	**0.23 ± 0.04**	**0.95 ± 0.02**	**0.93 ± 0.02**	**0.99 ± 0.01**	**0.98 ± 0.02**	**0.93 ± 0.03**	**0.91 ± 0.03**	3.73 ± 1.53	12.0 ± 6.51
	GE	1.55 ± 0.18	0.24 ± 0.09	0.27 ± 0.09	0.92 ± 0.12	0.90 ± 0.12	0.98 ± 0.02	0.97 ± 0.03	0.88 ± 0.18	0.86 ± 0.18	3.13 ± 1.53	8.40 ± 3.81
Ionosphere	NN-SGE	**1.48 ± 0.18**	**0.21 ± 0.07**	**0.32 ± 0.05**	**0.93 ± 0.11**	**0.87 ± 0.10**	**0.94 ± 0.04**	**0.90 ± 0.05**	**0.93 ± 0.18**	**0.89 ± 0.17**	3.53 ± 1.36	12.1 ± 5.79
	GE	1.82 ± 0.28	0.33 ± 0.10	0.38 ± 0.07	0.79 ± 0.20	0.76 ± 0.18	0.86 ± 0.18	0.83 ± 0.09	0.78 ± 0.32	0.76 ± 0.31	2.50 ± 1.41	7.33 ± 5.33
Sonar 50k	NN-SGE	**1.85 ± 0.18**	**0.34 ± 0.06**	**0.44 ± 0.04**	**0.84 ± 0.12**	**0.73 ± 0.09**	**0.91 ± 0.04**	**0.82 ± 0.05**	**0.78 ± 0.23**	**0.64 ± 0.20**	3.07 ± 1.39	13.3 ± 6.42
	GE	2.01 ± 0.23	0.38 ± 0.08	0.45 ± 0.06	0.76 ± 0.16	0.68 ± 0.12	0.88 ± 0.06	0.81± 0.05	0.70 ± 0.29	0.61 ± 0.25	2.53 ± 1.20	9.40 ± 5.73
Sonar 250k	NN-SGE	**1.62 ± 0.16**	**0.27 ± 0.04**	**0.42 ± 0.05**	**0.92 ± 0.04**	**0.78 ± 0.05**	**0.93 ± 0.04**	**0.84 ± 0.05**	**0.90 ± 0.05**	**0.74 ± 0.07**	4.23 ± 1.33	21.93 ± 0.53
	GE	1.82 ± 0.16	0.33 ± 0.06	0.45 ± 0.04	0.86 ± 0.08	0.73 ± 0.06	0.91 ± 0.04	0.81 ± 0.05	0.82 ± 0.16	0.68 ± 0.14	3.93 ± 1.78	11.37 ± 4.45

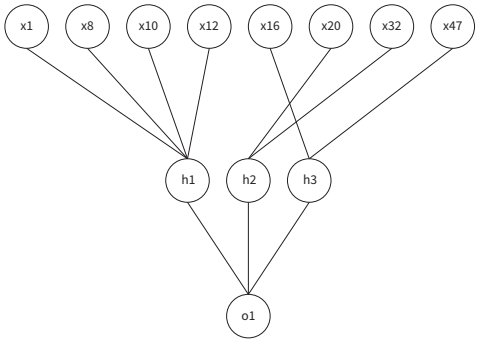

Fig. 4. Example of one of the best performing ANNs (in terms of fitness) evolved using NN-SGE for the classification of the sonar dataset. Weights and bias values are omitted for simplicity reasons. x, h and o represent input, hidden and ouput neurons, respectively.

TABLE III

GRAPHICAL OVERVIEW OF THE STATISTICAL RESULTS. SEE TEXT.

		Flame	WDBC	Ionosphere	Sonar 50k	Sonar 250k
Fitness		+++	∼	+++	++	+++
RMSE	Train	++	∼	+++	++	+++
	Test	∼	∼	+++	∼	++
Accuracy	Train	++	∼	+++	++	+++
	Test	++	∼	++	∼	++
AUROC	Train	++	∼	+++	++	++
	Test	++	∼	+++	∼	++
F-measure	Train	+++	∼	+++	∼	+++
	Test	++	∼	+++	∼	++

of the problems domain reaching solutions that perform better. An example of one of the best evolved networks for the sonar dataset is depicted in Figure 4.

Concerning the comparison with state-of-the-art GGP approaches to evolve ANNs, NN-SGE is able to evolve solutions that surpass the ones achieved by previous works. Tsoulos et al. [14] report an accuracy of 0.9544 in the WDBC dataset and 0.9034 in the ionosphere. The results are incomplete, since they only show the results attained by the best networks. NN-SGE mean accuracy values for WDBC and ionosphere are 0.93 and 0.87, respectively. However, the best found networks have test accuracies up to 0.97 and 0.96 for the WDBC and ionosphere, respectively. In [15], Soltanian et al. report an average test accuracy of 0.899 in the ionosphere benchmark. Despite lower than NN-SGE, the authors use backpropagation to finetune the weights of the evolved networks and longer experiments (in terms of needed evaluations). Later, we show that by fine tuning the best evolved networks of each run similar results are achieved. More recently, Ahmadizar et al. [16] combined a GA with GE, obtaining a RMSE of 0.4398 and an accuracy of 0.7201 in the sonar benchmark and 0.8694 in the ionosphere dataset. The proposed evolutionary approach takes into account the generalisation ability of the evolved networks, by measuring the classification accuracy on the testing set, which is considered in the fitness function. By doing so, the evolutionary engine is provided with all the dataset information, and no data is kept aside from the evolutionary process. Thus, this results should be compared with our training ones, which are superior. Additionally, we also use less computational resources in the training phase than previous approaches, since we perform less evaluations. Tsoulos et al. report having used 1 million evaluations (500 individuals, 2000 generations), Soltaian et al. used from 92000 to 250000 evaluations, and Ahmadizar et al. used 250000 evaluation (500 individuals, 500 generations).

Statistical Analysis

To verify if the differences between the approaches are meaningful we perform a statistical analysis. We start by checking whether the samples follow a Normal Distribution using the Kolmogoro-Smirnov and Shapiro-Wilk tests, with a significance level $\alpha = 0.05$. The tests revealed that we cannot say that NN-SGE does not follow a Normal Distribution. However, for GE the test revealed that it does not follow a normal distribution. Based on the results of these tests, we will assume that our data does not follow any distribution, and will use non-parametric tests to perform the pairwise comparison for each one of the recorded metrics.

We used the MannWhitney U test with the same level of significance $\alpha = 0.05$. Table III uses a graphical overview to present the results of the statistical analysis: $\sim$ indicates no statistical difference between NN-SGE and GE and $+$ signals that NN-SGE is statistically better than GE. The effect size is denoted by the number of $+$ signals, where $+$, $++$ and $+++$ correspond respectively to low ($0.1 \leq$ r < 0.3), medium ($0.3 \leq$ r < 0.5) and large (r ≥ 0.5) effect sizes. A $-$ signals scenarios where the NN-SGE is worse than GE.

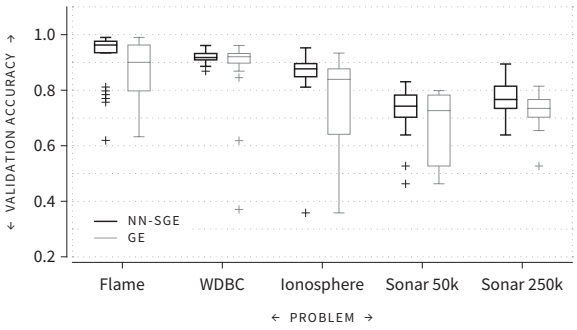

Fig. 5. Box plots of the test accuracies for all used datasets.

Looking at the results, it is possible to see that NN-SGE is never worse than GE. Moreover, the approaches are equivalent in 15 occasions: on the test RMSE of the flame dataset, for all comparisons on the WDBC, and on the test and f-measure results of the sonar 50k dataset. For all the other comparisons NN-SGE outperforms GE, and the effect size is medium in 16 occasions, and large in 14 occasions. These results confirm the viability of the proposed approach. Figure 5 depicts the test accuracies for the best individuals of each approach, for all the problems considered.

Analysing the Evolved Networks

To better understand the impact of evolving the weights and topology of an ANN we fine-tune the networks built by SGE using backpropagation (BP) in two different scenarios: (i) considering the evolved topology and the weights as the initial weights for the BP algorithm (BP NN-SGE); and (ii) considering the topology but not the evolved weights (BP NN-SGE (top.)). In addition, we also compare the obtained networks with the results obtained by training Fully Connected Neural Networks (FCNNs) with one hidden-layer and number of nodes equal to the ones used by the best networks evolved using NN-SGE. The results are depicted in Table IV and are averages of 30 networks, one for each evolutionary run. For convenience, the first row (NN-SGE) shows the results prior to the application of the BP algorithm (copied from Table II). The BP algorithm is applied until convergence, up to a maximum of 1000 epochs, with the default parameters of pybrain [25]: (i) learning rate (lr): 0.01; (ii) lr decay: 1; and (iii) no momentum.

Focusing on the results obtained by applying BP to the evolved networks (BP NN-SGE and BP NN-SGE (top.)) it is clear that the evolution of both the topology and weights is important for achieving better results. If we do not consider the weights the performance of the networks is inferior to the ones that used the evolved weights. This difference is confirmed by the statistics, which show that BP NN-SGE is always statistically superior to BP NN-SGE (top.) ($\alpha = 0.05$). Moreover, the advantage of evolving the weights is also noticeable in the number of training epochs: when the evolved weights are used, the number of BP epochs that are needed for the learning algorithm to converge is lower than when weights are initialised at random. This behaviour is observable in all the conducted experiments.

The results obtained by the handcrafted FCNNs are consistently worse than those obtained by BP NN-SGE. BP NN-SGE is statistically superior to the handcrafted 29 times (marked with two asterisks) and there is no statistical difference in the remaining cases. These results prove that evolution is helpful and facilities the process of finding effective networks to solve the considered problems.

Comparing BP NN-SGE with the baseline (NN-SGE) it is noticeable that in the first two datasets (flame and WDBC) the results obtained without fine-tune are slightly superior to those resulting from further training using BP. After a careful examination we conclude that this is a result of the implementation of the BP algorithm that splits the training data into two disjoint sets. Therefore it is not guided taking into account all data instances, and because the flame and WDBC problems have lower complexity when comparing with the ionosphere and sonar, evolution is able of attaining a near-perfect tuning of the weights.

V. Conclusions and Future Work

In this paper we propose a novel methodology for the evolution of the topologies and parameters of ANNs for multiple classification tasks. The proposed approach is based on SGE, where the mutation operator was changed to consider the different production rules under different probabilities. These probabilities are based on the number of integers for that production rule that are used in the genotype to phenotype mapping. The fitness aims at minimising the RMSE in the classification task at hand; however, in order to avoid overfitting, we consider the RMSE per class, which is combined with the exponential function. By doing so, low errors have less impact than greater ones.

NN-SGE is tested in four classification datasets, with an increasing complexity: flame, WDBC, ionosphere and sonar. Results show that NN-SGE is able to create more effective ANNs than GE: the most similar Grammar-based GP approach. The performance of the evolved ANNs is better in terms of fitness, RMSE, accuracy, AUROC and f-measure in the train and test sets, which proves that NN-SGE is capable of evolving consistent ANNs, that perform well beyond the training data. Notwithstanding, we performed a statistical analysis to assess the significance of the results. The analysis revealed that NN-SGE is consistently statistically superior when compared with GE. We also compared the best NN-SGE evolved networks with hand-designed ones, and the results showed that NN-SGE is consistently statistically superior to them.

Next steps to expand on this work will focus primarily on the study of the quality and impact of the feature selection in the obtained results and on the generalisation of NN-SGE to the evolution of ANNs with more than one hidden-layer. Therefore, we will compare the selected features with those selected by well established feature selection approaches (e.g., correlation feature selection) and investigate methodologies for

TABLE IV

COMPARISON BETWEEN THE BEST NN-SGE NETWORKS AND FULLY CONNECTED NEURAL NETWORKS (FCNNS). * MEANS THAT BP NN-SGE IS STATISTICALLY SUPERIOR TO BP NN-SGE (TOP.) AND ** THAT IT IS STATISTICALLY SUPERIOR TO BP NN-SGE (TOP.) AND FCNN. SEE TEXT.

		RMSE		Accuracy		AUROC		F-measure		Epochs
		Train	Test	Train	Test	Train	Test	Train	Test	
Flame	NN-SGE	0.16 ± 0.13	0.22 ± 0.13	0.96 ± 0.08	0.93± 0.09	0.98 ± 0.04	0.96 ± 0.08	0.96 ± 0.08	0.94 ± 0.09	-
	BP NN-SGE	**0.19 ± 0.14****	**0.23 ± 0.14****	**0.93 ± 0.13****	**0.91 ± 0.14****	**0.94 ± 0.13****	**0.92 ± 0.16****	**0.94 ± 0.10****	**0.93 ± 0.13****	558 ± 482
	BP NN-SGE (top.)	0.43 ± 0.08	0.43 ± 0.08	0.70 ± 0.11	0.69 ± 0.11	0.63 ± 0.27	0.62 ± 0.25	0.81 ± 0.06	0.80 ± 0.06	633 ± 438
	FCNN	0.40 ± 0.08	0.40 ± 0.08	0.74 ± 0.11	0.75 ± 0.11	0.68 ± 0.30	0.71 ± 0.28	0.81 ± 0.08	0.81 ± 0.08	749 ± 406
WDBC	NN-SGE	0.19 ± 0.03	0.23 ± 0.04	0.95 ± 0.02	0.93 ± 0.02	0.99 ± 0.01	0.98 ± 0.02	0.93 ± 0.03	0.91 ± 0.03	-
	BP NN-SGE	**0.26 ± 0.07****	**0.28 ± 0.06****	**0.90 ± 0.08****	**0.88 ± 0.08****	**0.95 ± 0.09****	**0.94 ± 0.09****	**0.82 ± 0.22****	**0.80 ± 0.23****	343 ± 425
	BP NN-SGE (top.)	0.36 ± 0.09	0.36 ± 0.09	0.79 ± 0.12	0.78 ± 0.12	0.85 ± 0.19	0.84 ± 0.19	0.55 ± 0.40	0.54 ± 0.39	697 ± 408
	FCNN	0.46 ± 0.06	0.46 ± 0.06	0.66 ± 0.09	0.66 ± 0.09	0.56 ± 0.14	0.56 ± 0.14	0.11 ± 0.27	0.11 ± 0.27	657 ± 358
Ionosphere	NN-SGE	0.21 ± 0.07	0.32 ± 0.05	0.93 ± 0.11	0.87 ± 0.10	0.94 ± 0.04	0.90 ± 0.05	0.93 ± 0.18	0.89 ± 0.17	-
	BP NN-SGE	**0.20 ± 0.05****	**0.31 ± 0.05**	**0.95 ± 0.03****	**0.89 ± 0.03***	0.94 ± 0.04*	0.90 ± 0.05*	**0.97 ± 0.02****	**0.91 ± 0.02***	259 ± 307
	BP NN-SGE (top.)	0.31 ± 0.06	0.35 ± 0.06	0.88 ± 0.08	0.83 ± 0.08	0.87 ± 0.14	0.84 ± 0.16	0.91 ± 0.05	0.89 ± 0.05	863 ± 274
	FCNN	0.23 ± 0.03	0.31 ± 0.04	0.94 ± 0.02	0.88 ± 0.03	**0.96 ± 0.02**	**0.91 ± 0.05**	0.95 ± 0.01	0.91 ± 0.02	818 ± 233
Sonar 50k	NN-SGE	0.34 ± 0.06	0.44 ± 0.04	0.84 ± 0.12	0.73 ± 0.09	0.91 ± 0.04	0.82 ± 0.05	0.78 ± 0.23	0.64 ± 0.20	-
	BP NN-SGE	**0.33 ± 0.06****	**0.43 ± 0.04***	**0.84 ± 0.12****	**0.74 ± 0.09***	**0.91 ± 0.05****	**0.82 ± 0.05***	**0.79 ± 0.23****	**0.67 ± 0.19***	99 ± 160
	BP NN-SGE (top.)	0.48 ± 0.04	0.49 ± 0.03	0.61 ± 0.11	0.59 ± 0.10	0.67 ± 0.15	0.63 ± 0.14	0.42 ± 0.32	0.40 ± 0.31	519 ± 477
	FCNN	0.39± 0.06	0.43 ± 0.04	0.77 ± 0.12	0.70 ± 0.11	0.84 ± 0.13	0.78 ± 0.11	0.70 ± 0.25	0.63 ± 0.22	775 ± 391
Sonar 250k	NN-SGE	0.27 ± 0.04	0.42 ± 0.05	0.92 ± 0.04	0.78 ± 0.05	0.93 ± 0.04	0.84 ± 0.05	0.90 ± 0.05	0.74 ± 0.07	-
	BP NN-SGE	**0.26 ± 0.06****	**0.41 ± 0.05***	**0.92 ± 0.04****	**0.78 ± 0.05****	**0.93 ± 0.03****	**0.85 ± 0.05****	**0.91 ± 0.04****	**0.75 ± 0.07****	118 ± 223
	BP NN-SGE (top.)	0.45 ± 0.05	0.47 ± 0.04	0.66 ± 0.14	0.62 ± 0.12	0.70 ± 0.20	0.67 ± 0.18	0.50 ± 0.31	0.46 ± 0.28	624 ± 472
	FCNN	0.40 ± 0.07	0.44 ± 0.05	0.74 ± 0.16	0.69 ± 0.13	0.79 ± 0.18	0.76 ± 0.16	0.72 ± 0.16	0.67 ± 0.11	654 ± 425

updating the grammatical production rule that defines the neurons in the previous layer (<*features*>). Later, the approach will be tested on the evolution of different ANN types, such as Convolutional Neural Networks or AutoEncoders.

ACKNOWLEDGMENTS

This research is partially funded by: Fundação para a Ciência e Tecnologia (FCT), Portugal, under the grant SFRH/BD/114865/2016. We would also like to thank Tiago Martins for all the patience making the charts herein presented.

REFERENCES

[1] S. S. Tirumala, "Implementation of evolutionary algorithms for deep architectures," in *Proceedings of the 2nd International Workshop on Artificial Intelligence and Cognition (AIC)*, 2014, pp. 164–171.

[2] C. Ryan, J. Collins, and M. O'Neill, *Grammatical evolution: Evolving programs for an arbitrary language*. Berlin, Heidelberg: Springer Berlin Heidelberg, 1998, pp. 83–96.

[3] C. Ryan, M. Keijzer, and M. Nicolau, "On the avoidance of fruitless wraps in grammatical evolution," in *Genetic and Evolutionary Computation Conf.* Springer, 2003, pp. 1752–1763.

[4] N. Lourenço, F. B. Pereira, and E. Costa, "Unveiling the properties of structured grammatical evolution," *Genetic Programming and Evolvable Machines*, pp. 1–39, 2016.

[5] X. Yao, "Evolving artificial neural networks," *Proceedings of the IEEE*, vol. 87, no. 9, pp. 1423–1447, 1999.

[6] R. S. Sutton, "Two problems with backpropagation and other steepest-descent learning procedures for networks," in *Proc. 8th annual conf. cognitive science society*, 1986, pp. 823–831.

[7] D. Whitley, T. Starkweather, and C. Bogart, "Genetic algorithms and neural networks: Optimizing connections and connectivity," *Parallel computing*, vol. 14, no. 3, pp. 347–361, 1990.

[8] F. Gomez, J. Schmidhuber, and R. Miikkulainen, "Accelerated neural evolution through cooperatively coevolved synapses," *Journal of Machine Learning Research*, vol. 9, no. May, pp. 937–965, 2008.

[9] L.-O. Fedorovici, R.-E. Precup, F. Dragan, and C. Purcaru, "Evolutionary optimization-based training of convolutional neural networks for OCR applications," in *System Theory, Control and Computing (ICSTCC), 2013 17th International Conf.* IEEE, 2013, pp. 207–212.

[10] O. E. David and I. Greental, "Genetic algorithms for evolving deep neural networks," in *Proceedings of the 2014 Conf. companion on Genetic and evolutionary computation companion*. ACM, 2014, pp. 1451–1452.

[11] M. Rocha, P. Cortez, and J. Neves, "Evolution of neural networks for classification and regression," *Neurocomputing*, vol. 70, no. 16, pp. 2809–2816, 2007.

[12] K. O. Stanley and R. Miikkulainen, "Evolving neural networks through augmenting topologies," *Evolutionary computation*, vol. 10, no. 2, pp. 99–127, 2002.

[13] F. Gruau, "Genetic synthesis of boolean neural networks with a cell rewriting developmental process," in *International Workshop on Combinations of Genetic Algorithms and Neural Networks, COGANN-92*. IEEE, 1992, pp. 55–74.

[14] I. Tsoulos, D. Gavrilis, and E. Glavas, "Neural network construction and training using grammatical evolution," *Neurocomputing*, vol. 72, no. 1, pp. 269–277, 2008.

[15] K. Soltanian, F. A. Tab, F. A. Zar, and I. Tsoulos, "Artificial neural networks generation using grammatical evolution," in *21st Iranian Conf. on Electrical Engineering (ICEE)*. IEEE, 2013, pp. 1–5.

[16] F. Ahmadizar, K. Soltanian, F. AkhlaghianTab, and I. Tsoulos, "Artificial neural network development by means of a novel combination of grammatical evolution and genetic algorithm," *Engineering Applications of Artificial Intelligence*, vol. 39, pp. 1–13, 2015.

[17] T. Si, A. De, and A. K. Bhattacharjee, "Grammatical swarm for artificial neural network training," in *International Conf. on Circuit, Power and Computing Technologies (ICCPCT)*. IEEE, 2014, pp. 1657–1661.

[18] J.-Y. Jung and J. A. Reggia, "Evolutionary design of neural network architectures using a descriptive encoding language," *IEEE transactions on evolutionary computation*, vol. 10, no. 6, pp. 676–688, 2006.

[19] M. Lichman, "UCI machine learning repository," 2013. [Online]. Available: http://archive.ics.uci.edu/ml

[20] L. Fu and E. Medico, "FLAME, a novel fuzzy clustering method for the analysis of dna microarray data," *BMC bioinformatics*, vol. 8, no. 1, p. 1, 2007.

[21] W. N. Street, W. H. Wolberg, and O. L. Mangasarian, "Nuclear feature extraction for breast tumor diagnosis," in *IS&T/SPIE's Symposium on Electronic Imaging: Science and Technology*. International Society for Optics and Photonics, 1993, pp. 861–870.

[22] V. G. Sigillito, S. P. Wing, L. V. Hutton, and K. B. Baker, "Classification of radar returns from the ionosphere using neural networks," *Johns Hopkins APL Technical Digest*, vol. 10, no. 3, pp. 262–266, 1989.

[23] R. P. Gorman and T. J. Sejnowski, "Analysis of hidden units in a layered network trained to classify sonar targets," *Neural networks*, vol. 1, no. 1, pp. 75–89, 1988.

[24] M. Bekkar, H. K. Djemaa, and T. A. Alitouche, "Evaluation measures for modelsassessment over imbalanced datasets," *Iournal Of Information Engineering and Applications*, vol. 3, no. 10, 2013.

[25] S. Tom, B. Justin, W. Daan, Y. Sun, F. Martin, S. Frank, R. Thomas, and S. Jürgen, "Pybrain," *Journal of Machine Learning Research*, vol. 11, pp. 743–746, 2010.