Statistically-driven Coral Reef metaheuristic for automatic hyperparameter setting and architecture design of Convolutional Neural Networks

Alejandro Martín*, Raúl Lara-Cabrera*, Víctor Manuel Vargas†,
Pedro Antonio Gutiérrez†, César Hervás-Martínez†, David Camacho*
*Universidad Politécnica de Madrid, Madrid, Spain
Email: {alejandro.martin,raul.lara,david.camacho}@upm.es
†Universidad de Córdoba, Córdoba, Spain
Email: {vvargas,pagutierrez,chervas}@uco.es

***Abstract*—The adjustment of the hyperparameters and network structure of Convolutional Neural Networks (CNNs) composes an important step towards building effective, but still efficient learning models. The selection of the best configuration is a problem-dependent task that involves to explore an enormous and complex search space. Due to this reason, the use of heuristic-based search fits perfectly within this task, seeking to obtain a near to optimal solution in a complex and large exploratory space. This paper presents SCRODeep, a self-adapting algorithm based on a statistically-driven Coral Reef Optimisation algorithm (SCRO), for the selection of the most adequate CNNs architecture in a particular domain. This metaheuristic has been designed to navigate through a search space where the architecture (defining the particular set of layers, including convolutional or pooling layers), and the hyperparameters of the network (i.e. activation functions, number of units or the kernel initializer, among others) are represented, but where the connections weights and bias are inferred using typical CNNs optimisation algorithms. In contrast to other approaches, where the use of a metaheuristic implies in turn to fix a series of hyperparameters (i.e. the mutation probability in a genetic algorithm), our approach follows a self-parametrisation perspective, thus removing the necessity of fixing these values. The method has been tested in the design of CNNs for image classification, showing that SCRODeep is able to find competitive solutions, while the complexity of the architectures found is constrained.**

***Index Terms*—Convolutional Neural Networks, Coral Reef based optimisation, architecture definition, optimisation**

I. Introduction

Recently, machine learning has taken the lead as the methodology to follow in order to solve complex problems, especially when dealing with large volumes of data. In the era of the information society, we are constantly generating data during our day-to-day activities. This situation has brought to light a large number of problems which, due to their difficulty, cannot be solved efficiently with specific algorithms. On the other hand, machine learning algorithms have proved to solve these problems efficiently.

Deep neural networks (DNNs) can be found within the machine learning ecosystem. They can be seen as an improvement of classical artificial neural networks, which exploits new paradigms of parallel and distributed computation as well as the current reduced cost of computation. Due to its features, DNNs are able to tackle complex problems, comprising thousands of features. In fact, DNN ability to adapt themselves to non-linear spaces and also their capabilities of building strong classification and prediction systems are within the sources of their popularity. However, the training process is more complex compared to other machine learning algorithms such as support vector machines or random forests. Another undesirable feature is that the models built by DNNs are not self-explanatory, so it is not possible to reason about the knowledge gathered. The same advantages and disadvantages can be associated with Convolutional Neural Networks (CNNs), architectures specialised in visual related tasks.

In general, the optimisation of DNNs or CNNs can be performed at three different levels: architecture, parameters (connection weights and bias) and hyperparameters, such as activation function or the batch size. This configures a large space of possibilities, particularly complex in the case of Convolutional Neural Networks. These models involve multi-layered architectures each with its own hyperparameters and parameters and a large number of neurons in the last layers. While the definition of the parameters (the connection weights) during the training process is typically performed using a backpropagation algorithm, the architecture and the hyperparameters are fixed by hand, either based on a trial and error process or based on previous experience. Several approaches have been followed to improve this process, many of them relying on evolutionary computation. However, new features and specifications, as well as an increased complexity, require from new specialised methods.

In swallow artificial neural networks, it is usually necessary to train and evaluate the network multiple times to find a suitable architecture and set of hyperparameters and to account for random initial weights. Currently, hyperparameter search is often conducted via rules of thumb and general guidelines, coupled with manual experiments [1], but it remains as a major challenge when working with CNNs and DNNs [2], [3]. Additionally, the optimal set of hyperparameter values changes during the training process (i.e. the learning rate

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

is reduced), which calls for a dynamic optimisation [4]. Therefore, the design of CNNs architectures can be expressed as a hyperparameter optimisation problem [5], [6], in which the number of layers, neurons or strides, among others, can be optimised using techniques such as evolutionary algorithms [2] or particle swarm optimisation [7], among others.

One of the biggest challenges to face lies in the long time necessary to evaluate a given CNN configuration, as it implies to train the whole model. This is especially the case of deep models with a potentially high number of filters on each convolutional layer. Therefore, the automatic design of CNN architectures has recently become a relevant research topic, exploring the use of evolutionary algorithms for the design of CNNs [8]. On the other hand, the performance of many contemporary machine learning algorithms depends crucially on the specific initialisation of the hyperparameters [6]. Learning algorithms in CNNs require the practitioner to manually set the values of many hyperparameters such as the general architecture, the learning rate, regularisation parameters, batch size, a fraction of units to drop in the dropout layers, the number of filters or the stride size in the convolutional layers, to name a few. In order to assess the performance of a particular configuration, the model featuring that configuration has to be trained and evaluated. However, in CNNs, the training of a model can take a considerable amount of time and the search space is often very high-dimensional.

As an example of the above, the architecture of the CNN model used in Simonyan, K. and Zisserman [9] is associated with filters with a very small receptive field: 3×3, but in one of the configurations they also use 1×1 convolution filters. The convolution stride is fixed to 1 pixel and the spatial padding is 1 pixel for 3×3 conv. layers. Spatial pooling is carried out by five max-pooling layers (performed over a 2×2 pixel window, with stride 2), but not all the convolutional layers are followed by max-pooling. The determination of these hyperparameters is tedious but finding a near optimal combination is crucial in order to reach high performance levels.

In this paper, we propose SCRODeep, a self-adaptive algorithm for hyperparameter optimisation of convolutional neural networks, but which can be easily extended in order to deal with other architectures such as Recurrent Neural Networks. SCRODeep is based on a statistically-driven coral reef optimisation algorithm [10], a individual codification which encodes the hyperparameters and network structure of a deep learning model, a set of specifically designed reproduction and mutation operators and finite state machine, in charge of establishing valid transitions between layers and that allow to build feasible learning models. The self-parametrisation ability of the SCRO metaheuristic allows to avoid the necessity of fixing additional hyperparameters.

Finally, this paper is organised in the following sections: Section II introduces the background of the problem while Section III presents SCRODeep. Section IV deals with the experimental setup and the obtained results. Finally, Section V enumerates the conclusions drawn and also potential future work lines.

II. Background

According to the first authors referring to the concept of Artificial neural networks (ANNs), they are a computational model vaguely inspired on biological neurons [11]. Since their origin, ANNs have been used to solve a wide range of different problems, including control systems [12], classification [13], face recognition [14] and finance [15]. Due to the progress on computational capabilities, there was a natural transition to larger number of neurons and more complex architectures in the 90s with the emergence of DNNs [16] and specific architectures such as CNNs

As with ANNs, this computing model is very popular in a wide range of scientific fields, mainly motivated by the good results this model is able to achieve as well as due the large number of frameworks and programming libraries that can be found to develop DNNs. Frameworks such as Tensorflow [17] and PyTorch [18], to name a few, have contributed to the rise of these models by making the task easier for developers and scientists. One of the fields that has benefited most from deep networks has been computer vision and image recognition [19], [20]. Within this scope, images pass through multiple layers that form a deep architecture in order to extract features from the image and classify them. Other examples of the dominance of deep networks include time-series forecasting [21], video recognition [22], malware detection and classification [23], and audio recognition [24].

Regardless of the domain in which they are used, one of the most critical moments when configuring and designing a deep neural network is to properly select the hyperparameters and architecture of the network. While this task is often performed by hand following a trial and error scheme, a plausible alternative is to use meta-heuristics and bio-inspired algorithms to make optimal parameter selection. For example, Xin Yao [25] studied the application of evolutionary computing for this purpose, coming to the conclusion that these techniques can be applied in the different phases that comprise the training of an artificial neural network: synaptic weights, architecture and learning rules.

Many approaches can be found in the literature when dealing with the tuning of both the architecture and parameters of classical neural networks [26]. For instance, the approach by [27] includes the definition of the architecture altogether with the adaptation of the weights. Moreover, the problem should be formulated so it can be solved by means of evolutionary algorithms, in which the number of neurons is included within the individual of the population [28]. Keeping with the evolutionary algorithms, [29] presented a hyper-heuristic approach based on evolutionary algorithms to adjust the polynomial type of the layers, the number of nodes on each, and the number of layers in the architecture. This type of methods has also been used to boost the performance [30] as well as including connection weights in the evolutionary search [31].

EvoDeep [32] is a method devoted to evolve the hyperparameters and the architecture of a Convolutional Neural Network in order to maximise its classification accuracy, as

well as maintaining a valid sequence of layers. Its core is an evolutionary algorithm whose population is composed of individuals that encode not only the hyperparameters of the CNN, but also the sequence of layers as well as their hyperparameters. Furthermore, the algorithm included a Finite-State Machine to ensure that all the architectures generated are valid sequences of layers, as they have restrictions on their inputs and outputs types. The experimental results showed that EvoDeep was able to build valid CNNs architectures that, in turn, achieved good accuracy when using a dataset of handwritten images. However, EvoDeep also requires the adjustment of different hyperparameters of the evolutionary algorithm (such as the mutation and crossover probabilities), which can difficult its application.

In this paper, we present a novel method for the optimisation of the hyperparameters and network structure of Convolutional Neural Networks by leveraging a statistically-driven CRO (SCRO) [10] algorithm. This approach includes a self-adaptive fine-tuning of the hyperparameters of the evolutionary process in order to automate the whole optimisation process.

III. SCRODeep: Convolutional Neural Networks parametrisation using a statistically-driven Coral Reef Optimisation algorithm

The strengths of Deep Learning (DL) models are offset by their inherent intricacy and variable architecture. While training a traditional machine learning algorithm, such as the popular Random Forest, is a simple and direct task, in which normally just the number of trees has to be tweaked, DL necessitates an architecture which typically relies on different particularities of the problem. For instance, DL applied to computer vision includes the specification of many layers in a stipulated order accountable for the assessment of multiple data alterations.

This architecture consists of many layers distributed to characterise the non-linear relationships that resolve a specific problem. Each layer has an undefined number of neurons, outputs, and distinct methods both for initialisation and activation. These hyperparameters must be set up prior to the training process of the neural network, thereby creating a large search space with the optimal combination of settings that are unknown and dependent on the problem.

There are various restrictions on the design of a multi-layer architecture. For instance, a specific layer's input form has to match the previous layer's output. This might occur when a layer expects an input vector (i.e. a fully connected layer) to restrict the preceding layers. In this instance, a reshape layer can not be left of a fully connected layer as it always delivers an output of at least 2 dimensions. The parametrization of each additional layer of the model, which must also fulfil certain properties, is another restriction to take into consideration.

Hyperparameters and architectures in neural networks form a broad search area in which many configurations are defined. Since maximising the network accuracy in the performance of a particular task is expected, the selection of the correct configuration can be seen as an optimisation process. An evolutionary algorithm was used in this paper to perform a meta-heuristic search in order to reach a configuration, which maximises precise classification.

A. General description of the SCRO algorithm

The SCRO metaheuristic [10] is a modification of the original Coral Reef Optimisation (CRO) approach [33], [34]. CRO simulates the process of coral reproduction (where sexual and asexual reproduction operators are considered) and the coral reef formation (where corals fight for space). These operators are executed in a loop until a stopping criterion is reached. The SCRO version of this algorithm includes a statistically-driven parametrisation in order to avoid the manual adjustment of the different hyperparameters involved in the evolutionary process.

The main steps of SCRO are included in Algorithm 1, which will be described in the following subsections. Moreover, the specific adaptations of SCRO for dealing with the configuration of CNN training are included in sections III-B and III-E.

Algorithm 1 Statistical coral reef optimisation algorithm

Input: Dataset.
Output: CNN.
1: Initialization of the algorithm.
2: **while not** Stop Condition **do**
3: Asexual reproduction.
4: Sexual reproduction (external and internal).
5: Larvae settlement.
6: Evaluate the new population (coral reef).
7: Predation process.
8: **end while**
9: **return** Best solution.

In SCRO, the main ingredient is that the algorithm analyses the fitness values in order to decide how the different operators are applied, dynamically adapting the behaviour to the state of the population and avoiding the adjustment of the hyperparameters in each optimisation problem. In this way, the quality of a solution is given by $f \in [0, 1]$. The fitness of all corals define the population state (potential solutions to the problem) in the reef, $\{f_{1j}, f_{2j}, \ldots, f_{Nj}\}$, where the population size is N_j corals during the j-th generation of the algorithm. This is assumed to be a random sample.

The main assumption is that the distribution of the fitness is approximately Gaussian. In this way, the variance of the population can be estimated as:

$$S^2_{f_j} = \frac{\sum_{i=1}^{N_j}(f_{ij} - \bar{f}_j)^2}{N_j - 1}, j = 1, \ldots, M, \quad (1)$$

where f_{ij} is the fitness of the i-th individual in the j-th generation, N_j is the number of individuals of the j-th generation, and $\bar{f} = \sum_{i=1}^{N_j} f_{ij}/N_j$ is the average fitness value of all the individuals of the generation.

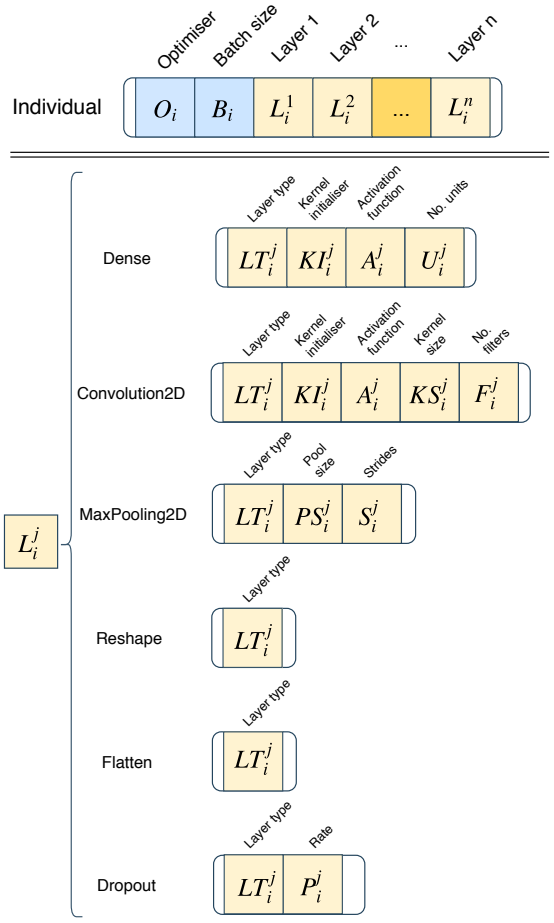

Fig. 1: Encoding of the individual, including the different hyperparameters and layer types.

B. Coral Encoding and fitness function

According to our approach, the corals must represent all the required hyperparameters to train a CNN (see Fig. 1). Hence, the individuals of SCRODeep are composed of an array of global hyperparameters that encode the network's common behaviour, and an arbitrary number of layers that make up the network's architecture. Each of the latter can be of a different kind, which also fixes its hyperparameters. In other words, each layer's hyperparameters are defined by its layer type (see Fig. 1).

Regarding the former, there is a global operator O_i that specifies which of the six optimisers is used. The global operator B_i represents how many samples the neural network receives at a time, updating its weights accordingly. Note that a dynamic stopping criterion has been applied, in a way that the training process stops if the best accuracy obtained is not improved over a fixed number of iterations.

The fitness function is calculated according to the final performance, in terms of accuracy in this research, of training the CNN model encoded by one specific individual.

C. Evolutionary operators

The evolutionary process performed by the SCRO metaheuristics makes use of different operators in order to evolve the population iteratively. Below there is a summary of these operations:

1) Initialisation of the algorithm: After generating N random solutions to the problem, those corals whose fitness verifies Equation 2 are deleted.

$$f_{i1} \notin (\bar{f}_1 - S_{f_1}, 1] \tag{2}$$

2) Asexual reproduction: Asexual reproduction is based on two consecutive processes: fragmentation (caused by external agents) and budding. For the former, the candidate corals are those whose fitness value satisfies:

$$f_{ij} \in (\bar{f}_j + S_{f_j}, 1]. \tag{3}$$

After that, a coral is randomly selected from the candidates and, finally, a mutation is applied to it.

3) Sexual reproduction: SCRO considers two types of sexual reproduction:

1) External sexual reproduction is similar to a crossover operator in an evolutionary algorithm. In SCRO, the algorithm uses those corals whose fitness verifies:

$$f_{ij} \in (\bar{f}_j - S_{f_j}, 1]. \tag{4}$$

2) Internal sexual reproduction, which applies a random mutation to the rest of corals, i.e.:

$$f_{ij} \in [0, \bar{f}_j - S_{f_j}]. \tag{5}$$

4) Settlement: The next stage is that the newly generated corals try to settle and grow in the reef. The process is based on trying to accommodate the coral in a random position of the reef. If the position is empty, the coral is automatically settled. If not, the coral is settled provided that its fitness is better. The process is repeated until two attempts.

5) Depredation: After settlement is made, those individuals whose fitness function verifies:

$$f_{ij} \in [0, \bar{f}_j - 2S_{f_j}], \tag{6}$$

are eliminated.

D. Operators to evolve CNN architectures

Due to the particularities that CNN architectures entail and due to the designed encoding of the individuals, it has been necessary to develop specific mutational operators which provide valid individuals who can train varied network models. The operators employed in EVODeep have been adapted in this case to be part of the SCRO algorithm used. It is noteworthy that the operators functions at two levels: global hyperparameters and layer levels. The reason is the encoding of the individuals, which can be made of a varied number of layers with different hyperparameters each, depending on their type. Following is the description of both the mutation and recombination operators.

1) Mutation: As it has been mentioned, mutation works at the two levels: global hyperparameters and layers. At the first, every individual's global hyperparameter changes according to a consistent approach, in which each value resets to a random value with probability $p = 0.5$ (provided the global probability of mutation is fulfilled). The new random value is created based on the range of values and the list of possible values in the case of numerical and categorical hyperparameters, respectively.

With respect to layers, the mutation performs as follows:

1) A random point of insertion between two consecutive layers is selected.
2) The operator inserts a valid sequence of n layers, with $n = \{1, 2, 3\}$ and uniformly selected at random, provided the length of the final sequence of layers does not exceed the maximum number of layers.
3) Every hyperparameter for each layer is mutated in a similar way as the global hyperparameters, that is, values are reset to a random value with a certain probability.

2) Recombination: The recombination, again, works at two levels: global and layers. At the global level, individuals are recombined following a uniform crossover. That is, hyperparameters are swapped pairwise with a probability of $p = 0.5$. On the other hand, layers are recombined following a cut-and-splice approach that works as follows: it randomly selects two points p_1 and p_2 that fulfil these conditions:

- $1 < p_1 < n$ and $1 < p_2 < m$, with n and m are the number of layers of each individual, so the first and last layer are always located in the correct place
- The sequence of layers is still valid after swapping parts
- The maximum number of layers is not exceeded after swapping

Finally, it is necessary to perform a recombination of the internal hyperparameters of the layers, as they are left untouched with the aforementioned mechanism. In this case, it applies a layer level crossover that works as follows: beginning with the first layer, two hyperparameters p_i and p'_j from two layers L_l and L'_l placed in the same position l are swapped until the penultimate layer of the shortest individual is reached. The last layer of each individual is also crossed, as they always share the same layer type.

E. Restricting the algorithm: building a valid sequences of layers

Similarly to EVODeep, all possible transitions between layers are modelled through a Finite-State Machine (FSM). When individuals are crossed and mutated, the FSM is used to avoid invalid structures or to incorporate new consistent layers. In addition, this FSM is also employed to generate all possible paths with a minimum and a maximum length, from which one is selected randomly in order to initialise the first population.

IV. Experiments

A series of experiments allow to test SCRODeep, proving its ability to automatically adjust both the hyperparameters and architecture of a Convolutional Neural Network model. Furthermore, the statistically-driven approach allows to leave the whole workflow process in hands of the algorithm. The following subsections describe the dataset used in the experiments, the experimental settings and the results obtained.

Parameter	Range of values
Kernel initializer KI_i	Uniform, LeCun uniform, Normal, Zero, Glorot normal, Glorot uniform, He normal, He uniform
Activation function A_i	ReLU, Softmax, Softplus, Softsign, Tanh, Sigmoid, Hard sigmoid, Linear
Units U_i	{10, 20, 30, ..., 500}
Rate (dropout) P_i	{0.1, 0.2, 0.3, ..., 0.8}
No. filters F_i	{5, 10, 15, ..., 50}
Kernel size KS_i	{3, 5, 7, ..., 15}
Pool size PS_i	{2, 3, 4, 5, 6}
Strides S_i	{2, 3, 4, 5, 6}
Optimizer O_i	Adam, SGD, RMSProp, Adagrad Adamax, Nadam
Batch size B_i	{100, 200, 300, ..., 1000}
No. layers N	{3, 4, 5, ..., 9}

TABLE I: Range of values for each hyperparameter involved in the heuristic search. Layer level and global hyperparameters are shown in the table.

A. Dataset

The well known MNIST dataset [35] was used to run the experiments. The reasons under this decision lie in the large number of methods proposed in the literature, which allow to make an objective comparison of the results. Moreover, this dataset was also used for training and testing EvoDeep [32], thus allowing to make a further comparison against SCRODeep. The MNIST dataset comprises a database of 60,000 training examples and 10,000 test examples of handwritten digits represented with grey levels in 28x28 pixel pictures.

B. Experimental setting

SCRODeep was run 30 times in order to obtain a representative number of solutions to fairly evaluate the proposed method. While the heuristic search is in charge of automatically providing the relation of hyperparameters and network architecture which maximises the result of the model in terms of accuracy, it is required to provide a range of values for each hyperparameter involved in order to narrow the search space, thus avoiding to reach values which can be excluded in advance and to limit the time involved in performing a whole execution of the search algorithm.

Table I provides a summary of the range of values tested for each hyperparameter of the different layers considered and also of the global hyperparameters. In the first case, different kernel initialisers, activation functions, number of units or number of filters are some of the hyperparameters for which different values are explored during search process. In the case of the global hyperparameters, different optimisers, batch sizes (restricted to a maximum of 1,000 due to memory limitations) and size of the network in terms of number of layers. In the

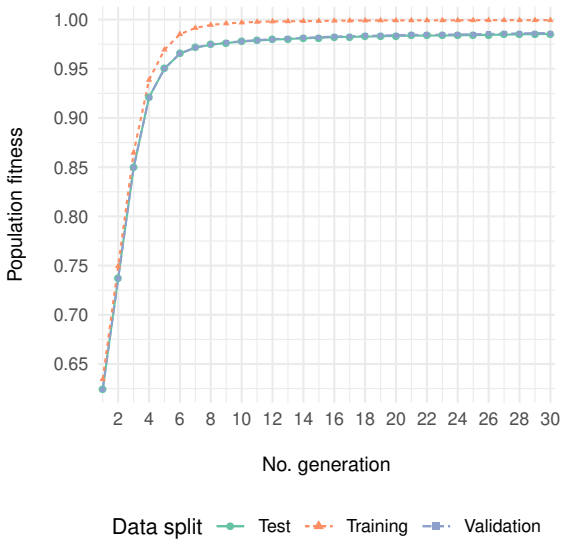

Fig. 2: Fitness evolution over generations for the three data splits for all executions performed.

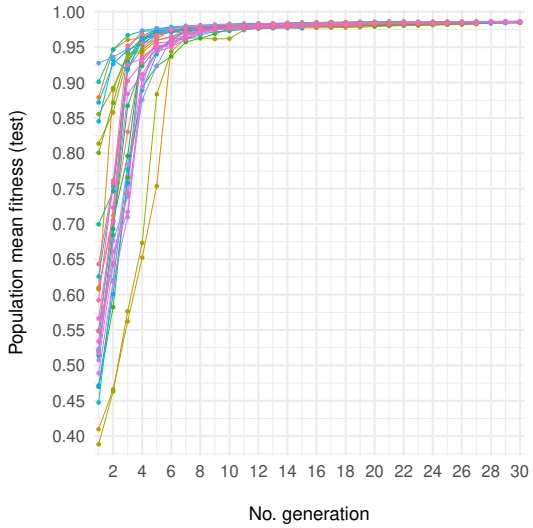

Fig. 3: Fitness evolution over generation in the test split showing data of all execution run.

case of the number of epochs, a stop criteria was set in order to stop the training step when validation accuracy did not improve in the last 10 epochs, establishing a maximum of 100 epochs. In general, a wide range of values was established for each hyperparameter, attempting not to limit the spectrum of feasible solution to deal with the issue at hand.

For the evaluation, the MNIST dataset was split into two slices, providing 50% of the examples for training and 50% for testing purposes. A further division was applied to the training dataset, allocating 10% of this batch of instances to a validation set. This last set of instances serves as an external measure of the quality of a particular coral or model, and it is used as the fitness function.

Regarding the SCRO parametrisation, although all hyperparameters are statistically settled, the reef size requires to be manually fixed. In these experiments it was to 6×6, a sufficient size for the algorithm to give an answer in a decent time. In other words, the size of the reef has been fixed by seeking a solution that is a compromise between computing speed and the quality of the individuals. Nevertheless, greater values can be considered if necessary.

C. Results

SCRODeep seeks to obtain the hyperparameters of a Convolutional Neural Network model. In this section, the results obtained after executing this algorithm with the MNIST dataset are described, showing how the approach proposed is able to define the hyperparameters and sequence of layers which maximise the performance of the model.

First of all, the use of the validation set was evaluated in order to check if it is a valid procedure to guide the heuristic search and if the results obtained in this portion can be extrapolated to the test set. Figure 2 includes the mean fitness value of the reef resulted at each generation for the three data splits. They show a very similar trend, almost identical in the case of the accuracy achieved in the test and validation sets. This evidences that the validation set used constitutes a useful and also unbiased tool for guiding the heuristic search towards strong models able to achieve high accuracy values.

All executions performed are represented in Fig. 3. In this plot it is possible to observe that just 6 generations are sufficient to converge a population with an average accuracy (in the test set) of more than 92%. From this point, the accuracy slightly improves towards 99%. Similarly, Fig. 4 represents the accuracy in the test dataset in a box plot, which allows to better appreciate the distribution of solutions as the search process progresses. The first three generations introduce a wide variety of individuals into the reef, showing high deviation levels, and where a considerable amount of solutions is not able to achieve good accuracy levels. From this point, diversity, in terms of accuracy levels, is reduced and the whole reef is composed by accurate solutions successfully improved in the following iterations.

The search for new and improved solutions is also expected to increase their complexity, thus increasing the time needed to evaluate the new generated individuals at each successive generation. SCRODeep performs an incremental search procedure, starting with solutions of low complexity whose evaluation takes a short period of time. In following generations, the operators allow to increase the complexity of the solutions (i.e. increasing the number of layers) in a controlled manner. Solutions reaching a counterproductive complexity showing no sign of improvement will be discarded, confining the heuristic search process to avoid unnecessarily complex models.

The results obtained from the best execution (the one reaching the highest value in the validation set) are shown in Table II. SCRODeep reaches 98.36% accuracy on average and 98.67% in the best run. These values show an improvement if

	Runs	Measure	Training	Validation	Test
SCRODeep	30	**Min**	99.82%	98.11%	98.11%
		Mean	99.95% ± 0.06	98.38% ± 0.16	**98.36% ± 0.15**
		Max	100%	98.59%	98.67%
EvoDeep	30	**Min**	99.20%	97.77%	97.56%
		Mean	99.78% ± 0.21	98.26% ± 0.36	98.24% ± 0.37
		Max	100.00%	98.71%	98.71%

TABLE II: Summary of the results obtained with SCRODeep considering the best individual obtained in terms of accuracy in the validation set for each execution. The lower section allows to make a comparison against EvoDeep.

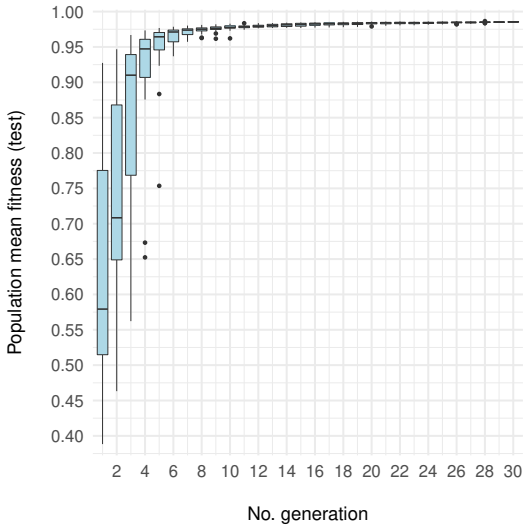

Fig. 4: Fitness evolution over generations in the test split grouped for all executions.

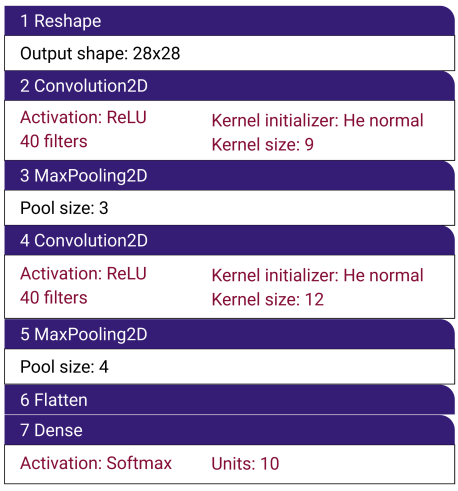

Fig. 5: Architecture and hyperparameters of the best individual found.

compared to EvoDeep.

The variances obtained by SCRODeep are significantly lower than those obtained by EvoDeep using a Levene test [36] (p-value $= 0.001, F = 14.873$). As Correct Classification Rates (CCR) for both EvoDeep and SCRODeep are not normal, we performed a Wald-Wolfowitz test [37] for small not-normal samples with different variances and small differences between average values. According to this test, there are statistically significant differences (p-value $= 0.038, Z = -2.073$) in the cumulative distributions of SCRODeep versus EvoDeep. This result stresses the enhancement provided by SCRODeep, automatically adjusting the hyperparameters and layers without the need for defining a set of hyperparameters of the metaheuristic.

Finally, Fig. 5 represents the structure of layers and hyperparameters of the best execution found in terms of accuracy in the validation set. It is composed of two cycles of Convolution2D and MaxPooling2D, a flatten layer a final dense layer of 10 units, directly connected to the outputs. At the global hyperparameters level, a batch size of 300 instances and the Adamax optimiser are used. In comparison with state-of-the-art models using the same layer types, this architecture is similar to LeNet-4 [35], although with some differences. The accuracy achieved is also similar, since LeNet-4 performs classification with 1.1 error rate.

V. Conclusions and Future Works

Deep learning has been assumed as a powerful tool able to deal with varied, complex and large problems. The wide range of techniques falling into this scope have allowed to address issues with precise and effective models. Nevertheless, they have a high degree of dependence with the particular problem definition. Thus, there is a plethora of architectures which have proven to be successful in specific domains. The design of the structure or hyperparameters of these models is not a trivial exercise. SCRODeep aims to address the task in which these factors are decided by performing a metaheuristic search where individuals represents potentials sets of hyperparameters and network structure seeking to maximise the accuracy of the model. The method proposed improves a previously presented method, called EvoDeep, by automating the whole search process eliminating the need for adjusting hyperparameters of the search algorithm itself. By using a statistically-driven version of a Coral Reef Optimisation algorithm, SCRODeep allows to automatically define the most proper model to use. The experiments show that the approach proposed is able to

provide competitive models able to reach high accuracy rates with statistical difference. In future work, we aim to enhance the possibilities of SCRODeep by including new layers able to deal with more complex problems and also to implement techniques able to reduce the number of evaluations, given that this is the most computationally expensive element in the algorithm.

ACKNOWLEDGMENT

This work has been co-funded by the following grants: Comunidad Autónoma de Madrid under grant S2018/TCS-4566 (CYNAMON: Cybersecurity, Network Analysis and Monitoring for the Next Generation Internet); Spanish Ministry of Science and Education and Competitivity (MINECO) and European Regional Development Fund (FEDER) under grants TIN2017-85727-C4-3-P (DeepBio), TIN2017-85887-C2-1-P and TIN2017-90567-REDT and Consejería de Economía, Conocimiento, Empresas y Universidad of the Junta de Andalucía (Spain) under grant UCO-1261651.

REFERENCES

[1] J. S. Bergstra, R. Bardenet, Y. Bengio, and B. Kégl, "Algorithms for hyper-parameter optimization," in *Advances in Neural Information Processing Systems 24*, J. Shawe-Taylor, R. S. Zemel, P. L. Bartlett, F. Pereira, and K. Q. Weinberger, Eds. Curran Associates, Inc., 2011, pp. 2546–2554.

[2] I. Loshchilov and F. Hutter, "Cma-es for hyperparameter optimization of deep neural networks," *arXiv preprint arXiv:1604.07269*, 2016.

[3] H. Mendoza, A. Klein, M. Feurer, J. T. Springenberg, and F. Hutter, "Towards automatically-tuned neural networks," in *Workshop on Automatic Machine Learning*, 2016, pp. 58–65.

[4] M. Jaderberg, V. Dalibard, S. Osindero, W. M. Czarnecki, J. Donahue, A. Razavi, O. Vinyals, T. Green, I. Dunning, K. Simonyan *et al.*, "Population based training of neural networks," *arXiv preprint arXiv:1711.09846*, 2017.

[5] J. Bergstra and Y. Bengio, "Random search for hyper-parameter optimization," *Journal of Machine Learning Research*, vol. 13, no. Feb, pp. 281–305, 2012.

[6] J. Bergstra, D. Yamins, and D. D. Cox, "Making a science of model search: Hyperparameter optimization in hundreds of dimensions for vision architectures," in *Proceedings of the 30th International Conference on International Conference on Machine Learning - Volume 28*, ser. ICML'13. JMLR.org, 2013, p. I–115–I–123.

[7] F. Ye, "Particle swarm optimization-based automatic parameter selection for deep neural networks and its applications in large-scale and high-dimensional data," *PLOS ONE*, vol. 12, no. 12, p. e0188746, dec 2017.

[8] T. Hinz, N. Navarro-Guerrero, S. Magg, and S. Wermter, "Speeding up the hyperparameter optimization of deep convolutional neural networks," *International Journal of Computational Intelligence and Applications*, vol. 17, no. 02, p. 1850008, 2018.

[9] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2014.

[10] A. M. Durán-Rosal, P. A. Gutiérrez, S. Salcedo-Sanz, and C. Hervás-Martínez, "A statistically-driven coral reef optimization algorithm for optimal size reduction of time series," *Applied Soft Computing*, vol. 63, pp. 139–153, 2018.

[11] W. S. McCulloch and W. Pitts, "A logical calculus of the ideas immanent in nervous activity," *The bulletin of mathematical biophysics*, vol. 5, no. 4, pp. 115–133, Dec 1943.

[12] K. Hunt, D. Sbarbaro, R. Żbikowski, and P. Gawthrop, "Neural networks for control systems—a survey," *Automatica*, vol. 28, no. 6, pp. 1083 – 1112, 1992.

[13] G. P. Zhang, "Neural networks for classification: a survey," *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, vol. 30, no. 4, pp. 451–462, Nov 2000.

[14] E. Hjelmås and B. K. Low, "Face detection: A survey," *Computer Vision and Image Understanding*, vol. 83, no. 3, pp. 236–274, 2001.

[15] A. Bahrammirzaee, "A comparative survey of artificial intelligence applications in finance: artificial neural networks, expert system and hybrid intelligent systems," *Neural Computing and Applications*, vol. 19, no. 8, pp. 1165–1195, Nov 2010.

[16] J. Schmidhuber, "Deep learning in neural networks: An overview," *Neural networks*, vol. 61, pp. 85–117, 2015.

[17] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard *et al.*, "Tensorflow: a system for large-scale machine learning." in *OSDI*, vol. 16, 2016, pp. 265–283.

[18] A. Paszke, S. Gross, S. Chintala, and G. Chanan, "Pytorch," 2017.

[19] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *CoRR*, vol. abs/1409.1556, 2014.

[20] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

[21] Z. Wang, W. Yan, and T. Oates, "Time series classification from scratch with deep neural networks: A strong baseline," in *2017 International Joint Conference on Neural Networks (IJCNN)*, May 2017, pp. 1578–1585.

[22] T. Afouras, J. S. Chung, A. Senior, O. Vinyals, and A. Zisserman, "Deep audio-visual speech recognition," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pp. 1–11, 2018.

[23] Z. Yuan, Y. Lu, and Y. Xue, "Droiddetector: android malware characterization and detection using deep learning," *Tsinghua Science and Technology*, vol. 21, no. 1, pp. 114–123, Feb 2016.

[24] W. Xiong, L. Wu, F. Alleva, J. Droppo, X. Huang, and A. Stolcke, "The microsoft 2017 conversational speech recognition system," in *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, April 2018, pp. 5934–5938.

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

[26] M. Srinivas and L. Patnaik, "Learning neural network weights using genetic algorithms-improving performance by search-space reduction," in *Neural Networks, 1991. 1991 IEEE International Joint Conference on*. IEEE, 1991, pp. 2331–2336.

[27] J. R. Koza and J. P. Rice, "Genetic generation of both the weights and architecture for a neural network," in *Neural Networks, 1991., IJCNN-91-Seattle International Joint Conference on*, vol. 2. IEEE, 1991, pp. 397–404.

[28] F. H.-F. Leung, H.-K. Lam, S.-H. Ling, and P. K.-S. Tam, "Tuning of the structure and parameters of a neural network using an improved genetic algorithm," *IEEE Transactions on Neural networks*, vol. 14, no. 1, pp. 79–88, 2003.

[29] J. Gascón-Moreno, S. Salcedo-Sanz, B. Saavedra-Moreno, L. Carro-Calvo, and A. Portilla-Figueras, "An evolutionary-based hyper-heuristic approach for optimal construction of group method of data handling networks," *Information Sciences*, vol. 247, pp. 94–108, 2013.

[30] 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.

[31] A. Abraham, "Meta learning evolutionary artificial neural networks," *Neurocomputing*, vol. 56, pp. 1–38, 2004.

[32] A. Martín, R. Lara-Cabrera, F. Fuentes-Hurtado, V. Naranjo, and D. Camacho, "EvoDeep: A new evolutionary approach for automatic deep neural networks parametrisation," *Journal of Parallel and Distributed Computing*, vol. 117, pp. 180 – 191, 2018.

[33] S. Salcedo-Sanz, J. Del Ser, I. Landa-Torres, S. Gil-López, and A. Portilla-Figueras, "The coral reefs optimization algorithm: an efficient meta-heuristic for solving hard optimization problems," in *Proceedings of the 15th International Conference on Applied Stochastic Models and Data Analysis (ASMDA2013), Mataró*, 2013, pp. 751–758.

[34] S. Salcedo-Sanz, J. Del Ser, I. Landa-Torres, S. Gil-López, and J. Portilla-Figueras, "The coral reefs optimization algorithm: a novel metaheuristic for efficiently solving optimization problems," *The Scientific World Journal*, vol. 2014, 2014.

[35] Y. LeCun, L. Bottou, Y. Bengio, P. Haffner *et al.*, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[36] H. Levene, "Robust tests for equality of variances," *Contributions to probability and statistics. Essays in honor of Harold Hotelling*, pp. 279–292, 1961.

[37] R. C. Magel and S. H. Wibowo, "Comparing the powers of the wald-wolfowitz and kolmogorov-smirnov tests," *Biometrical Journal*, vol. 39, no. 6, pp. 665–675, 1997.