Hybrid Encodings for Neuroevolution of Convolutional Neural Networks: A Case Study*

Gustavo-Adolfo Vargas-Hákim
Artificial Intelligence Research Institute, University of Veracruz
Xalapa, Mexico
vargashakimg@gmail.com

Efrén Mezura-Montes
Artificial Intelligence Research Institute, University of Veracruz
Xalapa, Mexico
emezura@uv.mx

Héctor-Gabriel Acosta-Mesa
Artificial Intelligence Research Institute, University of Veracruz
Xalapa, Mexico
heacosta@uv.mx

ABSTRACT

The Neuroevolution of Convolutional Neural Networks have yield into highly competitive results in the field of visual recognition in contemporary years. Some of the most recent advances in this field have been related to the design of neural encodings to represent these highly complex Deep Learning structures. Hybrid encodings have shown potential at distributing the representation of these networks into different sub-structures and thus improving the search. In this paper, we propose a compact hybrid encoding, which is used in an evolutionary framework called Deep Genetic Algorithm (DeepGA). We assess the performance of our simple representation against a hybrid encoding based on DenseBlocks, to evaluate how certain encodings might bias the search towards larger CNNs, in both single- and multi-objective scenarios. Our case study is the classification of lung conditions in chest X-ray images.

CCS CONCEPTS

• **Computing methodologies** → **Genetic algorithms**; **Neural networks**;

KEYWORDS

Neuroevolution, Encodings, Convolutional Neural Networks

ACM Reference Format:
Gustavo-Adolfo Vargas-Hákim, Efrén Mezura-Montes, and Héctor-Gabriel Acosta-Mesa. 2021. Hybrid Encodings for Neuroevolution of Convolutional Neural Networks: A Case Study. In *2021 Genetic and Evolutionary Computation Conference Companion (GECCO '21 Companion), July 10–14, 2021, Lille, France.* ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/3449726.3463133

1 INTRODUCTION

For almost a decade, there has been an emergence of Deep Neural Networks architectures. In particular, Convolutional Neural Networks (CNN) have excelled at different tasks, specializing in the field of visual recognition [20, 21, 27]. Nonetheless, the successful application of CNNs has important drawbacks: (1) even when the practitioner possesses expertise in Deep Learning, the design process of CNN structures tends to be iterative and time-consuming [41], (2) the usage of well-known architectures, as well as their manual design, oftentimes leads to excessive computational costs with respect to the area of application.

Neuroevolution (NE) relieves these issues by utilizing Evolutionary Computation (EC) techniques for the optimization of Artificial Neural Networks (ANN) [37]. Primeval methods focused on the optimization of the weights of the networks [30], while in recent years, more powerful mechanisms have arisen, able to also evolve the neural architectures [28, 42, 49]. These advancements have reached different application domains ranging from learning and controls [19] to visual classification [46] and image segmentation [51].

Latterly, Neuroevolution has matured considerably for the automatic design of CNNs. These advancements have lead to converge into a series of established neural encodings, i.e. the mechanisms through which CNNs are represented and manipulated by Evolutionary Algorithms (EA) and Swarm Intelligence (SI) algorithms. As has been discussed by Eiben & Smith [17], the encoding of potential solutions is the first and most important step in any EA or SI algorithm design. Furthermore, the impact of the neural encoding in any NE method has been discussed [39], being the large search spaces an important factor to consider [12].

In this work, we present a new flexible hybrid encoding that can potentially help discovering competitive CNNs with less complexity, in comparison with another more modular hybrid encoding. We pay special attention in controlling the size of the networks, measured in terms of number of trainable parameters. Our method, DeepGA, encapsulates a simple Genetic Algorithm-based Neuroevolution framework, that is able to manipulate hybrid encodings in both single- and multi-objective settings. It has been found that DeepGA helps designing highly competent CNNs in terms of accuracy and complexity, outperforming several hand-crafted *state-of-the-art* architectures.

The rest of this paper is organized as follows. An introduction and a literature review on encodings for CNNs is presented in Section 2. The proposed DeepGA framework is introduced in Section 3. The experimental design and methods are shown in Section 4, while the results are discussed in Section 5. Conclusions and future work are in Section 6.

*Produces the permission block, and copyright information

GECCO '21 Companion, July 10–14, 2021, Lille, France

ACM ISBN 978-1-4503-8351-6/21/07…$15.00
https://doi.org/10.1145/3449726.3463133

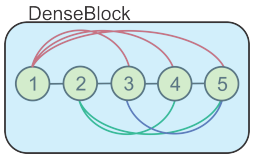

Figure 1: A DenseBlock composed of five convolutional layers.

2 NEURAL ENCODINGS

The first part in the design of a Neuroevolution algorithm consists in defining a neural encoding, which is a computational representation of an ANN. EAs and SI algorithms manipulate the encodings to navigate through the search space. Despite their importance, empirical studies on the effects of different encodings have not been formally addressed.

Stanley and Miikkulainen presented Neuroevolution of Augmented Topologies (NEAT) [38], which introduced the notions of direct and indirect encodings. Although these concepts were originally applied to feed-forward neural networks, they can be easily extrapolated to CNNs.

In indirect encodings, the characteristics present in a CNN are not explicitly represented. A set of rules are needed to transform the encoding (*genotype*) into an actual CNN (*phenotype*). In spite of being compact, these encodings can be less flexible and *obscure* to the users. There are two representatives of indirect encodings for CNNs. First, Binary Encodings [6, 49, 51], that use binary strings to represent the network's elements. This encoding excels at describing connectivity between layers, being 1 an active connection and 0 an ignored connection. Second, Grammar Encodings [4–6], which base on formal grammars in Backus-Naur notation. These approaches have proved to be flexible, although they require the definition of grammar rules in order to automatically build a language that later is transformed into a working CNN.

In direct encodings, all the characteristics of a CNN are explicitly represented in the encoding, and the variation operators have direct access to them. There is no difference between the *genotype* and the *phenotype*. There are two representative encodings that belong to this family. First, Graph-based Encodings [16, 18, 23, 34], which utilize graphs or trees to represent the CNNs. Usually, the layers are used as nodes, while edges define the connectivity between them. These encodings are highly flexible, yet hard to manipulate, as cycles must usually be avoided by the variation operators. Second, there are Block-chained Encodings [25, 40–42], which are based on a sequence of *blocks*. A block is an abstract representation of a layer or a set of layers. A low-level block may represent a convolutional, pooling or fully-connected layer, containing hyperparameters such as filter size, kernel size, pooling type, stride, and so on. Some authors introduced higher-level blocks, which contain more than one layer. One example is the DenseBlock (see Fig. 1), that inspires from the well-known DenseNet architecture [21].

In a DenseBlock, the output of each layer connects not only to the next immediate layer, but also to all the subsequent non-consecutive layers through *skip connections*. The condition is that all the outputs have the same spatial resolution. DenseBlocks have an additional hyperparameter called *growth rate*, which relates to how many

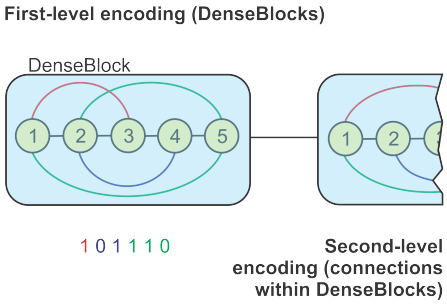

Figure 2: The hybrid encoding proposed by Wang *et al.* [47], consisting of a sequence of DenseBlocks in the first level, and binary strings in the second level. Each bit represents the connectivity from previous layers, from the 3–rd layer onward.

feature maps result from a convolutional layer. As an example, if the growth rate has a value of 3, each layer would contain 3 filters and thus would produce the same number of feature maps. This means that the k-th layer would receive $3(k-1)$ feature maps from the previous layer and from skip connections altogether.

Finally, Hybrid Encodings [28, 47, 50] are more recent representations that combine two or more of the preceding methods. The objective of hybrid encodings is to distribute the representation of CNNs between more than one structure. A remarkable example is the encoding presented by Wang *et al.* [47], that combines Block-chained and Binary Encodings, and is the only approach that has explored this configuration. Fig. 2 shows an example of this representation.

In this hybrid encoding, the first-level comprises a sequence of DenseBlocks, each represented by their number of convolutional layers and their growth rate. All the convolutional layers use 3×3 filters, with zero padding and stride of 1. Each convolution is followed by batch normalization [22] and ReLU. Transition layers are always used between DenseBlocks, and consist of a convolutional layer of 3×3 as filter size, with a number of filters equal to half of the growth rate of the previous DenseBlock, and a max pooling layer with a kernel of size 2×2. Transition layers decrease the spatial resolution of the outputs of a DenseBlock.

On the other hand, the second-level encoding is a binary string that defines the dense connectivity patterns of a DenseBlock. The bits of the binary start from the third layer onward, and represent a skip connection from all the previous, non-immediate layers. The third layer might receive one skip connection from the first layer only, the fourth layer can receive a skip connection from the first and second layers, and so on. This separation in the encoding allows the authors to use a hybrid between Particle Swarm Optimization and Genetic Algorithm to evolve the CNNs. We refer to this encoding as *Wang encoding* for the rest of this paper.

In this work, we propose to study a hybrid encoding based on simpler blocks. For our purposes, a block consists of only one convolutional layer of variable number of filters and variable filter sizes. Stride of 1 and zero padding are fixed and batch normalization and ReLU are always utilized. Within the same block, the pooling operation can be enabled or disabled, being max pooling and average

Table 1: Evolvable hyperparameters for convolutional blocks and fully-connected blocks.

	Hyperparameter	Values
Convolutional Block	Conv. Filter Sizes	{2, 3, 4, 5, 6, 7, 8}
	No. of Conv. Filters	{2, 4, 8, 16, 32}
	Pooling Types	{Max, Avg, Off}
	Pooling Kernel Sizes	{2, 3, 4, 5}
FC Block	No. of Neurons	{4, 8, 16, 32, 64, 128}

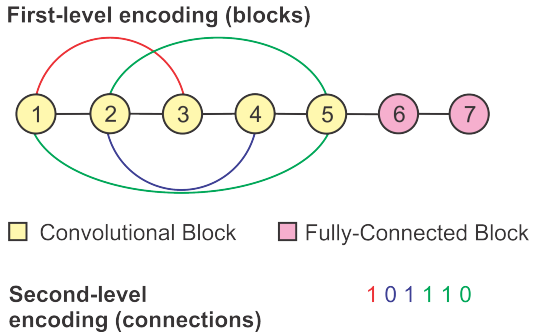

Figure 3: The proposed encoding based on simpler convolutional blocks. A single binary string determines the skip connections received from the third block onward. Each bit represents the connectivity from previous layers, from the 3–rd layer onward.

pooling the two enabled possible options. The stride value is always 2 and the kernel size is a searched hyperparameter.

Finally, the feature extraction section is followed by fully-connected (FC) blocks, which consist of fully-connected layers characterized by their number of neurons, with ReLU as the fixed option. Table 1 summarizes the evolvable hyperparameters of this encoding.

In this encoding, only one binary string per individual is required, unlike Wang encoding, where each DenseBlock is associated to an array of bits. Fig. 3 illustrates the proposed representation.

As convolutional layers can have different filter sizes, and the pooling operations can also differ in their spatial resolution, an adjustment mechanism needs to be used to avoid feature maps concatenation incompatibilities. Max pooling or zero padding are applied to adjust the feature maps to be concatenated depending on their sizes. Fig. 4 presents this procedure in detail.

The rationale behind proposing a hybrid encoding with simpler blocks lies in the abstraction of the encoding. A single DenseBlock might contain more convolutional layers than the total number of convolutional blocks inside the proposed encoding. It is therefore intuitively expected that the search process using the former encoding, would be biased towards deeper, more expensive CNNs in comparison to the latter approach. This assumption is here evaluated to provide for more empirical evidence on the impact of the modularity of neural encoding.

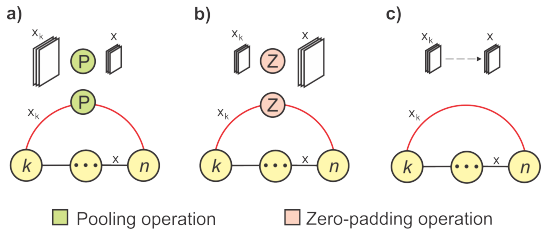

Figure 4: Spatial resolution adjustment for skip connections. a) When the incoming feature maps x_k exceed the current output x, max pooling with stride of 1 is applied. The kernel size is calculated so as to fit the result to x. b) When x_k is smaller than the current output x, zero padding is used on x_k until it matches the size of x. c) When both x_k and x match in their spatial resolution, no operation is applied.

3 DEEP GENETIC ALGORITHM

We propose the Deep Genetic Algorithm (DeepGA), which is able to manipulate hybrid representations combining block-chained and binary encodings. The Genetic Algorithm is chosen due to its balance between exploitation (through crossover) and exploration (through mutation). Different operators are designed to work with the proposed hybrid encoding and the Wang encoding.

3.1 Parent Selection

Parent selection is the process of choosing the best individuals of the population (based on fitness) for recombination. The selection of parents is performed in such a way that the crossover of such parents promotes the improvement of the population.

Tournament is a common approach for parent selection. A simple approach consists on selecting the individual with the highest fitness from a randomly selected group of individuals (tournament). As discussed in [29], a larger tournament generates more selection pressure on the contestants.

In DeepGA, an stochastic Tournament Selection is slightly modified for exploration purposes. The individual with highest fitness might be the best option towards an improvement in the offspring. Nonetheless, the possibility of another individual being the best option for recombination should not be discarded. For this reason, a probability of choosing the best individual as a parent is set to 0.8. If a random uniform number does not exceed this probability, an individual will be chosen at random. The expectation is that 80% percent of the selections are based on fitness, meanwhile the remaining selections explore using other potential solutions. This parameter was chosen to set a significant margin to explore other solutions. Furthermore, a moderate selection pressure is maintained by using a tournament of five individuals.

3.2 Crossover

The crossover operator combines two parent solutions to generate offspring. Offspring solutions should inherit information from their parents only in order to exploit around the currently visited area of the search space. In the proposed encoding, the crossover is accomplished in two parts: (1) first-level encoding crossover, where

Table 2: Evolvable hyperparameters for DenseBlocks and fully-connected blocks in the Wang encoding.

	Hyperparameter	Values
DenseBlock	No. of Conv. Layers	[3, 5]
	Growth Rate	[3, 12]
FC Block	No. of Neurons	{4, 8, 16, 32, 64, 128}

a number of blocks are exchanged between parents, and (2) second-level encoding crossover, where a portion of the binary strings are exchanged between parents.

In the first-level crossover the smallest parent is identified; m is the floor function of half the number of convolutional blocks, and n is the floor function of half the number of fully-connected blocks. The crossover exchanges the last m convolutional blocks of both parents, as well as their last n fully-connected blocks.

In the second-level crossover, c is the floor function of half of the bits in the smallest parent's binary string. The last c bits of both parents are therefore exchanged. Fig. 5 presents an example of the entire crossover.

The same kind of operation is applied when using the Wang encoding. However, when a DenseBlock exchange is performed, the binary strings are also exchanged between parents. It can be seen that encodings that are more modular can also involve higher-level operations.

3.3 Mutation

The mutation operation promotes exploration through the search space. Unlike crossover, where the changes are applied to many of the best individuals, the mutation performs unbiased alterations to the offspring.

In DeepGA, two mutations are applied: (1) the first-level mutation, which affects the blocks, and (2) the second-level mutation, which affects the connections. The second mutation selects a random bit and flips its value. Given a random uniform number $U_1(0,1)$, the first mutation can have two possible forms: (a) restarting a block ($U_1 \leq 0.5$), or (b) adding a new block ($U_1 > 0.5$). If a random number $U_2(0,1)$ exceeds the threshold of 0.5, mutation (b) adds a fully-connected block. Otherwise, a convolutional block is added. When the proposed encoding is used, the addition of a convolutional block also carries the insertion of new bits in the second-level encoding. The new bits' positions depend on where the new block was randomly inserted (see Fig. 6). The addition of a fully-connected block requires no further changes.

Restarting a block, which is the first possible mutation, consists in randomly resetting the hyperparameters (see Table 1).

When using the Wang encoding, these mutations are used on a more modular basis. When adding a new block, no additional bits need to be added, as each DenseBlock contains its own binary string. A DenseBlock or a fully-connected block could be added based on the same stochastic approach. The second possible mutation resets the block's configuration. Table 2 shows the evolvable hyperparameters for the Wang encoding.

3.4 Fitness Function

Disregarding of the chosen encoding, DeepGA seeks to maximize the classification performance of the CNNs while reducing their computational complexity. Here, the computational cost of a CNN is measured with respect to the number of trainable parameters, i.e., weights and bias values. To handle the fulfillment of these requirements, we propose two different fitness mechanisms.

First, a linear weighted fitness function that takes classification accuracy and a proportion of parameters has been designed. This fitness function has the advantage of being configurable for the needs of the problem. Eq. 1 introduces the fitness function:

$$f(cnn) = (1 - w) * accuracy(cnn) + w * \frac{MP - NP}{MP} \quad (1)$$

where cnn is a CNN architecture, w is the complexity weight in $[0, 1]$, MP is a user-defined maximum number of parameters allowed in the networks, and NP is the current number of parameters in cnn. As w grows, more importance will be given to the network's complexity. A series of preliminary experiments aided to define $w = 0.3$, for having a good performance in dealing with both objectives. The smallest model against out method is compared has 1.164 [32], thus MP is set to 2 million parameters, expecting networks around this value.

The second fitness mechanism is Multi-Objective Optimization (MOO). Unlike the linear aggregation fitness function, where accuracy and complexity are mixed in a single objective, the Multi-Objective DeepGA (MODeepGA) handles both functions (classification error and number of parameters) independently. MOO consists of finding a good approximation of the optimal Pareto Front, i.e., a set of competitive trade-offs between objectives.

The Pareto Optimal Front is composed by a set of non-dominated solutions. Let a multi-objective minimization problem be composed of m objective functions. A solution x is said to dominate a solution y (denoted as $x \prec y$), if and only if, $f_i(x) \leq f_i(y)$ for all $i \in [1, ..., m]$ and $f_i(x) < f_i$ for at least one $i \in [1, ...m]$. Thus, a solution x^* is part of the Pareto Optimal Set P^* if there does not exist other solution x such that $x \prec x^*$. The Pareto Optimal Front is then $PF^* = \{f(x) | x \in P^*\}$.

In MODeepGA, the solutions are sorted based on Pareto dominance. Non-dominance sorting [15] is used to distinguish between dominated and non-dominated solutions. All the non-dominated CNNs are clustered together in the first Pareto Front. The remaining networks are again sorted with respect to non-dominance to extract the second Pareto Front. The procedure is repeated until all the solutions are grouped and sorted into their respective fronts. At the end of the optimization, a set of trade-off solutions is expected to be produced. Backpropagation is always required for fitness evaluation, which occurs by training two individuals at the same time on two NVIDIA Titan RTX GPUs, meanwhile all the genetic operators take place in series on the CPU.

3.5 General Framework

DeepGA is built using the previously mentioned components, following the architecture of an standard Genetic Algorithm. Algorithm 1 elaborates on the complete DeepGA framework.

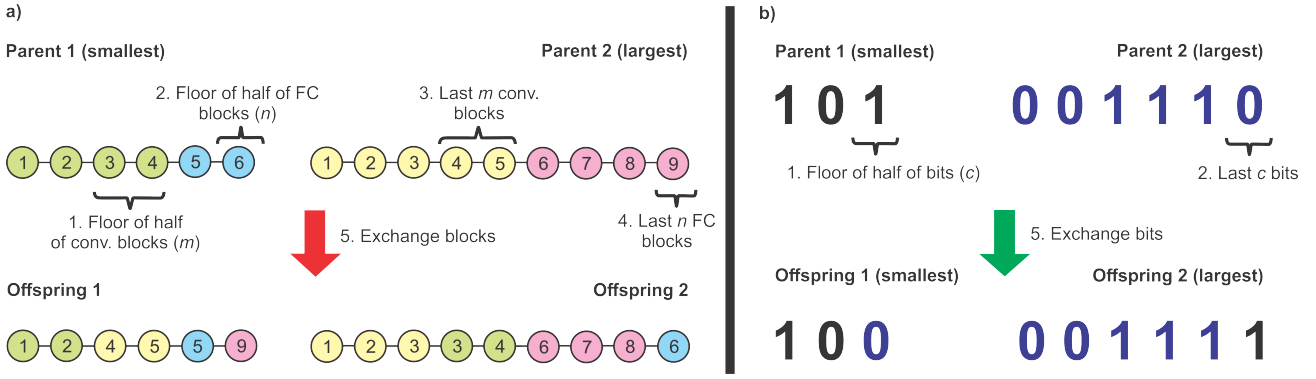

Figure 5: Crossover operation in DeepGA for the proposed encoding. Convolutional blocks are in green and yellow, meanwhile fully-connected (FC) blocks are in blue and pink. a) Is the exchange of blocks in the first-level encoding, and b) is the exchange of bits in the second-level encoding.

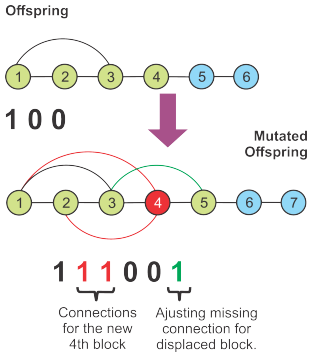

Figure 6: The second mutation operation in DeepGA for the proposed encoding. The adjusting bits for displaced convolutional block k are always placed at the end of the k-th binary sub-string.

Algorithm 1: DeepGA

Input: A population P of N individuals. The number of generations T, crossover rate $CXPB$, mutation rate $MUPB$, tournament size $TSIZE$.
Output:
Initialize population (training the networks).
$t \leftarrow 1$
while $t \leq T$ **do**
  Select $N/2$ parents with probabilistic tournament selection
  Offs $\leftarrow \{\}$
  while $|Offs| < N/2$ **do**
    Select two random parents p_1 and p_2.
    if *random(0,1)* $\leq CXPB$ **then**
      $O_1, O_2 \leftarrow$ Crossover(p_1, p_2) // Crossover
      if *random(0,1)* $\leq MUPB$ **then**
        Mutation(O_1, O_2) // Mutation
      fitness(O_1, O_2) (Eq. 1 //Evaluation
    $P \leftarrow P \cup$ Offs
    Select the best N individuals in P as survivals.
  end
end

MODeepGA differs only in the fitness evaluation. Instead of using Eq. 1, the classification error (instead of accuracy) and the number of parameters are stored independently. The non-dominance sorting is used on the union of the population and offspring, to select the first N individuals.

The following section defines the experimental settings to evaluate DeepGA and MODeepGA with both hybrid encodings.

4 EXPERIMENTAL SETTING

Our case study is motivated by the emergence of the Coronavirus Disease 2019 (COVID-19). The standard test bases on the Reverse-Transcription Polymerase Chain Reaction (RT-PCR) [2], which has proven to be hampered and ineffective in several instances [8]. In our approach, we utilize Chest X-ray (CXR) images as an alternative and complementary test to classify between this and other lung conditions, as CNNs have demonstrated to be highly competent with these kind of imagery [7]. The scarcity of abundant data in this kind of niche problems makes necessary to utilize Neuroevolution approaches that search for more compact architectures. Furthermore, computational resources in many healthcare centers might be limited, which hampers the deployment of large models.

A total of 2754 CXR images were collected from different repositories [14, 43, 48]. The images are evenly distributed in COVID-19, bacterial/viral pneumonia, and healthy patients (918 images each). As suggested in [31], the images went through the following preprocessing: (1) transformation to grayscale, (2) casting to *float32*, (3) histogram equalization, (4) gamma correction ($\gamma = 0.5$), (5) resizing to 256×256 pixels, and (6) normalization within $[0, 1]$. Fig. 7 shows an example of three classes of images.

The training of the CNNs during the Neuroevolution consisted in only 10 epochs (as suggested by [42]), using the Adam optimizer [26] with a learning rate of 1×10^{-4}. The training utilized

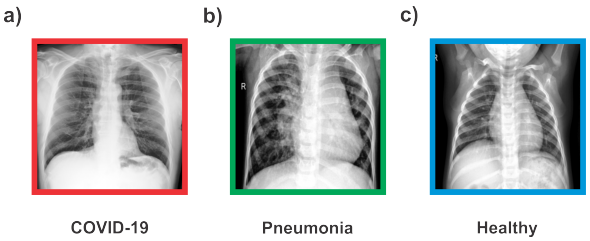

Figure 7: Chest X-ray images of a) a patient with COVID-19, b) a patient with viral/bacterial pneumonia, and c) a healthy patient.

Table 3: Parameters of DeepGA and MODeepGA: population size (N), number of generations (T), crossover rate ($CXPB$), mutation rate ($MUPB$), and tournament size (S).

	N	T	CXPB	MUPB	S
DeepGA	20	50	0.7	0.5	4
MODeepGA		20			

70% of the image dataset, meanwhile the accuracy/error was computed using the remaining 30% for testing. The parameters of DeepGA (see Table 3) were manually tuned after a series of experiments; particularly, the mutation rate (*MUPB*) and the crossover rate (*CXPB*) received special attention. These two parameters were adjusted until a sufficient diversity through exploration was observed. The computational costs related to Neuroevolution impeded the utilization of automatic parameter tuning methods. The population size and number of generations were set in order to maintain a practically feasible computation time with the available resources. The source code of DeepGA and MODeepGA was developed in Python 3.7 and PyTorch, and is available at https://github.com/GustavoVargasHakim/DeepGA.git along with the utilized dataset.

To assess the impact of the proposed neural encoding and the Wang encoding on the performance of DeepGA, two sets of experiments were carried out. First, 30 executions were run with each of the two encodings in the single-objective version of DeepGA. At the end of the 50 generations, the fitness score, the accuracy, and the number of parameters of the best individual are reported. Additionally, the best CNN at the end of each execution is independently trained under 5-fold crossvalidation with a random seed of 0, in order to compute the specificity and sensitivity of the three classes. The Kolmogorov-Smirnov Test at 95% is applied to all samples to discard normality, and afterwards, all these metrics are compared between the two encodings using the 95%-confidence Wilcoxon Rank Sum test. Additionally, our results are compared against a number of hand-crafted CNNs that were used for the same type and number of classes (see Section 5).

In the second round of experiments, 30 executions of MODeepGA were run using each encoding. The quality of the final bi-objective (classification error and number of parameters) Pareto front obtained is measured by using the Hypervolume indicator. Hypervolume is chosen due to its direct relation with the optimality of the Pareto fronts as well as to the diversity of its solutions

Table 4: Mean ± std. dev. of the fitness function, the accuracy, and the number of parameters (# Params). Based on the Wilcoxon test, our encoding outperforms Wang encoding (+), or achieves equal results (=).

Metric	Our Encoding	Wang Encoding	p-value	Result
Fitness	0.9671 ± 0.0037	0.9619 ± 0.0019	1.47×10^{-7}	+
Accuracy	0.9593 ± 0.0056	0.9504 ± 0.0037	1.03×10^{-6}	+
# Params	28944.13 ± 8974.11	29770.23 ± 10872.72	0.709	=

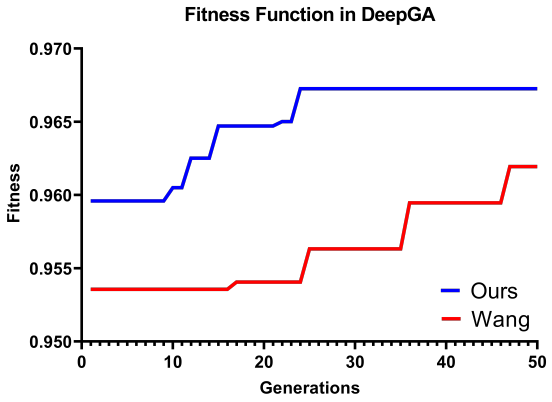

Figure 8: Convergence plot of the fitness function of the median executions using our encoding (blue) and the Wang encoding (red).

[36]. To compute this indicator, both objective functions in the final Pareto Fronts are normalized in order to use the Nadir point [9] as reference point $\mathbf{r} = (1, 1)$. The Hypervolume samples are passed through the Kolmogorov-Smirnov test in order to compare both encodings with the 95%-confidence Wilcoxon Rank Sum test.

5 RESULTS AND DISCUSSION

In this section, the obtained results from the two experimental rounds (single-objective and multi-objective) of DeepGA are presented and analyzed.

5.1 Single-Objective Results

The Kolmogorov-Smirnov Test was performed on the resulting samples of fitness function, accuracy, and number of parameters from DeepGA. In all cases, the p-values were smaller than 0.05, thus the samples were considered as not normal. The Wilcoxon Rank Sum Test is applied to verify statistical differences between analogous samples of our proposed encoding and the Wang encoding. Table 4 shows fitness function, the accuracy, and the number of parameters of each encoding.

From Table 4, it is concluded that our encoding aids in finding CNN architectures with overall higher fitness and accuracy. Interestingly, both representations discovered CNN topologies with an equivalent number of parameters. The convergence plots of the fitness function is shown in Fig. 8. The proposed encoding drives DeepGA to its highest value faster than the Wang encoding.

Table 5: Specificity and sensitivity comparison between encodings across the three classes. The Wilcoxon Rank Sum Test determined when our encoding surpasses the Wang encoding (+), when the Wang encoding surpasses our encoding (-), or when both yield equal results (=).

	Our Encoding	Wang Encoding	Results
COVID-19 Specificity	0.9615 ± 0.0074	0.9642 ± 0.0041	=
COVID-19 Sensitivity	0.965 ± 0.0015	0.9558 ± 0.0096	=
Pneumonia Specificity	0.9634 ± 0.0143	0.9626 ± 0.0057	=
Pneumonia Sensitivity	0.9478 ± 0.0198	0.9518 ± 0.0064	=
Healthy Specificity	0.9646 ± 0.0154	0.9627 ± 0.0052	+
Healthy Sensitivity	0.8951 ± 0.0206	0.9052 ± 0.0105	-

Albeit the accuracy results are higher with the proposed representation, DeepGA obtains equal specificity and sensitivity results for COVID-19 and pneumonia images, as displayed in Table 5. Our encoding lies behind the Wang encoding in terms of healthy patients' sensitivity, but improves on the specificity.

5.2 Multi-Objective Results

The Hypervolume indicator is computed at the end of each of the 30 executions of MODeepGA with both encodings. The final Pareto Fronts are represented by 20 individuals with two objective functions. After normalizing these functions, the Hypervolume is calculated using a deterministic implementation [11], which returns a value in the range $[0, 1]$, where 1 corresponds to an optimal performance, and 0 corresponds to the opposite.

With the proposed encoding, the resulting mean Hypervolume is 0.9468 ± 0.1536, meanwhile with the Wang encoding, the means Hypervolume resulted in 0.9084 ± 0.1289. With p−values of 3.18×10^{-18} and 3.79×10^{-16}, respectively, the Kolmogorov Smirnov Test probes that the samples do not belong to normal distributions. Therefore, the Wilcoxon Rank Sum Test is utilized to demonstrate, with a p−value of $3.79e-16$, that the proposed encoding surpasses the Wang encoding in optimality, in terms of this metric. The point $p_{ref} = (0, 0)$ and an euclidean knee-based selection chose one CNN from each encoding basing on the median executions. With our encoding, the selected architecture had 134131 parameters and 95.1632% of accuracy, whilst with the Wang encoding, the CNN had 163861 parameters and 95.28% of accuracy.

In comparison to the single-objective version of DeepGA, which uses a weighted linear function to evaluate the individuals, the MODeepGA takes better advantage of the proposed encoding. This can be observed as the Hypervolume is higher when using our proposed representation, meaning a better convergence to more competitive trade-offs between objectives.

5.3 Architectures

The CNNs from the best and median executions of DeepGA, using the two encodings, are presented in Fig. 9. With the proposed encoding, it has been found that skip connections were not required in none of these two executions. Furthermore, the hyperparameters in both cases are very similar, which could provide for insights on the optimality of the problem.

Table 6: Comparison of accuracy (%) and number of parameters (millions) between the *state-of-the-art* hand-crafted CNNs and the architectures of the median execution of DeepGA with our encoding and the Wang encoding.

Model	Accuracy	Parameters
DarkCovidNet [32]	87.02	1.164
Bayes SqueezeNet [45]	98.26	1.263
Xception + ResNetv50 [33]	91.4	48.855
ResNet18 [31]	88.9	11
EfficientNet-B0 [3]	95.24	5.3
VGG-16 [10, 13]	82.81	138
Xception [1]	97.41	22.85
MobileNet [44]	99.2	14.174
CNN + LSTM [24]	**99.34**	99.34
DeepGA (Wang)	95.76	0.0562
DeepGA (Ours)	96	**0.0321**

The architectures of the median executions of both encodings are compared with several *state-of-the-art* hand-crafted CNNs, as shown in Table 6. Additionally, these results are plotted together in Fig. 10, where it can be seen that the architectures found by DeepGA are highly competitive in terms of accuracy, whilst reducing the number of parameters by at least two orders of magnitude with respect to previous works. Our dataset differs in size from those in previous works, however, the images are obtained from the same repositories and represent the same classes. In our approach, neither augmented nor synthetic data were required, and the training only required 10 epochs as used during the DeepGA.

6 CONCLUSIONS

In this work, a new GA-based Neuroevolution algorithm was proposed; DeepGA. An empirical study was conducted to assess two hybrid encodings based on simple blocks (our encoding) and DenseBlocks (Wang encoding), respectively, and binary strings. Furthermore, single-objective and multi-objective variants of DeepGA were utilized to evaluate the performance based on the chosen encoding.

The testing problem was the classification of lung conditions in chest X-ray images, which is of timely interest. The fitness evaluation comprised the classification accuracy of the networks, along with their number of parameters. The experiments demonstrated that the single-objective DeepGA using our encoding, generally discovers higher quality networks in terms of accuracy, specificity and sensitivity, but matching the Wang encoding in terms of number of parameters. In the multi-objective experiments, MODeepGA finds Pareto Fronts with better optimality when using our encoding compared to the Wang encoding, measured with the Hypervolume.

Future work includes: (1) integrating the DeepGA with the proposed hybrid encoding to be tested using benchmark datasets such as CIFAR-10 and CIFAR-100, in order to extrapolate the results found in this paper and to explore the scalability of DeepGA with larger datasets, (2) a more detailed study on multi-objective Neuroevolution based on different hybrid encodings, and (3) introducing

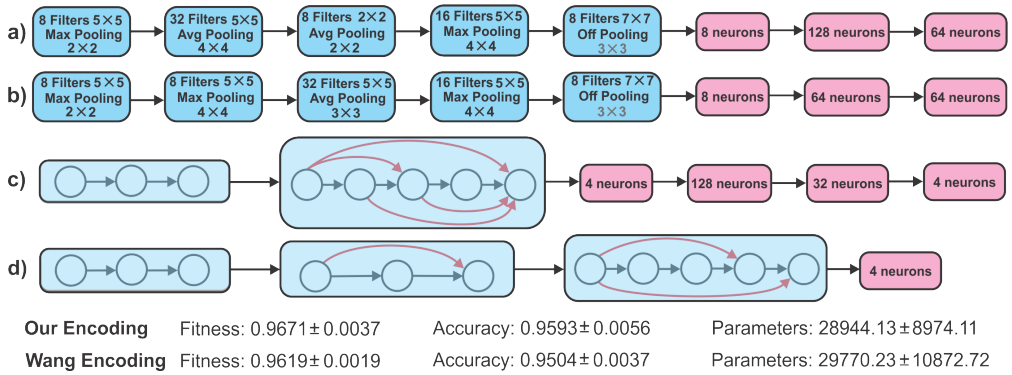

Figure 9: The CNN architectures with our encoding from the a) best and b) median executions with respect to fitness, and the architectures with the Wang encoding from c) the best and d) median executions.

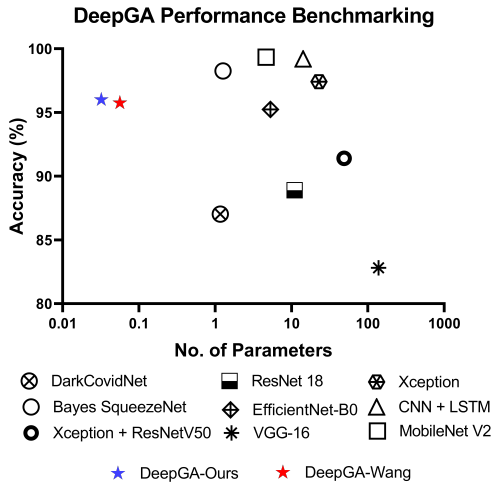

Figure 10: The benchmarking of DeepGA with our encoding (blue star) and the Wang encoding (red star) in terms of accuracy and the number of parameters (millions). The horizontal axis is in *log*10 scale.

explainability components, such as GradCam [35], to encourage the evolution of explainable CNNs.

7 ACKNOWLEDGMENT

The first author acknowledges support from the Mexican National Council of Science and Technology (CONACyT) through a scholarship to pursue graduate studies at University of Veracruz. The authors acknowledge the grant from the *Laboratorio de Supercómputo del Bajío* from the Center of Research in Mathematics (CIMAT) to access their high-performance computing facilities. Also, the authors would like to thank the Artificial Intelligence Research Institute for sponsoring the publication of this research work.

REFERENCES

[1] N. Narayan Das ad N. Kumar, V. Kumar, and D. Singh. 2020. Deep Learning System for COVID-19 Diagnosis Aid Using X-ray Pulmonary Images. *IRBM* (2020). https://doi.org/10.1016/j.irbm.2020.07.001

[2] Tao Ai, Zhenlu Yang, Chenao Zhan, Chong Chen, Wenzhi Lv, Qian Tao, Ziyong Sun, and Liming Xia. 2020. Correlation of Chest CT and RT-PCR Testing for Coronavirus Disease 2019 (COVID-19) in China: A Report of 1014 Cases. *Radiology* 296, 2 (2020). https://doi.org/10.1148/radiol.2020200642

[3] Aytac Altan and Seckin Karasu. 2020. Recognition of COVID-19 disease from X-ray images by hybrid model consisting of 2D curvelet transform, chaotic salp swarm algorithm and deep learning technique. *Chaos, Solitons & Fractals* 140 (2020). https://doi.org/10.1016/j.chaos.2020.110071

[4] Filipe Assuncao, Nuno Lourenco, Penousal Machado, and Bernardete Ribeiro. 2018. DENSER: deep evolutionary network structured representation. *Genetic Programming and Evolvable Machines* 20, 1 (2018), 5–35. https://doi.org/10.1007/s10710-018-9339-y

[5] Filipe Assuncao, Nuno Lourenco, Penousal Machado, and Bernardete Ribeiro. 2019. Fast DENSER: Efficient Deep NeuroEvolution. In *European Conference on Genetic Programming*. Springer, 197–212. https://doi.org/10.1007/978-3-030-16670-0_13

[6] Alejandro Baldominos, Yago Saez, and Pedro Isasi. 2018. Evolutionary convolutional neural networks: An application to handwriting recognition. *Neurocomputing* 283, 29 (March 2018), 38–52. https://doi.org/10.1016/j.neucom.2017.12.049

[7] Ivo M. Baltruschat, Hannes Nickisch, Michael Grass, Tobias Knopp, and Axel Saalbach. 2019. Comparison of Deep Learning Approaches for Multi-Label Chest X-Ray Classification. *Nature Scientific Reports* 9, 6381 (2019). https://doi.org/10.1038/s41598-019-42294-8

[8] Nick J. Beeching, Tom E. Fletcher, and Mike B.J. Beadsworth. 2020. Covid-19: testing times. *British Medical Journal* 369 (2020). https://doi.org/10.1136/bmj.m1403

[9] Julian Blank and Kalyanmoy Deb. 2020. A Running Performance Metric and Termination Criterion for Evaluating Evolutionary Multi- and Many-objective Optimization Algorithms. In *2020 IEEE Congress on Evolutionary Computation (CEC)*. IEEE. https://doi.org/10.1109/CEC48606.2020.9185546

[10] Luca Brunese, Francesco Mercaldo, Alfonso Reginelli, and Antonella Santone. 2020. Explainable Deep Learning for Pulmonary Disease and Coronavirus COVID-19 Detection from X-rays. *Computer Methods and Programs in Biomedicine* 196 (2020). https://doi.org/10.1016/j.cmpb.2020.105608

[11] Yi Cao. 2020. Hypervolume Indicator. (2020). https://www.mathworks.com/matlabcentral/fileexchange/19651-hypervolume-indicatorRetrievedJanuary16, 2021.

[12] Yukang Chen, Gaofeng Meng, Qian Zhang, Shiming Xiang, Chang Huang, Lisen Mu, and Xinggang Wang. 2019. RENAS: Reinforced Evolutionary Neural Architecture Search. In *Proceedings of IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, 4782–4791. https://doi.org/10.1109/CVPR.2019.00492

[13] Javier Civit-Masot, Francisco Luna-Perejón, Manuel Domínguez-Morales, and Anton Civit. 2020. Deep Learning System for COVID-19 Diagnosis Aid Using X-ray Pulmonary Images. *Applied Sciences* 10, 13 (2020). https://doi.org/10.3390/app10134640

[14] Joseph Paul Cohen, Paul Morrison, Lan Dao, Karsten Roth, Tim Q Duong, and Marzyeh Ghassemi. 2020. COVID-19 Image Data Collection: Prospective Predictions Are the Future. *arXiv 2006.11988* (2020). https://github.com/ieee8023/covid-chestxray-dataset

[15] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan. 2002. A fast and elitist multiobjective genetic algorithm: NSGA-II. *IEEE Transactions on Evolutionary Computation* 6, 2 (2002). https://doi.org/10.1109/4235.996017

[16] Travis Desell. 2017. Large scale evolution of convolutional neural networks using volunteer computing. In *Proceedings of the Genetic and Evolutionary Computation Conference Companion*. Association for Computing Machinery, 127–128. https://doi.org/10.1145/3067695.3076002

[17] A.E. Eiben and J.E. Smith. 2015. *Introduction to Evolutionary Computing* (2 ed.). Springer.

[18] Chrisantha Fernando, Dylan Banarse, Malcolm Reynolds, Frederic Besse, David Pfau, Max Jaderberg, Marc Lanctot, and Daan Wierstra. 2016. Convolution by Evolution: Differentiable Pattern Producing Networks. In *Proceedings of the Genetic and Evolutionary Computation Conference*. Association for Computing Machinery, New York, NY, USA, 109–116. https://doi.org/10.1145/2908812.2908890

[19] Faustino Gomez, Jürgen Schmidhuber, and Risto Miikkulainen. 2006. Efficient Non-Linear Control Through Neuroevolution. In *European Conference on Machine Learning 2006*. Springer, 654–662. https://doi.org/10.1007/11871842_64

[20] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep Residual Learning for Image Recognition. In *2016 Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, 770–778. https://doi.org/10.1109/CVPR.2016.90

[21] Gao Huang, Zhuang Liu, Laurens van der Maaten, and Kilian Q. Weinberger. 2017. Densely Connected Convolutional Networks. In *2017 Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, 4700–4708.

[22] Sergey Ioffe and Christian Szegedy. 2015. Batch normalization: accelerating deep network training by reducing internal covariate shift. In *Proceedings of the 32nd International Conference on International Conference on Machine Learning*, Vol. 37. JMLR, 448–456.

[23] William Irwin-Harris, Yanan Sun, Bing Xue, and Mengjie Zhang. 2019. A Graph-Based Encoding for Evolutionary Convolutional Neural Network Architecture Design. In *2019 IEEE Congress on Evolutionary Computation (CEC)*. IEEE, 546–553. https://doi.org/10.1109/CEC.2019.8790093

[24] Md. Zabirul Islam, Md. Milon Islam, and Amanullah Asraf. 2020. A combined deep CNN-LSTM network for the detection of novel coronavirus (COVID-19) using X-ray images. *Informatics in Medicine Unlocked* 20 (2020). https://doi.org/10.1016/j.imu.2020.100412

[25] Francisco Erivaldo Fernandes Junior and Gary G. Yen. 2019. Particle swarm optimization of deep neural networks architectures for image classification. *Swarm and Evolutionary Computation* 49 (2019), 62–74. https://doi.org/10.1016/j.swevo.2019.05.010

[26] Diederik P. Kingma and Jimmy Ba. 2015. Adam: A Method for Stochastic Optimization. In *Proceedings of the 3rd International Conference on Learning Representations (ICLR)*.

[27] Alex Kizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. 2012. ImageNet Classification with Deep Convolutional Neural Networks. In *Advances in Neural Information Processing Systems 25*, Vol. 25. Curran Associates, Inc., 1097–1105. https://doi.org/10.1109/WACV.2015.71

[28] Zhichao Lu, Ian Whalen, Vishnu Boddeti, Yashesh Dhebar, Kalyanmoy Deb, Erik Goodman, and Wolfgang Banzhaf. 2019. NSGA-Net: Neural Architecture Search using Multi-Objective Genetic Algorithm. In *Proceedings of the Genetic and Evolutionary Computation Conference*. Association for Computing Machinery, 419–427. https://doi.org/10.1145/3321707.3321729

[29] Brad L. Miller and David E. Goldberg. 1995. Genetic Algorithms, Tournament Selection, and the Effects of Noise. *Complex Systems* 9, 3 (1995).

[30] David J. Montana and Lawrence Davis. 1989. Training feedforward neural networks using genetic algorithms. In *Proceedings of the 11th International Joint Conference on Artificial Intelligence*. Morgan Kaufmann Publishers Inc., 762–767.

[31] Yujin Oh, Sangjoon Park, and Jong Chul Ye. 2020. Deep Learning COVID-19 Features on CXR using Limited Training Data Sets. *IEEE Transactions on Medical Imaging* 39, 8 (2020), 2688–2700. https://doi.org/10.1109/TMI.2020.2993291

[32] Tulin Ozturk, Muhammed Talo, Eylul Azra Yildirim, Ulas Baran Baloglu, Ozal Yildirim, and U. Rajendra Acharya. 2020. Automated detection of COVID-19 cases using deep neural networks with X-ray images. *Computers in Biology and Medicine* 121 (2020). https://doi.org/10.1016/j.compbiomed.2020.103792

[33] Mohammad Rahimzadeh and Abolfazi Attar. 2020. A modified deep convolutional neural network for detecting COVID-19 and pneumonia from chest X-ray images based on the concatenation of Xception and ResNet50V2. *Informatics and Medicine Unlocked* 19 (2020). https://doi.org/10.1016/j.imu.2020.100360

[34] Esteban Real, Sherry Moore, Andrew Selle, Saurabh Saxena, Yutaka Leon Suematsu, Jie Tan, Quoc V. Le, and Alexey Kurakin. 2017. Large-scale evolution of image classifiers. In *Proceedings of the 34th International Conference on Machine Learning*, Vol. 70. JMLR, 2902–2911. https://doi.org/10.5555/3305890.3305981

[35] Ramprasaath R. Selvaraju, Michael Cogswell, Abhisehk Das, Ramakrishna Vedantam, Devi Parikh, and Dhruv Batra. 2017. Grad-CAM: Visual Explanations From Deep Networks via Gradient-Based Localization. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*. IEEE, 618–626.

[36] Ke Shang, Hisao Ishibuchi, Linjun He, and Lie Meng Pang. 2020. A Survey on the Hypervolume Indicator in Evolutionary Multi-objective Optimization. *IEEE Transactions on Evolutionary Computation* (2020). https://doi.org/10.1109/TEVC.2020.3013290

[37] Kenneth O. Stanley, Jeffrey Clune, Joel Lehman, and Risto Miikkulainen. 2019. Designing neural networks through neuroevolution. *Nature Machine Intelligence* 1, 1 (Jan. 2019), 24–35.

[38] Kenneth O. Stanley and Risto Miikkulainen. 2002. Evolving Neural Networks Through Augmenting Topologies. *Evolutionary Computation* 10, 2 (May 2002), 99–127. https://doi.org/0.1162/106365602320169811

[39] Marcin Suchorzewski. 2011. Evolving scalable and modular adaptive networks with Developmental Symbolic Encoding. *Evolutionary Intelligence* 4, 3 (May 2011), 145–163. https://doi.org/10.1007/s12065-011-0057-0

[40] Yanan Sun, Handing Wang, Bing Xue, Yaochu Jin, Gary G. Yen, and Mengjie Zhang. 2020. Surrogate-Assisted Evolutionary Deep Learning Using an End-to-End Random Forest-Based Performance Predictor. *IEEE Transactions on Evolutionary Computation* 24, 2 (2020), 350–364. https://doi.org/10.1109/TEVC.2019.2924461

[41] Yanan Sun, Bing Xue, Mengjie Zhang, and Gary G. Yen. 2019. Evolving Deep Convolutional Neural Networks for Image Classification. *IEEE Transactions on Evolutionary Computation* 24, 2 (2019), 394–407. https://doi.org/10.1109/TEVC.2019.2916183

[42] Yanan Sun, Bing Xue, Mengjie Zhang, and Gary G. Yen. 2020. Automatically Designing CNN Architectures Using Genetic Algorithm for Image Classification. *IEEE Transactions on Cybernetics* (2020), 1–5. https://doi.org/10.1109/TCYB.2020.2983860

[43] S. Tabik, A. Gómez-Ríos, J. L. Martín-Rodríguez, I. Sevillano-García, M. Rey-Area, D. Charte, E. Guirado, J. L. Suárez, J. Luengo, M. A. Valero-González, P. García-Villanova, E. Olmedo-Sánchez, and F. Herrera. 2020. COVIDGR dataset and COVID-SDNet methodology for predicting COVID-19 based on Chest X-Ray images. (2020). arXiv:eess.IV/2006.01409

[44] Mesut Togacar, Burhan Ergen, and Zafer Comert. 2020. COVID-19 detection using deep learning models to exploit Social Mimic Optimization and structured chest X-ray images using fuzzy color and stacking approaches. *Computers in Biology and Medicine* 121 (2020). https://doi.org/10.1016/j.compbiomed.2020.103805

[45] Ferhat Ucar and Deniz Korkmaz. 2020. COVIDiagnosis-Net: Deep Bayes-SqueezeNet based diagnosis of the coronavirus disease 2019 (COVID-19) from X-ray images. *Medical Hypotheses* 140 (2020). https://doi.org/10.1016/j.mehy.2020.109761

[46] Phillip Verbancsics and John Harguess. 2015. Image Classification Using Generative Neuro Evolution for Deep Learning. In *2015 IEEE Winter Conference on Applications of Computer Vision*. IEEE, 488–493. https://doi.org/10.1109/WACV.2015.71

[47] Bing Wang, Yanan Sun, Bing Xue, and Mengjie Zhang. 2019. A Hybrid GA-PSO Method for Evolving Architecture and Short Connections of Deep Convolutional Neural Networks. In *Proceedings of the Pacific Rim International Conference on Artificial Intelligence*. Springer, 650–663. https://doi.org/10.1007/978-3-030-29894-4_52

[48] Linda Wang, Alexander Wong, Zhong Qiu Lin, Paul McInnis, Audrey Chung, Hayden Gunraj, James Lee, Matt Ross, Blake VanBerlo, Ashkan Ebadi, Kim-Ann Git, and Abdul Al-Haimi. 2020. Actualmed COVID-19 Chest X-ray Dataset Initiatives. (2020). https://github.com/agchung/Actualmed-COVID-chestxray-dataset

[49] Lingxi Xie and Alan Yuille. 2017. Genetic CNN. In *The IEEE International Conference on Computer Vision (ICCV)*. IEEE, 1379–1388.

[50] Zhaohui Yang, Yunhe Wang, Xinghao Chen, Boxin Shi, Chao Xu, Qi Tian, and Chang Xu. 2020. CARS: Continuous Evolution for Efficient Neural Architecture Search. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE.

[51] Yao Zhou, Gary G. Yen, and Zhang Yi. 2020. Evolutionary Compression of Deep Neural Networks for Biomedical Image Segmentation. *IEEE Transactions on Neural Networks and Learning Systems* 31, 8 (2020), 2916–2929. https://doi.org/10.1109/TNNLS.2019.2933879