Neural networks optimized by genetic algorithms in cosmology

Isidro Gómez-Vargas[*]
Instituto de Ciencias Físicas, Universidad Nacional Autónoma de México, 62210, Cuernavaca, Morelos, México

Joshua Briones Andrade[†]
Facultad de Ciencias, Universidad Nacional Autónoma de México, Ciudad de México, México

J. Alberto Vázquez[‡]
Instituto de Ciencias Físicas, Universidad Nacional Autónoma de México, 62210, Cuernavaca, Morelos, México

(Received 13 September 2022; accepted 13 January 2023; published 8 February 2023)

The applications of artificial neural networks in the cosmological field have shone successfully during the past decade, this is due to their great ability of modeling large amounts of datasets and complex nonlinear functions. However, in some cases, their use still remains controversial because their ease of producing inaccurate results when the hyperparameters are not carefully selected. In this paper, to find the optimal combination of hyperparameters to artificial neural networks, we propose to take advantage of the genetic algorithms. As a proof of the concept, we analyze three different cosmological cases to test the performance of the architectures achieved with the genetic algorithms and compare them with the standard process, consisting of a grid with all possible configurations. First, we carry out a model-independent reconstruction of the distance modulus using a type Ia supernovae compilation. Second, the neural networks learn to infer the equation of state for the quintessence model, and finally with the data from a combined redshift catalog the neural networks predict the photometric redshift given six photometric bands (urgizy). We found that the genetic algorithms improve considerably the generation of the neural network architectures, which can ensure more confidence in their physical results because of the better performance in the metrics with respect to the grid method.

DOI: 10.1103/PhysRevD.107.043509

I. INTRODUCTION

Throughout cosmology there exists a variety of numerical and statistical techniques that allow to study complex theoretical models and to process large amounts of observational measurements. However, despite the increasing amount of observational evidence, there are still several tensions with unknown physical explanations which encourage the search for new analysis tools to extract valuable information from the data, and to use the computational resources more efficiently. During the past decades, machine learning incorporated effective alternatives into the data analysis field, in particular the deep learning. For instance, the artificial neural networks (ANNs) have been successfully used to carry out a broad diversity of different tasks such as regression, classification, image processing and time series, among many others [1]. In cosmology, they have achieved great results in performing model-independent reconstructions (nonlinear regression) [2–6], in the process of speeding up numerical calculations [7–11] or in the classification of different objects [11–14]. Nevertheless, in most of these studies the architecture of the network is built up by generating a grid with a large number of possible combinations of hyperparameter values in order to select the best one, which could be computationally expensive; and in other scenarios the architecture is constructed on an empirical way, which may lead to inaccurate results.

Despite their great advantages, ANNs have two main drawbacks. First, the fact that artificial networks have thousands or millions of parameters (called weights) generates a hard interpretation of them. Second, ANNs have also several hyperparameters that must be picked out carefully (e.g., number of layers, nodes, activation function, batch size, etc.) in order to have acceptable predictions and therefore the results depend on their selection. That is, even though several combinations are able to generate a good neural network model, there are even more bad

[*]igomez@icf.unam.mx
[†]joshuabriones@ciencias.unam.mx
[‡]javazquez@icf.unam.mx

 © 2023 American Physical Society

combinations that will achieve incorrect predictions and, in cosmology, a spurious and imprecise physical interpretation. If the hyperparameters configuration is adequate, there is a good balance between the bias and variance of the neural model which implies the network is neither overfitted nor underfitted [15], therefore the predictions should be reliable and thus underestimate the weak interpretation of the multiple weights.

There are several strategies for finding the appropriate values of the hyperparameters in a neural network [16–19]. The standard approach is to propose a multidimensional grid form of several values of the hyperparameters [16], evaluate all possible combinations and determine, by comparison, which combination has the best performance. Newer approaches are based in mathematical optimization or metaheuristic algorithms, both of which consist in specialized algorithms to find the optimal value for a given function. Therefore, the search for the best combination of hyperparameters of neural networks is posed as an optimization problem. Genetic algorithms, by themselves, have been investigated in cosmology for quasar parametrization [20], nonparametric reconstructions [21,22], and string theory analysis [23], to name just a few. In other research areas, there are various cases in which artificial neural networks and genetic algorithms have been applied together; for example, in medicine [24–26], seismology [27] and in string theory [28], among others.

In several fields it is still uncommon to pay particular attention to the hyperparameter tuning during the construction of the neural network model. Regularly, an usual strategy is the hyperparameter grid and in several cases may be a good enough option, however there are better ways to optimize this selection. In this paper, we explore the use of the genetic algorithms (GAs), the most popular metaheuristic algorithm, and compare their performance with the standard grid of hyperparameters. In this context, the goal of the genetic algorithm is to find the best combination of neural network hyperparameters that minimizes the target function.

This paper is structured as follows. In the next section we provide a brief overview about neural networks, genetic algorithms and the hyperparameter tuning approaches. In Sec. III we explain the technical details about our implementation. Section IV, as a proof of the concept, contains three cosmological study cases, in the first one we made a model-independent reconstruction of the distance modulus using a type Ia supernovae compilation; in the second, we train neural network models using the analytical results of the equation of state of the quintessence model; and last, a prediction of the photometric redshift given some features from the catalog of Ref. [29]. In Sec. V we describe our final remarks about this research. In addition, in the Appendix we consider some randomness effects in the case study of Sec. IV A to verify the robustness of the genetic algorithms in the context of this work.

II. MACHINE LEARNING BACKGROUND

Machine learning is the field of artificial intelligence focused on the mathematical modeling of the data; it extracts the intrinsic properties of datasets by minimizing an objective function through many iterations until an acceptable combination of model parameters is found. In recent years, the most successful types of machine learning models are artificial neural networks, which have thousands or millions of parameters, called weights, which allow modeling any nonlinear function [30]. Finding the correct combination of hyperparameters, or in other words the best neural network architecture, is a hard task. The classic method for this is to generate several combinations of hyperparameters and evaluate all of them until the best one is found. In recent years, by taking advantage of the existing computing power, various optimization and parameter estimation techniques have been applied to find these hyperparameters more efficiently; in particular, the metaheuristic optimization algorithms (i.e., the genetic algorithm), which allow finding the best solution to an optimization problem without using derivatives.

In this section we describe very briefly artificial neural networks, genetic algorithms and the hyperparameter tuning.

A. Artificial neural networks

Artificial neural networks have been applied in various scientific fields because of their ability to model large and complex datasets. The universal approximation theorem guarantees that ANNs can model any nonlinear function [30], making them a powerful tool in modeling datasets where the intrinsic relationships of their variables are unknown and most of the time multidimensional. A complete review of neural networks is beyond the scope of this article; there are great references in the literature to delve into this topic in a formal way [1,31,32]; for a basic introduction, their main algorithms and a cosmological context, see [11].

Inspired by nature, an artificial neural network (ANN) consists of a computational model that aims to emulate the synapse through interconnected layers of units called neurons or nodes, which make up its basic information processing elements. In the simplest type of network, the feedforward neural network, there are three types of layers: an input layer that receives the initial information, hidden layers responsible for extracting patterns and producing nonlinearity effects, and finally the output layer that presents the results of the prediction.

The intrinsic parameters of ANNs are known as hyperparameters, which, unlike the weights, are not adjusted during the training and must be configured in advance. Examples of hyperparameters are the number of layers, the number of nodes per layer, the number of epochs and the activation function. In addition, since ANNs use a gradient descent and a backpropagation algorithm [33],

the parameters of these algorithms can also be hyperparameters of the ANN model, e.g., batch size and learning rate, among others. In practice, some hyperparameters are fixed and others remain as free parameters that are found by a tuning strategy.

B. Genetic algorithms

Genetic algorithms are inspired by genetic populations which consider any possible solution of an optimization problem as an individual, or chromosome. They are popular for their ability to solve large-scale nonlinear and nonconvex optimization problems [34] and for difficult search situations [35]. It is beyond the scope of this article for a full background of genetic algorithms, but for interested readers we recommend the following references [36–38].

Inspired by genetic populations, the first step of the algorithm is to generate several individuals, within the search space, and define them as a population. Then, through operations such as crossover, mutation and selection over several iterations (called generations) the population gets closer to the optimum of a target function. In any problem approached with genetic algorithms, it is necessary to select the fitness, or target function to optimize, to define the search space and to set the genetic operators (selection, crossover and mutation).

Selection operation is the criterion used to determine which individuals, at given generation, will survive or reproduce in the next generation. There are several types of selection methods such as roulette wheel, stochastic universal sampling, ranking or tournament. In this paper, we use the tournament selection, for which, in each round two or more individuals, randomly selected, are confronted and the one with the highest fitness, or target function, wins and survives.

The crossover operation, also called recombination, interchanges the genes of two individuals generating an offspring, i.e., a new individual. Usually, the probability of crossover is high; but zero means that the two parents pass to the next generation without doing anything else. There are several crossover types, however in this work we focused in the uniform crossover: for each gene of the new individual it chooses randomly one of the parents.

On the other hand, mutation refers to a random change in one or more gene values in an individual. We use the bit flip mutation [39], which consists in the random selection of one gene to be flipped, i.e., to be switched to different value, for example 0 to 1 or vice versa. We need to avoid high values for the mutation probability, because it can become a random walk instead of an effective exploration of the search space.

At the beginning, it is necessary to assign probability values to the crossover, to the mutation operators and for each iteration two individuals can have a crossover or a single individual a mutation with these probabilities. The value of elitism indicates how many individuals are bound to pass to the next generation, so it is a positive integer value. Thus, in a few words the genetic algorithm works as follows: it generates an initial population within the search space and, generation by generation, the individuals are modified by the operators, and by evaluating the objective function the individuals are approaching the optimum of the target function.

C. Neural network hyperparameter tuning

To find a good combination of the neural network hyperparameters, we focus on two approaches: the classic grid of hyperparameters and the genetic algorithms. Here are some highlights of both.

1. Conventional grid

The typical approach to finding a correct combination of hyperparameters in the ANN is to go through an array of possible hyperparameter values and evaluate each combination to choose the best that minimizes the neural network loss function [16]. This involves training as many times as there are combinations, and it is very computationally expensive. Another technique that attempts to reduce this cost is a random search, in which hyperparameter combinations are randomly sampled; however, it still has the same problems and depends on the size of the search space for its efficiency. In both approaches the best solution is always within the initial set of combinations and, however, a lot of configurations have to be evaluated.

2. Using genetic algorithms

The search for the hyperparameters of an ANN can be considered an optimization problem. Because of the increased number of hyperparameters, the search space is likely to be complex and high dimensional. Classical optimization methods involving derivatives can be very difficult to implement in this kind of scenario, therefore genetic algorithms are a very interesting way to solve this problem.

The crucial step in using genetic algorithms in hyperparameter fitting is to define the fitness function, or target function. For the case of neural networks, the loss function can be used as a fitness. The loss function during neural network training aims to be minimized, therefore the task of genetic algorithms is to find the best combination of hyperparameters that minimizes the target function. Several research works, unrelated to cosmology, have already combined these two powerful tools and in most cases have promising results [40–43].

III. METHODOLOGY

We use TENSORFLOW [44] to program the ANN models and the DEAP library [45–47] to implement the genetic algorithms, both in PYTHON. We developed a PYTHON

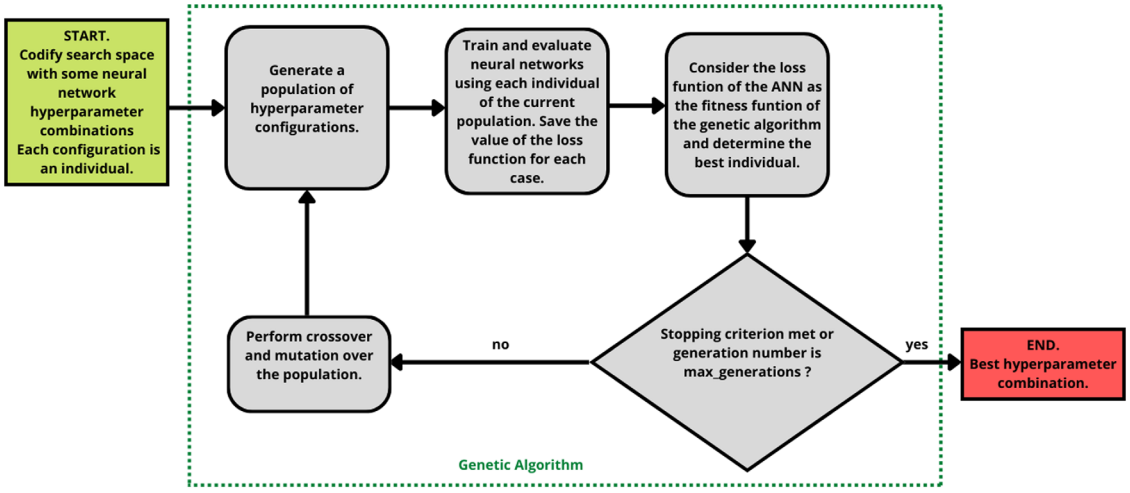

FIG. 1. Diagram of neural network hyperparameter tuning with a genetic algorithm.

library called NNOGADA[1] in which a simple genetic algorithm searches the best hyperparameters for a neural network.

In this framework, the target function of the genetic algorithm must be some metric of the artificial neural network; typically it would be the loss function, but for classification problems it could be accuracy, precision or something similar. In the case of the loss function, the problem would be a minimization and in the case of a classification metric, a maximization. Throughout all cases in our study, we use the loss function and therefore have minimization problems.

The first step is to define the hyperparameters of the neural network model to be found. In practice, there are some hyperparameters that have values recommended by the literature or by experienced users on certain types of problems. In this work, as variable hyperparameters we choose the number of layers, the number of nodes, the learning rate and the batch size. With these we define the search space of the optimization problem. The gradient descent algorithm in our neural network models is ADAM [48], and its hyperparameter that we tune is precisely the learning rate. Second, it is necessary to define the possible values for each hyperparameter, where the hyperparameter grid algorithm will search for the best combination, and those that the genetic algorithm will use to generate the first population.

In the case of the genetic algorithm, the possible values of the free hyperparameters must be encoded in a way that the genetic algorithm can understand them (binary, hexadecimal, etc.). Next, it is necessary to define the population size and the number of generations; also the probability of mutation, crossover and elitism. Once this configuration of the genetic algorithm is established, we can use the neural model as a function to optimize. To have a fair comparison, we chose these parameters of the genetic algorithms to have a number of neural network evaluations similar to the hyperparameter grid cases; the formal selection of the parameters of the genetic algorithm is out of the scope of this paper.

For the genetic algorithms, in all of the following case studies in Sec. IV, we set the tournament method [49] for selection with size 2, binary coding, using elitism with a hall of fame size equal to 1. We use the uniform crossover operator and bit flip mutation operator. We varied and tested different values of crossover and mutation probabilities, population size and number of generations with the goal to analyze their effect in the performance of the genetic algorithms. Figure 1 summarizes the implementation of the hyperparameter tuning with genetic algorithms.

In all cases, we split the datasets into training, validation and test sets. The first is used to adjust the weights during training, the validation set is used to evaluate the performance of each neural network model during training and, finally, the test set contains information not used in training but useful to measure the generalization capability of the neural networks. For the neural networks of the same example, we fix the number of epochs and report the lowest loss function for each case.

Once the neural networks were trained, to evaluate the performance of the models, we use the mean squared error (MSE), a measurement of the difference between the neural networks prediction and the real values. It is defined as follows:

$$\mathrm{MSE} = \frac{1}{n}\sum_{i}^{n}(Y_i - \hat{Y}_i)^2, \tag{1}$$

where Y_i is a vector with predictions of the ANN, $\hat{Y}_i$ a vector with the expected values, and n is the number of predictions (or the length of Y_i and $\hat{Y}_i$). Using the same

[1]Repository for NNOGADA (Neural Networks Optimized by Genetic Algorithms for Data Analysis) available at: https://github.com/igomezv/nnogada.

notation, we also use the mean absolute error (MAE) as a metric:

$$\frac{1}{n}\sum_{i}^{n}|(Y_i - \hat{Y}_i)|. \tag{2}$$

The last metric we use for our regression cases of study is the coefficient of determination, also called $\boldsymbol{R^2}$, which provides information about the goodness of fit of a model and it is defined as

$$R^2 = 1 - \frac{\sum_i^n (Y_i - \hat{Y}_i)^2}{\sum_i^n (Y_i - \bar{Y})^2}, \tag{3}$$

where $\bar{Y}$ is the mean of the observed data.

IV. CASES OF STUDY

We have chosen three different cosmological problems to test the hyperparameter tuning with genetic algorithms and in all of them we use the type of neural networks known as feedforward networks (also called multilayer perceptrons); however, the method can be easily implemented in other cosmological scenarios and with more complex neural network architectures such as convolutional or recurrent. In the following examples, we compare the performance of a hyperparameter grid and the genetic algorithms for finding an acceptable combination of hyperparameters in neural network models. We test these two strategies in three different cosmological contexts, similar to the problems presented in [11]. Table I shows our choice of possible hyperparameter values for each of the following subsections; we use these hyperparameters to construct the grids and the initial populations for the genetic algorithms.

A. Reconstruction of distance modulus

Model-independent cosmological reconstructions refer to the use of statistical or computational techniques to generate a model of a cosmological observable without assuming an underlying theory. In cosmology, several techniques have been used such as histogram density estimators [50], principal component analysis [51,52], smoothed step functions [53], Gaussian processes [54–57], extrapolation methods [58], Bayesian nodal free-form methods [59,60], evolutionary algorithms [21,22,61] and, recently, neural networks [2–4,6].

In this example, we perform a reconstruction of the distance modulus $\mu(z)$, using data from the joint lightcurve analysis (JLA), with 740 type Ia supernovae [62], which already have been approached with this type of computational models [2–4], sometimes with a grid of hyperparameters for tuning the neural network architecture and others without any criterion. We assume a spatially flat universe, for which the relationship between the luminosity distance d_L and the comoving distance $D(z)$ is given by

$$d_L(z) = \frac{1}{H_0}(1+z)D(z), \quad \text{with} \quad D(z) = H_0 \int \frac{dz}{H(z)}. \tag{4}$$

Thus, the observable quantity is computed by the distance modulus $\mu(z) = 5\log\, d_L(z) + 25$.

The JLA SNeIa compilation attributes are the redshift of the measurement, the distance modulus and its statistical error; in addition, it has a covariance matrix with the systematic errors. We employ the diagonal of the covariance matrix, and with associate errors we add them to the statistical error. Then our neural network should have only one node in the input layer, corresponding to the redshift z and two nodes in the output layer, for the distance modulus and the error (statistical plus systematic).

The hyperparameters have been searched by training several architectures using the SNIa from the JLA compilation, varying the number of layers, number of nodes, batch size and learning rate, for the values shown in the second column of Table I. Then, the grid of hyperparameters evaluates 128 different neural network architectures to determine the best. For the genetic algorithms, as can be seen in Table II, cases A, B and C are configurations with different values for the mutation probability, crossover probability, population size and number of generations.

In all cases the models were trained along 200 epochs, considering the rectifier linear unit (ReLU) activation function in all the hidden layers and the linear function in the last one. In Table II can be noticed the result for the hyperparameter tuning using a grid and the genetic algorithms (cases A, B and C). For the test set, we use the mean squared error (MSE), the mean absolute error (MAE) and the R^2 test to evaluate the performance of the neural network models. We can notice an improvement on the performance for the results of the genetic algorithms, even

TABLE I. Hyperparameters for the three cases of Sec. IV.

	Section IV A	Section IV B	Section IV C
Number of layers	[1, 2, 3, 4]	[1, 2, 3, 4]	[3, 4]
Number of nodes	[50, 100, 150, 200]	[50, 100, 150, 200]	[100, 200]
Learning rate	$[10^{-4}, 10^{-3}]$	$[10^{-4}, 10^{-3}]$	$[10^{-4}, 10^{-3}]$
Batch size	[2, 4, 8, 16]	[8, 16]	[8, 16, 32, 64]

TABLE II. Results of neural nets training with JLA compilation.

	Grid	Genetic A	Genetic B	Genetic C
Population size	⋯	5	5	5
Max generations	⋯	10	10	15
Crossover	⋯	0.5	0.5	0.5
Mutation	⋯	0.5	0.2	0.4
Hyperparameter results				
Layers	4	2	4	4
Nodes	200	100	100	000
Learning rate	0.0001	0.0001	0.0001	0.0001
Batch size	16	2	4	4
Metrics				
MSE	0.0371	0.0311	0.0314	0.0336
MAE	0.1165	0.09978	0.0985	0.1059
R^2	0.4968	0.6735	0.6797	0.7251
Evaluations	128	41	34	50

when it required less evaluations of neural network architectures. Because in this problem the numerical precision is relevant, we can see in Fig. 2 that indeed the genetic algorithms obtain better reconstructions for the distance modulus, particularly in the lower redshifts values. In the Appendix we validate these results through more repetitions of the algorithms.

Our results suggest that, in previous cosmological works in which the reconstructions are performed with neural networks, there is a possibility that a better architecture could be found using genetic algorithms instead of the use of the hyperparameter grid, and evidently for cases where no strategy is employed to find the correct architecture.

B. Quintessence equation of state

In this case, we test the capability of neural networks to modeling the equation of state (EOS) of an scalar field ϕ for the quintessence model, which is computed from a set of differential equations. This is a common and nontrivial problem because it involves a shooting method to find the right initial conditions (see an example in [63]). In particular, the EOS of the quintessence model is defined as follows:

$$w(z) = \frac{\dot{\phi}^2 - 2V(\phi)}{\dot{\phi}^2 + 2V(\phi)}, \tag{5}$$

with a potential V and the derivative of the field $\dot{\phi}$, that, in a Friedman-Robertson-Walker background, satisfy the Klein-Gordon equation:

$$\ddot{\phi} + 3H\dot{\phi} + \frac{\partial V(\phi)}{\partial \phi} = 0, \tag{6}$$

where H is the Hubble parameter,

$$H^2 = \frac{1}{3}\left[\frac{1}{2}\dot{\phi}^2 + V(\phi) + \rho_M\right]. \tag{7}$$

For simplicity, we use the scalar field potential $V = \frac{1}{2}m_\phi^2\phi^2$, with m_ϕ as the mass of the field in units of $[3H_0]$, and ρ_M being the matter content (baryons, dark matter) that satisfies the continuity equation.

We compute the EOS for several redshifts, fixing $H_0 = 68.2$ and $\Omega_m = 0.3$ and varying values for the mass of the field m_ϕ, to generate a dataset of 2800 combinations. We use a modified version of the SIMPLEMC code [64,65] to calculate the EOS, including the method to search the initial conditions of the dynamical system [63]. Therefore, we train the neural networks to predict the EOS of the quintessence model given the redshift and the mass of the field.

Varying the hyperparameters shown in Table I (batch size, learning rate, number of layers and number of nodes), the combinations found with the grid method and the genetic algorithms are included in Table III.

All neural network models were trained with 100 epochs and using the ReLU activation function for all the hidden layers, and a linear function for the last layer. In Table III,

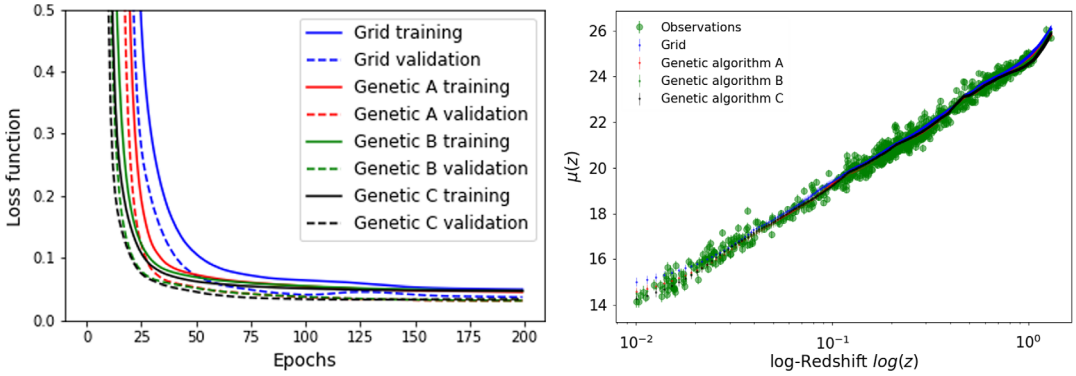

FIG. 2. Left: loss function behavior for the training of the neural networks using the JLA dataset. Right: distance modulus reconstruction.

TABLE III. Results of ANN training for the quintessence EOS.

	Grid	Genetic
Population size	⋯	8
Max generations	⋯	5
Crossover	⋯	0.8
Mutation	⋯	0.2
	Hyperparameter results	
Layers	4	4
Nodes	200	200
Learning rate	0.001	0.001
Batch size	16	8
	Metrics	
MSE	0.0008	1.0953×10^{-5}
MAE	0.0174	0.0020
R^2	0.9873	0.9998
Evaluations	64	34

we can see the results of the hyperparameter grid and a genetic algorithm. We can notice that the neural network obtained from the genetic algorithm search has a better performance in the MSE, MAE and R^2 metrics; in the case of MSE and MAE it improves with an order of magnitude, which is very significant. In addition, the genetic algorithms only used 34 neural networks to find the best, while the grid method evaluated 64 different architectures.

Figure 3 shows the behavior of the loss function for both methods and the ANN predictions of the EOS in comparison with the theoretical values. It is noticeable that the neural network based on the genetic algorithm performs better and their results are closer to the analytical predictions.

C. Photometric redshift prediction

In Ref. [29], the authors use a random forest regression to test a catalog that combines photometry from Canada-France-Hawaii Telescope Legacy Survey (CFHTLS), Y-band photometry from the Subaru Suprime camera, and spectroscopic redshifts from DEEP2, DEEP3 [66] and 3D-HST surveys [67]. In this study case, we use the same catalog, see details therein, to test the analysis with neural networks instead of random forest. We generate the six same variables used in [29] based on u,g,r,i,z and Y (see Table IV). Therefore, the input of the neural networks is these six variables and the output is the photometric redshift.

To find the best architecture of the network we use a grid of hyperparameters and genetic algorithms with the combinations shown in Table I. In this case, we vary three hyperparameters: number of layers, number of nodes by layer (the same for all the layers) and the learning rate, such as shown in Table I. The number of epochs is 200. The loss function fixed is the mean squared error, rectifier linear unit (ReLU) as an activation function in the hidden layers and the linear function in the last layer.

For this case, the neural network model has six nodes in the input layer (the attributes of Table IV) and one node in the output layer, corresponding to the photometric redshift. Using the hyperparameter grid varying the number of layers, number of nodes, batch size and learning rate (with the values shown in Table I), we have a total of 32 combinations. We choose three different configurations for the genetic algorithms regarding to the crossover and mutation probabilities, population size and generations.

Table V contains the results of the hyperparameter tuning with the grid and different configurations of genetic algorithms (cases A and B). We found that the genetic algorithms A and B, considering the values of the MSE, MAE and R^2, are better than the grid method. In both cases, the MSE and MAE are around half of the values obtained by the grid; and the R^2 score is also significantly higher.

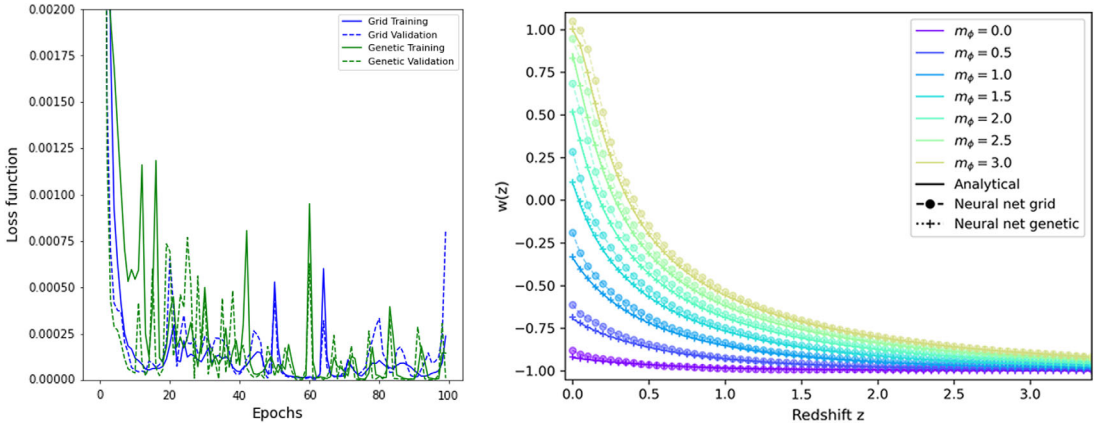

FIG. 3. Left: loss function behavior for the neural networks learning the equation of state for the quintessence model. Right: comparison of the analytical EOS for the quintessence model using different mass values and the predictions with the neural networks.

TABLE IV. Attributes of the combined catalog of Ref. [29].

Attribute	Description
u, g, r, i, z	UV, green, red, near-infrared, far-infrared filters from CFHTLS [68].
Y	Y-band photometry from the SuprimeCam at the Subaru telescope [69]
Redshift	Photometric redshift

TABLE V. Results of ANN training for photometric redshift.

	Grid	Genetic A	Genetic B
Population size	···	5	8
Max generations	···	10	5
Crossover	···	0.5	0.5
Mutation	···	0.2	0.4
Hyperparameter results			
Layers	4	4	4
Nodes	200	200	100
Learning rate	0.001	0.001	0.001
Batch size	64	16	16
Metrics			
MSE	0.0356	0.0154	0.0193
MAE	0.1074	0.0583	0.0571
R^2	0.7057	0.8608	0.8192
Evaluations	32	31	25

During the training, in the two cases, the genetic algorithms have a lower value in the loss function; we plot them in Fig. 4. The second panel of Fig. 4 shows the predicted redshift versus the true redshift and the three results are visually similar to those presented in Ref. [29], and the predictions of the neural networks proposed by the genetic algorithms have less dispersion than that corresponding to the grid method.

V. DISCUSSION

Throughout the three cases in this work, we conclude that the use of the genetic algorithms is an interesting strategy to find a correct neural network architecture in a cosmological context.

It is noteworthy that when genetic algorithms use higher mutation probability values, fewer neural network architectures are evaluated, more variance is induced in individuals between subsequent generations and this may allow a good combination of hyperparameters to be found more quickly. We can notice the effect of the mutation probability in the number of evaluations, smaller mutation probability leads to more neural network evaluations. However, we must remember that higher values of the mutation probability can become a random walk and should be avoided.

We have observed that the hyperparameter grid evaluates each combination just once, regardless of its proximity to the optimal value. In contrast, the genetic algorithms evaluate more times the configurations that are within a region of the search space where the optimum is likely to be, they may even evaluate the same point more than once; for example, the mutation or crossover of two different individuals could generate the same new point. This behavior makes the solution found by the genetic algorithms more reliable because the individuals of the last population have been tested better than the configurations

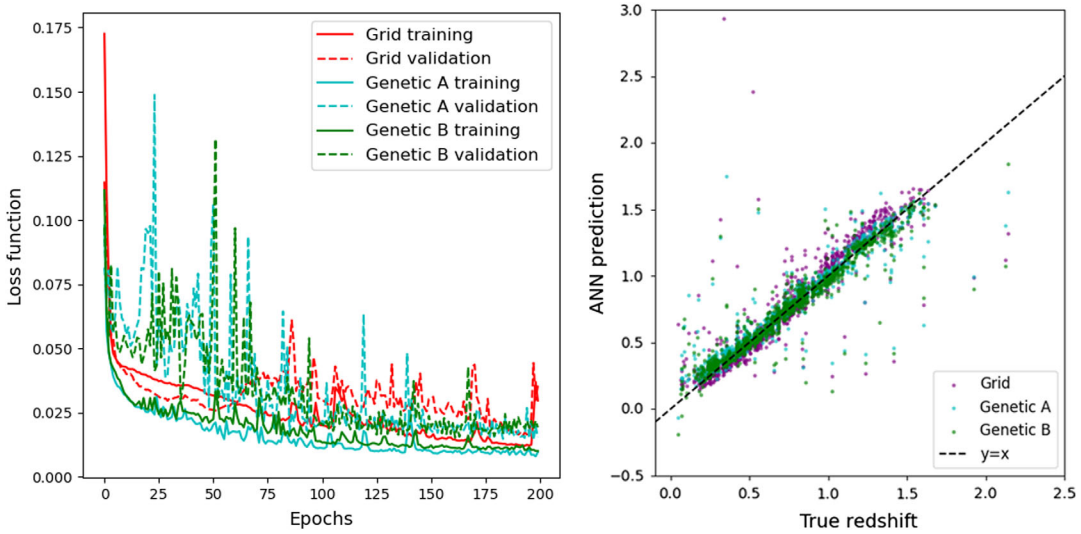

FIG. 4. Left: behavior of the loss functions. Right: predicted photometric redshifts by the ANNs in comparison with the true spectroscopic redshifts.

evaluated within the hyperparameter grid framework. In general, as can be noticed in the more extensive exploration of Table VI, the genetic algorithms found better solutions than the grid method.

We tested the genetic algorithms with a relatively small population size and number of generations and, in the majority of the analyzed cases, their performances were very competitive and even better than the traditional grid; we validate this observation with the complementary analysis in the Appendix. In this paper we have tested the hyperparameter tuning with genetic algorithms only with feedforward neural networks and in three very specific examples; nevertheless, this same methodology can be used in other cosmological applications and with any other types of neural networks. Moreover, in cosmological scenarios where more numerical precision, more complex architectures or larger search space size (i.e., more number of hyperparameters) are required, then the genetic algorithms are still expected to perform very well.

Another remark is that the process of running a genetic algorithm to find the hyperparameters of a neural network can be slow, if we increase the initial population and decrease the mutation probability, it can be similar to the hyperparameter grid method (see Table VII). However, in the current times of precision cosmology we believe it is a necessary cost to obtain better neural network models for the observational evidence.

ACKNOWLEDGMENTS

J. A. V. acknowledges the support provided by SEP-CONACYT sectoral funds Investigación Básica A1-S-21925, Ciencias de Frontera CONACYT-PRONACES/304001/202 and UNAM-DGAPA-PAPIIT IA104221. I. G. V. acknowledges support of a CONACYT postdoctoral fellowship, the support of the ICF-UNAM, R. Medel Esquivel for the discussions on genetic algorithms and Viviana Acquaviva for her workshop on machine learning at the VIII Essential Cosmology for the Next Generation.

TABLE VI. Ten tests for each method using the JLA SNe-Ia dataset.

Test Number	Layers	Nodes	Learning rate	Batch size	Evaluations	MSE	MAE	R^2
				Grid				
1	4	200	0.00010	16	128	0.0371	0.1165	0.4968
2	2	50	0.00100	16	128	0.0370	0.1141	0.6914
3	4	150	0.00010	16	128	0.0361	0.1133	0.5394
4	3	200	0.00100	16	128	0.0350	0.1093	0.4285
5	4	100	0.00100	16	128	0.0448	0.1222	0.4994
6	3	200	0.00100	16	128	0.0807	0.1714	0.4531
7	4	150	0.00010	16	128	0.0409	0.1189	0.6572
8	3	200	0.00100	16	128	0.0678	0.1478	0.6657
9	3	200	0.00100	16	128	0.0369	0.1101	0.4174
10	4	100	0.00100	16	128	0.0340	0.1035	0.6679
Average ± standard deviation					128 ± 0.0	0.0450 ± 0.016	0.1227 ± 0.021	0.5517 ± 0.109
				Genetic A				
1	2	100	0.00010	2	41	0.0311	0.0998	0.6735
2	4	50	0.00010	2	41	0.0363	0.1069	0.5894
3	2	200	0.00010	2	36	0.0315	0.0996	0.6893
4	2	100	0.00010	2	36	0.0313	0.0997	0.6866
5	4	150	0.00010	4	39	0.0340	0.1049	0.7091
6	3	100	0.00010	2	33	0.0363	0.1070	0.6357
7	2	100	0.00010	2	38	0.0316	0.0998	0.6515
8	1	50	0.00100	2	37	0.0323	0.1023	0.4249
9	2	200	0.00010	2	37	0.0316	0.0997	0.6787
10	2	200	0.00100	8	31	0.0411	0.1157	0.6956
Average ± standard deviation					37 ± 3	0.0337 ± 0.003	0.104 ± 0.005	0.6434 ± 0.084
				Genetic B				
1	2	200	0.00010	2	34	0.0314	0.0985	0.6797
2	4	200	0.00010	2	36	0.0470	0.1285	0.7054
3	4	150	0.00010	2	36	0.0407	0.1177	0.6659

(Table continued)

TABLE VI. *(Continued)*

Genetic B								
4	3	100	0.00010	4	32	0.0333	0.1009	0.6912
5	4	100	0.00100	4	31	0.0515	0.1449	0.0920
6	4	100	0.00010	4	28	0.0342	0.1075	0.6933
7	3	100	0.00010	4	35	0.0343	0.1029	0.6792
8	3	200	0.00100	16	31	0.0386	0.1102	0.6795
9	4	50	0.00010	4	29	0.0333	0.1021	0.7204
10	2	100	0.00010	2	22	0.0305	0.0982	0.6779
Average ± standard deviation					31 ± 4	0.0375 ± 0.007	0.1111 ± 0.015	0.6285 ± 0.189
Genetic C								
1	4	100	0.00010	4	50	0.0336	0.1059	0.7251
2	1	200	0.00100	4	38	0.0339	0.1043	0.3435
3	4	50	0.00010	4	51	0.0330	0.1021	0.4566
4	2	100	0.00010	2	48	0.0314	0.0980	0.6970
5	2	200	0.00100	8	52	0.0357	0.1120	0.5074
6	1	200	0.00100	4	55	0.0319	0.1015	0.3628
7	4	100	0.00010	4	52	0.0328	0.1039	0.6673
8	2	150	0.00010	2	48	0.0317	0.0989	0.6083
9	3	100	0.00010	4	40	0.0329	0.1022	0.6479
10	2	100	0.00010	2	49	0.0338	0.1006	0.6747
Average ± standard deviation					48 ± 5	0.0331 ± 0.001	0.1029 ± 0.004	0.5691 ± 0.141

APPENDIX: NOTE ABOUT THE RANDOMNESS

The reader may have noticed that the performance of the genetic algorithms, in the above three examples, is always better than the hyperparameter grid. A valid question may be why; if the grid considers all combinations of a proposed set of hyperparameters and hence the individuals considered in the genetic algorithms are a subset of the grid. The cause of this apparent problem is the randomness of all the processes involved; even fixing the random seed, the values of the weights in the neural networks and the value of the mean square error at the end of training are very similar, but not exactly the same due to their random nature and the way computers generate the pseudorandom numbers. On the other hand, the hyperparameter grid only evaluates each combination once, while the genetic algorithm can evaluate the same individual several times over generations, so the genetic algorithm obtains the best solution even considering the small fluctuations related to the randomness of the process.

In order to validate the previous statement, we have performed 10 times the example of Sec. IV A. Genetic algorithms A, B and C refer to the values of Table II, in which all of them have a crossover probability of 0.5 and a mutation probability of 0.5, 0.2 and 0.4, respectively. Results are shown in Table VI. In general, we can notice

TABLE VII. Tests for a genetic algorithm with 15 individuals as population, mutation 0.2, crossover 0.5 and with ten maximum number of generations.

Run Number	Layers	Nodes	Learning rate	Batch size	Evaluations	MSE	MAE	R^2
1	3	200	0.0001	4	108	0.0313	0.0996	0.6414
2	4	50	0.0001	4	110	0.0320	0.0997	0.6893
3	1	100	0.001	4	106	0.0331	0.1016	0.606
4	1	100	0.001	4	95	0.0328	0.1028	0.4797
5	3	100	0.0001	4	112	0.0330	0.1027	0.6823
6	1	100	0.001	4	92	0.0322	0.1021	0.4873
7	3	100	0.0001	4	99	0.0318	0.1000	0.6242
8	3	100	0.0001	4	103	0.0338	0.1024	0.6372
9	1	100	0.001	4	83	0.0347	0.1027	0.6965
10	4	100	0.0001	4	92	0.0326	0.1047	0.6929
Average ± standard deviation					100 ± 9	0.0327 ± 0.001	0.1018 ± 0.002	0.6237 ± 0.080

that the performance of the genetic algorithms is better than the grid method and with less variance in their metrics.

A remarkable fact is the decrease in the number of evaluations for neural network architectures. However, because of the randomness, in a few cases the grid method can be better than the genetic algorithm. This issue can be tackled with more generations and a higher population in the genetic algorithm; for this purpose, see Table VII, the test uses the configuration of genetic algorithm B for Sec. IV A but with a population size of 15 and 10 as maximum number of generations. We can notice that the standard deviation decreases and in all ten runs the genetic algorithm is better than the grid method, shown in Table VI, and yet this genetic algorithm executes less evaluation of neural network architectures.

[1] Ian Goodfellow, Yoshua Bengio, Aaron Courville, and Yoshua Bengio, *Deep Learning* (MIT Press, Cambridge, MA, 2016), Vol. 1.

[2] Celia Escamilla-Rivera, Maryi A Carvajal Quintero, and Salvatore Capozziello, A deep learning approach to cosmological dark energy models, J. Cosmol. Astropart. Phys. 04 (2022) (016).

[3] Guo-Jian Wang, Xiao-Jiao Ma, Si-Yao Li, and Jun-Qing Xia, Reconstructing functions and estimating parameters with artificial neural networks: A test with a Hubble parameter and SNe Ia, Astrophys. J. Suppl. Ser. **246**, 13 (2020).

[4] Isidro Gomez Vargas, Ricardo Medel Esquivel, Ricardo Garcia, and J. Alberto Vazquez, Neural network reconstructions for the Hubble parameter, growth rate and distance modulus, arXiv:2104.00595.

[5] Hai-Nan Lin, Xin Li, and Li Tang, Non-parametric reconstruction of dark energy and cosmic expansion from the pantheon compilation of type ia supernovae, Chin. Phys. C **43**, 075101 (2019).

[6] Konstantinos Dialektopoulos, Jackson Levi Said, Jurgen Mifsud, Joseph Sultana, and Kristian Zarb Adami, Neural network reconstruction of late-time cosmology and null tests, J. Cosmol. Astropart. Phys. 02 (2022) 023.

[7] Philip Graff, Farhan Feroz, Michael P. Hobson, and Anthony Lasenby, Bambi: Blind accelerated multimodal Bayesian inference, Mon. Not. R. Astron. Soc. **421**, 169 (2012).

[8] Hector J Hortua, Riccardo Volpi, Dimitri Marinelli, and Luigi Malago, Accelerating MCMC algorithms through Bayesian deep networks, arXiv:2011.14276.

[9] Alessio Spurio Mancini, Davide Piras, Justin Alsing, Benjamin Joachimi, and Michael P. Hobson, Cosmopower: Emulating cosmological power spectra for accelerated Bayesian inference from next-generation surveys, Mon. Not. R. Astron. Soc. **511**, 1771 (2022).

[10] Isidro Gómez-Vargas, Ricardo Medel Esquivel, Ricardo García-Salcedo, and J. Alberto Vázquez, Neural network within a Bayesian inference framework, J. Phys. Conf. Ser. **1723**, 012022 (2021).

[11] Juan de Dios Rojas Olvera, Isidro Gómez-Vargas, and Jose Alberto Vázquez, Observational cosmology with artificial neural networks, Universe **8**, 120 (2022).

[12] Nathanaël Perraudin, Michaël Defferrard, Tomasz Kacprzak, and Raphael Sgier, Deepsphere: Efficient spherical convolutional neural network with healpix sampling for cosmological applications, Astron. Comput. **27**, 130 (2019).

[13] Emille EO Ishida, Machine learning and the future of supernova cosmology, Nat. Astron. **3**, 680 (2019).

[14] A. Moller and T. De Boissiere, Supernnova: An open-source framework for Bayesian neural network-based supernova classification, Mon. Not. R. Astron. Soc. **491**, 4277 (2020).

[15] Stuart Geman, Elie Bienenstock, and René Doursat, Neural networks and the bias/variance dilemma, Neural Comput. **4**, 1 (1992).

[16] Hugo Larochelle, Dumitru Erhan, Aaron Courville, James Bergstra, and Yoshua Bengio, An empirical evaluation of deep architectures on problems with many factors of variation, *Proceedings of the 24th International Conference Machine Learning* (ACM, New York, USA, 2007), pp. 473–480.

[17] Frank Hutter, Holger H. Hoos, Kevin Leyton-Brown, and Thomas Stützle, Paramils: An automatic algorithm configuration framework, J. Artif. Intell. Res. **36**, 267 (2009).

[18] Rémi Bardenet, Mátyás Brendel, Balázs Kégl, and Michèle Sebag, Collaborative hyperparameter tuning, *Proceedings of the 30th International Conference Machine Learning* (PLMR, Georgia, USA, 2013), Vol. 28, pp. 199–207.

[19] Xiang Zhang, Xiaocong Chen, Lina Yao, Chang Ge, and Manqing Dong, Deep neural network hyperparameter optimization with orthogonal array tuning, *International Conference on Neural Information Processing* (Springer International Publishing, Cham, 2019), pp. 287–295.

[20] Piotr Wasiewicz and Krzysztof Hryniewicz, Optimization of quasar parametrization using genetic algorithms, Photonics Applications in Astronomy, Communications, Industry, and High-Energy Physics Experiments **11176**, 944 (2019).

[21] Rubén Arjona, Alessandro Melchiorri, and Savvas Nesseris, Testing the ΛCDM paradigm with growth rate data and machine learning, J. Cosmol. Astropart. Phys. 05 (2022) 047.

[22] Rubén Arjona, Machine learning meets the redshift evolution of the CMB temperature, J. Cosmol. Astropart. Phys. 08 (2020) 009.

[23] Alex Cole, Andreas Schachner, and Gary Shiu, Searching the landscape of flux vacua with genetic algorithms, J. High Energy Phys. 11 (2019) 045.

[24] Lothar Terfloth and Johann Gasteiger, Neural networks and genetic algorithms in drug design, Drug Discov. Today **6**, 102 (2001).

[25] Amin Kabir Anaraki, Moosa Ayati, and Foad Kazemi, Magnetic resonance imaging-based brain tumor grades classification and grading via convolutional neural networks and genetic algorithms, Biocybern. Biomed. Eng. **39**, 63 (2019).

[26] Amin Kabir Anaraki, Moosa Ayati, and Foad Kazemi, Magnetic resonance imaging-based brain tumor grades classification and grading via convolutional neural networks and genetic algorithms, Biocybern. Biomed. Eng. **39**, 63 (2019).

[27] Gloria Curilem, Jorge Vergara, Gustavo Fuentealba, Gonzalo Acuna, and Max Chacón, Classification of seismic signals at villarrica volcano (chile) using neural networks and genetic algorithms, J. Volcanol. Geotherm. Res. **180**, 1 (2009).

[28] Fabian Ruehle, Evolving neural networks with genetic algorithms to study the string landscape, J. High Energy Phys. 08 (2017) 038.

[29] Rongpu Zhou, Michael C. Cooper, Jeffrey A. Newman, Matthew L. N. Ashby, James Aird, Christopher J. Conselice, Marc Davis, Aaron A. Dutton, S. M. Faber, Jerome J. Fang *et al.*, Deep ugrizy imaging and deep2/3 spectroscopy: A photometric redshift test bed for lsst and public release of data from the deep3 galaxy redshift survey, Mon. Not. R. Astron. Soc. **488**, 4565 (2019).

[30] Kurt Hornik, Maxwell Stinchcombe, and Halbert White, Universal approximation of an unknown mapping and its derivatives using multilayer feedforward networks, Neural Netw. **3**, 551 (1990).

[31] Christopher M. Bishop and Nasser M. Nasrabadi, *Pattern Recognition and Machine Learning* (Springer, New York, 2006), Vol. 4.

[32] Michael A. Nielsen, *Neural Networks and Deep Learning* (Determination Press, San Francisco, 2015), Vol. 25.

[33] David E. Rumelhart, Geoffrey E. Hinton, and Ronald J. Williams, Learning representations by back-propagating errors, Nature (London) **323**, 533 (1986).

[34] Kerry Gallagher and Malcolm Sambridge, Genetic algorithms: A powerful tool for large-scale nonlinear optimization problems, Comput. Geosci. **20**, 1229 (1994).

[35] S. N. Sivanandam and S. N. Deepa, *Genetic Algorithms* (Springer, New York, 2008).

[36] Colin R. Reeves, Genetic algorithms for the operations researcher, INFORMS J. Comput. **9**, 231 (1997).

[37] Sourabh Katoch, Sumit Singh Chauhan, and Vijay Kumar, A review on genetic algorithm: Past, present, and future, Multimedia Tools Appl. **80**, 8091 (2021).

[38] E. Wirsansky, *Hands-On Genetic Algorithms with Python: Applying Genetic algorithms to Solve Real-World Deep Learning and Artificial Intelligence Problems* (Packt Publishing, Birmingham, UK, 2020).

[39] Francisco Chicano, Andrew M. Sutton, L. Darrell Whitley, and Enrique Alba, Fitness probability distribution of bit-flip mutation, Evolutionary computation **23**, 217 (2015).

[40] Geoffrey F. Miller, Peter M. Todd, and Shailesh U. Hegde, Designing neural networks using genetic algorithms, in *ICGA: Proceedings of the Third International Conference on Genetic Algorithms* (Morgan Kaufmann Publishers, USA, 1989), Vol. 89, pp. 379–384.

[41] David J. Montana, Lawrence Davis *et al.*, Training feedforward neural networks using genetic algorithms, in IJCA: Proceedings of the 11th International Joint Conference on Artificial (Morgan Kaufmann Publishers, USA, 1989), Vol. **89**, pp. 762–767.

[42] Nurshazlyn Mohd Aszemi and P. D. D. Dominic, Hyperparameter optimization in convolutional neural network using genetic algorithms, Int. J. Adv. Comput. Sci. Appl. **10**, 269 (2019).

[43] Changxi Ma, Wei Hao, Fuquan Pan, and Wang Xiang, Road screening and distribution route multi-objective robust optimization for hazardous materials based on neural network and genetic algorithm, PLoS One **13**, e0198931 (2018).

[44] Martín Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard *et al.*, TensorFlow: A system for large-scale machine learning, in *12th USENIX Symposium on Operating Systems Design and Implementation (OSDI 16)* (USENIX, Georgia, USA, 2016), pp. 265–283.

[45] Félix-Antoine Fortin, François-Michel De Rainville, Marc-André Gardner, Marc Parizeau, and Christian Gagné, DEAP: Evolutionary algorithms made easy, J. Mach. Learn. Res. **13**, 2171 (2012).

[46] François-Michel De Rainville, Félix-Antoine Fortin, Marc-André Gardner, Marc Parizeau, and Christian Gagné, Deap: A python framework for evolutionary algorithms, in *Proceedings of the 14th Annual Conference Companion on Genetic and Evolutionary Computation* (ACM, New York, USA, 2012), pp. 85–92.

[47] François-Michel De Rainville, Félix-Antoine Fortin, Marc-André Gardner, Marc Parizeau, and Christian Gagné, Deap: Enabling nimbler evolutions, ACM SIGEVOlution **6**, 17 (2014).

[48] Diederik P. Kingma and Jimmy Ba, Adam: A method for stochastic optimization, arXiv:1412.6980.

[49] Anton Valentinovich Eremeev, A genetic algorithm with tournament selection as a local search method, J. Appl. Ind. Math. **6**, 286 (2012).

[50] Varun Sahni and Alexei Starobinsky, Reconstructing dark energy, Int. J. Mod. Phys. D **15**, 2105 (2006).

[51] Ranbir Sharma, Ankan Mukherjee, and H. K. Jassal, Reconstruction of late-time cosmology using principal component analysis, Eur. Phys. J. Plus **137**, 219 (2022).

[52] Luis A. Escamilla and J. Alberto Vazquez, Model selection applied to non-parametric reconstructions of the dark energy, arXiv:2111.10457.

[53] Francesca Gerardi, Matteo Martinelli, and Alessandra Silvestri, Reconstruction of the dark energy equation of state from latest data: The impact of theoretical priors, J. Cosmol. Astropart. Phys. 07 (2019) 042.

[54] Christopher K. I. Williams and Carl Edward Rasmussen, *Gaussian Processes for Machine Learning* (MIT Press, Cambridge, MA, 2006), Vol. 2.

[55] Ryan E. Keeley, Arman Shafieloo, Gong-Bo Zhao, Jose Alberto Vazquez, and Hanwool Koo, Reconstructing the universe: Testing the mutual consistency of the pantheon and SDSS/eBOSS BAO data sets with Gaussian processes, Astron. J. **161**, 151 (2021).

[56] Benjamin L'Huillier, Arman Shafieloo, David Polarski, and Alexei A. Starobinsky, Defying the laws of gravity I: Model-independent reconstruction of the universe expansion from growth data, Mon. Not. R. Astron. Soc. **494**, 819 (2020).

[57] Purba Mukherjee and Narayan Banerjee, Revisiting a non-parametric reconstruction of the deceleration parameter from combined background and the growth rate data, Phys. Dark Universe **36**, 100998 (2022).

[58] Ariadna Montiel, Ruth Lazkoz, Irene Sendra, Celia Escamilla-Rivera, and Vincenzo Salzano, Nonparametric reconstruction of the cosmic expansion with local regression smoothing and simulation extrapolation, Phys. Rev. D **89**, 043007 (2014).

[59] J. Alberto Vazquez, M. Bridges, M. P. Hobson, and A. N. Lasenby, Reconstruction of the dark energy equation of state, J. Cosmol. Astropart. Phys. 09 (2012) 020.

[60] Sonke Hee, J. A. Vázquez, W. J. Handley, M. P. Hobson, and A. N. Lasenby, Constraining the dark energy equation of state using Bayes theorem and the Kullback-Leibler divergence, Mon. Not. R. Astron. Soc. **466**, 369 (2017).

[61] Savvas Nesseris and Juan Garcia-Bellido, A new perspective on dark energy modeling via genetic algorithms, J. Cosmol. Astropart. Phys. 11 (2012) 033.

[62] M. Betoule, R. Kessler, J. Guy, J. Mosher, D. Hardin, R. Biswas, P. Astier, P. El-Hage, M. Konig, S. Kuhlmann *et al.*, Improved cosmological constraints from a joint analysis of the SDSS-II and SNLS supernova samples, Astron. Astrophys. **568**, A22 (2014).

[63] J. Alberto Vázquez, David Tamayo, Anjan A. Sen, and Israel Quiros, Bayesian model selection on scalar ϵ-field dark energy, Phys. Rev. D **103**, 043506 (2021).

[64] J. A. Vazquez, I. Gomez-Vargas, and A. Slosar, Updated version of a simple mcmc code for cosmological parameter estimation where only expansion history matters (2021), https://github.com/ja-vazquez/SimpleMC.

[65] Éric Aubourg, Stephen Bailey, Julian E. Bautista, Florian Beutler, Vaishali Bhardwaj, Dmitry Bizyaev, Michael Blanton, Michael Blomqvist, Adam S. Bolton, Jo Bovy *et al.*, Cosmological implications of baryon acoustic oscillation measurements, Phys. Rev. D **92**, 123516 (2015).

[66] Michael C. Cooper, Roger L. Griffith, Jeffrey A. Newman, Alison L. Coil, Marc Davis, Aaron A. Dutton, S. M. Faber, Puragra Guhathakurta, David C. Koo, Jennifer M. Lotz *et al.*, The deep3 galaxy redshift survey: The impact of environment on the size evolution of massive early-type galaxies at intermediate redshift, Mon. Not. R. Astron. Soc. **419**, 3018 (2012).

[67] Gabriel B. Brammer, Pieter G. Van Dokkum, Marijn Franx, Mattia Fumagalli, Shannon Patel, Hans-Walter Rix, Rosalind E. Skelton, Mariska Kriek, Erica Nelson, Kasper B. Schmidt *et al.*, 3D-HST: A wide-field grism spectroscopic survey with the Hubble space telescope, Astrophys. J. Suppl. Ser. **200**, 13 (2012).

[68] P. Hudelot, J. C. Cuillandre, K. Withington *et al.*, Vizier online data catalog (2012), p. 2317.

[69] Satoshi Miyazaki, Yutaka Komiyama, Maki Sekiguchi, Sadanori Okamura, Mamoru Doi, Hisanori Furusawa, Masaru Hamabe, Katsumi Imi, Masahiko Kimura, Fumiaki Nakata *et al.*, Subaru prime focus camera–suprime-cam, Publ. Astron. Soc. Jpn. **54**, 833 (2002).