Layers Sequence Optimizing for Deep Neural Networks using Multiples Objectives

Giuseppe Neto*, Péricles B.C. Miranda*, George D.C. Cavalcanti†, Tapas Si‡, Filipe Cordeiro*, Mayara Castro*
*Departamento de Computação, Universidade Federal Rural de Pernambuco
†Centro de Informática, Universidade Federal de Pernambuco
‡Department of Computer Science and Engineering, Bankura Unnayani Institute of Engineering

***Abstract*—Selecting the best architecture for a Deep Neural Network (DNN) is a non-trivial task since there is a massive amount of possible configurations (layers and parameters) and great difficulty in how to choose them. To make this task more independent of human interaction, this work addresses the DNN architecture selection problem as a multi-objective optimization task with different criteria in a combinatorial context. For this, we defined a new way to represent the architecture of DNN (layer sequence) as a solution in the optimization process. The proposed method attempts to find the best composition and sequence of layers for the DNN architecture satisfying two criteria: accuracy and F_1score. The method was evaluated for performance and compared to the exhaustive and random approaches and state-of-the-art DNN algorithms. The results obtained showed that the proposed method is capable of achieving results close to the optimum, and competitive when compared to those results reached by state of the art algorithms.**

***Index Terms*—Deep Neural Networks, Multi-objective optimization**

I. Introduction

The task of solving problems like the identification of objects in images and transcription of voice into text uses more and more a class of techniques called Deep Neural Networks (DNN) [1]. These techniques have been very successful in recent years, reaching out to humans in different tasks. However, these algorithms depend on their architecture (number of layers, the type of each layer, positioning of the layers, and parameters) to obtain a satisfactory performance, which raises the question: What is the most suitable DNN architecture for a given classification problem?

The three most widely used methods for architecture selection in DNNs are (1) manual search, (2) grid search, and (3) random search [2]. Manual search refers to the process of a researcher manually selecting architectures. This search requires in-depth knowledge about the problem and algorithm, being difficult for a non-specialist to define a good setup. Grid search is a computational procedure that evaluates all possible combinations of architectures for the DNN and is not efficient in searching solutions in a high-dimensional space. Finally, the random search attempts to find reasonable solutions by evaluating random solutions from the set of candidates. In this case, when the problem has a high-dimensional space, the chance of finding a suitable architecture for DNN by chance (randomly) is small. For this reason, there is a growing interest in the automatic and non-exhaustive selection of the architecture of DNNs using intelligent algorithms [3].

Some works have dealt with the DNN architecture selection as an optimization problem. Some of these works performed a single objective optimization of the different DNN's parameters, trying to maximize accuracy or minimize the root mean squared error (RMSE) [2], [4], [5]. Instead of considering only parameter optimization, other works optimized the DNN's layer composition and sequence [6]–[8], also considering one objective function.

Handling the current task as a single-objective optimization problem is not adequate [9]. The optimization of DNN's architecture is naturally a multi-objective problem where distinct objective functions need to be satisfied, such as accuracy, precision, recall, and others [9]. Few studies investigated the DNN's architecture selection as a multi-objective problem.

Liu et al. [10] proposed a Multi-Objective Convolutional Neural Network (MOCNN) algorithm for face labeling. The training process of this algorithm has an internal optimization procedure, which minimizes the losses of unary and pairwise terms, respectively, through a unified convolutional network. Miseikis et al. [11] adapted the MOCNN for robot localization and 3D position estimation in 2D camera images. Yang et al. [12] proposed the Multi-objective Pruning Evolutionary algorithm (MOPEA), which optimized the pruning of preexisting DNN architectures considering the error rate, computational cost, and sparsity as objective functions.

This work addresses the DNN architecture selection problem as a multi-objective optimization task with different criteria in a combinatorial context. For this, we formulated a new way to represent the architecture of DNN (layer sequence) as a solution in the optimization process. Thus, the proposed method attempts to find the best composition and sequence of layers for the DNN architecture. In this research, the optimization procedure maximizes the accuracy and the F_1-score simultaneously. The maximization of the accuracy favors the DNN to achieve the desired results. While the maximization of the F_1-score guarantees the relevance and significance of the results.

The contributions of this works are two-fold: i) the proposal of a multi-objective optimization to choose the best architecture for a DNN; ii) a novel candidate solution representation that characterizes the sequence of the layers. Moreover, an experimental study was carried out using different classes of problems and the results show that the proposed approach is more efficient and effective than the grid search (exhaustive),

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

the random search, and state-of-the-art algorithms.

This article is organized as follows: Section II introduces DNN and Multi-objective Optimization. Section III presents the related works. Section IV describes the proposed method. Section V presents the experimental methodology used to evaluate the proposed method. Section VI presents the results of the experimental study. Finally, Section VII highlights the conclusions and future work.

II. Background

This section presents fundamental concepts about DNN and Multi-objective optimization for a better understanding of the proposed work.

A. Deep learning

Different from conventional machine-learning algorithms, deep learning belongs to the class of methods called representative-learning methods. These methods allow a machine to be fed with raw data and to discover the representations needed for detection or classification automatically [1]. Deep-learning methods are representation-learning methods with multiple levels of representation, achieved by composing simple but non-linear modules that each modify the representation at one level (starting with the raw input) into a representation at a higher, slightly more abstract level. With the composition of enough such transformations, very complex functions can be learned.

For classification tasks, higher layers of representation amplify aspects of the input that are important for discrimination and suppress irrelevant variations [1]. Figure 1 shows an example of a DNN with different layers, which is a sequence of layers, where the i^{th} layer receives its input from the layer $i-1$, and its output serves as the input for the layer $i+1$. So, each layer trains a distinct set of features based on the previous layer. The more advanced the network, the more complex the characteristics that the layers may recognize since they aggregate and recombine characteristics of previous layers.

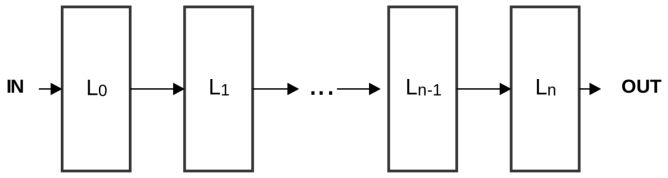

Fig. 1. Example of a DNN with different layers. Source: [3]

The architecture of a DNN is composed of different types and numbers of layers. The types of layers and where each one is positioned generate a great influence on the performance of the network. Herein, we discuss three popular kinds of layers: Convolutional, Pooling, and Dense. The Convolutional layer has a set of learnable filters. Convolution is the simple employment of a filter to an input that results in an "activation". Repeated application of the same filter to an input returns a map of activations called a feature map, indicating the locations and strength of a detected feature in input, such as an image. Pooling layers are responsible for calculating the mean or maximum local and making sub-samples of the data from the previous layer by reducing the size and selecting the most relevant features of the input [1]. The most common type of pooling used is the Max Pooling, which is a square-sized filter and stride chosen according to the problem, which results in each sub-selection of the image at the highest value within the selected region. A Dense layer is just a regular layer of neurons in a neural network. Each neuron receives input from all the neurons in the previous layer, thus densely connected. The layer has a weight matrix, a bias vector, and the activations of the previous layer.

B. Multi-objective Optimization

Multi-objective optimization (MOO) problems refer to problems that contain more than one conflicting objectives. MOO algorithms attempt to find solutions, in a search space of candidate solutions $\mathcal{S}$, which satisfy all the conflicting objectives ($\vec{f} = (f_1, f_2, ..., f_n)$) at the same time, where n is the number of objectives.

Each solution into $\mathcal{S}$ can be represented by a vector of decision variables ($\vec{x} = (x_1, x_2, ..., x_k)$), where k is the number of decision variables. Each solution has its quality (fitness) assessed through the execution of each objective function on it. Thus, $\vec{f}(\vec{x}) = (f_1(\vec{x}), f_2(\vec{x}), ..., f_n(\vec{x}))$ represents a vector of fitness values belonged to the solution $\vec{x}$.

A general MOO minimization problem can be defined as:

$$\text{minimize } \vec{f}(\vec{x}) := [f_1(\vec{x}), f_2(\vec{x}), ..., f_n(\vec{x})], \tag{1}$$

subject to:

$$g_i(\vec{x}) \leq 0 \quad i = 1, 2, ..., q, \tag{2}$$

$$h_j(\vec{x}) = 0 \quad j = 1, 2, ..., s, \tag{3}$$

where $\vec{x} = (x_1, x_2, ..., x_k)$ is the vector on the decision search space; and $g_i(\vec{x})$ and $h_j(\vec{x})$ are the constraint functions and $q + s$ is the number of constrains of the problem.

As in MOO, each solution's quality is represented by a vector of fitness values, the comparison between two different solutions to know which one is the best is different when compared to a single-objective optimization. In an MOO, the solutions are usually compared to each other through the Pareto dominance [13]. Given two vectors $\vec{x}, \vec{y} \in \mathbb{R}^n$, $\vec{x}$ dominates $\vec{y}$ (denoted by $\vec{x} \prec \vec{y}$) if $\vec{x}$ is better than $\vec{y}$ in at least one objective and $\vec{x}$ is not worse than $\vec{y}$ in any objective. $\vec{x}$ is not dominated if does not exist another current solution $\vec{x}_i$ in the current population, such that $\vec{x}_i \prec \vec{x}$. The set of non-dominated solutions in the objective space is known as Pareto front. Figure 2 shows some candidate solutions in a two dimensional optimization problem (two objective functions f_1 and f_2). Those solutions in the circle format are considered non-dominated among themselves, dominating the squared solutions. These circled solutions compose the Pareto front and are considered the best solutions among all [14].

For a better understanding of how MOO algorithms works, Algorithm 1 presents general steps of the whole search process. It is important to mention that there are MOO algorithms

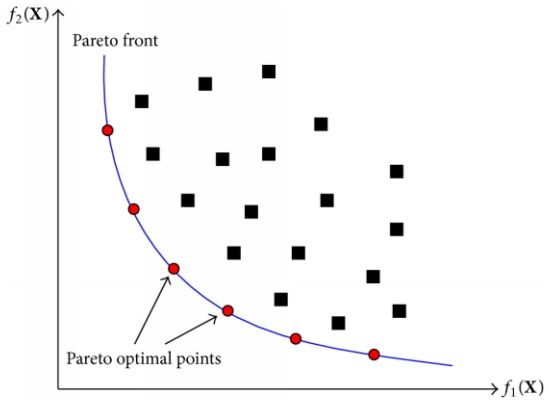

Fig. 2. Pareto front in a two dimensional optimization problem. Source: [14]

with different inspirations, and the algorithm discussed here is an evolutionary population-based approach that uses Pareto Dominance. Initially, a population of pop_size solutions is generated randomly. Next, each solution has its fitness values assessed by each objective function, and the Pareto front, empty at first, is updated with non-dominated solutions from $candidates$. After, an iterative procedure begins, and while the stop condition is not satisfied, evolutionary operators of selection, crossover, and mutation are applied. Once offspring are generated from the selected candidates, each one of them has their fitness assessed. Those promising offspring replace dominated solutions from $candidates$ so that the population size pop_size is the same. Finally, the Pareto front is updated. The output of the MOO algorithm is the resultant Pareto front.

Algorithm 1: MOO algorithm.

```
PARAMETERS:
pop_size: number of candidates per generation.
front: Non-dominated solutions found.
    candidates = random_initialization(pop_size)
    evaluate_fitness(candidates)
    front = []
    update_front(front, candidates)
1: while !stop_criterion do
    |   parents = select(candidates)
    |   offspring = crossover(parents)
    |   mutate(offspring)
    |   evaluate_fitness(offspring)
    |   candidates = replace(candidates, offspring)
    |   update_front(front, candidates)
   end
   return front
```

III. Related Work

It is important to mention that the optimization of Artificial Neural Networks (ANNs) architecture and parameters is not a recent problem. Different optimization algorithms already have been successfully employed for optimizing ANNs [9]. However, architectures of DNNs have become increasingly complex, there are still few studies that attempt to optimize them, and some gaps need to be better-investigated [9]. In this context, next, we present important works which tackled the problem of optimizing DNNs.

A. Single-objective Optimization Approaches

David and Greental [4] used a Genetic Algorithm (GA) to optimize the weights of the DNN's encoding layer, and adopted the root mean squared error (RMSE) as an objective function. Young et al. [2] used a GA to optimize the kernel size and the number of filters of each of the convolutional layers and considered the error rate as an objective function. Lorenzo et al. [5] used a Particle Swarm Optimization (PSO) algorithm to optimize the parameters of a SimpleNET (existing model). The parameters considered were the size and number of receptive fields of convolutional layers and the stride and receptive field size of the max-pooling layer. They used the classification error rate as an objective function. The previous works optimized parameters of DNNs considering a predefined DNN architecture.

The following works considered not only the optimization of DNN parameters (as the previous works) but also their layer composition and sequence. Diniz et al. [6] proposed an approach that uses Grammar-Based Genetic Programming (GGP) technique to optimize convolutional neural network architectures (parameters and layers) considering the accuracy as an objective function. Miikkulainen et al. [7] proposed the CoDeepNEAT, a Neuroevolution technique to optimize DNN architectures. This approach evolves the 15 hyperparameters, and the architecture of Convolutional Neural Networks, considering as fitness function the mean across three metrics (BLEU, METEOR, and CIDEr) normalized by their baseline values. Assunção et al. [8] recently proposed an approach to develop DNNs applied to convolutional neural networks. They combined GA with the Grammatical Evolution to evolve a sequential list of layers and their parameters. This work used the accuracy as an objective function.

B. Multi-objective Optimization Approaches

Treating the selection of DNN architecture problem as a single-objective optimization problem is not adequate. The current problem is naturally a multi-objective problem where different objective functions need to be satisfied. Unlike the works mentioned above, next, we present works that dealt with the DNN selection as a multi-objective problem. Liu et al. [10] used the Multi-Objective Convolutional Neural Network (MOCNN) algorithm. This algorithm is not evolutionary. The training process has an internal optimization procedure, which minimizes the losses of unary and pairwise terms, respectively, through a unified convolutional network. This work applied the

MOCNN for face labeling. Miseikis et al. [11] also used the MOCNN for robot localization and 3D position estimation in 2D camera images. This work considered four specific objectives related to the problem. Finally, Yang et al. [12] used a Multi-objective Evolutionary Algorithm (MOEA) to optimize the pruning of preexisting DNN architectures according to the multi-objective trade-off among error rate, computational cost, and sparsity.

Few studies investigated the DNN architecture selection as a multi-objective problem. Differently, from the studies above, we propose a multi-objective evolutionary method for the automatic selection of DNN architectures, able to build novel DNN architectures satisfying multiple criteria. In this optimization process, our method returns a DNN architecture with an optimized sequence of layers. Table I summarizes each related work, contrasting it with the proposed work. The aim is to facilitate understanding of the contribution of the proposed approach in relation to others.

IV. Proposal

In this work, we treat the problem of selecting a suitable architecture for DNNs as an optimization task with multiple criteria. Our goal is to find the best composition and sequence of layers for the DNN architecture, satisfying different objectives. The current problem is a combinatorial problem and demands a space of solutions of high dimension. Because of the nature of the problem, we decided to adopt a widely used evolutionary algorithm, the Nondominated Sorting Genetic Algorithm II (NSGA-II). NSGA-II is a multi-objective optimization algorithm adequate for combinatorial problems (details of this algorithm can be found in [15]). This algorithm showed to be a good choice for this work because it achieved much better performance compared to other restricted multi-objective optimizers [15].

The problem of selecting architectures for DNNs is a difficult task because a single architecture can be composed of different numbers and types of layers, each one with different hyper-parameters, and the order, in which the layers are placed, matters. Thus, the main contribution of this work is an approach that attempts to find the best sequence of layers for DNN architectures, without human intervention, using multi-objective optimization algorithms for a given problem.

Next, we present how the DNN architecture selection problem was formulated to be solved by a multi-objective evolutionary algorithm.

A. Individual Representation

An individual or chromosome C in an evolutionary algorithm is a solution to the problem one wants to solve. In the problem in question, it is intended to allocate a sequence of layers:

$$C = (L_1, L_2, ..., L_s), \tag{4}$$

where L_i is the i^{th} layer in the chromosome, and s is the total number of layers. In this case, each chromosome represents a possible sequence of layers (candidate) for the DNN architecture.

In this work, the architecture optimization process of a DNN considers the number of layers, order, and type. The chromosome is a vector of integers of size s, and each position can assume a value into the range $[0, 3]$. The value 0 (zero) indicates that in that position of the vector, no layer was chosen, and values 1, 2, and 3 represent the Max Pooling (MaxPool), Convolutional (Conv), and Dense layer types respectively. All these layers are configured with their default parameters. It is important to mention that the first and last layers were fixed as Convolutional and Dense respectively. This restriction was defined to avoid incorrect architectures. Figure 3 shows an example of chromosome (considering $s = 10$). As can be seen, the first and last layers (black slots) are fixed as Convolutional and Dense layers respectively, and the other eight slots have values in the range $[0, 3]$. In this example, the chromosome has five values different from zero, so it has five layers exactly in the same order presented in the vector.

Still considering the example in Figure 3, where each chromosome is represented by a vector of size $s = 10$ (being $s - 2$ variable and 2 fixed layers), there are $65,538$ possible combinations of architectures. The goal of the MOO algorithm is to search in the solutions space those non-dominated ones that satisfy the objectives. As the calculation of the vector of fitness of each solution is the execution of the DNN itself configured with the sequence of layers represented by the solution in the input problem, the optimization becomes very costly, mainly when the value of s is large.

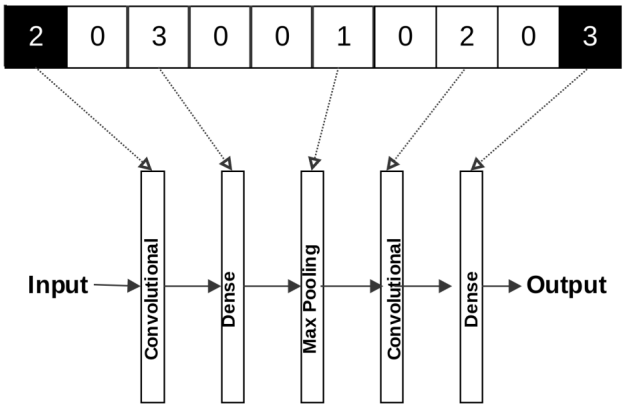

Fig. 3. Example of chromosome where $s = 10$.

B. Individual Evaluation

As presented in Section II-B, in a multi-objective scenario, each individual, $\vec{x}$, has a vector of fitness values assessed by a vector of objective functions, $\vec{f}$. The number and which objective functions will be considered in the optimization process depends on the aspects considered relevant by the expert. This paper aims to optimize the architecture of DNN in terms of layers when applied to classification problems. Thus, different quality measures can be considered as objective functions, such as accuracy, precision, recall, F_1-score, and others.

TABLE I
CONTRASTING RELATED WORKS WITH THE PROPOSAL.

Reference	Approach	#Objectives	Objective Function	Configuration
David et al. (2014) [4]	GA	1	RMSE	-Weights of the deeplearning's coding layer
Young et al. (2015) [2]	GA	1	RMSE	-Kernel's size -Number of filters of the convolutional layers
Lorenzo et al. (2017) [5]	PSO	1	RMSE	-*Stride* -Size of the receptive field of the convolutional layers -Number of receptive fields of the convolutional layers -Size of the receptive field of the *max pooling* layer
Diniz et al. (2018) [6]	GGP	1	Accuracy	-Layers -Layers' parameters
Miikkulainen et al. (2019) [7]	CoDeepNEAT	1	Average among BLEU, METEOR and CIDEr	-15 hyperparameters -Layers
Assunção et al. (2018) [8]	GA + GGP	1	Accuracy	-Layers -Layers' parameters
Yang et al. (2019) [12]	MOEA	3	RMSE Computational cost Dispersion	-Pruning heuristic
Proposal	MOEA	2	Accuracy F_1 score	-DNN sequence of layers

V. EXPERIMENTAL ENVIRONMENT

This section presents the datasets used to assess the proposed method, experimental settings, and experimental methodology. All executions were performed on a laptop with an Intel Core i5-7300U with 4M cache memory, 2.5 GHz clock speed, and up to 3.5 GHz turbo, having two logical threads per physical core and 8GB RAM. The language used to implement the proposed approach was Python™; the following libraries were adopted: Scikit learn [16] for the machine learning use, Keras [17] for building deep learning algorithms and Platypus [18] for the use of multi-objective optimization algorithms. These libraries are robust and widely used for research projects.

A. Datasets

For the experiments, two datasets are used: CIFAR 10 [19] and MNIST [20]. These data sets are composed of several images and are commonly used as classification problems to evaluate neural networks. Their popularity is because the two bases are robust and have real images.

CIFAR 10: Data set of 60,000 32x32 color training images, labeled in 10 categories.
MNIST: Data set of 70,000 28x28 10-digit grayscale images (0-9). The dataset was built from NIST Special Dataset 3 and Special Dataset 1, which contain binary images of handwritten digits.

B. Experimental Setup

In this work, we used the NSGA-II as the MOO algorithm, and it is available in the platypus library [18]. The MOO algorithm was set with its default parameters. The population size equal to 10 and the maximum number of generations equal to 50 (totalizing 500 fitness evaluations) were empirically defined, taking into account the high computational cost of the problem. The chromosome size was defined as $s = 10$, where 8 layers are variable (totalizing $65,538$ possible candidates). The decision of using up to ten layers is supported by the idea of creating architectures simpler than those used in the literature; and that still be able to achieve competitive results. Table II shows the default parameters of each layer type. For the execution of the DNN, standard values such as $batch\_size = 200$, $epochs = 50$, $learningrate = 0.1$, $impulse = 0.8$ and $shuffle = True$ were used. For the experiments, the proposed method was executed ten times to generate an average performance.

TABLE II
DEFAULT PARAMETERS OF EACH TYPE OF DNN'S LAYER

Layer's Type	***Parameters***
Conv2D	$filter = 5$ $kernelsize = 4$ $padding = "same"$
Max Pooling2D	$poolsize = [2, 2]$ $strides = 2$
Dense	$units = 5$ $activation = "softmax"$

C. Evaluation Metrics

To evaluate the fitness of an individual, we executed it in a given problem and applied a 10-fold cross-validation procedure repeated 10 times. The performance of a DNN architecture cannot be assessed considering only the accuracy, because it may be unrepresentative [21]. Therefore, we adopted two criteria to evaluate the DNN architecture's performance:

Accuracy (Acc): Total hit rate of the classifier independently of the classes of the examples. This criterion is evaluated through the following equation:

$$Acc = (TP + TN)/(TP + FP + FN + TN), \quad (5)$$

where TP and TN mean true positive and negative, and FP and FN mean true positive and negative.

F_1-score (F_1): F_1 (also called f-measure) is the harmonic average between Recall and Precision (TP / (TP + FP)). The trade-off between Recall and Precision only has a high value when both metrics have high values. F_1 is assessed by the following equation:

$$F_1 = (2 \times Recall \times Precision)/(Recall + Precision). \quad (6)$$

The values of each criterion range from 0 to 1 and the multi-objective optimization algorithm tries to find a solution that better satisfies each criterion.

D. Literature Methods

In order to evaluate the performance of the proposed method, we considered six different methods to compare:

- Optimization methods (both in their multi-objective versions, considering the same objective functions (Acc and F_1):
  - Grid search.
  - Random search.
  - MOPEA [12].
- State-of-the-art DNN algorithms:
  - Mini GoogLeNet (GNet) [22].
  - Mini VGGNet-16 (VGGNet) [23].
  - ResNet-20 [24].

The grid search and random search were chosen as an upper and lower bound. The MOPEA is an evolutionary method, proposed in [12], that optimizes the pruning procedure in DNN architectures, and is considered a strong competitor of the proposed approach. In addition, we also considered three state-of-the-art algorithms. The GNet is a DNN composed by 22 deep layers; VGG-16 has 13 convolutional layers, 5 max pooling layers and 3 dense layers; and ResNet-20 has 21 convolutional layers, 1 max pooling layers, 1 dropout layer and a fully-connected neural network at the end of the network. These algorithms have a complex architecture, achieving promising results in different classes of problems. Our intention is to investigate if the solutions found by the proposed method achieve competitive results compared to these approaches. All state-of-the-art algorithms used the same settings adopted for the execution of DNNs in the proposed method (see Section V-B).

All algorithms were executed using the same datasets as the proposed technique, and accuracy and F_1-score were assessed through a 10-fold cross-validation procedure repeated ten times. It is important to mention that the random search and MOPEA used the same methodology used by the proposed method (a simulation with 500 fitness evaluations and executed ten times to generate an average performance). In the case of the exhaustive method, all $65,538$ candidate architectures were evaluated.

VI. Results and Discussion

In this section, we present: 1) results of the proposed technique and the compared methods, and 2) an analysis of the architectures produced by the proposed approach and their layer composition.

A. Proposal vs Literature Methods

For practical comparisons of the results, we decided that instead of comparing the resulting *Paretos* found by the optimization methods (grid search, random search and proposed technique), only one solution of each Pareto front would be selected. This practice is prevalent in experiments involving multi-objective methods [25]. To choose one DNN architecture, from a set of non-dominated solutions, considering the two performance criteria mentioned previously, we applied the Borda count method [26]. This method is a single-winner election method in which each criterion ranks each algorithm, and then an average rank is returned, where the algorithm in the first place is the winner. The purpose of this analysis is to verify if one non-dominated solution (DNN architecture) found by the proposed method can improve the classification process. Also, it allows us to compare the results obtained by the optimization approaches with those of the state-of-the-art.

Tables III and IV show the results of each approach considering the CIFAR 10 and MNIST dataset respectively. These results include the average best fitness value and standard deviation reached in each objective (Acc and F_1), and the average performance time, measured in seconds, after ten runs. Also, the same tables show the best solution results achieved by the exhaustive approach.

TABLE III
COMPARATIVE PERFORMANCE ANALYSIS BETWEEN THE PROPOSED TECHNIQUE AND THE LITERATURE METHODS IN THE CIFAR DATASET

Algorithm	Acc	F_1	Time (s)
Exhaustive	0.92	0.49	90,000.00
Random	0.82 ($\pm$0.0804)	0.04 ($\pm$0.0772)	452.34
MOPEA	0.88 ($\pm$0.0103)	0.33 ($\pm$0.0132)	65,020.25
GNet	0.87 ($\pm$0.0354)	0.29 ($\pm$0.0213)	130
VGGNet	0.88 ($\pm$0.0378)	0.28 ($\pm$0.0272)	150
ResNet	0.88 ($\pm$0.0276)	0.29 ($\pm$0.0135)	100
Proposal	**0.91** ($\pm$0.0018)	**0.47** ($\pm$0.0102)	41,766.47

The first line of Tables III and IV shows the optimal values for Acc and F_1 achieved by the exhaustive approach in each problem, CIFAR 10 and MNIST. To find the optimal values, the exhaustive method evaluates all possible configurations for the DNN model, a task whose cost is extremely high. On the other hand, the random method has a smaller execution time (452.34s in CIFAR 10 and 364.22s in MNIST) when compared to the exhaustive method, however, due to a large number of possible solutions, the possibility of finding a good architecture for DNN by chance (randomly) is small. As can be

TABLE IV
COMPARATIVE PERFORMANCE ANALYSIS BETWEEN THE PROPOSED TECHNIQUE AND THE LITERATURE METHODS IN THE MNIST DATASET

Algorithm	**Acc**	**F_1**	**Time (s)**
Exhaustive	0.99	0.97	92,000.00
Random	0.88 ($\pm$0.0835)	0.01 ($\pm$0.1724)	364.22
MOPEA	0.95 ($\pm$0.0376)	0.85 ($\pm$0.0817)	40,357.89
GNet	0.94 ($\pm$0.0214)	0.85 ($\pm$0.0167)	130
VGGNet	0.95 ($\pm$0.0165)	0.86 ($\pm$0.0275)	145
ResNet	0.95 ($\pm$0.01877)	0.83 ($\pm$0.0217)	110
Proposal	**0.99** ($\pm$0.0027)	**0.96** ($\pm$0.0070)	36,597.99

seen, the random method achieved $Acc = 0.82$ and $F_1 = 0.04$ for CIFAR 10 and $Acc = 0.88$ and $F_1 = 0.01$ for MNIST, with high values of standard deviation. Regarding the MOPEA, can be seen it overcame the results achieved by the random approach, costing $65,020.25$s for CIFAR 10 and $40,357.89$s for MNIST.

The three state-of-the-art algorithms achieved close results in terms of Acc and F_1, presenting a similar standard deviation. As these algorithms were meticulously crafted by an expert, their results easily surpassed the random search, and reached close results to the MOPEA, in both CIFAR10 and MNIST.

Regarding the proposed method, we can see it was able to find good solutions when compared to the exhaustive approach. For CIFAR 10, the proposal found a solution with $Acc = 0.91$ and $F_1 = 0.47$, close to the optimal solution. The same for MNIST, where the proposal found a solution with $Acc = 0.99$ and $F_1 = 0.96$. Concerning the other methods, our proposal was superior in both Acc and F_1 (with the lowest standard deviation values) in both problems. The statistical significance of the results was evaluated as suggested in [27]. The null hypothesis is that there is no difference between the mean values of the six approaches (excluding the grid search). The alternative hypothesis is that there is at least one difference between the mean values. After, the *Friedman Aligned-Ranks* non-parametric test was conducted, and the null hypothesis was rejected, with a p-value $= 2.2 \times 10^{-5}$. This result shows that there is a statistical difference between the means. Finally, since the Friedman's test rejected the null hypothesis, a posthoc test was performed to identify which differences are significant. We used the *Finner* procedure with p-value correction (as multiple comparisons are being made), defined the proposed approach as the control algorithm, and compared it with the other five approaches: Random search, MOPEA, GNet, VGGNet, and ResNet. The results of the posthoc test showed that the proposal was superior regarding both Acc and F_1 to all the competitors.

In terms of computational cost, the proposal achieved solutions close to the optimal (found by the grid search), but in much less time when compared to the exhaustive approach. When compared to the MOPEA and random approach (using the same stop criterion, 500 fitness evaluations), the proposed method overcame the results achieved by the MOPEA in less time, but, regarding the random approach, our method had a much higher execution time than the random, in CIFAR 10 and MNIST problems. Could it be that if the random method had the same execution time, it could outweigh the proposed approach? Here, we also performed this analysis. Table V shows the results obtained by the random method, being executed for the same execution time of the proposal in CIFAR 10 and MNIST, respectively. As can be seen in this table, even if we run the random method many more times, the proposed method continues to surpass it in all the objectives.

TABLE V
COMPARATIVE PERFORMANCE ANALYSIS BETWEEN THE PROPOSED TECHNIQUE AND THE RANDOM SEARCH (BOTH WITH THE SAME EXECUTION TIME) IN CIFAR 10 AND MNIST.

CIFAR 10		
Acc	Random	0.88 ($\pm$0.0282)
	Proposal	0.91($\pm$0.0018)
F_1	Random	0.03 ($\pm$0.0255)
	Proposal	0.47 ($\pm$0.0102)
MNIST		
Acc	Random	0.90 ($\pm$0.0197)
	Proposal	0.99 ($\pm$0.0027)
F_1	Random	0.10 ($\pm$0.1880)
	Proposal	0.96 ($\pm$0.0070)

The results showed the potential of the proposal in the construction of simpler DNN architectures than the high end DNNs, capable of obtaining competitive results. Given this, the proposed method becomes an alternative for experts to automate the definition of the architecture of DNNs for a given classification problem.

B. Frequency of layer type per slot

In this section, we present the frequency of the most used layer type per slot selected by the proposed approach during the simulations.

TABLE VI
MOST FREQUENT LAYER TYPE PER SLOT IN THE ARCHITECTURES.

	CIFAR10	***MNIST***
Slot 1	Conv (100.0%)	Conv (100.0%)
Slot 2	Dense (94.2%)	Empty (87.8%)
Slot 3	Empty (93.0%)	Conv (96.1%)
Slot 4	Conv (91.6%)	Conv (91.0%)
Slot 5	Conv (89.2%)	Conv (87.4%)
Slot 6	Conv (67.0%)	MaxPool (86.8%)
Slot 7	MaxPool (52.6%)	MaxPool (84.6%)
Slot 8	MaxPool (95.0%)	Conv (93.6%)
Slot 9	Conv (83.2%)	Conv (88.5%)
Slot 10	Dense (100.0%)	Dense (100.0%)

In this work, we limited the architecture to ten slots ($s = 10$). Table VI presents the most used layer type per slot in the architectures, for both problems. To calculate the frequency, we considered the Paretos of all generations. As the first and last slot were fixed with the Convolutional and Dense layers, their frequencies in the assessed architectures are 100%.

It is important to highlight that the best architectures for CIFAR 10 and MNIST, generated by the proposal (whose results were presented in Section VI-A) are composed of the same layers as the ones presented in Table VI. For both problems, the best DNN architectures are composed of only nine layers (the third layer for CIFAR 10 and the second layer for MNIST are "Empty"), and, as shown in Section VI-A, these DNNs obtained better results than state-of-the-art DNNs which have a complex composition of layers.

In both problems, the DNN architectures with nine layers proved to be the most promising, since they were present in the Pareto front at least 75% of the time during the simulations. The average number of DNN layers of the solutions present in the Pareto is 8.3 layers.

VII. Conclusion and Future Works

In this work, the selection of architectures for DNNs is treated as a multi-objective optimization problem. To tackle this problem, an evolutionary algorithm, called NSGA-II, was adopted and adjusted to select DNN architectures considering two different criteria: Accuracy and F_1 score. The main contributions of this work are: i) the proposal of a multi-objective optimization method to find the best architecture for a DNN; ii) a novel candidate solution representation that characterizes the sequence of the layers.

The results showed that the proposed method was able to achieve a performance close to the exhaustive method but in much less time, and overcame the random search and state-of-the-art algorithms in all problems. In the future, it is intended to optimize the parameterization along with the architecture, as this configuration also influences the performance of a DNN. In addition, we plan to investigate further the impact of the number of layers s. In this work $s = 10$ was adopted empirically, but we believe that the optimal value of s may be problem-dependent.

References

[1] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *nature*, vol. 521, no. 7553, p. 436, 2015.

[2] S. R. Young, D. C. Rose, T. P. Karnowski, S.-H. Lim, and R. M. Patton, "Optimizing deep learning hyper-parameters through an evolutionary algorithm," in *Proceedings of the Workshop on Machine Learning in High-Performance Computing Environments*. ACM, 2015, p. 4.

[3] T. Elsken, J. H. Metzen, and F. Hutter, "Neural architecture search: A survey," *arXiv preprint arXiv:1808.05377*, 2018.

[4] O. E. David and I. Greental, "Genetic algorithms for evolving deep neural networks," in *Proceedings of the Companion Publication of the 2014 Annual Conference on Genetic and Evolutionary Computation*. ACM, 2014, pp. 1451–1452.

[5] P. R. Lorenzo, J. Nalepa, M. Kawulok, L. S. Ramos, and J. R. Pastor, "Particle swarm optimization for hyper-parameter selection in deep neural networks," in *Proceedings of the Genetic and Evolutionary Computation Conference*. ACM, 2017, pp. 481–488.

[6] J. B. Diniz, F. R. Cordeiro, P. B. Miranda, and L. A. T. da Silva, "A grammar-based genetic programming approach to optimize convolutional neural network architectures," in *Anais do XV Encontro Nacional de Inteligência Artificial e Computacional*. SBC, 2018, pp. 82–93.

[7] R. Miikkulainen, J. Liang, E. Meyerson, A. Rawal, D. Fink, O. Francon, B. Raju, H. Shahrzad, A. Navruzyan, N. Duffy *et al.*, "Evolving deep neural networks," in *Artificial Intelligence in the Age of Neural Networks and Brain Computing*. Elsevier, 2019, pp. 293–312.

[8] F. Assunção, N. Lourenço, P. Machado, and B. Ribeiro, "Evolving the topology of large scale deep neural networks," in *European Conference on Genetic Programming*. Springer, 2018, pp. 19–34.

[9] J. D. Schaffer, D. Whitley, and L. J. Eshelman, "Combinations of genetic algorithms and neural networks: A survey of the state of the art," in *[Proceedings] COGANN-92: International Workshop on Combinations of Genetic Algorithms and Neural Networks*. IEEE, 1992, pp. 1–37.

[10] S. Liu, J. Yang, C. Huang, and M.-H. Yang, "Multi-objective convolutional learning for face labeling," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2015, pp. 3451–3459.

[11] J. Miseikis, I. Brijacak, S. Yahyanejad, K. Glette, O. J. Elle, and J. Torresen, "Multi-objective convolutional neural networks for robot localisation and 3d position estimation in 2d camera images," in *2018 15th International Conference on Ubiquitous Robots (UR)*. IEEE, 2018, pp. 597–603.

[12] C. Yang, Z. An, C. Li, B. Diao, and Y. Xu, "Multi-objective pruning for cnns using genetic algorithm," *arXiv preprint arXiv:1906.00399*, 2019.

[13] H. Ishibuchi, N. Tsukamoto, and Y. Nojima, "Evolutionary many-objective optimization: A short review," in *Evolutionary Computation, 2008. CEC 2008.(IEEE World Congress on Computational Intelligence). IEEE Congress on*. IEEE, 2008, pp. 2419–2426.

[14] W. Wang, L.-L. Zhang, J.-J. Chen, and J.-H. Wang, "Parameter estimation for coupled hydromechanical simulation of dynamic compaction based on pareto multiobjective optimization," *Shock and Vibration*, vol. 2015, 2015.

[15] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan, "A fast and elitist multiobjective genetic algorithm: Nsga-ii," *IEEE transactions on evolutionary computation*, vol. 6, no. 2, pp. 182–197, 2002.

[16] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg *et al.*, "Scikit-learn: Machine learning in python," *Journal of machine learning research*, vol. 12, no. Oct, pp. 2825–2830, 2011.

[17] F. Chollet *et al.*, "Keras," https://keras.io/, 2015, accessed: 2019-06-12.

[18] D. Hadka, "Platypus: A free and open source python library for multiobjective optimization," *Available on Github, vol. https://github.com/Project-Platypus/Platypus*, 2017.

[19] A. Krizhevsky and G. Hinton, "Learning multiple layers of features from tiny images," Citeseer, Tech. Rep., 2009.

[20] L. Deng, "The mnist database of handwritten digit images for machine learning research [best of the web]," *IEEE Signal Processing Magazine*, vol. 29, no. 6, pp. 141–142, 2012.

[21] H. He and E. A. Garcia, "Learning from imbalanced data," *IEEE Trans. on Knowl. and Data Eng.*, vol. 21, no. 9, pp. 1263–1284, Sep. 2009. [Online]. Available: http://dx.doi.org/10.1109/TKDE.2008.239

[22] H. Haenssle, C. Fink, R. Schneiderbauer, F. Toberer, T. Buhl, A. Blum, A. Kalloo, A. B. H. Hassen, L. Thomas, A. Enk *et al.*, "Man against machine: diagnostic performance of a deep learning convolutional neural network for dermoscopic melanoma recognition in comparison to 58 dermatologists," *Annals of Oncology*, vol. 29, no. 8, pp. 1836–1842, 2018.

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

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

[25] M. Orouskhani, M. Teshnehlab, and M. A. Nekoui, "Evolutionary dynamic multi-objective optimization algorithm based on borda count method," *International Journal of Machine Learning and Cybernetics*, pp. 1–29, 2017.

[26] D. Black, "Partial justification of the borda count," *Public Choice*, vol. 28, no. 1, pp. 1–15, 1976.

[27] S. García, A. Fernández, J. Luengo, and F. Herrera, "Advanced non-parametric tests for multiple comparisons in the design of experiments in computational intelligence and data mining: Experimental analysis of power," *Information Sciences*, vol. 180, no. 10, pp. 2044–2064, 2010.