2021 IEEE Congress on Evolutionary Computation (CEC) | 978-1-7281-8393-0/21/$31.00 ©2021 IEEE | DOI: 10.1109/CEC45853.2021.9504822

A Multi-Objective Grammatical Evolution Framework to Generate Convolutional Neural Network Architectures

Cleber A.C.F. da Silva*, Daniel Carneiro Rosa*, Péricles B.C. Miranda*, Filipe R. Cordeiro*,
Tapas Si‡, André C.A. Nascimento*, Rafael F.L. Mello*, Paulo S. G. de Mattos Neto†
*Departamento de Computação, Universidade Federal Rural de Pernambuco, Recife, Brazil
†Centro de Informática, Universidade Federal de Pernambuco, Recife, Brazil
‡Department of Computer Science and Engineering, Bankura Unnayani Institute of Engineering, Bankura, India

***Abstract*—Deep Convolutional Neural Networks (CNNs) have reached the attention in the last decade due to their successful application to many computer vision domains. Several handcrafted architectures have been proposed in the literature, with increasing depth and millions of parameters. However, the optimal architecture size and parameters setup are dataset-dependent and challenging to find. For addressing this problem, this work proposes a Multi-Objective Grammatical Evolution framework to automatically generate suitable CNN architectures (layers and parameters) for a given classification problem. For this, a Context-free Grammar is developed, representing the search space of possible CNN architectures. The proposed method seeks to find suitable network architectures considering two objectives: accuracy and F_1-score. We evaluated our method on CIFAR-10, and the results obtained show that our method generates simpler CNN architectures and overcomes the results achieved by larger (more complex) state-of-the-art CNN approaches and other grammars.**

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

I. Introduction

Convolutional Neural Networks (CNNs) is a class of Deep Learning (DL) methods that have shown increasing performance dealing with different computer vision tasks, such as image classification and object detection [1], to different areas of applications [2]. With the rising interest in the area, several architectures have been proposed, and the size and complexity of the CNNs have increased, resulting in a large number of parameters and design options that are dataset-dependent [2]. The first proposed CNN, LeNet [3], had five layers and around 60 million parameters. Modern architectures, such as ResNet-152 [4] are deeper, having 152 layers and around the same amount of parameters. The main challenge about using a CNN for a given classification task is the diversity of existing deep architectures to choose and the large number of parameters, making it difficult to optimize the network. The architecture design, which defines the types and arrangement of layers and parameters that compose the CNN, directly impact its performance and complexity [2]. Frank and Carbin [5] show that it is possible to find neural subnetworks with a reduced number of parameters without compromising accuracy, using pruning techniques. However, pruning techniques have limitations because the final solution depends on the original deep CNN and sequence of layers. As the CNN algorithms get more complex, defining a suitable CNN architecture (layers and parameters) is not a trivial task and requires in-depth knowledge about the problem and algorithm. To solve this problem, researchers have focused their efforts on automating the design of such deep neural networks. Some works have dealt with the CNN architecture selection as an optimization problem. Among these works, some optimized the CNN's architecture using NeuroEvolution [6] and Genetic Algorithms (GAs) [7]. However, the results achieved in [8], [9], [10] showed that Grammatical Evolution (GE) is a promising approach for the evolution of CNNs. GE is a Genetic Programming (GP) algorithm that uses context-free grammar to evolve variable-length programs (in our case, CNNs). The grammar can incorporate *a priori* knowledge about the problem domain to guide the GE search better, avoiding syntactic incorrectness of the generated programs [11]. Thus, GE has potential advantages to algorithm design since it is flexible to represent and evolve more complex architectures [11].

Although [8], [9], [10] have shown GE's potential on the generation of CNNs, the problem at hand was modeled as a single-objective problem whose objective function is the CNNs' accuracy maximization. According to [7], generating CNN models with a good balance between generalization, accuracy, recall, and precision is hard. Thus, treating this task as a single-objective optimization problem is not adequate because different aspects (objectives) need to be satisfied.

In this work, we propose a Multi-Objective GE-based framework to optimize CNN's architectures. The contributions are two-fold: we implemented a multi-objective search engine and created a novel grammar that focuses on evolving low complex CNN models. The proposal was compared to state-of-the-art CNN algorithms and with two grammar-based approaches for CNN generation ([9] and [10]). All neural networks had their effectiveness evaluated according to the

978-1-7281-8393-0/21/$31.00 ©2021 IEEE

Authorized licensed use limited to: Hebei University of Technology. Downloaded on February 02,2023 at 14:47:30 UTC from IEEE Xplore. Restrictions apply.

accuracy and F_1-score values obtained after testing the CNN. The execution of CNN's training and test was made using the image classification dataset CIFAR-10. The results showed that the proposal generated competitive and low-complexity CNN models when compared to state-of-the-art algorithms. Besides, it overcomes the results achieved by the models created by grammar-based approaches in all metrics.

The paper is organized as follows: Section II introduces basic concepts for a better understanding of the current work. Section III reports related work. In section IV, we present the proposed work. The experimental environment is presented in Section V. Section VI discuss results, and, finally, Section VII contains the conclusion and future work of the study.

II. Background

A. Convolutional Neural Networks

CNNs are responsible for extracting features from the input space, such as images, reducing the data representation to a representative feature vector, which is applied to a sequence of fully-connected layers, such as in the Multilayer Perceptron (MLP) [2]. CNNs are a composition of a sequence of multiple layers of neurons, which computes a nonlinear operation on a linear transformation of the preceding layer's outputs. The types of layers in a CNN can be mainly classified in convolutional, pooling, batch normalization or fully connected layers . The convolutional layers and fully connected layers have weights that need to be trained, while the pooling layers transform the activation using a fixed function. Pooling layers are important parts of the CNN and they implement a spatial dimensionality reduction operation designed to reduce the number of trainable parameters for the next layers and allow them to focus on larger areas of the input pattern. Given an feature map M_l, a pooling layer pool size $P_m \in \mathbb{N}$ and stride $\alpha_m \in \mathbb{N}$ implements a channel-wise operation, using the operations of average or max. The most used type of pooling function is the Max pooling [1]. The pooling layer reduces the number of parameters by down-sampling the representation.

The fully connected (FC) layers are present in the end of the CNN and they are the traditional layers present in MLP. As the output of the convolutional layers is a n-dimensional vector, it is applied a reshaping operation, which transforms the tensor to a one-dimensional vector. The flattened vector is feature vector which is used in the sequence of FC layers and outputs the logits corresponding to each class.

The batch normalization layers have been used in modern networks and are responsible for normalizing the input between layers. It is also part of the design choice when building an architecture. Other types of layers are not considered in the scope of this work can be easily extended by just adding in the grammar.

In the CNN architecture, there is no pre-defined rule of how the convolutional layers and pooling layers are disposed. It can be built using blocks of convolutional layers followed by pooling layers, or using interspersed convolutional and pooling layers. The number of layers, how the layers are disposed and the hyperparameters of each layers are design choices that are data-dependent and have a large exploration space.

B. Grammatical Evolution

Finding a suitable CNN architecture for a given task is a challenging problem, and Grammatical Evolution (GE) has shown to be an alternative for the automatic creation of networks [10]. GE is a GP algorithm that employs a context-free grammar to evolve variable-length programs [11]. The GE algorithm has three components: a search engine, grammar, and the mapping process. The search engine is an optimizer (e.g. GA) that evolves the solutions (in genotype representation) iteratively. The genotypes are represented as variable-length binary strings (binary array), where every slot of the array (codon) can receive numerical values of 8-bits. Throughout the search engine execution, each genotype has its fitness value assessed. For this, the mapping process converts each genotype (binary array) to a corresponding phenotype (a program), and the fitness function calculates the genotype's fitness using the mapped program (phenotype) as input. The conversion from genotype to phenotype, performed by the mapping process, utilizes a grammar in the Backus-Naur Form (BNF). The BNF grammar is domain-dependent and comprises specific rules to create the programs [11].

In GE, context-free grammars are represented by a tuple $\{N, T, P, S\}$, where N is the set of non-terminals, T the set of terminals, P the set of production rules and S the initial symbol. The production rules have the form $x \models y$, where $x \in N$ and $y \in \{T \cup N\}$ and there is a set of possibilities for x, each possible production is delimited by the disjunctive symbol $|$ [11]. Figure 1 shows an example of a context-free grammar that generates arithmetic expressions.

$$
\begin{aligned}
&N = \{\langle\text{exp}\rangle, \langle\text{var}\rangle, \langle\text{op}\rangle\}\\
&T = \{x, +, -, /, *\}\\
&S = \langle\text{exp}\rangle\\
&P = \{\langle\text{exp}\rangle \models \langle\text{var}\rangle \mid ((\langle\text{exp}\rangle\langle\text{or}\rangle\langle\text{exp}\rangle)\\
&\qquad \langle\text{op}\rangle \models + \mid - \mid / \mid *\\
&\qquad \langle\text{var}\rangle \models \text{x}\}
\end{aligned}
$$

Fig. 1. An example of context-free grammar.

In the example of Figure 1, the terms $\langle\text{exp}\rangle$, $\langle\text{var}\rangle$ and $\langle\text{op}\rangle$ are non-terminals. On the other hand, the terms x and the arithmetic operators $+$, $-$, $/$ and $*$ are terminals.

The mapping process receives the genotype as input to map start symbols to terminal symbols. This procedure reads the codon's value of the binary array, and the returned integer value indexes a specific rule in the BNF grammar:

rule = codon's value
MOD
(number of rules of the current non terminal item).

As an example, consider the following genotype (18, 102, 203, 57, 62), and the first codon (value 18) is associated with

Authorized licensed use limited to: Hebei University of Technology. Downloaded on February 02,2023 at 14:47:30 UTC from IEEE Xplore. Restrictions apply.

the terminal <exp>. As the <exp> has 2 options and 18 MOD 2 = 0, the first rule is added to the phenotype. The mapping process reads the genotype from left to right until there are no more non-terminals. In case there are non-terminals, and the end of the binary array is reached, the process restart from the first codon. This procedure's output is the phenotype or program. For illustration, a possible phenotype, for the current example, is $x * (x + x)$. Once the phenotype is generated, it is provided for the fitness function, and its fitness value is assessed. We highlight that although the evaluation of a solution is in the phenotype format, the application of crossover and mutation operators occurs in the genotype format.

C. Multi-objective Optimization

Multi-objective optimization (MOO) problems have more than one objective function to be optimized. For a given search space of candidate solutions, $\mathcal{S}$, MOO algorithms aim to seek solutions that better 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 candidate into $\mathcal{S}$ can be defined as a vector of decision variables ($\vec{x} = (x_1, x_2, ..., x_k)$), where k is the number of decision variables. For fitness evaluation, each solution is provided as input for every objective function into $\vec{f}$, which return the respective fitness value. Thus, $\vec{f}(\vec{x}) = (f_1(\vec{x}), f_2(\vec{x}), ..., f_n(\vec{x}))$ represents a vector of all fitness values belonging 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 a candidate solution; 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.

Different from mono-objective optimization, in MOO, each solution's quality is determined by a vector of fitness values instead of a single one. Thus, the solutions are generally compared to each other through the Pareto dominance [12]. Given two vectors $\vec{x}$, $\vec{y} \in \mathbb{R}^n$, $\vec{x}$ dominates $\vec{y}$ (denoted by $\vec{x} \prec \vec{y}$) if $\vec{x}$ overcomes $\vec{y}$ in at least one objective and $\vec{x}$ is not worse than $\vec{y}$ in any other objectives. $\vec{x}$ is not dominated if there is no solution $\vec{x}_i$ in the current population, such that $\vec{x}_i \prec \vec{x}$. All non-dominated solutions into the objective space are known as the Pareto front.

III. Related Work

The application of grammar-based algorithms, e.g., GE, for the generation of artificial neural networks (ANNs) is not novel [13]. However, with the rise of deep neural networks (DNNs), the architectures have become even more complex and deeper [2]. As a consequence, the optimization process became even more challenging. Then, recently, different studies have developed and improved grammar-based approaches to better tackle the generation of DNNs problem. These studies are presented next.

Suganuma et al. [14] used the Cartesian genetic programming (CGP) to define layers for CNNs automatically. This work has two limitations: the layers' parameters are fixed and do not consider the optimization layers (e.g., dropout and batch normalization) in the grammar. The CNN structure is optimized only to maximize the accuracy, and the experiments were performed considering the CIFAR-10 dataset.

Evans et al. [15] proposed a genetic programming solution to evolve CNNs, called ConvGP. In this work, only convolutional and pooling layers (ReLU is used as activation function) and aggregation functions are considered. The classifier is produced by generating a function built with arithmetic operators such as +, -, / and *. Other parameters related to the possible layers were fixed. The CNN structure is optimized to maximize the accuracy, and the approach was evaluated on four benchmark image datasets.

Assunção et al. [8] proposed DENSER (Deep Evolutionary Network StructurEd Representation), that combines the basic principles of GAs with Dynamic Structured Grammatical Evolution (DSGE). DENSER was used for the generation of CNN architectures attempting to maximize the accuracy. The experiments considered the datasets CIFAR-10, CIFAR-100, MNIST, and FashionMNIST for evaluation. The results showed that DENSER is a promising approach, being competitive regarding state-of-the-art algorithms. However, it does not allow to add of optimization layers as the dropout layer.

Diniz et al. [9] used GE for the evolution of CNN's, proposing a simple grammar for the generation of individuals in a mono-objective optimization process. The grammar only allows creating networks with convolutional, pooling, and dense layers. The experiments were performed in CIFAR 10, and the results showed the method is promising, generating simpler architectures with competitive results when compared to state-of-the-art CNNs.

Lima et al. [10] also proposed a novel grammar for CNN generation using a mono-objective GE to maximize accuracy. The authors created a rich grammar with recursive configurable parameters that generate CNNs with infinite layers and consequently an infinite search space. However, the grammar does not allow to add of some optimization layers as batch normalization. A limitation of this work is that a complex grammar with an infinite number of solutions may contribute to the non-convergence of the GE algorithm.

This work proposes a GE-based framework composed of 1) a novel grammar for CNN generation and 2) a search engine set with a multi-objective algorithm to guide the optimization process. The proposed grammar is compact and incorporates components that have high relevance in the CNN's performance. The reason for creating a compact grammar is to reduce the space of candidate solutions to favor the search for promising CNNs, and to avoid the generation of complex architectures. Besides, differently from the other works, we treat the CNN architecture generation as a multi-objective problem with two objective functions to be optimized: accuracy and

Authorized licensed use limited to: Hebei University of Technology. Downloaded on February 02,2023 at 14:47:30 UTC from IEEE Xplore. Restrictions apply.

F_1-score. Table I summarizes the main characteristics of each approach, contrasting them with the proposed work.

TABLE I
CONTRASTING RELATED WORKS WITH PROPOSAL.

Refer.	***Grammar***	***Search Eng.***	**Dataset(s)**
Suganuma et al. [14]	Convolutional, max and average pooling, and concatenation, summation and softmax functions	CGP	CIFAR 10
Evans et al. [15]	Convolutional, max pooling, aggregation functions, +, -, /, *	GP	Cars, JAFFE, Faces, Pedestrian
Assunção et al. [8]	Convolutional, max pooling, fully-connected, softmax, batch normalization, pool-type, paddings, activation functions (linear, relu, sigmoid), learning rates	GA	CIFAR-10, CIFAR-100, MNIST and FashionMNIST
Diniz et al. [9]	Convolutional, max pooling, fully-connected	GA	CIFAR-10
Lima et al. [10]	Convolutional, max and average pooling, dropout, fully-connected and # of neurons, paddings, learning rate, filters, activation functions (relu, selu, elu, tanh, sigmoid, linear)	GA	CIFAR-10, MNIST
Proposal	Convolutional, max pooling, dropout, fully-connected and # of neurons, batch normalization, learning rate	NSGA-II	CIFAR 10

IV. Proposed Approach

This work proposes a GE-based framework for the generation of suitable CNN architectures. The proposal has three components: Context-Free Grammar (CFG) in BNF, search engine, and mapping process. The mapping process follows the same procedure of the original GE, presented in section II-B, and will not be detailed again. The novel grammar and search engine are detailed next.

A. CFG in BNF

Figure 2 presents the proposed CFG in BNF used to generate CNN architectures. An amount of eleven tags composes the grammar, and each one is explained next.

$$
\begin{aligned}
\langle \text{CNN} \rangle &\models \langle \text{BLOCK} \rangle.\texttt{flatten}.\langle \text{FC} \rangle.\langle \text{DROPOUT} \rangle.fc\ \langle \text{LR} \rangle \\
\langle \text{BLOCK} \rangle &\models (\langle \text{CONV} \rangle \langle \text{POOL} \rangle) * \langle \text{M} \rangle \\
\langle \text{CONV} \rangle &\models (\texttt{conv}.\langle \text{BNORM} \rangle) * \langle \text{Z} \rangle \\
\langle \text{POOL} \rangle &\models \texttt{pool}.\langle \text{DROPOUT} \rangle \mid \lambda \\
\langle \text{FC} \rangle &\models (\texttt{fc}\langle \text{UNITS} \rangle) * \langle \text{K} \rangle \\
\langle \text{BNORM} \rangle &\models \texttt{bnorm} \mid \lambda \\
\langle \text{DROPOUT} \rangle &\models \texttt{dropout} \mid \lambda \\
\langle \text{LR} \rangle &\models 0.1 \mid 0.01 \mid 0.001 \mid 0.0001 \\
\langle \text{UNITS} \rangle &\models 64 \mid 128 \mid 256 \mid 512 \\
\langle \text{K} \rangle &\models 0 \mid 1 \mid 2 \\
\langle \text{Z} \rangle &\models 1 \mid 2 \mid 3 \\
\langle \text{M} \rangle &\models 1 \mid 2 \mid 3
\end{aligned}
$$

Fig. 2. CFG in BNF applied to study.

The tag <CNN> is the main tag, and it is from there that the translations begin. To facilitate the understanding of how the translation starts from <CNN>, Figure 3 presents the grammar's generation flow of CNNs. <CNN> is defined by the expression <BLOCK>.flatten.<FC>.<DROPOUT>.fc <LR>, which means that the tag <BLOCK> is the next to be translated.

The tag <BLOCK> is defined by the expression (<CONV> <POOL>) * <M>. It means that a set of <CONV> and <POOL> layers can be added M times. The next tag to be translated is <CONV>. The tag <CONV> is defined by the expression (conv.<BNORM>) * <Z>~~)~~, which defines the number (Z) of convolutional layers (conv) with <BNORM> that will be added. The tag <BNORM> can be translated to a batch normalization technique (bnorm) or none (λ). In case bnorm is chosen, each of the Z convolutional layers is set with batch normalization. After translating the <CONV> tag, the next is <POOL>. The tag <POOL> is defined by the expression (pool.<DROUPOUT> | λ). pool.<DROUPOUT> means that a pooling layer (pool) is added and followed by the <DROUPOUT> tag, which can add a dropout layer or not (λ). It is important to say that the <POOL> can also be translated to λ, and no pooling layer is instantiated.

The tag <FC> is defined by the expression (fc*<UNITS>) * <K>). It means that K fully connected layers (fc) will be added with a determined number of neurons (<UNITS>). <K> can assume the values 0, 1, or 2. In the case $K = 0$, no fc layers are added at this moment. The next tag of <CNN> expression is <DROPOUT>, which can be translated in a dropout layer or not (λ). Next, an fc layer with a softmax activation is added to produce the final classification. Finally, the tag <LR> can assume four different values, representing the CNN's learning rate.

As it can be seen, the grammar is flexible, allowing the creation of small networks with one convolutional layer or networks with a large number of convolutional, dropout, pooling, and fully-connected layers, with their respective parameters. Nonetheless, the values for the variables Z, K, and M were determined empirically to control the uncontrolled growth of the architecture. M and Z's upper-bound was defined as 3 to avoid CNN architectures with a high computational cost and complexity (number of parameters and layers). However, all three hyperparameters can be set with more and higher values for the generation of deeper architectures. It is important to highlight that only sequential topologies, formed by stacked layers, can be generated.

Other parameters such as padding, activation functions, optimizer, stride, filters, and others remained fixed. Details about other layers' parameters is presented in Table II. In convolutions, the ReLU activation function was used because it has a lower computational cost than others such as Sigmoid and Hyperbolic Tangent, and still presents good results [16]. In the pooling layers, Max Pooling was used to highlight the most important aspects of the images. We use the optimizer Adam, as it presents good results quickly and satisfactorily

Authorized licensed use limited to: Hebei University of Technology. Downloaded on February 02,2023 at 14:47:30 UTC from IEEE Xplore. Restrictions apply.

compared to other optimizers [17]. The error function used was Categorical Cross-entropy, and the number of epochs for optimization and batch size was defined as 70 and 128, respectively.

TABLE II
CNN PARAMETERS.

Parameter	***Value***
Kernel size	3 x 3
# of filters	Starts with 32; duplicates for every 2 convolutions.
Stride	1
Max-pooling shape	2 x 2
Dropout	0.25 after max-pooling layer; 0.50 after Fully Connected layer.
Activation function	ReLU for Convolutions; Softmax for the last layer.
Optimizer	Adam optimizer
Loss function	Categorical cross-entropy
# Epochs	70
Batch size	128
Data augmentation	zoom_range=0.2 horizontal_flip=True
Early stopping	monitor=val_accuracy, mode=max patience=10, baseline=0.5

B. Search Engine

The generation of CNN models with a good balance between generalization, accuracy, recall, and precision is hard. Treating the problem at hand as a single-objective optimization problem is not adequate [7], because different aspects (objectives) need to be satisfied. Thus, the framework's search engine was set with a multi-objective algorithm to optimize CNN architectures.

Due to the nature of the problem, the NSGA-II, a widely used multi-objective evolutionary algorithm (MOEA), was chosen for the optimization (Details can be found in [12]). This algorithm has reached better performance than other restricted multi-objective optimizers [9], being a right choice for this work.

The NSGA-II is an MOEA based on GA with a strong elitism strategy. Its pseudocode is shown in Algorithm 1. Before the iterative process, the population *P* is initialized randomly with size N', and each individual has its fitness values assessed (lines 1 and 2). Next, the individuals are ranked based on Pareto-sort (line 3). Then, a novel offspring population is generated through selection, crossover, and mutation operators (line 4). After that, the iterative process begins (lines 5-12). For each generation, the algorithm sorts the individuals from parent and offspring populations regarding the non-dominance, producing several fronts (lines 7-8). The sorting procedure works as follows. The first front is composed of all non-dominated solutions. After removing the first front solutions, all novel non-dominated solutions are included in the second front. This procedure of removing the current non-dominated solutions and inserting the novel ones in the next front is repeated until all individuals are classified. After that, a crowding distance based-sort is performed over the same front to promote diversity (line 9). The crowding distance (CD) represents the distance between a given solution and its neighbors. Once each solution has its CD value, they are decreasingly sorted. This strategy aims to benefit boundary solutions placed in regions of the search space with fewer neighbors.

Algorithm 1: NSGA-II's pseudocode.

```
INPUT: $N', g, f_k(X) \triangleright N'$ members evolved in $g$ generations to
  solve $f_k(X)$
1: Generate random population P with size $N'$
2: Evaluate objectives values;
3: Assign Rank (level) based on Pareto - sort
4: Generate Child Population: Selection, Crossover and Mutation
5: for $i \leftarrow 1$ to $g$ do
6:     for each Parent and child in P do
7:         Assign Rank (level) based on Pareto - sort
8:         Generate sets of nondominated solutions
9:         Determine Crowding distance
10:        Loop (inside) by adding solutions to next generation
             starting from the first front until $N'$ individuals
       end
11:    Select points on the lower front with high crowding distance
12:    Generate next generation: Selection, Crossover and Mutation
   end
13: return the Best Pareto front
```

In line 11, the outputs of the sorting procedures (front and CD) are used by a selection method. Solutions with high CD are selected on the lower front and provided to evolutionary operators (line 12). In the selection, a binary tournament is used, individuals are selected from the lower front, and in the case of the same fronts, the solution with greater CD is chosen. After that, crossover and mutation operators are employed, and a new generation is created. This process continues until the stop criterion is reached. The NSGA-II's output is the best Pareto front found. In this work, the NSGA-II is adopted to optimize CNN architectures through the maximization of two objective functions: accuracy and F_1-score.

Accuracy is the number of correctly predicted examples out of all the examples, and can be calculated by the following equation: $Accuracy = \frac{TP+TN}{TP+FP+FN+TN}$, where, TP and TN mean true positives and negatives, and FP and FN mean false positives and negatives. Accuracy is one of the most common metrics to assess the algorithm's performance on a classification task. Nonetheless, it is commonly used when the class distribution is similar, when the TP and TN are more important. However, most real-life classification datasets are imbalanced. Thus, we chose an additional metric to assess the quality of the candidate solutions.

F_1-score is the harmonic average between the sensitivity (Recall)= $\frac{TP}{TP+TN}$ and the precision= $\frac{TP}{TP+FP}$. This metric is obtained by the following equation: F_1-score = $\frac{2 \times Recall \times Precision}{Recall+Precision}$. As it can be seen, F_1-score is adopted when the FN and FP are crucial. In cases where datasets have an imbalanced class distribution or presents a high overlapping

Authorized licensed use limited to: Hebei University of Technology. Downloaded on February 02,2023 at 14:47:30 UTC from IEEE Xplore. Restrictions apply.

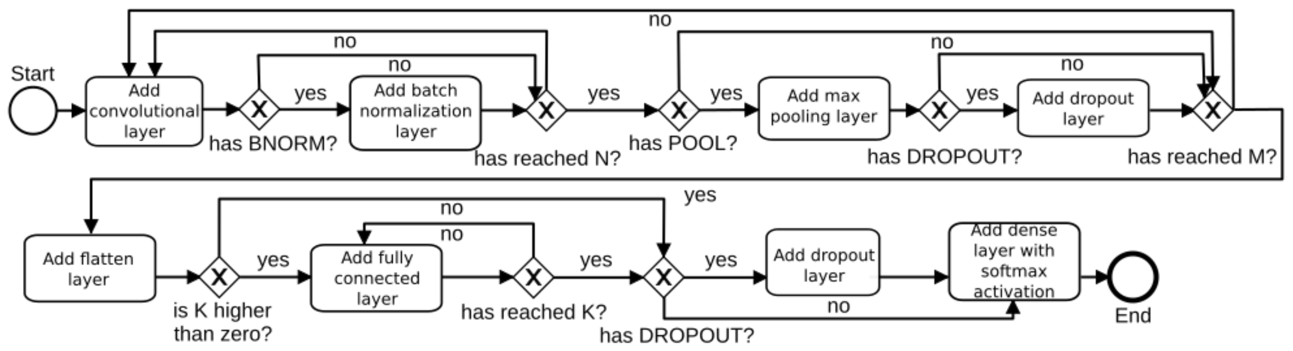


Fig. 3. Architecture creation flow.

of examples of different classes, F_1-score is a better metric to evaluate classification models.

V. Experimental Environment

This section presents the dataset used in the experiments, details about the adopted configurations and the comparative methods, as well as the evaluation metrics used. All executions of the experiment were performed in the Google Colab environment with GPU enabled. The language used to implement the proposed approach, and the comparative methods were Python ™.

A. Dataset

The CIFAR-10 [18] is a largely used classification dataset for deep neural networks evaluation. The CIFAR-10 has about $60,000$ real-life images of 32×32 pixels, all colored and divided into ten categories ($6,000$ images per class). There are $50,000$ training images and $10,000$ test images. For evaluation, the dataset is divided into five training batches and a single test batch; each batch has $10,000$ images. The test batch is composed of $10,000$ randomly-selected images, $1,000$ from each class. The training batches include the resting images in random order. Thus, some training batches may contain more images from one class than another. In each training batch, there is a total of $5,000$ images.

B. Experimental Setup

The PonyGE2 framework was adopted to run the GE, and the CNNs were created and executed using Keras open-source library. Besides, Scikit-Learn was used for normalizing the dataset and training networks. The framework' search engine used the parameters as follows: the simulations were performed on a population of 50 individuals and 30 generations. The selection operator adopted is the binary tournament (using a size equals 2), native to the NSGA-II. The crossover operator used is one-point, with a rate of 75%. The applied mutation is Int Flip per Codon, with an of 1%.

Regarding the CNNs execution, during optimization, each individual was trained for 70 epochs (for computational reasons) with a batch size of 128. During the process, the early stop feature was configured to stop training networks when there is no improvement.

C. Compared Methods

After the framework's execution, the best Pareto front is returned. As a Pareto front is composed of different non-dominated solutions, we selected a random CNN architecture to compare. Figure 4 presents the architecture of the selected CNN architecture, which is represented by the expression (((conv.bnorm)*3).pool.dropout)*3).flatten.((fc 256)*1)).dropout.fc 0.001. The selected CNN presents three sets of convolutional layers accompanied by batch normalization. Each set of three convolutions was followed by max-pooling layers along with dropout. After that, there is a single fully connected layer set with 256 neurons followed by a dropout layer. Finally, the CNN is set with a learning rate equals to 0.001.

The framework's selected CNN architecture was compared to eight different approaches: 1) GoogLeNet [19], 2) VGGNet [20], 3) ResNet [21], 4) GoogLeNet-TL, 5) VGGNet-TL, 6) ResNet-TL, 7) Generation of CNNs using grammar proposed by Diniz et al. [9] and 8) Generation of CNNs using grammar proposed by De Lima et al. [10]. All CNN algorithms involved in the experiments had their performance compared in terms of accuracy and F_1-score.

The GoogLeNet, VGGNet, and ResNet, were executed using their default parameters. Besides, we also compared the framework's CNN architecture with GoogLeNet, VGGNet, and ResNet using transfer learning (TL). The learning was extracted from the ImageNet dataset [22]. In these cases, the training using CIFAR10 images was performed only in the last network layers.

In this work, we also compared the framework's CNN architecture performance against CNNs generated by the grammars proposed by [9] and [10]. As the competitors are mono-objective GE-based approaches, we created two additional instances of the proposed framework: one set with the grammar proposed by Diniz et al. [9], and the other using the grammar developed by De Lima et al. [10]. The search engine used in both additional instances is the same used in the proposed framework. After the execution of each additional instance, a random CNN architecture was selected from the respective best Pareto front returned. The CNN architecture selected from the instance using Diniz et al.'s grammar is called (Diniz et

Authorized licensed use limited to: Hebei University of Technology. Downloaded on February 02,2023 at 14:47:30 UTC from IEEE Xplore. Restrictions apply.

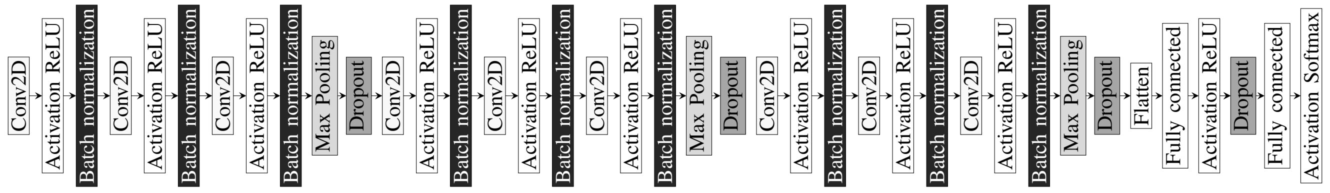


Fig. 4. Random CNN architecture from the best Pareto front returned by the proposed framework.

al.), and that selected from the instance using De Lima et al.'s grammar is called (De Lima et al.).

All competitor CNN algorithms were executed using the same dataset as the framework's selected CNN with the same training/test radio. Besides, all of them were trained using 100 epochs (without early stop), batch size equals 128, and learning rate of 0.001. Finally, All CNN algorithms were executed three times to produce mean and standard deviation.

VI. Results and Discussion

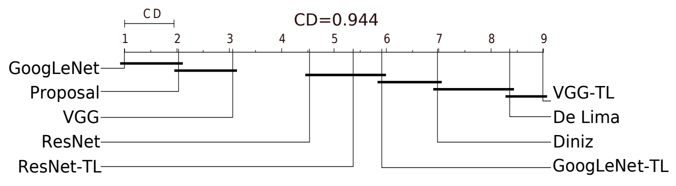
Table III presents the results obtained by the approaches in the CIFAR-10 database. Columns 2 and 3 highlight the mean and standard deviation of accuracy and F_1-score obtained, respectively. Columns 4 and 5, on the other hand, show the mean and standard deviation of accuracy and F_1-score achieved by the approaches submitted to the data augmentation technique. This technique creates new image variants through zooms and horizontal flips, further favoring CNN's performance. Finally, column 6 presents the number of parameters that compose each CNN architecture, and columns 7 and 8 present each model's time and the size after training.

As one can see, among the state-of-the-art algorithms, GoogLeNet obtained better accuracy and F_1-score compared to VGGNet, ResNet, and their versions using TL, in scenarios with and without data augmentation. Regarding the CNNs generated by grammar-based approaches, the proposal has more parameters, but regarding the results, the proposal reached much better accuracy and F_1-score values than Diniz et al. and De Lima et al., with differences greater than ten percentual points. In general, compared to the state-of-the-art CNNs, the proposal reached results similar to that of GoogLeNet, surpassing even the other architectures in accuracy and F_1-score. In addition to achieving competitive results, the proposed CNN architecture comprises only $4,458,282$ parameters, 79.57% fewer than GoogLeNet. Also, the proposed approach's model is more compact (in size) and has a shorter training time than those of the state-of-the-art. This shows that the models produced can also be used in contexts where memory savings are an important requirement.

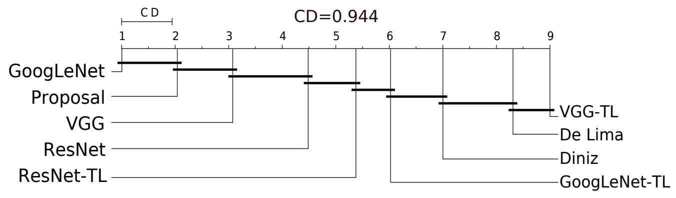
To perform a fair comparison, we evaluated the statistical significance of the results. The null hypothesis is that there is no statistical difference between the results of the nine approaches. The non-parametric hypothesis test used was the *Friedman Aligned-Ranks* significance level of $\alpha = 0.05$, and the null hypothesis was rejected with a p-value $= 4.02 \times 10^{-7}$. The rejection of the null hypothesis means that at least one result differs statistically from the others. Due to the rejection of the null hypothesis, the Finner post hoc test was performed to identify groups of results that show statistical similarity after a statistical test of multiple comparisons (such as the Friedman Aligned-Ranks test). The two approaches are significantly different if their corresponding average ranks differ by the Critical Difference (CD) value.

Figure 5 presents the comparison results using a graphical representation called CD diagram. This diagram ranks the approaches and group those that are statistically similar (considering the CD value). As it can be seen in Figure 5-(a), the proposal reached accuracy results statistically equal to GoogLeNet (in the first place). The VGG model also reached competitive results of accuracy being grouped with the proposed model. The other models did not achieve the same success as the top three and were overcome. Figure 5-(b) shows the proposal obtained F_1-score results statistically equal to GoogLeNet (in the first place), the VGG model is in the third place again. The other models are overcome statistically by the proposal.

The results showed the potential of the proposed framework for the generation of competitive CNN models. The model produced by the framework overcame the results achieved by Diniz et al. and De Lima et al., both in accuracy and F_1-score. Besides, the framework's CNN model obtained results statistically equivalent to the best state-of-the-art CNN model, GoogLeNet, and overcame all other models, but with much fewer parameters. Thus, the current proposal becomes an alternative for specialists to build competitive CNN models for classification problems.



(a) Accuracy



(b) $F_1 score$

Fig. 5. Critical difference diagrams for accuracy and F_1-score.

Authorized licensed use limited to: Hebei University of Technology. Downloaded on February 02,2023 at 14:47:30 UTC from IEEE Xplore. Restrictions apply.

TABLE III
COMPARATIVE PERFORMANCE ANALYSIS BETWEEN THE PROPOSED TECHNIQUE AND THE LITERATURE METHODS IN THE CIFAR-10.

CNN	**Accuracy**	$F_1 score$	**Accuracy***	$F_1 score$*	**# of params**	**Time (s)**	**Size (MB)**
GoogLeNet	87.22 (±0.26)	87.29 (±0.26)	91.13 (±0.34)	91.19 (±0.29)	21,823,274	39,519.14	250.80
VGGNet	86.04 (±0.33)	86.26 (±0.31)	89.22 (±0.34)	89.40 (±0.40)	45,294,194	46,163.14	270.44
ResNet	82.66 (±0.94)	82.73 (±0.88)	89.46 (±1.52)	89.51 (±1.52)	23,585,290	4,265.16	518.50
GoogLeNet-TL	83.19 (±0.07)	83.44 (±0.11)	84.65 (±0.25)	84.81 (±0.27)	21,823,274	6,841.96	84.23
VGGNet-TL	60.48 (±0.37)	60.55 (±0.37)	60.47 (±0.17)	60.59 (±0.18)	134,301,514	16,145.51	90.62
ResNet-TL	84.67 (±0.14)	84.80 (±0.17)	86.19 (±0.51)	86.39 (±0.49)	23,585,290	2,642.19	272.79
Diniz et al.	76.76 (±0.58)	76.84 (±0.60)	82.81 (±0.67)	82.89 (±0.66)	926,826	2,173.38	10.66
De Lima et al.	70.37 (±0.27)	70.72 (±0.35)	74.04 (±0.55)	74.03 (±0.53)	284,234	2,207.29	3.29
Proposal	87.10 (±0.47)	87.29 (±0.43)	90.26 (±0.16)	90.36 (±0.13)	4,458,282	2,701.07	51.18

* Data augmentation applied.

VII. CONCLUSION AND FUTURE WORKS

In this work, a multi-objective grammatical evolution framework is proposed to generate optimized and low-complexity CNN architectures. For this, a compact grammar was created for CNN evolution, and the NSGA-II was set as the multi-objective algorithm of the search engine module to maximize accuracy and F_1-score. The proposal was compared to eight different CNN models: three state-of-the-art algorithms, their versions using transfer learning, and other two CNN models generated by grammar-based approaches. The results showed the model generated by the proposal reached equivalent accuracy and F_1-score values to the GoogLeNet, model that reached the best results, and with 79.57% fewer parameters. The other five state-of-the-art CNN models were overcome statistically by the proposed model in all metrics. Compared to the CNN models generated by the competitor grammars, the proposal reached results even better, with differences greater than ten percentual points in all metrics.

As future work, we intend to update the grammar with other important layers and parameters, set the search engine with other multi-objective algorithms, and analyze their impact on the evolution. Also, we plan to investigate the adoption of other grammar-based algorithms with different individual representations (e.g., tree), and their influence on the quality of the CNNs.

REFERENCES

[1] L. Liu, W. Ouyang, X. Wang, P. Fieguth, J. Chen, X. Liu, and M. Pietikäinen, “Deep learning for generic object detection: A survey,” *International journal of computer vision*, vol. 128, no. 2, pp. 261–318, 2020.

[2] Y. LeCun, Y. Bengio, and G. Hinton, “Deep learning,” *Nature Publishing Group*, vol. 521, no. 7553, p. 436, 2015.

[3] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner, “Gradient-based learning applied to document recognition,” *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[4] 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.

[5] J. Frankle and M. Carbin, “The lottery ticket hypothesis: Finding sparse, trainable neural networks,” *arXiv preprint arXiv:1803.03635*, 2018.

[6] 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*, 2019, pp. 293–312.

[7] G. Neto, P. B. Miranda, G. D. Cavalcanti, T. Si, F. Cordeiro, and M. Castro, “Layers sequence optimizing for deep neural networks using multiples objectives,” in *2020 IEEE Congress on Evolutionary Computation (CEC)*. IEEE, 2020, pp. 1–8.

[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*, 2018, pp. 19–34.

[9] 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 *Encontro Nacional de Inteligência Artificial e Computacional*, 2018, pp. 82–93.

[10] R. H. R. de Lima, A. Pozo, and R. Santana, “Automatic design of convolutional neural networks using grammatical evolution,” in *Brazilian Conference on Intelligent Systems (BRACIS)*, 2019, pp. 329–334.

[11] M. O’Neill and C. Ryan, “Grammatical evolution by grammatical evolution: The evolution of grammar and genetic code,” in *European Conference on Genetic Programming*. Springer, 2004, pp. 138–149.

[12] 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.

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

[14] M. Suganuma, S. Shirakawa, and T. Nagao, “A genetic programming approach to designing convolutional neural network architectures,” in *Proceedings of the Genetic and Evolutionary Computation Conference*, 2017, pp. 497–504.

[15] B. Evans, H. Al-Sahaf, B. Xue, and M. Zhang, “Evolutionary deep learning: A genetic programming approach to image classification,” in *Congress on Evolutionary Computation (CEC)*, 2018, pp. 1–6.

[16] X. Glorot, A. Bordes, and Y. Bengio, “Deep sparse rectifier neural networks,” in *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*, vol. 15, 2011, pp. 315–323.

[17] D. P. Kingma and J. Ba, “Adam: A method for stochastic optimization,” 2017.

[18] A. Krizhevsky and G. Hinton, “Learning multiple layers of features from tiny images,” University of Toronto, Toronto, Ontario, Tech. Rep. 0, 2009.

[19] H. A. 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.

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

[21] 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.

[22] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei, “ImageNet Large Scale Visual Recognition Challenge,” *International Journal of Computer Vision (IJCV)*, vol. 115, no. 3, pp. 211–252, 2015.

Authorized licensed use limited to: Hebei University of Technology. Downloaded on February 02,2023 at 14:47:30 UTC from IEEE Xplore. Restrictions apply.