A New Grammar for Creating Convolutional Neural Networks Applied to Medical Image Classification

Cleber A.C.F da Silva, Péricles B.C. Miranda, Filipe R. Cordeiro,
Department of Statistics and Informatics, Federal Rural University of Pernambuco (UFRPE), Recife, Brazil
{cleber.cabral, pericles.miranda, filipe.rolim}@ufrpe.br

***Abstract*—In the last decade, the adoption of Deep Convolutional Neural Networks (CNNs) has been successfully applied to solve computer vision tasks, such as image classification in the medical field. However, the several architectures proposed in the literature are composed of an increasing number of parameters and complexity. Therefore, finding the optimal trade-off between accuracy and model complexity for a given data set is challenging. To help the search for these suitable configurations, this work proposes using a new Context-Free Grammar associated with a Multi-Objective Grammatical Evolution Algorithm that generates suitable CNNs for a given image classification problem. In this structure, the new grammar maps every possible search space for the creation of networks. Furthermore, the Multi-Objective Grammatical Evolution Algorithm used optimizes this search taking into account two objective functions: accuracy and F_1-score. Our proposal was used in three medical image datasets from MedMNIST: PathMNIST, OCTMNIST, and OrganMNIST_Axial. The results showed that our method generated simpler networks with equal or superior performance from state-of-the-art (more complex) networks and others CNNs also generated by grammatical evolution process.**

***Index Terms*—Grammatical Evolution, Deep Neural Networks, Multi-Objective Optimization**

I. Introduction

The area of computer vision has attracted a lot of attention in the last decade due to how Convolutional Neural Networks (CNNs) have added performance in solving problems such as image classification and object detection [1], acting in different application areas [2]. Several CNN architectures have been proposed in the literature in the last years, with an increasing number of parameters, depth, and complexity. Nonetheless, choosing the optimal architecture and its respective parameters is a challenging and time-consuming task due to the diversity of possible arrangements of its structure. State-of-the-art (SOTA) CNNs, like ResNet-152 [3] have 60 million trainable parameters. However, they have been originally proposed to achieve the highest performance on large-scale data sets, such as ImageNet [4], which has millions of images and thousands of classes. For small data sets, which specific characteristics, such as in medical data, a smaller model with fewer parameters might be the best option and achieve the same results. The challenge is how to find the best model and parameters for a given data set.

Frank and Carbin [5] demonstrated that it is possible to obtain, through pruning techniques, sub-nets with a reduced number of parameters without compromising the accuracy of the model. However, pruning techniques have some limitations, such as the fact that the final solution depends directly on the original CNN and the sequence of its layers.

To solve this problem, researchers have been adopting the automation of network design combined with optimization. Some works have optimized these CNNs using Neuro-Evolution [6] and Genetic Algorithms [7]. Nonetheless, the results obtained in [8]–[12] showed that Grammatical Evolution (GE) is a promising approach for the optimization of CNNs. GE is a Genetic Programming (GP) algorithm that uses context-free grammars to evolve programs (in our case, CNNs). The grammar can incorporate knowledge about the problem domain to guide the GE search better, avoiding the syntactic error of the generated programs [13]. Thus, GE has potential advantages to improve algorithm design, to represent and involves more complex architectures [13].

Aiming to use the potential of GE, this article proposes the use of a new Grammar combined with a Multi-Objective Evolutionary Algorithm for the optimization of CNN architectures. The contributions of this work are three: (1) we created a new Context-Free Grammar for mapping CNNs architectures that allows the addition of regularization layers (absent in most related works), (2) we implemented a Multi-Objective Search to optimize these architectures, and finally, (3) we applied this optimization process to a recently published medical imaging dataset. The best architectures found were compared to state-of-the-art CNNs (deeper and more complex) and others networks generated by grammars. Their efficiency was evaluated according to the accuracy and F_1-score values obtained after testing the CNNs. The training and testing of the networks were carried out using three medical imaging datasets that compose MedMNIST [14]: PathMNIST, OCTMNIST, and OrganMNIST_Axial. The results showed that our proposal generated competitive and low complexity CNN models compared to other state-of-the-art networks and CNNs generated by others grammars.

The article is organized as follows: Section II introduces the basic concepts for a better understanding of this work. Section III reports related works found. In Section IV, we present the proposed work. The experimental environment and methodology are presented in Section V. Section VI discusses the experimental results, and finally Section VII contains the conclusion and future work of the study.

II. Background

In this section, we will present the fundamental concepts of Convolutional Neural Networks, Grammatical Evolution, and Multi-Objective Optimization.

A. Convolutional Neural Networks

CNNs are made up of multiple layers [2], where each is responsible for a specific task, processing data from previous layers. The way these layers are arranged and their respective types act directly on how CNN will perform. There are usually three types of layers most popularly used: convolutional layers, pooling layers, and fully connected layers. Convolutional layers act as a composition of filters that identify the main characteristics of a given input. Pooling layers implement the data dimensionality reduction operation, focusing on reducing the number of trainable network parameters where, generally, pooling layers follow convolutional layers. Finally, the fully connected layers are nothing more than the neural network itself. They interconnect a series of neurons that receive the data processed by the previous layers and make their respective classifications. There are others types of layers responsible for the regularization of the network training process, which are the batch normalization layers and dropout layers. Batch normalization layers are responsible for normalizing the input between layers. They are usually found between the convolutional and pooling layers. Dropout layers are adopted to avoid overfitting in the model, randomly disabling the percentage of neurons present in the network. They are commonly used after pooling layers or fully connected layers.

B. Grammatical Evolution

Grammatical Evolution (GE) [15] is an alternative for performing the difficult task of creating optimized CNNs. GE is a Genetic Programming algorithm [16] that uses Context-Free Grammar to evolve certain programs of varying lengths [17]. It is composed of three components: a search engine, a grammar, and a mapping process.

The search engine uses the principles of evolutionary algorithms to iteratively evolve possible solutions. Each solution is represented by a genotype (generally a binary array of varying lengths), containing the individual's information. Each compartment of this array is called a codon and can receive 8-bit numeric values. During the execution of the search engine, each genotype is evaluated through its fitness value. In our case, the genotype must be converted into a phenotype, an executable program that is CNN itself, to have the aptitude evaluated. This conversion is done by the mapping process that uses a Context-Free Grammar in the Backus-Naur Form (BNF) [17].

The Context-Free Grammar is 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 rules of production, 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 of the possible ones productions is delimited by the disjunctive symbol '|' [18]. Figure 1 shows an example of a context-free grammar that generates arithmetic expressions, where a possible grammar-generated individual would be $x{*}(x{+}x)$. Thus, we can generate a whole population where individuals are valid according to what was described in the grammar.

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

Fig. 1. Example of a Context-Free Grammar.

C. Multi-Objective Optimization

Multi-Objective Optimization (MOO) problems have more than one objective function to be optimized. For a given candidate solution search space, $\mathcal{S}$, MOO algorithms aim to find solutions that best satisfy all conflicting objectives ($\vec{f} = (f_1, f_2, ..., f_n)$) at the same time, where n is the number of goals. Each candidate solution within $\mathcal{S}$ 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 each objective function within $\vec{f}$, where it returns the respective fitness value. Therefore, $\vec{f}(\vec{x}) = (f_1(\vec{x}), f_2(\vec{x}), ..., f_n(\vec{x}))$ represents a vector of all fitness values belonging to the solution vector $\vec{x}$.

Unlike single-objective optimization, in MOO each quality of the solution is determined by a vector of fitness values instead of a single value. Thus, the solutions are usually compared to each other through the Pareto Dominance [19]. Given two vectors $\vec{x}$, $\vec{y} \in \mathbb{R}^n$, $\vec{x}$ dominates $\vec{y}$ (denoted by $\vec{x} \prec \vec{y}$) if $\vec{x}$ surpasses $\vec{y}$ by at least one goal and $\vec{x}$ is not worse than $\vec{y}$ in any other goals. $\vec{x}$ 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 within the objective space are known as the Pareto Front.

III. Related Works

Using algorithms based on grammars is not a new approach [15], [17], as well as its use in the construction of neural networks [20], [21]. However, over time, optimizing CNNs became a laborious task, and recently different studies used the grammatical evolution technique to accelerate this process.

Soltanian et al. (2013) [11] applied grammar-based evolutionary algorithms linked to generating MLPs. The networks were optimized, seeking to maximize accuracy over four datasets (Breast cancer, Heart-Cleveland, Ionosphere, and Iris) and achieved results superior to other techniques.

Assunção et al. (2018, 2019) [8], [22] used Neuro-Evolution to propose DENSER (*Deep Evolutionary StructurEd Representation*), which combines the use of evolutionary algorithms with the *Dynamic Structured Grammatical Evolution* (DSGE).

DENSER was used to generate valid CNNs seeking to optimize them by maximizing the accuracy (mono-objective) in four datasets: MNIST, FashionMNIST, CIFAR-10, and CIFAR-100. The results showed that the technique is quite promising, producing CNNs competitive with state-of-the-art networks in the literature. However, the grammar used in the process does not allow some regularization layers to be added to networks (e.g., dropout layers). Another point to be highlighted is that this grammar adds many CNN configuration parameters to its structure (e.g. stride, kernel size, number of filters, and values between ranges), thus the number of terminal combinations grows, greatly increasing the process search space, where it may cause the algorithm to not converge.

Diniz et al. (2018) [9] also used the evolutionary principles associated with context-free grammar to optimize neural networks. Its grammar represented topologies that contained convolutional, pooling, and fully connected layers. The associated genetic algorithm also sought to optimize the networks by maximizing accuracy, but only in one dataset, the CIFAR-10. The results showed that the method is promising, generating light and low complexity architectures. However, its grammar did not allow the addition of some regularization layers (e.g., dropout, batch normalization) to the network design, and seeks only the accuracy maximization. In addition, the grammar used generates a very small search space, with only 54 possible individuals.

Suganuma et al. (2017) [23] used an evolutionary algorithm associated with *Cartesian Genetic Programming* (CGP) to generate CNNs according to a fixed structure. Its structure allowed six types of layers to be added to the model (convolutional layers, max-pooling, average pooling, sum, concatenation, and ResBlocks). The tests were performed on the CIFAR-10 and CIFAR-100 datasets, and the evolutionary algorithm was guided to seek the maximization of accuracy. The results showed that the technique is promising, generating models that are competitive with state-of-the-art networks. Although, the method did not allow the addition of regularization layers to the model and guided the evolutionary algorithm in a single-objective process.

In our later work (Da Silva et al. (2021) [12]) we used a process for creating and optimizing CNNs similar to the current one, maximizing accuracy and F_1-score through NSGA-II. The tests were performed on the CIFAR-10 dataset. The grammar used can generate a search space with 2592 different individuals and allowed the creation of CNNs competitive with state-of-the-art networks and other networks also generated by grammars. However, when performing a more critical analysis, we realized that the associated grammar has a limitation regarding the arrangement of convolutional and batch normalization layers, where they tend to be repeated between convolutional blocks, limiting these layers' diversity. Thus, the grammar proposed in this work seeks to solve this problem.

This work proposes a novel context-free grammar associated with an evolutionary algorithm focused on maximizing two goals: accuracy and F_1-score, while related works focus on single-objective optimization. The grammar is composed of layers that add high relevance to the resolution of classification problems and allow the addition of regularization layers, absent from many of the related works. Furthermore, our grammar generates a search space with 6426 different individuals and avoids a possible non-convergence due to an infinite number of possibilities that can be seen in Assunção et al.. In addition, our new grammar manages to generate the same individuals present in the grammars of Diniz et al. [9] and Da Silva et al. [12], thus, those grammars' search spaces are contained in our grammar.

Another important point is the fact that our grammar adds more granularity options in the construction sequence of convolutional blocks, ranging from 1 to 3, with each one being able to have particular configurations. In addition, our proposal applied the techniques described in a dataset of medical images, the MedMNIST [14], a point that was not performed by any other related work. This dataset has just been published and the accuracy and F_1-score data obtained in this research can serve as a comparison for future works. Table I presents the comparative table between our proposal and the others mentioned above.

TABLE I
COMPARISON BETWEEN RELATED WORKS AND THE PROPOSED TECHNIQUE.

Reference	***Grammar***	***Algorithm***	**Dataset(s)**
Soltanian et al. [11]	*Multi-Layer Perceptron* and arithmetic operators.	GA	Breast cancer, Heart-Cleveland, Ionosphere and Iris
Assunção et al. [8], [22]	Convolutional layers, max pooling, fully connected, softmax, batch normalization, pooling type, activation functions and learning rate.	GA	CIFAR-10, CIFAR-100, MNIST and FashionMNIST.
Diniz et al. [9]	Convolutional, max pooling and fully connected layers.	NSGA-II	CIFAR-10
Suganuma et al. [23]	Convolutional layers, max pooling, average pooling, sum, concatenation and ResBlocks.	CGP	CIFAR-10 and CIFAR-100.
Da Silva et al. [12]	Convolutional layers, max pooling, dropout, fully connected, batch normalization and learning rate.	NSGA-II	CIFAR-10
Proposal	Convolutional layers, max pooling, dropout, fully connected, batch normalization and learning rate.	NSGA-II	PathMNIST, OCTMNIST and OrganMNIST Axial.

IV. PROPOSAL

This work proposes a new Context-Free Grammar associated with a Multi-Objective Evolutionary Algorithm for the generation of CNNs that can solve classification problems in a medical image dataset.

A. Context-Free Grammar

Figure 2 shows the grammar, in Backus-Naur (BNF) format, proposed for the creation of CNNs. It consists of a total of 15 tags and its permutations allow the creation of a search space with 6426 different individuals.

The tag ⟨cnn⟩ defines the entire structure of the CNN, containing the convolution blocks, the data flattening layer, the fully connected layers, and, finally, the learning rate. The tag ⟨blocks⟩ represents the convolutional blocks of the network. It is defined by ⟨nblocks⟩⟨block⟩, where ⟨nblocks⟩ is the number of times the ⟨block⟩ tag will repeat in the network , which varies from 1 to 3 according to line 14. Continuing on, the tag ⟨block⟩ defines the set of convolutive layers, ⟨convs⟩, followed or not by the layer of *pooling* ⟨pooling⟩. In line 4, the convolutional block is defined, composed of 1, 2, or 3 convolutional layers. Each convolutional layer, line 5, may or may not be followed (represented by the symbol λ) by batch normalization layers ⟨bnorm⟩ (line 9). The tag ⟨pooling⟩ shows that pooling layers can be followed by dropout layers or not. The tag ⟨flatten⟩, defines the data flattening layer. Continuing, the tag ⟨fc⟩ defines the fully connected layers that may or may not exist in the grammar. If they exist, they can have four values of the number of neurons and can repeat 1 or 2 times, according to the tag ⟨nfcs⟩. Also, fully connected layers may or may not be followed by dropout layers. More details are described in line 8. Finally, the tag ⟨lr⟩ defines the learning rate at which the CNN will be optimized, which can take three values as defined in line 12.

$$\langle\text{cnn}\rangle \models \langle\text{blocks}\rangle\langle\text{flatten}\rangle\langle\text{fc}\rangle\langle\text{lr}\rangle \tag{1}$$

$$\langle\text{blocks}\rangle \models \langle\text{nblocks}\rangle\langle\text{block}\rangle \tag{2}$$

$$\langle\text{block}\rangle \models \langle\text{convs}\rangle\langle\text{pooling}\rangle \tag{3}$$

$$\langle\text{convs}\rangle \models \langle\text{conv}\rangle \mid \langle\text{conv}\rangle\langle\text{conv}\rangle \mid \langle\text{conv}\rangle\langle\text{conv}\rangle\langle\text{conv}\rangle \tag{4}$$

$$\langle\text{conv}\rangle \models \texttt{(Conv}\langle\text{bnorm}\rangle\texttt{),} \tag{5}$$

$$\langle\text{pooling}\rangle \models \texttt{(MaxPool}\langle\text{dropout}\rangle\texttt{),} \mid \lambda \tag{6}$$

$$\langle\text{flatten}\rangle \models \texttt{(Flatten),} \tag{7}$$

$$\langle\text{fc}\rangle \models \texttt{(Fc}\langle\text{nfcs}\rangle\langle\text{units}\rangle\langle\text{dropout}\rangle\texttt{),} \mid \lambda \tag{8}$$

$$\langle\text{bnorm}\rangle \models \texttt{BNorm} \mid \lambda \tag{9}$$

$$\langle\text{dropout}\rangle \models \texttt{Dropout} \mid \lambda \tag{10}$$

$$\langle\text{lr}\rangle \models \texttt{(Lr}\langle\text{rates}\rangle\texttt{)} \tag{11}$$

$$\langle\text{rates}\rangle \models \texttt{0.01} \mid \texttt{0.001} \mid \texttt{0.0001} \tag{12}$$

$$\langle\text{units}\rangle \models \texttt{64} \mid \texttt{128} \mid \texttt{256} \mid \texttt{512} \tag{13}$$

$$\langle\text{nblocks}\rangle \models \texttt{1} \mid \texttt{2} \mid \texttt{3} \tag{14}$$

$$\langle\text{nfcs}\rangle \models \texttt{1} \mid \texttt{2} \tag{15}$$

Fig. 2. Context-Free Grammar in Backus-Naur (BNF) syntax used in the study.

As can be seen, the proposed grammar is flexible. It allows the creation of small to large networks, from those with only convolutional layers to more complex ones with convolutional, pooling, batch normalization, dropout, and fully connected layers. The grammar allows tags like ⟨convs⟩, ⟨nblocks⟩ and ⟨nfcs⟩ to be altered to generate deeper, and therefore more complex, networks. Some important parameters in the construction of CNNs remained fixed, such as activation function, padding, stride, and the number of convolutional filters. In the convolutional layers, the activation function used was ReLU because it has low computational power and still presents good results compared to others such as sigmoid and hyperbolic tangent [24]. The pooling layers have been set to the maximum type, thus highlighting the most important aspects of the images. The optimizer used in training was Adam for bringing fast and satisfactory results compared to other optimizers [25]. The optimizer's learning rate comes from the tag ⟨lr⟩ present in the grammar. The error function used was the *Categorical Cross-Entropy* [26]. The CNN models were configured to be trained in 70 epochs with a batch size of 128. More details about these parameters can be seen in Table II.

TABLE II
CNN FIXED PARAMETERS.

Parameter	***Value***
Kernel size	3 x 3
# of filters	Starts with 32; duplicates for every two convolutions.
Stride	1
Pooling size	2 x 2
Dropout rate	0.25 after pooling layers; 0.50 after fully connected layers.
Activation function	ReLU for convolutions; Softmax for the last model layer.
Optimizer	Adam
Loss function	Categorical Cross-Entropy
# of epochs	70
Batch size	128

Early stopping	monitor=val_accuracy, mode=max
	patience=10, baseline=0.5

B. Multi-Objective Evolutionary Algorithm

According to Neto et al. [7] finding CNN models with a good balance between accuracy, generalization, and precision is a challenging task. Thus, it is not suitable to optimize CNNs using only single-objective optimization algorithms due to the several aspects that must be considered when choosing the ideal CNN. Therefore, we decided to adopt the use of a widely used Multi-Objective Evolutionary Algorithm (MOEA), the NSGA-II [27]. This algorithm has achieved good performance compared to other multi-objective optimizers [9].

NSGA-II is a MOEA with a strong elitist strategy. Its pseudocode can be seen in the Algorithm 1, where initially, before the iterative process, a new population P with size N' is randomly generated. Each individual has their aptitude values evaluated and, after that, they are sorted based on the Pareto order (lines 1-3). Once this is done, the selection, crossing, and mutation operators are applied, generating a new offspring population. The iterative process is started, and for each generation, the algorithm classifies individuals according to non-dominance, thus producing some Pareto fronts (lines 7 and 8). A classification based on Agglomeration Distance (AD) is performed upfront to promote diversity. The AD represents the distance between a solution and its neighbors. Since each solution has its distance assessed, they are ranked descending. This strategy aims to benefit border solutions placed in regions of the search space with fewer neighbors.

In line 11, the output of the classification procedure is used by the selection method. High DA solutions are selected on the lower front and provided for evolutionary operators (row 12). In selection, a binary tournament is used. Individuals are selected from the lower front, and, in the case of equal fronts, the solution with the highest AD is chosen. After that, the

Algorithm 1: NSGA-II pseudocode.

```
Input: $N'$, $g$, $f_k(X)$ ▷ $N'$ members evolved in $g$ generations to solve $f_k(X)$
Generate a random population $P$ with size $N'$
Evaluates objectives values;
Assigns a ranking (level) based on the ordered Pareto front
Generates a daughter population: Selection, Crossover and Mutation
for $i \leftarrow 1$ to $g$ do
    for each Parent and child in $P$ do
        Assigns a ranking (level) based on the ordered Pareto front
        Generate undominated solution sets
        Determines Agglomeration Distance
        Internal loop adding solutions for the next generation starting from the front first up to $N'$ individuals
    end
    Select points on the lower front and with high agglomeration distance
    Generates the next generation: Selection, Crossover and Mutation
end
return The best Pareto front
```

crossover and mutation operators are employed, and a new generation is created. This process continues until the stop criterion is met. The NSGA-II output is the best Pareto front found. In this work, NSGA-II is adopted to optimize CNNs architectures by maximizing two objective functions: accuracy and F_1-score.

1) Accuracy: It is the total hit rate of the classifier used, regardless of the classes in the example. This variable is obtained through the expression $Accuracy = \frac{TP+TN}{TP+FP+FN+TN}$, where TP and TN mean true positive and negative, and FP and FN mean false positive and negative. Accuracy is one of the most common metrics for evaluating the performance of an algorithm in classification tasks. However, it is commonly used when the distribution of classes is similar when TP and TN are more important. However, most real-life datasets are unbalanced. Therefore, we chose an additional metric to assess the quality of candidate solutions.

2) F_1-score: It is the harmonic mean between the sensitivity ($Recall = \frac{TP}{TP+TN}$) and the precision ($Precision = \frac{TP}{TP+FP}$). It is primarily used to compare the performance between two classifiers. The relationship between Recall and Precision is that F_1-score only has high values if both metrics also have high values. In cases where the datasets do not have a balanced distribution of classes or have a large overlap of examples from different classes, F_1-score is the best metric to assess the ranking of models. The F_1-score is obtained through the expression $F_1score = \frac{2 \times Recall \times Precision}{Recall+Precision}$.

V. Methodology

This section presents the data set used in the experiments, details on the methodology and configurations adopted, comparative methods, and the metrics used to evaluate the solutions. All experiments were run using the Google Colab [1] environment with the use of TPU enabled. The language used to implement the proposed approach and comparative methods was Python™.

A. Dataset

The dataset used in the experiments was MedMNIST [14], which is a collection of ten subsets of pre-processed medical images released under the Creative Commons license. All images are set to the dimension of 28×28 pixels. From these ten subsets, we chose three of them: the ones with more training, validation, and testing data, in addition to being divided into multiple classes necessary for the classification task. They are PathMNIST, OCTMMNIST, and OrganMNIST Axial. Figure 3 shows examples of images contained in the datasets.

1) PathMNIST: Set composed of histological images of colorectal cancer stained with hematoxylin and eosin. Nine tissue types are involved, resulting in a multi-class classification task. Images are $3 \times 28 \times 28$ in size. In total, there are $89,996$, $10,004$ and $7,180$ training, validation and testing images, respectively.

2) OCTMNIST: Set composed of Optical Coherence Tomography (OCT) images valid for retinal diseases. Images are divided into four classes, leading to a multi-class classification task. Images are $1 \times 28 \times 28$ in size. In total, there are $97,477$, $10,832$ and $1,000$ training, validation and testing images, respectively.

3) OrganMNIST Axial: Set composed of 3D computed tomography images of the Liver Tumor Segmentation Reference (LiTS). 3D images are transformed into grayscale and recalled in planes, in this case, the axial plane. Images are $1 \times 28 \times 28$ in size. In total, there are $34,581$, $6,491$, $17,778$ training, validation and testing images, respectively. In this work, we are not using the data augmentation technique on training the images.

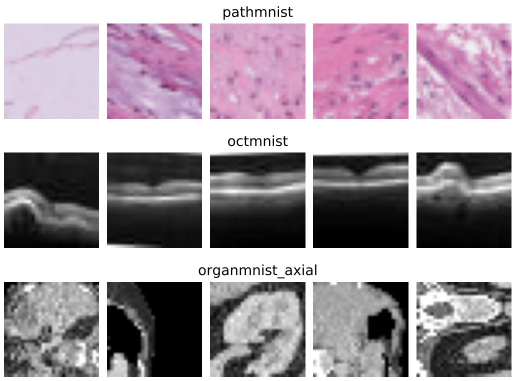

Fig. 3. Examples of images contained in the datasets.

B. Setup Experimental

For the grammatical evolution execution, we use the PonyGE2 framework [28]. To build and execute the convolutional neural networks, we use the open-source library Keras [29]. In addition, for data normalization and assistance in the training process, we used the Scikit-Learn library [30].

The multi-objective evolutionary algorithm was configured with the following parameters: the process was performed on

[1] https://colab.research.google.com/

a population of 50 individuals through 30 generations. The selection operator adopted was the *Binary Tournament* (of size 2), native to NSGA-II. For the crossing of individuals, we used the *One Point Crossover* operator, with a probability of 75%. The mutation operator used was the *Int Flip Per Codon*, with a probability of 1%. Table III summarizes the parameters used in the evolutionary algorithm.

TABLE III
EVOLUTIONARY ALGORITHM PARAMETERS.

Parameter	***Value***
Number of generations	30
Population size	50
Mutation operator	*Int Flip Per Codon*
Crossover operator	*One Point Crossover*
Mutation rate	0.01
Crossover rate	0.75
Tournament size	2
Elite size	1
Selection Algorithm	*nsga2_selection*
Replacement Algorithm	*nsga2_replacement*

In CNN's training, the individuals were trained throughout 70 epochs with a batch size of 128. To identify individuals that produce bad networks and thus stop their training, we apply the early stop configuration available in Keras to identify such networks. The fitness values were monitored, and when a network, in the first ten epochs, does not reach a proficiency of at least 50% in its metrics, its training process is interrupted. This early stop procedure was defined empirically.

C. Comparative Methods

After executing the process for each dataset, the NSGA-II returns a Pareto front that can contain one or more non-dominated solutions. In our case, in all datasets, the Pareto front returned only one individual who was not dominated by any other network. To carry out the comparison, we selected this individual from these Pareto fronts for each associated dataset. We will call these networks *PathMNIST Proposal*, *OCTMNIST Proposal* and *OrganMNIST Proposal* for the PathMNIST, OCTMNIST and OrganMNIST Axial datasets, respectively. The layer arrangement of each generated CNN can be seen in Table IV.

Each selected CNN was compared with six different networks, four available in in Keras's *applications* module: 1) InceptionV3 [31], 2) ResNet50V2 [32], 3) EfficientNetB1 [33] and 4) DenseNet169 [34]; and two approaches that create CNNs through an optimization process associated with a grammar: Diniz et al. [9], and Da Silva et al. [12]. The CNNs generated through an optimization process were obtained from the execution of the evolutionary process associated with their respective grammars, under the same parameters used in the proposal process. These grammars were chosen due to the ease of implementation and reproduction of their processes.

All these CNNs had their performances compared in terms of accuracy and F_1-score and were trained for 100 epochs (without the early stop criterion) under the same parameters, the same training/validation/test ratio, batch size of 128, optimizer Adam [25] and learning rate of 0.001. The images trained by the networks coming from Keras had to be resized to a minimum size acceptable by all, which in this case was 75×75 pixels. Each CNN, for each dataset, was run 10 times to produce the mean and standard deviation of the collected metrics.

TABLE IV
CNN LAYER STRUCTURE GENERATED FOR EACH DATASET.

PathMNIST Proposal	***OctMNIST Proposal***	***OrganMNIST Proposal***

Conv(32)
Conv(32)
Conv(64)
MaxPool
Dropout(0.25)
	Conv(32)
	Conv(32)
	Conv(64)
	Batch Normalization
	MaxPool
	Dropout(0.25)
		Conv(32)
		Batch Normalization
		Conv(32)
		MaxPool
		Dropout(0.25)

Conv(64)
Conv(128)
Conv(128)
MaxPool
Dropout(0.25)
	Conv(64)
	Conv(128)
	Conv(128)
	Batch Normalization
	MaxPool
	Dropout(0.25)
		Conv(64)
		Batch Normalization
		Conv(64)
		MaxPool
		Dropout(0.25)

	Conv(256)
	Conv(256)
	Conv(512)
	Batch Normalization
	MaxPool
	Dropout(0.25)
		Conv(128)
		Batch Normalization
		Conv(128)
		MaxPool
		Dropout(0.25)

Flatten	Flatten	Flatten

Dense(128)
Dropout(0.5)
	Dense(64)
	Dropout(0.5)
		Dense(64)
		Dropout(0.5)

Dense('softmax', 9)	Dense('softmax', 4)	Dense('softmax', 11)
Adam(lr=0.001)	Adam(lr=0.001)	Adam(lr=0.0001)

VI. RESULTS

The results obtained can be seen in Table V. Columns 3 and 4 bring the means and standard deviations for the accuracy metrics and F_1-score, for each CNN and respective dataset. Besides, columns 5 to 8 present the number of layers, number of parameters, networks' size in Megabytes, and the total training time for each network are listed.

We can see that the proposal performed better, for accuracy and F_1-score, than the other networks in the PathMNIST dataset. In the other two datasets, the winning CNN was DenseNet169 [34], followed by the proposal and Da Silva et al. [12]. Our proposal also beat the other networks generated by grammars. Thus, we can highlight the improvement of our grammar in relation to our previous work, Da Silva et al., in generating more specialized individuals.

It is important to highlight that despite the proposed CNNs are in second place; they present very competitive results associated with a minimalist design. The proposed networks for PathMNIST, OCTMNIST, and OrganMNIST respectively present 91.38%, 79.04%, and 97.14% less trainable parameters compared to the DenseNet169 network. Furthermore, they are more compact in size (MB), have fewer layers, and have a shorter training time. In this way, these produced models can also be used in contexts where memory savings are an important requirement.

To complement the previous analysis, we evaluated the statistical significance of the results to make a fair comparison among the CNNs. The null hypothesis is that there is no statistical difference between the averages of the five networks for each dataset and each metric, accuracy, and F_1-score. We used the non-parametric test of *Friedman* with a significance level of $\alpha = 0.05$, where the null hypothesis was rejected, in

TABLE V
COMPARATIVE PERFORMANCE ANALYSIS BETWEEN THE PROPOSED TECHNIQUE AND STATE-OF-THE-ART METHODS IN THE THREE DATASETS.

Dataset	CNN	Accuracy	F_1-score	# of layers	# of parameters	Size (MB)	Time (s)
PathMNIST	Proposal	**0.9023** ($\pm$0.0101)	**0.9015** ($\pm$0.0104)	15	1,091,113	12.56	1332.90 ($\pm$57.08)
	DenseNet169 [34]	0.8718 ($\pm$0.0180)	0.8709 ($\pm$0.0180)	169	12,657,865	146.44	6074.60 ($\pm$197.25)
	EfficientNetB1 [33]	0.7926 ($\pm$0.0480)	0.7921 ($\pm$0.0483)	116	6,586,768	76.48	3606.00 ($\pm$85.14)
	InceptionV3 [31]	0.8563 ($\pm$0.0431)	0.8557 ($\pm$0.0428)	159	21,821,225	250.78	3892.90 ($\pm$47.87)
	ResNet50V2 [32]	0.7629 ($\pm$0.1300)	0.7622 ($\pm$0.1301)	50	23,583,241	270.42	3002.30 ($\pm$124.47)
	Diniz et al. [9]	0.8095 ($\pm$0.0225)	0.8086 ($\pm$0.0226)	8	33,833	0.43	1233.10 ($\pm$74.95)
	Da Silva et al. [12]	0.8913 ($\pm$0.0080)	0.8909 ($\pm$0.0088)	15	1,091,113	12.56	1326.30 ($\pm$59.99)
	p-value	3.60×10^{-10}	3.83×10^{-10}				
OCTMNIST	Proposal	0.7595 ($\pm$0.0292)	0.7623 ($\pm$0.0297)	23	2,649,892	30.44	1446.00 ($\pm$54.72)
	DenseNet169 [34]	**0.7652** ($\pm$0.0304)	**0.7697** ($\pm$0.0309)	169	12,643,268	146.28	6129.10 ($\pm$116.98)
	EfficientNetB1 [33]	0.7448 ($\pm$0.0130)	0.7473 ($\pm$0.0132)	116	6,579,783	76.40	3699.40 ($\pm$58.20)
	InceptionV3 [31]	0.7274 ($\pm$0.0181)	0.7308 ($\pm$0.0183)	159	21,810,404	250.67	4223.20 ($\pm$113.96)
	ResNet50V2 [32]	0.7445 ($\pm$0.0219)	0.7489 ($\pm$0.0210)	50	23,566,724	270.24	3057.80 ($\pm$77.88)
	Diniz et al. [9]	0.6752 ($\pm$0.0136)	0.6781 ($\pm$0.0136)	6	78,244	0.93	1308.10 ($\pm$92.43)
	Da Silva et al. [12]	0.7585 ($\pm$0.0114)	0.7616 ($\pm$0.0115)	21	1,895,140	21.81	1463.30 ($\pm$65.65)
	p-value	3.60×10^{-10}	1.09×10^{-9}				
OrganMNIST Axial	Proposal	0.9283 ($\pm$0.0070)	0.9283 ($\pm$0.0070)	20	361,835	4.24	596.80 ($\pm$10.86)
	DenseNet169 [34]	**0.9374** ($\pm$0.0026)	**0.9372** ($\pm$0.0026)	169	12,654,923	146.41	2363.90 ($\pm$99.13)
	EfficientNetB1 [33]	0.9062 ($\pm$0.0205)	0.9063 ($\pm$0.0204)	116	6,588,750	76.51	1567.10 ($\pm$12.99)
	InceptionV3 [31]	0.9105 ($\pm$0.0286)	0.9106 ($\pm$0.0284)	159	21,824,747	250.85	1728.70 ($\pm$39.64)
	ResNet50V2 [32]	0.9132 ($\pm$0.0568)	0.9132 ($\pm$0.0567)	50	23,581,067	270.41	1290.00 ($\pm$37.94)
	Diniz et al. [9]	0.8727 ($\pm$0.0056)	0.8725 ($\pm$0.0055)	8	34,411	0.43	572.50 ($\pm$28.29)
	Da Silva et al. [12]	0.9222 ($\pm$0.0071)	0.9222 ($\pm$0.0070)	23	548,139	6.39	684.70 ($\pm$32.35)
	p-value	3.52×10^{-10}	4.22×10^{-10}				

both metrics in all datasets, as can be seen in Table V. The rejection of the null hypothesis indicates that at least one result is statistically different from the others. Thus, to identify the groups of results that present statistical similarity in multiple comparisons, we apply the post hoc *Nemenyi* to obtain the average ranks and thus calculate the Critical Difference (CD) value of the results. Figure 4 shows the comparison of these results through a graphical representation called *CD Diagram*.

For the PathMNIST dataset, our proposal obtained data statistically equal to the Da Silva et al., DenseNet169 and InceptionV3 networks, surpassing the EfficientNetB1, ResNetV2 and Diniz et al. networks. For the other two datasets, the graphs show that the networks generated by our approach are statistically equal to the first placed network, DenseNet169. Thus, these graphs indicate that in all data sets, both in accuracy and in F_1-score, the Proposed networks reached data statistically equal to the first placed. Therefore, our proposed grammar becomes an interesting alternative for constructing competitive and low-complexity CNN models for image classification problems.

VII. CONCLUSIONS AND FUTURE WORK

In this work, we propose a new grammar for creating convolutional neural networks applied to medical image classification, associated with a multi-objective evolutionary algorithm for CNN optimization, based on two metrics: accuracy and F_1-score. The proposed approach was assessed in medical images from three datasets present in MedMNIST: PathMNIST, OCTMNIST, and OrganMNIST_Axial. The discovered CNNs were compared to four other state-of-the-art networks in the literature and two networks also generated by a grammatical evolution approach, applied to the same classification problems. The results showed that our generated models reached a prominent position both in accuracy and in F_1-score in all datasets, being equivalent to networks like DenseNet169 when compared statistically. However, our networks used about 91.38%, 79.04% and 97.14% (*PathMNIST Proposal*, *OCTMNIST Proposal* and *OrganMNIST Proposal*, respectively) less parameters compared to DenseNet169, showing that our models are effective and efficient, with a simpler architecture, and consuming less computational power and training time.

Our work also showed that the networks generated by the proposed new grammar also beat the other networks generated through the aforementioned grammatical evolution process.

As future work, we plan to update the grammar, adding other regularization settings. We also intend to add other important metrics that help in the convergence of the multi-objective algorithm.

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*, vol. 521, no. 7553, p. 436, 2015.

[3] 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.

[4] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "Imagenet: A large-scale hierarchical image database," in *2009 IEEE conference on computer vision and pattern recognition*. Ieee, 2009, pp. 248–255.

[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*. Elsevier, 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.

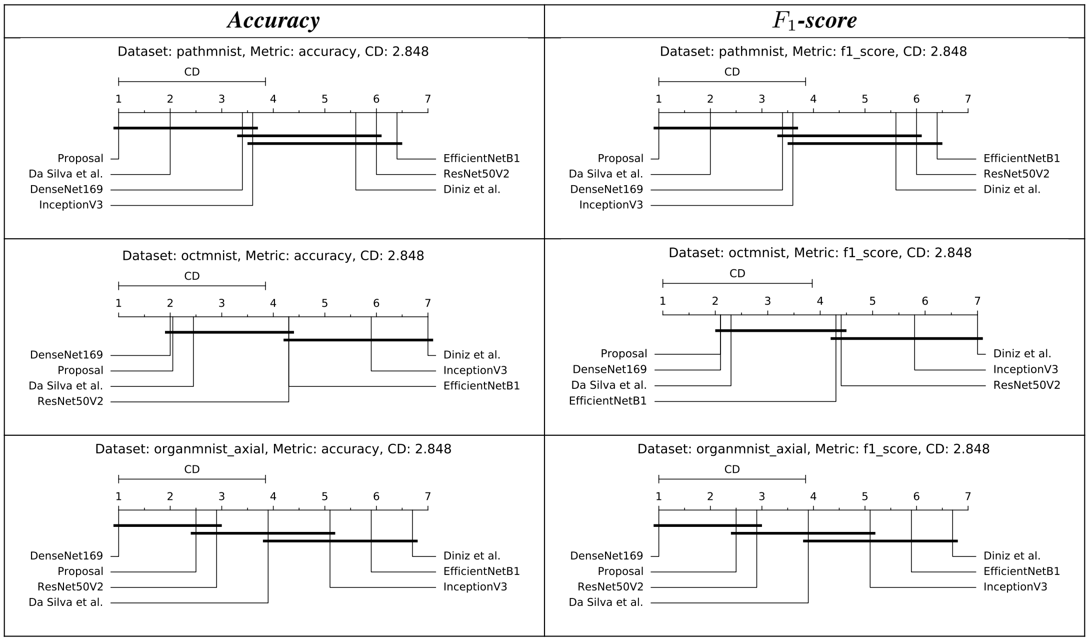

Fig. 4. Critical Difference diagrams for Friedman-Nemenyi statistical test.

[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. 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.

[10] R. H. R. de Lima, A. Pozo, and R. Santana, "Automatic design of convolutional neural networks using grammatical evolution," in *2019 8th Brazilian Conference on Intelligent Systems (BRACIS)*. IEEE, 2019, pp. 329–334.

[11] K. Soltanian, F. A. Tab, F. A. Zar, and I. Tsoulos, "Artificial neural networks generation using grammatical evolution," in *2013 21st Iranian Conference on Electrical Engineering (ICEE)*. IEEE, 2013, pp. 1–5.

[12] C. A. da Silva, D. C. Rosa, P. B. Miranda, F. R. Cordeiro, T. Si, A. C. Nascimento, R. F. Mello, and P. S. de Mattos Neto, "A multi-objective grammatical evolution framework to generate convolutional neural network architectures," in *2021 IEEE Congress on Evolutionary Computation (CEC)*. IEEE, 2021, pp. 2187–2194.

[13] 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.

[14] J. Yang, R. Shi, and B. Ni, "Medmnist classification decathlon: A lightweight automl benchmark for medical image analysis," 2021.

[15] M. O'Neill and C. Ryan, "Grammatical evolution," *IEEE Transactions on Evolutionary Computation*, vol. 5, no. 4, pp. 349–358, 2001.

[16] J. Koza, J. Koza, and J. Rice, *Genetic Programming: On the Programming of Computers by Means of Natural Selection*, ser. A Bradford book. Bradford, 1992. [Online]. Available: https://books.google.com.br/books?id=Bhtxo60BV0EC

[17] C. Ryan, J. J. Collins, and M. O. Neill, "Grammatical evolution: Evolving programs for an arbitrary language," in *European Conference on Genetic Programming*. Springer, 1998, pp. 83–96.

[18] G. Pappa and A. Freitas, "Evolving rule induction algorithms with multi-objective grammar-based genetic programming," *Knowl. Inf. Syst.*, vol. 19, pp. 283–309, 06 2009.

[19] 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.

[20] I. Tsoulos, D. Gavrilis, and E. Glavas, "Neural network construction and training using grammatical evolution," *Neurocomputing*, vol. 72, no. 1-3, pp. 269–277, 2008.

[21] 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.

[22] F. Assunção, N. Lourenço, P. Machado, and B. Ribeiro, "Denser: deep evolutionary network structured representation," *Genetic Programming and Evolvable Machines*, vol. 20, no. 1, pp. 5–35, 2019.

[23] 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.

[24] 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.

[25] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," 2017.

[26] Z. Zhang and M. R. Sabuncu, "Generalized cross entropy loss for training deep neural networks with noisy labels," 2018.

[27] 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.

[28] M. Fenton, J. McDermott, D. Fagan, S. Forstenlechner, E. Hemberg, and M. O'Neill, "Ponyge2: Grammatical evolution in python," in *Proceedings of the Genetic and Evolutionary Computation Conference Companion*. ACM, 2017, pp. 1194–1201.

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

[30] 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.

[31] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, "Rethinking the inception architecture for computer vision," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 2818–2826.

[32] K. He, X. Zhang, S. Ren, and J. Sun, "Identity mappings in deep residual networks," in *European conference on computer vision*. Springer, 2016, pp. 630–645.

[33] M. Tan and Q. Le, "Efficientnet: Rethinking model scaling for convolutional neural networks," in *International Conference on Machine Learning*. PMLR, 2019, pp. 6105–6114.

[34] G. Huang, Z. Liu, L. Van Der Maaten, and K. Q. Weinberger, "Densely connected convolutional networks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 4700–4708.