Assessing Multi-Objective Search Engines for GE: A Case Study in CNN Generation

Amerson Chagas[1], **Daniel Rosa**[2], **Cleber Silva**[2],
Tapas Si[3], **Péricles B.C. Miranda**[2]

[1] Centro de Estudos e Sistemas Avançados do Recife (CESAR), PE

[2]Departamento de Computação
Universidade Federal Rural de Pernambuco

[3]Department of Computer Science & Engineering
University of Engineering & Management, Jaipur, India

{pericles.miranda}@ufrpe.br

Abstract. *In recent years, the number of available Convolutional Neural Networks (CNNs) has increased significantly, making it difficult to select an appropriate CNN for a specific problem. To address this challenge, researchers have proposed automated techniques for optimizing CNN architectures, with Grammatical Evolution (GE) being one of the most promising approaches. GE uses context-free grammar to generate programs (e.g., CNNs) and a search engine to find the best solutions. Although several grammars have been proposed for CNN generation, there has been no research evaluating the impact of different search engines in the GE optimization process. This study treats the CNN generation as a multi-objective problem by optimizing accuracy and F1-score, and evaluates seven different multi-objective optimizers listed in the literature as potential search engines. The goal is to investigate the strengths and weaknesses of each optimizer in CNN generation. The experiments were performed on the widely-used CIFAR-10 image classification dataset, and the results showed that selecting the right optimizer for the task is crucial and can have a significant impact on the final result, especially when the number of generations is limited.*

1. Introduction

The task of choosing suitable Convolutional Neural Networks (CNNs) and their parameters for a given classification problem is not trivial due to the great variety of algorithms and number of configurable parameters [Diniz et al. 2018]. Thus, many works have proposed solutions to facilitate or automate the selection/generation of CNN architectures to help experts make decisions. Recently, some works [Assunção et al. 2018, Diniz et al. 2018,de Lima et al. 2019,Neto et al. 2020,da Silva et al. 2021b,da Silva et al. 2021a,Lima et al. 2022, da Silva et al. 2023] have employed Grammatical Evolution (GE) [O'Neill and Ryan 2001] for the generation of CNN architectures. The CNNs produced by GE frameworks have reached promising results, overcoming state-of-the-art CNNs in some image classification problems.

GE is a method that produces programs adopting a Context-Free Grammar (CFG) and uses a search engine to seek promising ones. The CFG is composed of building rules that may include information about the problem and guide the creation of solutions

[O'Neill and Ryan 2004, Mariani et al. 2016]. Thus, the elaboration of inconsistent and ambiguous grammar negatively impacts GE's performance. Another relevant GE component is the search engine. The search engine implements an optimization algorithm (optimizer) to seek the optimal or near-optimal solutions produced by the grammar.

There is a relevant number of works found in the literature proposing novel grammars for CNN generation and comparing their performance in different image classification problems [Assunção et al. 2018, Diniz et al. 2018, de Lima et al. 2019, da Silva et al. 2021b, da Silva et al. 2021a, Lima and Pozo 2019, Lima et al. 2022, da Silva et al. 2023]. Nonetheless, according to our knowledge, there is no work investigating different optimizers and their impact on GE's performance in the problem at hand.

This work aims to experimentally investigate the impact that the choice of the search engine's optimizer has on GE's performance for the generation of CNN architectures. Thus, we propose a suitable and fair experimental methodology sustained by statistical analyses. As our intention is to evaluate different optimizers as search engine, we chose a single grammar (proposed by [da Silva et al. 2023]) from literature used for CNN generation. Herein, we treat the CNN generation as a multi-objective problem considering accuracy and F_1-score as objective functions. In the experiments, a total of seven different multi-objective optimizers listed in the literature were considered in the search engine role: Non-dominated Sorting Genetic Algorithm II (NSGA-II), Strength Pareto Evolutionary Approach 2 (SPEA2), Pareto Envelope-based Selection Algorithm II (PESA2), Decomposition-Based Evolutionary Algorithm (DBEA), S-metric Selection Multi-Objective Evolutionary Algorithm (SMSMOEA), Epsilon Multi-Objective Evolutionary Algorithm (eMOEA) and Pareto Archived Evolution Strategy (PAES). This research has two contributions: 1) We executed the GE using seven multi-objective optimizers to seek CNN solutions, and we evaluated their performances varying the number generations; 2) The performance of the CNNs found by the optimizers were assessed in terms of accuracy and F_1-score acquired when tested in three datasets: CIFAR-10, MNIST, and EusoSAT. The experimental results showed that the difference in results is noticeable, from one search engine to another and from one dataset to another. It is clear that the choice of the search engine is a very important question to consider before starting any evolutionary process. The algorithms PESA2, NSGA-II, SPEA2, and DBEA presented the best performances in all three datasets. On the other hand, the PAES, SMSMOEA, and eMOEA were not able to contribute to GE's performance.

This paper is organized in: Section 2 introduces relevant concepts for the solution understanding. Section 3 presents related works. Section 4 details this work's experimental methodology. Section 5 presents the results obtained. Finally, Section 6 gives the conclusions and some directions for future research.

2. Background

This section introduces some basic concepts about Convolutional Neural Networks, Grammatical Evolution, and Multi-Objective Optimization to help the understanding of this paper's main subjects.

2.1. Convolutional Neural Networks

A Convolutional Neural Network (CNN) is a widely used type of neural network in image processing, particularly for image classification and object detection. The network is

comprised of layers that extract features from images in different ways, and the order and configuration of these layers can affect the network's performance. The most common layers used in CNNs are convolutional, pooling, and fully connected layers. Convolutional layers filter input to highlight key features, while pooling layers reduce input size to decrease the number of trainable parameters in the network. Fully connected layers classify data extracted by previous layers through a sequence of connected neurons, and can be improved with the addition of batch normalization and dropout layers to normalize information and prevent overfitting. The topology of a CNN is formed by these layers, with output from one layer serving as input to the next until the network produces its final output.

2.2. Grammatical Evolution

Grammatical Evolution (GE) [O'Neill and Ryan 2001] is a technique that evolved from Genetic Programming [Koza et al. 1992] and utilizes Context-Free Grammars to generate programs of various sizes, including Convolutional Neural Networks [Ryan et al. 1998]. The use of grammar in the creation of individuals provides GE with an advantage over other techniques since it allows prior knowledge of the domain of the problem to be embedded in the grammar structure. This helps avoid errors in the creation of individuals and ensures that the search space is comprised of valid individuals.

A GE algorithm comprises three main components: the search engine, grammar, and a process of mapping individuals. The grammar defines the rules for creating individuals and specifies the search space for the problem. The search engine navigates this search space by implementing an evolutionary algorithm that seeks promising areas in the grammar-generated search space. Finally, the mapping process converts the individuals generated by the grammar into executable programs. Each individual contains its genetic information, or genotype, which is usually structured in a numerical array. The mapping process reads the genotype through the rules defined in the grammar, constructing an individual from the information.

The way how GE performs the optimization is discussed as follows. The first step involves defining a BNF grammar, and then creating a group of individuals or genotypes as the initial population for an evolutionary algorithm, which are chosen randomly. A mapping process is used to convert each individual from its genotype to phenotype. These initial procedures are completed before the framework's iterative process starts (search engine). After evaluating all individuals, genetic operators are applied to the best parents, resulting in the generation of new individuals in genotype format. The new individuals are mapped to phenotype, and their fitness values are recorded. Only the most exceptional individuals are more likely to be chosen for the subsequent iteration. The algorithm finishes when the stop criterion is met, and the best solution (CNN) is returned.

2.3. Multi-Objective Optimization

Multi-Objective Optimization (MOO) Algorithms, unlike mono-objective algorithms, are those that seek to optimize more than one objective function of a problem at the same time. Formally, they are defined by a set of objective functions $O = \{o_1, o_2, \ldots, o_m\}$, where m represents the number of objectives defined in the problem. This way, in a problem, a solution $\vec{d}$ that needs to be analyzed, represented by the vector $\vec{d} = (d_1, d_2, \ldots, d_n)$, where n is the number of decision variables of the solution, must be passed as input to each

objective function present in the problem, thus forming the vector $\vec{a}$ corresponding to the individual's fitness, formally represented by $\vec{a} = (a_1, a_2, \ldots, a_m)$, where $a_i (i \in [1, m])$.

Because MOOs have m objective functions, and, consequently, their fitness value is composed of a vector of the same size, we can not compare their candidate solutions only by a single value, as it is done in single-objective optimization. In this way, the candidate solutions of the problem are compared through Pareto dominance [Deb et al. 2002], which individually evaluates each element of the individuals' fitness vector. Pareto dominance defines that given two candidate solutions $\vec{a}$ and $\vec{b}$, $\vec{a}$ dominates over $\vec{b}$ if $\vec{a}$ beats $\vec{b}$ in at least one objective and $\vec{a}$ is no worse than $\vec{b}$ in any other objective. That is, in at least one of the objectives, inequality is strictly met. This dominance relation is formally written as $\vec{a} \prec \vec{b}$. Finally, the set of all non-dominated solutions of a given proposition forms what we call the Pareto front.

3. Related Work

This section lists several approaches found in the literature that use the principles of Grammatical Evolution to optimize CNNs. However, we will present which search engines were used in each approach, their objective functions, and other specificities.

Assuncção et al. [Assunção et al. 2018] proposed a mechanism for creating and optimizing deep CNNs through Neuro-Evolution principles called DENSER (*Deep Evolutionary StructurEd Representation*). The approach created and optimized deep CNNs through a robust context-free grammar linked to a basic genetic algorithm to solve image classification problems. CNNs were optimized, seeking to maximize only one objective function, the accuracy, thus being a mono-objective approach. Even being a mono-objective approach, the results showed that the technique used is promising, generating optimized CNN architectures that compete or surpass other state-of-the-art architectures. The datasets used in this approach were CIFAR-10, CIFAR-100, MNIST, and FashionMNIST.

De Lima et al. [de Lima et al. 2019] also used the principles of grammatical evolution to optimize CNNs in image classification problems. Like the previous approach, the search engine was configured with a basic genetic algorithm, navigating the search space by maximizing accuracy in a mono-objective approach. The grammar associated with the study has a recursive structure, allowing an infinite search space. The experiments were performed on the CIFAR-10 and MNIST datasets. The results showed that the technique could generate very competitive optimized models of CNNs, even requiring a greater number of generations of the evolutionary process.

On the other hand, Diniz et al. [Diniz et al. 2018] proposed an approach similar to the previous works. However, using a search engine equipped with another genetic algorithm, the NSGA-II [Deb et al. 2002]. This genetic search algorithm uses an elitist strategy to classify candidate solutions to a problem, further optimizing the process. The proposed approach used a very simple context-free grammar able to generate until 54 different types of CNNs architectures. Similar to previous approaches, the search engine was configured to optimize CNNs by maximizing (mono-objective) accuracy in image classification problems in the CIFAR-10 dataset. The results showed that the approach is capable of generating very simple and lightweight optimized architectures compared to other more complex state-of-the-art architectures.

Inspired by the work of [Diniz et al. 2018], da Silva et al. [da Silva et al. 2021b] proposed a new approach to optimizing CNNs, implementing improvements in a new grammar based on previous work. Furthermore, the search engine, still being NSGA-II, has now been configured for multi-objective optimization of CNNs through the maximization of two objective functions: Accuracy and F_1-score. The new grammar now allowed the creation of 2592 different CNN architectures. The improvements applied showed through the results that the approach is quite promising, generating optimized CNNs architectures that compete or even surpass state-of-the-art CNNs models and other approaches that also use the technique of grammatical evolution. The dataset used in the experiments was the CIFAR-10.

Da Silva et al. [da Silva et al. 2021a], identifying possible improvements in the grammar of the previous work, proposed a new, more specialized grammar for creating optimized CNN architectures. The approach's search engine continued to be the NSGA-II, navigating the search space through the same objective functions: Accuracy and F_1-score. Unlike previous work, tests were performed on three datasets (PathMNIST, OCTMNIST, and OrganMNIST Axial) from a new collection of medical images called MedMNIST. The results showed that the new approach was able to generate more specialized individuals than the previous approach, producing competitive models that compete or even surpass state-of-the-art architectures beyond the previous approaches.

Silva et al. [da Silva et al. 2023] expanded on the research presented by Silva et al. [da Silva et al. 2021b] by introducing a new grammar that has a more extensive search space of 6426 potential individuals, enabling greater flexibility in arranging, organizing, and deepening layers while still supporting the same layers as its predecessor. Most layer parameters are still set externally, and a multi-objective optimization approach guides the search space navigation process. The study aimed to tackle classification problems across four datasets: MNIST, KMNIST, CIFAR-10, and EuroSAT. The outcomes demonstrated that the novel grammar produced competitive individuals, and in some cases, outperformed state-of-the-art networks and the reference grammar regarding image classification problem.

As can be seen, the literature provides some works that addressed the problem of optimization of CNNs through Grammatical Evolution, focusing mainly on two types of genetic algorithms: basic genetic algorithm and NSGA-II. One thing in common to all works is that only one search engine was used in the experiments, per research. While [Assunção et al. 2018] and [Diniz et al. 2018] used GA, the others, [de Lima et al. 2019], [da Silva et al. 2021b] and [da Silva et al. 2021a] used NSGA-II. Since there is not much variation in the use of different search engines, this work's main goal is to identify how the use of other evolutionary algorithms, in addition to those mentioned, can influence the production of optimized models of CNNs.

4. Experimental Methodology

This section describes the performed experiments, performed metrics used to evaluate the evolutionary algorithms, and the adopted parameters.

4.1. Search Engines

Search engines are utilized to explore potentially promising solutions. In GE, a search engine is responsible for identifying promising solutions within a problem's search space

in an iterative manner. As discussed in Section 3, genetic algorithms were the most common optimization algorithms used as the search engine. Genetic algorithms bring in their heuristic operations of crossover, mutation, and selection, which seek to simulate the behavior of Darwin's theory in an optimization context, where a population of solutions evolves towards an optimally evolved individual [Deb et al. 2002]. Next, we introduce some well-known multi-objective optimization algorithms adopted in this experiment. They were selected for in this experiments because showed high potential in different optimization tasks.

Non-dominated Sorting Genetic Algorithm II (NSGA-II) The NSGA-II [Deb et al. 2002] is a multi-objective algorithm that had its origin from the NSGA algorithm. Compared to its previous version, it has some additions: Improvement of the sorting process of non-dominated solutions, the addition of elitism, and a greater capacity to expand the variety of individuals. It uses a Fast Non-Dominated Sorting method that receives as input a set of individuals and then returns a set of non-dominated individuals.

Strength Pareto Evolutionary Approach 2 (SPEA2) The SPEA2 [Zitzler et al. 2001] is an extended version of SPEA, that is is an evolutionary algorithm for multi-objective optimization problems. SPEA2 uses a neighborhood density estimation technique incorporated into the individuals' fitness function. This algorithm uses a k-Nearest Neighbor (kNN) like mechanism and a specialized ranking system to sort the members of the population. It selects the next generation of population from the combination of current population and off-springs created by genetic operators (mutation and crossover).

Pareto Envelope-based Selection Algorithm II (PESA2) PESA2 [Corne et al. 2001] is an evolutionary multi-objective optimization algorithm that has been widely applied in many fields. One feature of PESA-II is its grid-based fitness assignment strategy in environmental selection. It uses grids to make selections and create the next generation. PESA-II uses the mechanism of genetic algorithm, with a selection based on the Pareto envelope.

Decomposition-Based Evolutionary Algorithm (DBEA) The decomposition has been the mainstream approach in classic mathematical programming for multi-objective optimization and multi-criterion decision-making. DBEA's algorithm [Li 2021] is based on this mathematical model of decomposition. That is, the population size μ is dynamically changed during the search process. Multiple individuals far from each other in the solution space can be assigned to the same decomposed single-objective sub-problem. Only similar individuals in the solution space are compared based on their scalarized function values making a dedicated weight vector.

S-metric Selection Multi-Objective Evolutionary Algorithm (SMSMOEA) SMSMOEA is an evolutionary multi-objective algorithm that uses the hypervolume measure within its selection operator [Naujoks et al. 2005]. The hypervolume measure, or S-metric, is a distinguished quality measure for solution sets in Pareto optimization. Once the aim to reach a high S-metric value is appointed, it seems promising to incorporate it in the optimization algorithm directly. Solutions are rated according to their contribution to the dominated hypervolume of the current population.

Epsilon Multi-Objective Evolutionary Algorithm (eMOEA) eMOEA is a steady-state algorithm [Liu et al. 2007], meaning only one individual in the population is evolved per

step. This evolution is based on Pareto-dominance Relationship Fail to find the true Pareto fronts and uses a Pareto dominance archive to maintain a well-spread set of Pareto-optimal solutions. The concept is used to improve further both properties of the non-premature convergence towards Pareto-optimal sets and the diversity among the found solutions.

Pareto Archived Evolution Strategy (PAES) The PAES algorithm [Knowles and Corne 1999] was created to solve routing problems in the telecommunication industry. It got good performance results and became an important algorithm in the area. Although it does not have the selection step, the binary tournament, and the crossover operator, it became a fast option since it does not implement those steps to the evolutionary process, leading to faster convergence.

4.2. Metrics

In order to be able to evaluate the produced CNN models, two metrics were used: the accuracy and F_1-score. The accuracy represents the percentage of the properly classified samples in the face of the whole database. The formula below shows us how to calculate its value. TP represents the true positives, TN the true negatives, FP false positives, and FN false negatives.

$$Accuracy = \frac{TP + TN}{TP + FP + FN + TN}, \tag{1}$$

This metric is a good one to evaluate those datasets that contain a balanced set of images. That is, all classes have the same number of samples, or at least almost the same. Two examples of balanced datasets are CIFAR-10 and MNIST. However, in many real cases, it will be necessary to use unbalanced datasets. Given an unbalanced dataset, for example, with 90% of the samples belonging to a single class, and only 10% of the rest, if the network throws the same prediction according to the majority class, without giving any importance to the others, it would still have around 90% of stated precision, when in fact, this is not the correct way of classifying. For this reason, another metric was adopted to measure the cases where the datasets are unbalanced: the F_1-score metric.

The F_1-score represents the harmonic mean between recall and precision. The recall is calculated by dividing the number of true positives by the sum of true positives with true negatives. On the other hand, the precision is calculated by dividing the true positives by the sum of the true positives with the false positives, as can be seen in equation 2. After calculating both recall and precision, we can calculate the F_1-score value following equation 3.

$$\text{Recall (R)} = \frac{TP}{TP + TN} \qquad \text{Precision (P)} = \frac{TP}{TP + FP} \tag{2}$$

$$F_1\text{-score} = \frac{2 \times R \times P}{R + P}. \tag{3}$$

4.3. Datasets

In order to be able to measure the effectiveness of the used search engines, it is necessary to train each CNN in a dataset to get its' results. Besides, to have a more accurate analysis,

we decided to use 3 datasets, two balanced ones and an unbalanced one, intending to analyze how the evolutionary process will behave in situations from different domains.

CIFAR-10: It is a popular dataset used in many experiments involving classification issues. It is composed of 60000 images divided into 10 classes, where each class has 6000 samples. The whole dataset is divided into two subsets: train and test, composed of 50000 and 10000 images, respectively. All samples consist of 32×32 dimension images.

MNIST: It is also a widely used dataset for classification purposes. It has 70000 no-colored samples of 28×28 dimension. Like CIFAR-10, it is divided into 10 classes, where each class represents a handwritten version of each digit from the Hindu-Arabic number system. The dataset is also divided into 60000 training set images and 10000 testing set ones.

EuroSAT: It is a dataset composed of 27000 satellite images, divided into 10 classes that represent distinct geographic locations. All samples consist of 64×64 dimension images. Unlike CIFAR-10 and MNIST, EuroSAT does not divide the dataset into subsets. To perform our experiments, we divided it into three subsets: The training set, with 80% of the images, the validation set, with 20%, and the testing set, with 20% of the validation set's samples.

4.4. Adopted Grammar

To perform the experiments using grammatical evolution, A grammar needs to be given to lead the evolutionary process. In this paper's experiments, we intended to use a simple grammar that could generate competitive CNNs, compared to other state-of-the-art automatic CNN generation approaches. For this reason, we used the grammatical rules proposed in [da Silva et al. 2023] to generate the CNNs.

Figure 1 shows the grammar, in Backus-Naur (BNF) format, proposed for the creation of CNNs. It consists of a total of 11 tags with specific functions described below.

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

Figure 1. Adopted BNF context-free grammar for CNN generation.

The main tag, ⟨CNN⟩, defines the entire structure of a CNN. It contains convolutional blocks, a data flattening layer, fully connected layers, a dropout layer, and the learning rate used during network training. The tag ⟨BLOCK⟩ represents the convolutional blocks of the network. Its structure is defined by the expression (⟨CONV⟩⟨POOL⟩) $*$ ⟨M⟩, which means that ⟨CONV⟩ and ⟨POOL⟩ layers can be appended to the network structure ⟨M⟩ times. The value of ⟨M⟩ ranges from 1 to 3. The tag ⟨CONV⟩ represents the structure of a convolutional layer, which is defined by the expression `(conv.`⟨BNORM⟩) $*$ ⟨Z⟩, indicating that a convolutional layer followed by batch normalization can repeat ⟨Z⟩ times, ranging from 1 to 3. The ⟨BNORM⟩ tag represents the use of batch normalization and can be present (λ) or absent. The tag ⟨POOL⟩ represents the structure of a network pooling layer and is defined by the expression `pool.`⟨DROPOUT⟩ | λ, indicating that pooling layers may or may not exist and if they do, they can be followed by a dropout layer. The tag ⟨FC⟩ represents the fully connected layers and can be added ⟨K⟩ times to the network, according to the expression `(fc`⟨UNITS⟩) $*$ ⟨K⟩. The tag ⟨UNITS⟩ indicates the number of neurons in these layers and can take four different values. The ⟨BNORM⟩ and ⟨DROPOUT⟩ tags may or may not exist, as indicated by the symbol λ present in the grammar. Finally, the ⟨LR⟩ tag defines four options of learning rates for network training.

The proposed grammar allows the creation of flexible CNN architectures, from simple ones with only convolutional layers to more complex ones with convolutional layers, pooling, batch normalization, dropout, and fully connected layers. The ⟨K⟩, ⟨Z⟩, and ⟨M⟩ tags can be modified to generate deeper and more complex networks. It's worth mentioning that the grammar generates only sequential layer architectures that are formed by structured layers in the form of stacks. Some important parameters, such as activation function, padding, stride, and the number of convolutional filters, remained fixed in the construction of the CNNs. The activation function used was ReLU, and the pooling layers were set to the maximum type. The optimizer used was Adam, and the loss function was the Categorical Cross-Entropy. The CNN models were trained for 70 epochs with a batch size of 128. More information about these parameters can be found in Table 1.

This grammar' search space is limited because there are limited possibilities to generate the individuals. However, it is not necessary to train every possible individual of the grammar, but only those generated by the evolutionary process. Some architectures can appear more than once in the middle of the evolutionary process. When it happens, an API is used to help avoid duplicate training processes by returning the results of its metrics that were already stored to the algorithm.

4.5. Experimental Configuration

Two applications needed to communicate through an API to run the experiments: The one responsible for performing the training process and the other responsible for storing the obtained results. The MOEA Framework[1] was used to execute the search algorithms with the grammar mentioned previously. MOEA Framework is a free and open-source Java library for developing and experimenting with multiobjective evolutionary algorithms (MOEAs) and other general-purpose single and multi-objective optimization algorithms.

Once an individual is generated, it is dispatched to an API for training and storage

[1] http://moeaframework.org/

Table 1. CNN fixed parameters.

Parameters	*Values*
Number of epochs	70
Batch size	128
Number of convolutional filters	Starts with 32; duplicates for every two convolutions.
Activation function	ReLU for convolutions; Softmax for the last fully-connected layer.
Stride size	1
Kernel size	3 x 3
Pooling size	2 x 2
Dropout rate	0.25 after pooling layers; 0.50 after fully connected layers.
Optimizer	Adam Optimizer
Loss function	Categorical Cross-Entropy
Early stopping criteria	monitor=val_accuracy, mode=max patience=10, baseline=0.5

in a database. On the other side of the API, an algorithm coded in Python using Tensorflow awaits the arrival of individuals for the training procedure. Upon training completion, the program provides the accuracy and F1-score metrics to the API for database updating. The algorithms ran with a population size of 50 individuals and 30 generations. The mutation rate was fixed at 5%, and the crossover rate was maintained at 75% across all algorithms. It is worth mentioning that the default parameters of all multi-objective algorithms were adopted, which can be found in the MOEA framework. Ten simulations were performed for each algorithm to yield the average and standard deviation. Using the Borda count method [Orouskhani et al. 2017], a single CNN solution was selected from each Pareto produced by each method, considering accuracy and F1-score. The Borda count method ranks the CNNs generated by each objective function and computes the average rank of solutions. The CNN ranked first in the average rank is then selected. This methodology is commonly used in multi-objective method experiments to compare CNN results found by each competing method regarding classification performance [Orouskhani et al. 2017].

5. Results and Discussion

Tables 2, 3 and 4 shows the generated CNNs' average and standard deviation considering each adopted multi-objective algorithm as search engine. Thus, the result reached by the GE using NSGA-II as a search engine is named with the search engine's algorithm (NSGA-II). The same was adopted for the other algorithms. To make a fair comparison, statistical tests were performed to measure the difference between the used approaches. The null hypothesis was defined as follows: the means of the results of one or more algorithms are the same. Based on the available groups and the number of samples per group, the Friedman Aligned-Ranks parametric test was performed. To complement the statistical analysis, the Nemenyi's posthoc test was used to identify which approaches are statistically similar. Figure 2 gives the comparison results (accuracy and F_1-score), using a graph representation, which classifies the approaches and groups those who are statisti-

cally similar (considering the value of CD). This representation is called CD diagram.

Table 2. Comparative performance analysis between the search engines for CIFAR-10 dataset.

Algorithm	Accuracy	F_1-score
PESA2	**0.8639** ($\pm$0.0001)	0.8658 ($\pm$0.0006)
SPEA2	0.8638 ($\pm$0.0003)	**0.8659** ($\pm$0.0004)
NSGAII	0.8636 ($\pm$0.0004)	0.8654 ($\pm$0.0008)
DBEA	0.8620 ($\pm$0.0012)	0.8634 ($\pm$0.0011)
eMOEA	0.8496 ($\pm$0.0112)	0.8508 ($\pm$0.0112)
SMSEMOA	0.8451 ($\pm$0.0115)	0.8465 ($\pm$0.0113)
PAES	0.6743 ($\pm$0.1700)	0.6539 ($\pm$0.2194)

Table 3. Comparative performance analysis between the search engines for EuroSAT dataset.

Algorithm	Accuracy	F_1-score
NSGAII	**0.9549** ($\pm$0.0012)	**0.9553** ($\pm$0.0011)
SPEA2	0.9545 ($\pm$0.0028)	0.9550 ($\pm$0.0028)
PESA2	0.9523 ($\pm$0.0048)	0.9526 ($\pm$0.0048)
DBEA	0.9440 ($\pm$0.0069)	0.9444 ($\pm$0.0068)
eMOEA	0.9296 ($\pm$0.0190)	0.9299 ($\pm$0.0190)
SMSEMOA	0.9248 ($\pm$0.0122)	0.9257 ($\pm$0.0123)
PAES	0.7660 ($\pm$0.1879)	0.7648 ($\pm$0.1922)

Table 4. Comparative performance analysis between the search engines for MNIST dataset.

Algorithm	Accuracy	F_1-score
PESA2	**0.9965** ($\pm$0.0001)	**0.9966** ($\pm$0.0002)
SPEA2	**0.9965** ($\pm$0.0001)	0.9965 ($\pm$0.0002)
NSGAII	0.9964 ($\pm$0.0001)	0.9965 ($\pm$0.0002)
DBEA	0.9963 ($\pm$0.0002)	0.9963 ($\pm$0.0001)
eMOEA	0.9956 ($\pm$0.0007)	0.9957 ($\pm$0.0007)
SMSEMOA	0.9955 ($\pm$0.0007)	0.9955 ($\pm$0.0006)
PAES	0.9903 ($\pm$0.0031)	0.9904 ($\pm$0.0030)

Analyzing the results on CIFAR-10 dataset, presented in Table II, the PESA2 algorithm obtained the best means in terms of accuracy, followed by SPEA2 and NSGA-II. However, in terms of F_1-score, the SPEA2 algorithm obtained better results. The PAES algorithm obtained the worst results, both on accuracy and F_1-scores. The PESA2 got the lowest standard deviation value on accuracy, while SPEA2 got it on the F_1-score. On the other hand, PAES obtained the highest standard deviation values in both metrics. Both the null hypothesis of accuracy and F_1-score were rejected, with a p-value of $2.02e-10$ for Accuracy and $5.27e-10$ for F_1-score. From the result presented in Figure 2, it is possible to affirm that there is statistical evidence that the result for both the accuracy and F_1-score metrics are statistically similar between algorithms: PESA2, SPEA2, NSGAII and DBEA, and that PESA2 and SPEA2 overcome all other approaches.

Looking at the EuroSAT dataset's results, Table 3 shows that NSGA-II got the best values on both accuracy and F_1-score, followed by SPEA2 and PESA2, while the PAES algorithm got the worst results in both metrics. The lowest standard deviation value in both metrics came from NSGA-II, while PAES got the highest ones. Both the null hypothesis of accuracy and F_1-score were rejected, with a p-value of $4.65e-09$ for accuracy and $4.27e-0$ for F_1-score. From the result presented in Figure 2, it is possible to say that there is statistical evidence that the result for both the accuracy and F_1-score metrics are statistically equivalent among algorithms NSGA-II SPEA2, PESA2, and DBEA. The algorithms PAES, SMSMOEA and eMOEA remained in the last positions being overcome by the first-placed methods.

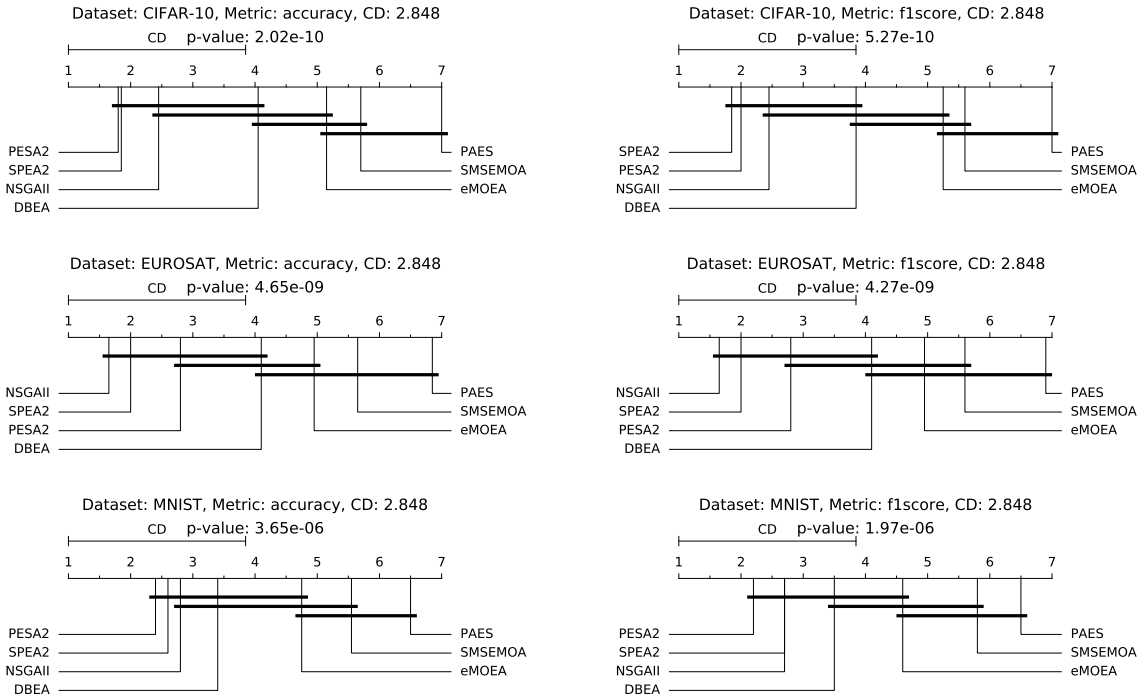

Figure 2. Critical Difference diagrams for Friedman-Nemenyi statistical test.

Finally, looking at the MNIST results in Table IV, the PESA2 and the SPEA2 obtained the best means in terms of accuracy, followed by NSGAII and DBEA. The PESA2 algorithm stood out from the others, followed by SPEA and NSGAII. The worst results in both metrics were obtained by PAES algorithm. The PESA2, SPEA2, and NSGAII algorithms got the lowest standard deviation values in accuracy, while DEBA obtained the lowest one in F_1-score, followed by PESA2, SPEA2, and NSGAII. The PEAS algorithm obtained the highest standard deviation value in both metrics. Both null hypotheses, from accuracy and F_1-score, were rejected, with a p-value of $3.65e-06$ for accuracy and $1.97e-06$ to F_1-score. From the results presented in Figure 2, it is possible to affirm that there is statistical evidence that the performances, in both accuracy and F_1-score metrics, are statistically equivalent between the PESA2, SPEA2, NSGA-II, and DBEA algorithms.

Based on the proposed analysis, it is possible to notice that the choice of a search engine can make a difference in the results obtained by GE for the CNN generation. Different engines may also have different results depending on the dataset used, which

reinforces the necessity to have more related works studying search engine's impact on generating neural networks. In general, the algorithms PESA2, SPEA2, NSGA-II, and DBEA performed well, contributing to GE's performance in all three datasets. On the other hand, the algorithms eMOEA, SMSMOEA, and PAES have not positively impacted GE's performance.

6. Conclusion and Future Works

Grammatical Evolution (GE) has been utilized to generate Convolutional Neural Network (CNN) architectures, with previous research focused on proposing new grammars with different building rules but limited experimentation with a single search engine. Our study aims to demonstrate the importance of the search engine algorithm by using different algorithms to evaluate GE's performance. We investigated the impact of search engine choice by adopting seven multi-objective optimization algorithms to build CNNs, using two objective functions (accuracy and F_1-score) across three distinct datasets. Results showed significant variation in performance across search engines and datasets, highlighting the importance of careful selection. PESA2, NSGA-II, SPEA2, and DBEA delivered the best performances across all datasets, while PAES, SMSMOEA, and eMOEA were not effective. As future work, we plan to adopt many-objective search engines, consider additional datasets to validate the results, and explore optimization algorithms with alternative inspirations such as Particle Swarm Optimization.

References

Assunção, F., Lourenço, N., Machado, P., and Ribeiro, B. (2018). Evolving the topology of large scale deep neural networks. In *European Conference on Genetic Programming*, pages 19–34.

Corne, D. W., Jerram, N. R., Knowles, J. D., and Oates, M. J. (2001). Pesa-ii: Region-based selection in evolutionary multiobjective optimization. In *Proceedings of the 3rd Annual Conference on Genetic and Evolutionary Computation*, GECCO'01, page 283–290, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc.

da Silva, C. A., Miranda, P. B., and Cordeiro, F. R. (2021a). A new grammar for creating convolutional neural networks applied to medical image classification. In *SIBGRAPI 2021 - 34th Conference on Graphics, Patterns and Images*.

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

da Silva, C. A., Rosa, D. C., Miranda, P. B., Cordeiro, F. R., Si, T., Nascimento, A. C., Mello, R. F., and de Mattos Neto, P. S. (2023). A novel multi-objective grammar-based framework for the generation of convolutional neural networks. *Expert Systems with Applications*, 212:118670.

de Lima, R. H. R., Pozo, A., and Santana, R. (2019). Automatic design of convolutional neural networks using grammatical evolution. In *Brazilian Conference on Intelligent Systems (BRACIS)*, pages 329–334.

Deb, K., Pratap, A., Agarwal, S., and Meyarivan, T. (2002). A fast and elitist multiobjective genetic algorithm: Nsga-ii. *IEEE transactions on evolutionary computation*, 6(2):182–197.

Diniz, J. B., Cordeiro, F. R., Miranda, P. B., and da Silva, L. A. T. (2018). A grammar-based genetic programming approach to optimize convolutional neural network architectures. In *Encontro Nacional de Inteligência Artificial e Computacional*, pages 82–93.

Knowles, J. and Corne, D. (1999). The pareto archived evolution strategy: A new baseline algorithm for pareto multiobjective optimisation. volume 1.

Koza, J., Koza, J., and Rice, J. (1992). *Genetic Programming: On the Programming of Computers by Means of Natural Selection*. A Bradford book. Bradford.

Li, K. (2021). Decomposition multi-objective evolutionary optimization: From state-of-the-art to future opportunities.

Lima, R. H., Magalhães, D., Pozo, A., Mendiburu, A., and Santana, R. (2022). A grammar-based gp approach applied to the design of deep neural networks. *Genetic Programming and Evolvable Machines*, 23(3):427–452.

Lima, R. H. and Pozo, A. T. (2019). Evolving convolutional neural networks through grammatical evolution. In *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO 2019)*, pages 179–180.

Liu, L., Li, M., and Lin, D. (2007). A novel epsilon-dominance multi-objective evolutionary algorithms for solving drs multi-objective optimization problems. In *Third International Conference on Natural Computation (ICNC 2007)*, volume 4, pages 96–100.

Mariani, T., Guizzo, G., Vergilio, S. R., and Pozo, A. T. (2016). Grammatical evolution for the multi-objective integration and test order problem. In *Proceedings of the Genetic and Evolutionary Computation Conference 2016 (GECCO 2016)*, pages 1069–1076.

Naujoks, B., Hochstrate, N., and Emmerich, M. (2005). Multi-objective optimisation using s-metric selection: application to three-dimensional solution spaces. volume 2, pages 1282 – 1289 Vol. 2.

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

O'Neill, M. and Ryan, C. (2001). Grammatical evolution. *IEEE Transactions on Evolutionary Computation*, 5(4):349–358.

Orouskhani, M., Teshnehlab, M., and Nekoui, M. A. (2017). Evolutionary dynamic multi-objective optimization algorithm based on borda count method. *International Journal of Machine Learning and Cybernetics*, pages 1–29.

O'Neill, M. and Ryan, C. (2004). Grammatical evolution by grammatical evolution: The evolution of grammar and genetic code. In *European Conference on Genetic Programming*, pages 138–149. Springer.

Ryan, C., Collins, J. J., and Neill, M. O. (1998). Grammatical evolution: Evolving programs for an arbitrary language. In *European Conference on Genetic Programming*, pages 83–96. Springer.

Zitzler, E., Laumanns, M., and Thiele, L. (2001). Spea2: Improving the strength pareto evolutionary algorithm.