Continuous Cartesian Genetic Programming based representation for Multi-Objective Neural Architecture Search for Image Classification

Cosijopii Garcia-Garcia, Alicia Morales-Reyes, Hugo Jair Escalante
Instituto Nacional de Astrofísica, Óptica y Electrónica
Luis Enrique Erro #1, Puebla, Mexico.
cosijopii@inaoe.mx, a.morales@inaoep.mx, hugojair@inaoep.mx

Abstract

We propose a novel approach for the challenge of designing less complex yet highly effective convolutional neural networks (CNNs) through the use of cartesian genetic programming (CGP) via neural architecture search (NAS). Our approach combines real-based and block-chained CNNs representations based on CGP for optimization in the continuous domain using multi-objective evolutionary algorithms (MOEAs). Two variants are introduced that differ in the granularity of the search space they consider. The proposed algorithms were evaluated using the non-dominated sorting genetic algorithm II (NSGA-II) in CIFAR-100 dataset.

Experimental results demonstrate that our approach is competitive with state-of-the-art proposals in terms of classification performance and model complexity.

1. Introduction

Deep neural networks (DNNs), particularly convolutional and recurrent neural networks, have recently gained considerable popularity for approaching a wide variety of problems [5, 6, 11, 21]. Regarding convolutional neural networks (CNNs), these are very effective computational models that have been thoroughly investigated in a wide range of image processing and computer vision-related tasks. This has been possible thanks to a number of factors including availability of large amounts of data, high-performance computing resources and research advances in the machine learning field [12].

Neural Architecture Search (NAS) is a field that aims at automating the design and configuration of CNN models [2]. NAS methods explore the space of CNN topologies looking for an architecture that meets certain criteria, for instance, achieving a minimum performance, or being light in terms of the number of parameters.

Nowadays, progress in NAS research has resulted in the identification of novel CNNs with favorable performance on representative image classification datasets, attracting the scientific community's attention to this topic [10].

In this paper, we introduce two multi-objective NAS methods based on Cartesian Genetic Programming (CGP), which aim to design CNN architectures for image classification. Our proposed CGP-NAS variants extend the original CGP-NAS [4] approach by considering a block-chain encoding. A first variant, called CGP-NASV1 is based on fixed CNN blocks to perform the NAS. While a second methodology, named CGP-NASV2, expands the search space to include the hyperparameters while relaxing the CGP-NASV1 fixed-blocks restriction. Our proposed NAS methods are evaluated in the context of image classification with CNNs. Experimental results in the CIFAR-100 dataset show that both proposals are competitive against several state-of-the-art references. The evolved CNNs architectures have a lower number of parameters as well as lower complexity measured in Multiply-Adds (MAdds) operations in comparison to the state-of-the-art and maintain a low classification error.

2. Continuous CGP based representation for multi-objective NAS

The automated configuration of CNN architectures has been mostly driven by methods that aim to optimize a single objective, commonly, accuracy. While effective architectures can be obtained by optimizing model's performance, such an approach has a number of limitations, the most important is perhaps the propensity of solutions to overfitting because model complexity is not restricted. Likewise, model complexity is tied with efficiency, hence, low compexity models should be preferred.

We propose the design of CNN architectures that simultaneously maximize model's accuracy and minimize its complexity. The goal is to restrict the capacity of the model with the aim of obtaining accurate models based on architectures of moderated complexity. Compared to alternative

solutions, ours is based on a CGP representation that allows us to operate in the real domain. This enables the usage of off-the-shell multi-optimization techniques.

The approached problem can be formulated as one of multi-objective optimization as follows:

$$\text{Minimize } \mathbf{F}(x) = (f_1(x; w^*(x), f_2(x))^T \quad (1)$$

$$\text{subject to: } w^*(x) \in \text{ argmin } \mathcal{L}(w; x) \quad (2)$$

Where f_1 is an objective associated to the classification error of the CNN architecture as defined by parameters w^* and f_2 is an objective associated to model complexity measured, in agreement with previous studies, with MAdds, as they express the total number of operations performed in each architecture. One should note that MAdds are a guideline for certain implementation scenarios, such as under mobile settings, where the complexity must be less than or equal to 600 MAdds [8,9]. Please note that in order to obtain an estimate of classification error it is necessary to optimize the weights w of the CNN architecture, so x (solution) depends on this optimization where normally some training algorithm such as stochastic gradient descent (SDG) is used. Two variants of our method are proposed. We first introduce a CGP-NASV1, a variant using a block-chained approach, see Figure 1. In each "Normal" block, an internal CGP representation itself handles the connections and functions, in the "Reduction" blocks, a spatial reduction is applied, this representation is re-encoded to a real-valued representation. Therefore, the full MOEAs searching potential within continuous domain is exploited.

In a second variant of CGP-NAS, we included the hyperparameters of the CNNs in the CGP representation, increasing the flexibility of model configurations that can be find, at the expense of increasing the search space.

2.1. CGP-NASV1 solution representation

CGP-NASV1 uses the block-chained encoding at the top level, it defines an template with some layers as shown in Figure 1. This representation connects blocks linearly, including those with specific tasks, in CGP-NASV1 the spatial reduction is performed by the pooling layers [10].

At the top-level in the block-chained, CGP-NASV1 places a CGP within a "Normal" block while the reduction blocks implements a max pooling layer. In all pooling blocks, the pooling size is fixed as a 2×2 kernel and a stride of 2. Figure 2 shows an example, after the last block a global average pooling and a fully connected layer are added. Similar guidelines were found in the state of the art review [10, 13]. We maintained the original idea from CGP-NAS of having two levels for representation [4]. In CGP-NASV1, each "Normal" block from the block-chained representation holds a CGP and it is represented as an integer vector. At the second level of representation, the integer vector is encoded to a real-based vector. Equation 3

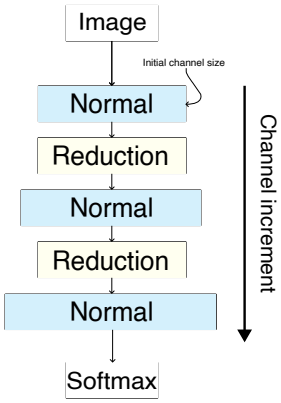

Figure 1. The general scheme of the representation based on chained blocks; each block internally focuses on special operations; in each normal block, for example, convolution operations are performed; on the other hand, in the reduction blocks, methods such as pooling are applied to spatially reduce the feature maps. As the number of blocks increases, the number of channels will gradually increase.

defines a range considering $func_k$ as the current function identifier and $func_{total}$ as the total number of functions in the function set. A uniform random number is generated to represent a function within this range in the real domain.

$$rfunc_k \in \left[\frac{func_k}{func_{total}}, \frac{func_k + 1}{func_{total}}\right] \quad (3)$$

Equation 4 defines a range for the function inputs in the real domain to map connections for that node. This operation is applied to all connections. An input value ($nodeinput$) and its node number ($nodeTerm$) are used to calculate this range.

$$rinput_j \in \left[\frac{nodeinput_j}{nodeTerm}, \frac{nodeinput_j + 1}{nodeTerm}\right] \quad (4)$$

$$func_k = \lfloor gene_i \times func_{total} \rfloor \quad (5)$$

$$input_j = \lfloor gene_i \times NodeTerm \rfloor \quad (6)$$

In order to decode solutions from the real to the integer domain, Equation 5 obtains the function identifier by multiplying the gene value by the total number of functions. Moreover, Equation 6 obtains the value of every connection by multiplying the gene value with the node number. The equations above are based on Clegg's work [1].

CGP-NASV1 represents CNN architectures in a divide and conquer approach. The use of a block-chained schema in synergy with CGP allows more control over the final architecture,

2.2. CGP-NASV2 representation

CGP-NASV2 proposal integrates the hyperparameters within the solutions representation of CGP-NASV1. Thus,

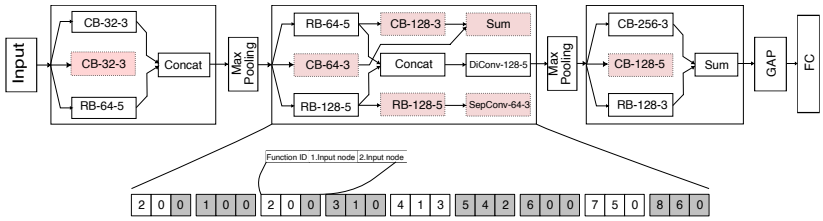

Figure 2. CGP-NASV1 using the block-chained representation

they are also evolved and optimized during the search. Changes in CGP-NASV2 are at the block-chained "low level" encoding. Therefore, there is a reduction in the number of functions within the function set. However, adding the hyperparameters to the search, necessarily increases the size of the integer vectors (and also the real-based vectors) to which the block-chained solutions are encoded.

In a deep neural architecture, there are hyperparameters such as the number of channels or filters and the kernel size. In CGP, these are explicitly included in the functions set if not considered within the solutions representation and in the evolutionary search. Therefore, a large function set must be defined in order to include all their possible combinations. Moreover, a large function set directly increases the CGP grid size to hold them uniformly. In CGP-NASV2, we add the hyperparameters explicitly to the vector representing the CGP. This reduces the number of functions since only the standard functions remain and not their hyperparameters configuration.

In Figure 3, a possible solution in CGP-NASV2 is shown. The number of channels and kernel size are defined as parameters associated to each block-chained. The idea is to associate these parameters as weights to each CGP node. Green positions at the integer vector encoding in Figure 3 represent the assigned hyperparameters within their corresponding CGP block. To convert the integer vector to the real-based vector, the same mechanism as the one used in CGP-NASV1 is applied adding only Equations 7 and 8 to encode and decode the hyperparameters.

$$rHyp_k \in \left[\frac{Hyp_k}{Hyp_{total}}, \frac{Hyp_k + 1}{Hyp_{total}}\right] \tag{7}$$

$$Hyp_j = \lfloor gene_i \times Hyp_{total} \rfloor \tag{8}$$

3. Experimental framework

In order to assess both proposed approaches, CGP-NASV1 and CGP-NASV2, first a direct performance comparison between the best evolved architecture based on the achieved accuracy as the performance metric is carried out. After, a multiple-criteria decision analysis is performed to evaluate those evolved CNN models that better achieve a more balanced trade-off performance considering both objectives, accuracy and complexity. In this section, we present the experimental settings for both experiments as well as the considered datasets. In the next section, we discuss the obtained results and thoroughly compared them against the original CGP-NAS [4] proposal and several state of the art approaches.

Table 1. CGP-NASV1 and CGP-NASV2 parameters

Parameters	CGP-NASV1	CGP-NASV2
Rows	5	10
Columns	25	4
Level-Back	1	1
Mutation probability	$Pm = 0.3$	
Crossover probability	$Pc = 0.9$, distribution index for simulated binary crossover $Dsc = 20$.	
Population	24	24
Generations	30	30

Table 1 shows CGP-NASV1 and CGP-NASV2 configuration in terms of the CGP and the evolutionary algorithm.

3.1. Comparison versus the State of the Art

Table 2 present a detailed comparison between the State of the Art works and the proposed algorithmic approaches CGP-NASV1 and CGP-NASV2 CIFAR-100 datasets. A total of 4 human designs, 5 NAS single-objective and 7 mutli-objective proposals are considered for comparison plus the original proposal CGP-NAS [4] is also included in both tables. From previous empirical assessments, it was determined that CGP-NASV2 provided the best overall results when compared to CGP-NASV1 and their baseline CGP-NAS.

In comparison with other methods, particularly those designed by humans, our proposal demonstrates superior performance in terms of both classification error and number of parameters in both datasets. When compared with single-objective methods, our proposal outperforms them in terms of classification error while showing a significant reduction in the number of parameters. It is important to note, however, that minimizing the number of parameters is not the primary objective of our proposal. When comparing with multi-objective methods in terms of classification er-

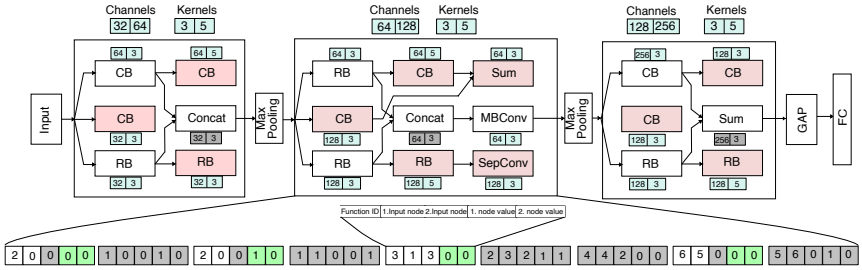

Figure 3. CGP-NASV2 block-chained representation with the hyperparameters directly encoded.

Table 2. Comparison on CIFAR-100 dataset: Classification error rate, the number of parameters and Multiply-adds (MAdds) are expressed in millions (1×10^6), GPU-days and GPU Hardware. The Knee solutions were selected using the Knee and boundary selection method [3].

Model	Error rate %	Params	MAdds	GPU-Days	GPU hardware
Single Objective Approaches					
CGP-CNN(ConvSet) (Suganuma et al., 2020) [15]	26.7	2.04	-	13	Nvidia 1080Ti
CGP-CNN(ResSet) (Suganuma et al., 2020) [15]	25.1	3.43	-	10.9	Nvidia 1080Ti
Large-Scale Evolution (Real et al., 2017) [14]	23.0	40.4	-	2750	-
AE-CNN (Sun et al., 2020) [16]	20.85	5.4	-	36	Nvidia 1080 Ti
Genetic-CNN (Xie et al., 2017) [19]	29.03	-	-	17	-
(Torabi et al., 2022) [18]	26.03	2.56	-	-	NVIDIA Tesla V100-SXM2
Multi-Objective Approaches					
NSGANetV1 (Lu et al., 2020) [8]	25.17	0.2	1290	27	Nvidia 2080 Ti
MOGIG-Net (Xue et al., 2021) [20]	24.71	0.7	-	14	-
EEEA-Net (Termritthikun et al., 2021) [17]	15.02	3.6	-	0.52	Nvidia RTX 2080 Ti
LF-MOGP(Liu et al., 2022) [7]	26.37	4.12	-	13	NVIDIA GeForce 3090
CGP-NAS(Garcia-Garcia et al., 2022) [4]	24.23 (26.41 ± 1.41)	5.43	1581	2.1	Nvidia Titan X
CGP-NASV1 - Knee solutions	31.47 (33.34 ± 1.6)	1.06 (0.82 ± 1.12)	45.70 (30.81 ± 9.7)	8.82	Nvidia 1080Ti
CGP-NASV1	21.76 (24.20 ± 1.8)	3.6 (7.25 ± 3.24)	791.85 (792.02 ± 342.6)	8.82	Nvidia 1080Ti
CGP-NASV2	**20.63** (22.49 ± 1.04)	5.9 (6.50 ± 1.7)	827 (850.74 ± 476.08)	11.28	Nvidia 1080Ti
CGP-NASV2 - Knee solutions	23.57 (28.43 ± 2.20)	0.49 (0.53 ± 0.13)	66.66 (55.66 ± 19.14)	11.28	Nvidia 1080Ti

ror, the EEEA-NET [17] method shows better performance, albeit at the cost of a higher number of parameters. In other metrics, our proposal outperforms the other methods presented. Our proposal aims to find solutions with a favorable trade-off between classification error and MAdds, resulting in architectures with reduced parameters and MAdds. In comparison with other proposals that use CGP as a method for architectures representation, such as CGP-CNN [15], Torabi [18], EvoApproxNas [13] and LF-MOGP [7] , our proposal demonstrates superior performance.

Works like EvoApproxNAS [13] focuses on optimizing values at the hardware level with a multi-objective approach, keeping the CGP representation intact and adding some block functions like bottleneck and residual inverted to the function set while keeping the CGP properties unchanged by only using mutations. On the other hand, none of these proposals modifies CGP at the representation level, instead attacking specific problems, for example by proposing new operators (Torabi [18]) or mechanisms to improve the search process (LF-MOGP [7]), which seems to be an important component since the results shown improve compared to other proposals, even though CGP-NASV2 shows a superior performance with canonical MOEAS.

4. Conclusions

CGP is known to be a robust method for architecture representation, and the use of real-based representation in our proposal improves the performance, which can be attributed to the relaxation of the search space and the better use of MOEAs, which are designed to operate in a continuous domain. In summary, CGP-NASV1 and CGP-NASV2 demonstrate the ability to identify solutions with a favorable trade-off between two objectives, specifically by achieving a lower number of MAdds when compared to other methods. It is important to note that depending on the specific application or task at hand, different methods for selecting solutions, such as those presented in this work, may be utilized to determine the most appropriate solution for the given scenario.

References

[1] Janet Clegg, James Alfred Walker, and Julian Frances Miller. A new crossover technique for Cartesian genetic programming. In *Proceedings of the 9th annual conference on Genetic and evolutionary computation - GECCO '07*, page 1580, New York, New York, USA, 2007. ACM Press. 2

[2] Thomas Elsken, Jan Hendrik Metzen, and Frank Hutter. Neural Architecture Search: A Survey. Technical report, 2019. 1

[3] Francisco E. Fernandes Jr. and Gary G. Yen. Pruning Deep Convolutional Neural Networks Architectures with Evolution Strategy. *Information Sciences*, 552:29–47, apr 2021. 4

[4] Cosijopii Garcia-Garcia, Hugo Jair Escalante, and Alicia Morales-Reyes. CGP-NAS. In *Proceedings of the Genetic and Evolutionary Computation Conference Companion*, volume 1, pages 643–646, New York, NY, USA, jul 2022. ACM. 1, 2, 3, 4

[5] Sorin Grigorescu, Bogdan Trasnea, Tiberiu Cocias, and Gigel Macesanu. A survey of deep learning techniques for autonomous driving. *Journal of Field Robotics*, 37(3):362–386, 2020. 1

[6] Morten Kolbk, Zheng-Hua Tan, Jesper Jensen, Morten Kolbk, Zheng-Hua Tan, and Jesper Jensen. Speech intelligibility potential of general and specialized deep neural network based speech enhancement systems. *IEEE/ACM Trans. Audio, Speech and Lang. Proc.*, 25(1):153–167, jan 2017. 1

[7] Qingqing Liu, Xianpeng Wang, Yao Wang, and Xiangman Song. Evolutionary convolutional neural network for image classification based on multi-objective genetic programming with leader–follower mechanism. *Complex and Intelligent Systems*, 2022. 4

[8] Zhichao Lu, Kalyanmoy Deb, Erik Goodman, Wolfgang Banzhaf, and Vishnu Naresh Boddeti. NSGANetV2: Evolutionary Multi-objective Surrogate-Assisted Neural Architecture Search. In *LNCS*, volume 12346 LNCS, pages 35–51, aug 2020. 2, 4

[9] Zhichao Lu, Gautam Sreekumar, Erik Goodman, Wolfgang Banzhaf, Kalyanmoy Deb, and Vishnu Naresh Boddeti. Neural Architecture Transfer. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pages 1–1, 2021. 2

[10] Zhichao Lu, Ian Whalen, Yashesh Dhebar, Kalyanmoy Deb, Erik Goodman, Wolfgang Banzhaf, and Vishnu Naresh Boddeti. Multi-Objective Evolutionary Design of Deep Convolutional Neural Networks for Image Classification. *IEEE Transactions on Evolutionary Computation*, pages 1–1, sep 2020. 1, 2

[11] Aritz D. Martinez, Javier Del Ser, Esther Villar-Rodriguez, Eneko Osaba, Javier Poyatos, Siham Tabik, Daniel Molina, and Francisco Herrera. Lights and shadows in Evolutionary Deep Learning: Taxonomy, critical methodological analysis, cases of study, learned lessons, recommendations and challenges. *Information Fusion*, 67:161–194, mar 2021. 1

[12] Risto Miikkulainen, Jason Liang, Elliot Meyerson, Aditya Rawal, Daniel Fink, Olivier Francon, Bala Raju, Hormoz Shahrzad, Arshak Navruzyan, Nigel Duffy, and Babak Hodjat. Evolving Deep Neural Networks. In *Artificial Intelligence in the Age of Neural Networks and Brain Computing*, pages 293–312. Elsevier, 2019. 1

[13] Michal Pinos, Vojtech Mrazek, and Lukas Sekanina. Evolutionary approximation and neural architecture search. *Genetic Programming and Evolvable Machines*, jun 2022. 2, 4

[14] Esteban Real, Sherry Moore, Andrew Selle, Saurabh Saxena, Yutaka Leon Suematsu, Jie Tan, Quoc Le, and Alexey Kurakin. Large-Scale Evolution of Image Classifiers. mar 2017. 4

[15] Masanori Suganuma, Masayuki Kobayashi, Shinichi Shirakawa, and Tomoharu Nagao. Evolution of deep convolutional neural networks using cartesian genetic programming, mar 2020. 4

[16] Yanan Sun, Handing Wang, Bing Xue, Yaochu Jin, Gary G. Yen, and Mengjie Zhang. Surrogate-Assisted Evolutionary Deep Learning Using an End-to-End Random Forest-Based Performance Predictor. *IEEE Transactions on Evolutionary Computation*, 24(2):350–364, apr 2020. 4

[17] Chakkrit Termritthikun, Yeshi Jamtsho, Jirarat Ieamsaard, Paisarn Muneesawang, and Ivan Lee. EEEA-Net: An Early Exit Evolutionary Neural Architecture Search. *Engineering Applications of Artificial Intelligence*, 104:104397, 2021. 4

[18] Ali Torabi, Arash Sharifi, and Mohammad Teshnehlab. Using Cartesian Genetic Programming Approach with New Crossover Technique to Design Convolutional Neural Networks. *Neural Processing Letters*, 2022. 4

[19] Lingxi Xie and Alan Yuille. Genetic CNN. In *2017 IEEE International Conference on Computer Vision (ICCV)*, pages 1388–1397. IEEE, oct 2017. 4

[20] Yu Xue, Pengcheng Jiang, Ferrante Neri, and Jiayu Liang. A Multi-objective evolutionary approach based on graph-in-graph for neural architecture search of convolutional neural networks. *International Journal of Neural Systems*, 31(9), sep 2021. 4

[21] Tom Young, Devamanyu Hazarika, Soujanya Poria, and Erik Cambria. Recent trends in deep learning based natural language processing [review article]. *IEEE Computational Intelligence Magazine*, 13(3):55–75, 2018. 1