ISSN 1870-4069

Induction of Convolutional Decision Trees with Differential Evolution for Image Segmentation

Jesús Arnulfo Barradas Palmeros, Efrén Mezura Montes,
Héctor Gabriel Acosta Mesa, Aldo Márquez Grajales,
Rafael Rivera López

Universidad de Veracruz,
Instituto de Investigaciones en Inteligencia Artificial,
Departamento de Sistemas y Computación,
Mexico

{jesus.arnulfo.bap, li.aldoma}@gmail.com, {emezura,
heacosta}@uv.mx, rrivera@itver.edu.mx

Abstract. Convolutional Neural Networks are the dominant approach for solving the image segmentation problem. However, they demand significant amounts of manually labeled data for training and suffer from lacking explainability. As an alternative, Convolutional Decision Trees take advantage of the interpretability and simplicity of decision tree models. Nevertheless, choosing between equivalent trees is a challenging task, given the trade-off between the model's precision and complexity. In this work, we propose using Differential Evolution as a global search metaheuristic for the induction of Convolutional Decision Trees applied to the image segmentation problem. Various tests were conducted on the Weizmann Horse dataset, where the elevated computational cost of determining the individuals' fitness value limited the search. Nonetheless, short and explainable models were induced with promising results for some parts of the dataset. In this way, Differential Evolution appears as an attractive tool for Convolutional Decision Trees induction, expecting future improvements.

Keywords: Convolution, decision trees, differential evolution, image segmentation.

1 Introduction

The image segmentation problem consists of assigning a semantic label to the pixels in an image. Differentiating an object from the image's background is essential in most image analysis systems. Thus, various image segmentation methods have been proposed in the literature [7]. However, in recent years, the high performance of Convolutional Neural Networks (CNN) as a Deep Learning technique has been proclaimed the dominant paradigm in computer vision [12].

Therefore, various approaches using CNN have been studied for the image segmentation problem, as shown in [3]. Despite the high-performance results reached by CNN in different tasks, they suffer from requiring lots of labeled data for training.

Additionally, CNN faces the difficulty of high training time, making some applications unsuitable for their use or demanding a specific hardware infrastructure [8]. Consequently, some challenges remain for CNN, such as their explainability, the efficient use of memory, and the speed to process a new instance on a real-time application [12].

Decision Trees (DT) are a classification model characterized by their simplicity and interpretability where internal nodes represent the test conditions and leaf nodes are the class labels. Nonetheless, the DT induction classic process uses a greedy recursive partitioning heuristic that suffers from adaptability in some applications. Alternatively, various approaches to using metaheuristics for decision tree induction have been proposed in the literature [15].

Convolutional Decision Trees (CDT) were proposed in [8] for image segmentation and feature learning problems, performing well and using a fraction of the time needed for training a CNN without a particular hardware configuration. This approach was tested on the Weizmann Horse dataset [2], obtaining an F1-score of 80.4% with a tree depth of 18. However, results showed that trees with short depths have less adequate performance.

Three main metaheuristic-guided DT induction strategies are described in [11]. The first consists of a recursive partition strategy where the metaheuristic finds a near-optimal partition. The second strategy uses the metaheuristic as a global search technique that looks for the complete model of a near-optimal DT.

An essential challenge in this approach is maintaining diversity in the population. Besides, the computational cost of the fitness value calculation increases considerably with high-dimension datasets. Finally, the third strategy uses a previously induced DT and continually optimizes it according to the metaheuristic.

A population-based metaheuristic used in literature for the induction of near-optimal DT is the Differential Evolution (DE) algorithm [15]. DE is one of the most popular metaheuristic search strategies and has been applied successfully for solving several optimization problems. Furthermore, DE is prominent in the algorithm simplicity where few parameters control the search process [6].

Two ways of using DE in DT induction are shown in [10, 15]. Perceptron Decision Trees incorporate a linear combination test condition on each internal node. DE is used to evolve the values of the tree's structure [10]. In [15], the DE-ADT$_{SPV}$ method uses DE as a global search strategy to find near-optimal parallel-axis DT coding the trees as real-value vectors.

Both works achieved decent results in the classification accuracy obtained by their resulting models. The use of DE in two different strategies for the induction of oblique decision trees is studied in [16]. The first is OC1-DE, where a recursive partition strategy is used with DE to find near-optimal partitions for each tree node. The second method is DE-ODT, which uses a global search strategy to find a near-optimal oblique decision tree.

The solutions representation corresponds to a real-value vector that codes the internal values of a tree. The length of the vector depends on the number of attributes given and the predefined depth of the tree. Both methods described showed their effectiveness in decision tree induction.

Based on the literature, it is seen that DE has been used in various DT induction processes. Nevertheless, to the best of our knowledge, DE has not been applied for the particular CDT induction case. Therefore, this paper uses DE as a global search strategy to induce CDT. Consequently, DT and DE characteristics were employed to construct an explainable model for the image segmentation problem.

The remaining structure of this document is divided into four sections. Section 2 includes the DE algorithm description. Implementation details are defined in section 3, whereas section 4 explains the experimentation and the results obtained during tests. Finally, section 5 contains the conclusions and suggested future work.

2 Differential Evolution (DE)

Differential Evolution (DE) is a population-based evolutionary algorithm for optimization of complex problems [13]. DE mainly works with real-valued vectors representing potential solutions to the problem. However, DE is also applied in the discrete and combinatory domain [14].

The basic strategy of DE is called DE/rand/1/bin [16, 1, 5]. The general DE procedure generates a trial vector for every individual x_i or target vector in the population. The first step consist of generating a noise vector using Equation 1 where r_0, r_1 and r_2 are individuals randomly selected from the population and F is a user-defined scale factor:

$$v_i = r_0 + F(r_1 - r_2). \tag{1}$$

Once v_i is computed, the trial vector is generated stochastically. Equation 2 expresses this process. If a random number ($rand_j$) is lower than a Crossing Rate (CR) defined by the user or the position (j) corresponds to one previously determined by chance, the component takes the value from v_i. Otherwise, it takes the value from x_i:

$$u_{i,j} = \begin{cases} v_{i,j} & \text{if } (\text{rand}_j \leq CR) \; or \; (j = J_{rand}); j = 1, ..., |x_i|, \\ x_{i,j} & \text{otherwise.} \end{cases} \tag{2}$$

Finally, the next step is determining which vector will be part of the next generation population. A binary tournament between trial and target takes place where the one with the better fitness value is chosen [16]. This selection method works as an elitism mechanism by always keeping the best individual in the population.

Another DE variant is called DE/best/1/bin, where the main difference is in the computation of v_i. Instead of choosing a random individual as r_0, the individual in the population with the highest fitness value is chosen [6].

3 Proposal Implementation

This work proposes an analysis of the DE effectiveness for CDT induction. The procedure implemented is based on the DE-ODT algorithm for oblique decision tree induction proposed in [16], where the values of each node are represented in a vector that is evolved.

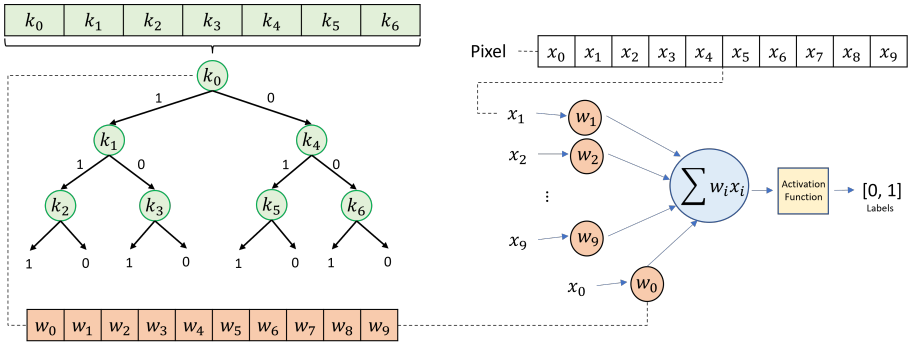

Fig. 1. Codification of a Convolutional Decision Tree internal convolution kernels and how a pixel-associated instance is processed.

In this proposal, the values coded in the vector represent internal convolution kernels for the tree. To determine which tree branch to take while classifying a dataset instance, we propose using a perceptron-like structure similar to [10]. The product between the instance values and the weights of a convolution kernel passes through an activation function returning a 0 or 1 label.

Based on that label, the tree node where the instance needs to go is decided. This procedure is repeated until a leaf node is reached, then a label is assigned to the instance. Figure 1 illustrates the proposal. DE/rand/1/bin and DE/best/1/bin are the two DE variants used in this work. Both versions need user-defined parameters such as scale factor F, crossing rate CR, population size, and how many generations will run the algorithm. Moreover, two additional parameters of the tree structure are required for this application: kernel size and tree depth.

3.1 Images Preprocessing

The images used for the tree induction procedure are preprocessed to obtain a vector associated with each pixel. Then, according to [10], pixels are coded as a vector of values given by the neighbor pixels. The vector length depends on the kernel size defined by the user. Additionally, a value of 1 is included in every instance to operate with the bias value of the proposed perceptron-like structure.

3.2 Coding Potential Solutions

A population's individual represents a solution consisting of a real-valued vector with the values of convolution kernels associated with the tree's internal nodes. The amount of weights needed by each kernel depends on kernel size ($s \times s$). Therefore, the weight amount is computed as s^2+1 where s is the kernel side length, and the one corresponds to the bias value.

Another aspect to be considered when a solution is coded is the number of kernels needed in the tree. This value depends entirely on tree depth and is determined by 2^d-1, where d is the depth defined by the user. Figure 2 shows an example of this codification.

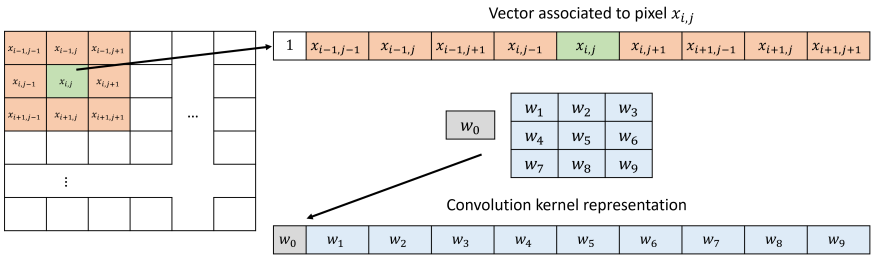

Fig. 2. Coding of pixel-associated instances and convolution kernels.

3.3 Fitness Value

The fitness value of each individual is determined by the F1-score metric based on the precision and recall metrics. To calculate this value, we need to compute the labels assigned to the training instances by the tree coded in each individual of the population and compare them with the actual labels of the instances. The resulting fitness value is between 0 and 1, and we try to maximize it using DE.

The initial population is generated with fixed length vectors of randomly chosen values from a uniform distribution with limits -255 and 255. After that, individuals are evaluated to compute their fitness value, and the DE procedure starts. Additionally, the Mean Point Distance metric [**?**] is used in every generation to measure diversity in the population.

3.4 Repair Operator

A repair operator is used to restrain the search space and keep the values of the tree between -255 and 255. While computing the v_i vector, if a value exceeds one of the limits imposed, a new value is calculated as two times the exceeded limit (-255 or 255) minus the value that infringed the restriction.

4 Experiments and Results

A single user-defined image was used to create short training and test sets as a controlled algorithm initial test. Then, a first algorithm parameters calibration was done, obtaining favorable results in pattern detection. These results identified that higher values of population size and the number of generations resulted in more suitable individuals at the end of the search. Nonetheless, both have a direct impact on the procedure's computational cost. After the initial tests, we executed 13 extended tests using the Weizmann Horse Dataset [2], which consists of 328 manually segmented horses images, allowing us to compare our results with the ones described in [8].

An image resizing procedure was applied to reduce the number of pixel-associated instances processed in the tree induction process. Moreover, multiple tests were conducted varying some algorithm parameters. In addition, population size and generation number values were adjusted to make each test last less than 24 hours. Finally, CR and F parameters were maintained at 0.9 in all executions.

Table 1. Tests of the proposed method for Convolutional Decision Trees induction.

Test	DE-var	Training	Popsize	Generations	Depth	F1-score	Accuracy	Time(hrs)
1	Rand	2/3	40	59	3	0.4296	0.6277	22.31
2	Rand	2/3	40	59	3	0.4421	0.6770	21.21
3	Rand	2/3	40	59	3	0.4671	0.5925	21.73
4	Best	2/3	40	60	3	0.4480	0.6546	18.65
5	Best	2/3	40	60	3	0.4465	0.6473	16.75
6	Best	2/3	40	60	3	0.4730	0.6877	18.32
7	Best	1/3	46	100	3	0.4513	0.6549	21.92
8	Best	1/10	80	200	3	0.4882	0.6798	23.17
9	Best	2/10	60	120	3	0.4819	0.6846	19.76
10	Best	2/10	50	100	5	0.4531	0.6025	16.72
11	Best	1/10	60	110	7	0.4696	0.6827	21.07
12	Best	1/20	50	130	13	0.4367	0.6504	16.32
13	Best	1/25	40	100	17	0.4255	0.5881	22.22

In the first six tests, the fraction of training data was maintained with similar population size, generation number, and a tree depth value of three. Also, the two DE versions previously mentioned in the document were employed. The main obstacle found was the time required for each test. Hence, the selected values of population size and the number of generations were limited.

After that, we tried different configurations of training set size to decrease the time consumed by the induction process, allowing us to increase parameters like the tree depth. Without reducing the training set, the resources and time demanded would have impeded testing deeper tree models. Table 1 shows our proposal's results under the above-mentioned considerations.

This table shows that DE/best/1/bin got better results and was faster than DE/rand/1/bin when tested in similar conditions. As a consequence, this DE variant was used during the remaining tests. Training set reduction did not significantly decrease the method's performance showing the model's capacity for generalization even though the induction process takes place with small amounts of data.

Test 8 resulting model got the highest F1-score while using only 10% of the data for training. More complex models did not imply better results in our tests, but the induction used even more reduced fractions of the dataset. Figure 3(a) presents the tree induced in test 9, whereas Figure 3(b) shows a comparison of the actual mask and the predicted mask of 12 test images from the same test.

We noticed that the model struggles with background and foreground textures, while horses' contour is detected in most images. In none of the test cases we achieved similar results to the 80.4% of F1-score obtained in [8] where their method for CDT induction takes 12 hours for training. Nevertheless, our results are comparable for short-depth trees.

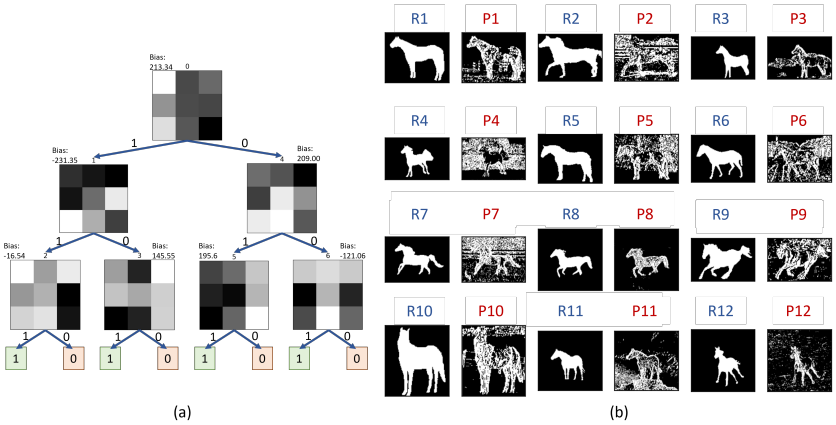

Fig. 3. (a) Tree induced in test 9. (b) A sample of the segmentation results obtained by the tree showed in (a). The letter R is used to identify the real segmented masks, and the letter P is for the predicted ones.

5 Conclusions and Future Work

In this work, a method for CDT induction with DE is proposed and compared with the original method proposed in [8]. The main difference is using a population-based metaheuristic to induce several trees instead of only one with a recursive partition strategy. Applying DE for the CDT induction faces the computational time problem when evaluating the capacity to classify the training instances by the population's individuals.

This classification ability is the fitness function of DE. Moreover, this task increases resource demand as the number of training instances is augmented. When an image dataset is used, the amount of pixel-associated instances is quite considerable, in the order of millions, making the labor of the model induction more complex.

In conclusion, we suffer from the search limits imposed by the computational cost required by the induction process in our proposal. This situation forced us to use less adequate parameters for the DE algorithm and reduce the training data fraction. However, DE was still capable of inducing short and explainable models. Given this, one thing to highlight is the method's capability to train with little data without additional processes to augment the training set.

For the model's explainability, one could successively apply convolutional operations to an image following the tree structure and analyze the results for each branch and leaf node. Nevertheless, deeper tree models reduce explainability, given the elevated model's number of branches and kernels.

In order to make the proposed procedure proficient, it is necessary to overcome the challenge of the computational cost of evaluating an individual. Without accomplishing this, the capabilities of using DE as a global search tool would still be limited for this type of problem. Future work could include trying a self-adapted DE scheme and exploring different parameter values for kernel size and tree depth.

Additionally, some improvements could be implementing techniques like windowing [9] to reduce the number of training instances and methods like pruning to enhance the resulting trees. For future reference, it is necessary to compare the proposal performance with diverse approaches for image segmentation, such as U-Net and other Convolutional Neural Networks methods [3].

Acknowledgments. The first author is funded by a Consejo Nacional de Ciencia y Tecnología (CONACYT) of Mexico.

References

1. Bilal, Pant, M., Zaheer, H., Garcia-Hernandez, L., Abraham, A.: Differential Evolution: A Review of More Than Two Decades of Research. Engineering Applications of Artificial Intelligence, vol. 90 (2020). DOI: 10.1016/j.engappai.2020.103479.
2. Borenstein, E., Sharon, E., Ullman, S.: Combining Top-down and Bottom-up Segmentation. In: Conference on Computer Vision and Pattern Recognition Workshop, pp. 46–46 (2004). DOI: 10.1109/CVPR.2004.314.
3. Cao, F., Bao, Q.: A Survey on Image Semantic Segmentation Methods with Convolutional Neural nNetwork. In: International Conference on Communications, Information System and Computer Engineering, pp. 458–462 (2020). DOI: 10.1109/CISCE50729.2020.00103.
4. Contreras-Varela, L.: Un estudio sobre diversidad en optimizacion evolutiva con restricciones, Universidad Veracruzana, Xalapa (2018)
5. Dabbagh, R. D. A., Neri, F., Idris, N., Baba, M. S.: Algorithmic Design Issues in Adaptive Differential Evolution Schemes: Review and Taxonomy. Swarm and Evolutionary Computation, vol. 43, pp. 284–311 (2018). DOI: 10.1016/j.swevo.2018.03.008.
6. Faiz-Ahmad, M. Mat-Isa, N. A., Hong-Lim W., Meng-Ang, K.: Differential Evolution: A Recent Review Based on State-of-the-art Works. Alexandria Engineering Journal, vol. 61, no. 5, pp. 3831–3872 (2022). DOI: 10.1016/j.aej.2021.09.013.
7. Hadjiiski, L., Samala, R., Chan, H. P.: Chapter 88 - Image Processing Analytics: Enhancements and segmentation. In: Molecular imaging, pp. 1727–1745 (2021). DOI: 10.1016/B978-0-12-816386-3.00057-0.
8. Laptev, D., Buhmann, J. M.: Convolutional Decision Trees for Feature Learning and Segmentation. In: Pattern Recognition: 36th German Conference, pp. 95–106 (2014). DOI: 10.1007/978-3-319-11752-2 8.
9. Limon, X., Guerra-Hernández, A., Cruz-Ramírez, N., Acosta-Mesa, H. G., Grimaldo, F.: A Windowing Strategy for Distributed Data Mining Optimized Through GPUs. Pattern Recognition Letters, vol. 93, pp. 23–30 (2017). DOI: 10.1016/j.patrec.2016.11.006.
10. Lopes, R. A., Freitas, A., Silva, R. P., Guimaraes, F. G.: Differential Evolution and Perceptron ˜ Decision Trees for Classification Tasks. In: Proceddings of 13th International Conference on Intelligent Data Engineering and Automated Leraning, pp. 550–557 (2012). DOI: 10.1007/ 978-3-642-32639-4 67.
11. Lopez, R. R., Reich, J. C., Montes, E. M., Chavez, M. A. C.: Induction of Decision Trees as Classification Models Through Metaheuristics. Swarm and Evolutionary Computation, vol. 69 (2022). DOI: 10.1016/j.swevo.2021.101006.
12. Minaee, S., Boykov, Y., Porikli, F., Plaza, A., Kehtarnavaz, N., Terzopoulos, D.: Image Segmentation Using Deep Learning: A Survey. In: IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 44, pp. 3523–3542 (2022). DOI: 10.1109/TPAMI.2021.3059968.

13. Opara, K. R., Arabas, J.: Differential Evolution: A Survey of Theoretical Analyses. Swarm and Evolutionary Computation, vol. 44, pp. 546–558 (2019). DOI: 10.1016/j.swevo.2018.06.010.
14. Price, K. V.: Differential evolution. In: Zelinka, I., Snásel, V., Abraham, A. (eds) Handbook ˘ of Optimization. Intelligent Systems Reference Library, vol. 38, pp. 187–214 (2013). DOI: 10.1007/978-3-642-30504-7 8.
15. Rivera-Lopez, R., Canul-Reich, J.: Construction of Near-Optimal Axis-parallel Decision Trees Using a Differential-evolution-based Approach. IEEE Access, vol. 6, pp. 5548–5563 (2018). DOI: 10.1109/ACCESS.2017.2788700..
16. Rivera-Lopez, R., Canul-Reich, J.: Differential Evolution Algorithm in the Construction of Interpretable Classification Models. Artificial intelligence-emerging trends and applications Chapter 3, pp. 49–73 (2018). DOI: 10.5772/intechopen.75694.