# Data sampling via Active Learning in Cartesian Genetic Programming for Biomedical Data

Yuri Lavinas*, Nathan Haut† and William Punch† and Wolfgang Banzhaf† and Sylvain Cussat-Blanc*
*IRIT - CNRS UMR5505, University of Toulouse
Email: lavinas.yuri.xp@alumni.tsukuba.ac.jp;
Email: sylvain.cussat-blanc@irit.fr
†Michigan State University, Department of Computer Science
Email: hautnath@msu.edu
Email: punch@msu.edu
Email: banzhafw@msu.edu

***Abstract*—In this contribution, we explore Cartesian Genetic Programming for image analysis of biomedical data. Producing large quantities of human-labeled biomedical data is an expensive task. Here, we introduce a way for CGP to use a small amount of training data, without loss in performance. To define the size of the training data, we utilize an Active Learning method to direct the algorithm towards informative samples. We examine how sampling a small set of data from the CELLPOSE dataset affects the performance of CGP. We also study the effects of restarting CGP with Active Learning. We found that using several restarts can lead to a more diverse set of the highest-performing solutions with fewer active nodes while maintaining similar performance to standard CGP.**

***Index Terms*—cartesian genetic programming, image analysis, image processing, data sampling**

## I. Introduction

As with most of the computer vision domain, the field of biomedical image analysis has recently been revolutionized by Neural networks and, in particular, Deep Learning approaches. They have shown to be very efficient in many image classification and segmentation applications such as in dermatology, radiology and pathology [1], [2], and in some cases even outperforming human experts in these tasks. However, Deep Learning approaches have two main challenging drawbacks. First, they are considered black-box approaches: the decisions taken by Deep Neural Networks are, nowadays, hard or maybe even impossible to explain to human experts, which is a requirement in critical applications such as medicine. Second, they require large amounts of annotated data. This task has to be done by experts who are often not readily available and is very time-consuming.

Recent work showed that Cartesian Genetic Programming (CGP) is an effective approach to address the above-mentioned limitation of Deep Learning [3], [4]. CGP evolves solutions based on a set of mathematical functions that, when executed, process given inputs to produce an expected output. The phenotype of these programs can be represented as graphs which might be evolved using a (1+$\lambda$) evolutionary strategy [5]. One of the main benefits of CGP is the use of a fixed-length integer-based genome to encode the functional graphs, not having the bloating effect encountered in many tree GP approaches. Therefore, small programs can be evolved which facilitates interpretability [6]. In the specific case of image processing, the function library contains computer vision functions that are combined and optimized in order to build an image processing pipeline adequate to the given task.

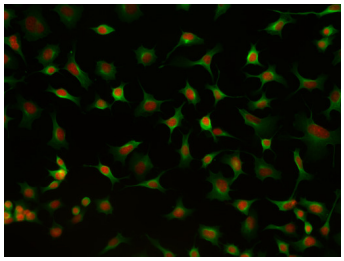
To evolve such solutions (or models), we require data that is manually annotated by a human specialist. However, the task of collecting and labeling data is expensive and requires significant time and effort. Thus, our goal is to improve the evolution of well-performing CGP solutions for biomedical image processing when using only a small amount of labeled data during evolution. Our reason is that the amount of data is extremely limited since biological data is composed of complex images that are costly to collect and annotate. One of the popular approaches available to reduce the amount of data used in Machine Learning is called Active Learning (AL) [7], [8]. The main idea of AL is to identify samples from a larger dataset to use during training and create models of equivalent performance with less overhead [9].

Therefore, we argue that AL is a potential methodology to be applied to CGP in the biomedical domain. The overall idea is to sample the most informative data points from an initial dataset, training on a smaller, selected dataset while achieving similar performance with faster convergence and to avoid overfitting by improving generality [7] with less overhead [9]. When using AL, we assume that the most expensive step in the application is the labeling process [10].

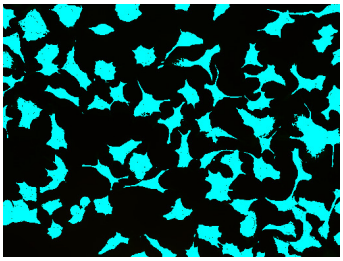
In this contribution, we study how extracting a small set of data samples from the CELLPOSE dataset affects the performance of CGP in comparison with the results from CGP using the whole dataset. Our contributions can be summarized as follows:

(1) We introduce AL to CGP for biological image segmentation to reduce the number of image points needed to evolve efficient programs.
(2) We show that AL boosts the convergence speed of CGP.
(3) We present arguments that AL encourages CGP to reduce the number of active nodes throughout evolution.

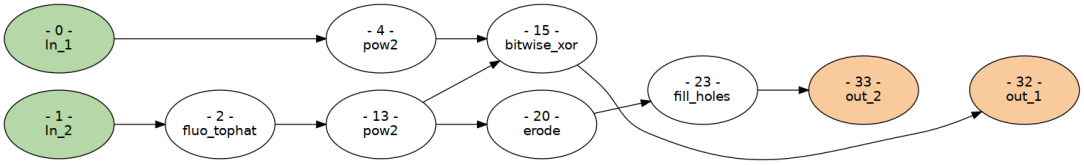
The paper is organized as follows: Section II introduces the necessary background. Section III explains relevant concepts.



(a) Image to segment.



(b) Segmentation mask.




(c) The 2 outputs are provided to the Watershed Transform to generate the mask in (b).

Fig. 1: One image from the CELLPOSE dataset, the final segmentation mask generated to this image from CGP with Active Learning, and the graph used to segment this image.

Section IV gives the experimental setup. Section V presents the experimental results of our analysis. Finally, Section VI concludes the paper and discusses further research.

## II. Preliminaries

Most work in the area of Computer Vision tasks uses black-box approaches, such as artificial Deep Neural Networks. However, these models have proven to be difficult for humans to analyze and interpret their outputs and they require a high amount of training data [3], [11]. One way to complement these black-box methods is to use inherently explainable methods. Genetic Programming is of this class and among such methods, we highlight Cartesian Genetic Programming, an evolutionary computation algorithm that evolves easy-to-interpret programs[1] [3], [4], [6], [12]–[15]. Additionally, GP approaches require less training data because, while Deep Neural Networks need to optimize image filters from scratch at the pixel level, GP uses high-level mathematical functions already designed and optimized by human engineers to perform specific tasks.

### A. Cartesian Genetic Programming

Cartesian Genetic Programming is a Genetic Programming variant [5] specialized in evolving graph phenotype. Such graphs are often direct and acyclic and are indexed by Cartesian coordinates. Evolution defines how to connect the nodes of the graphs and the instructions or functions of each node.

CGP has been successfully applied in multiple domains [12]–[14]. Specifically, CGP has been applied in Computer Vision tasks such as for controlling agents to play ATARI games [6], and in image processing tasks, such as biomedical image segmentation and object detection in robotics [3], [4], [15]. The solutions in CGP are generally optimized by using the (1+$\lambda$) Evolutionary Algorithm, although any other evolutionary algorithm could be used. Initially, a population of $\lambda$ individuals is randomly generated and evaluated on the problem in question. Then, evaluation is conducted by first generating the programs from the graphs and then measuring the performance of such programs on the task considered. The solution with the highest performance is maintained to the next generation step, influencing the next $\lambda$ individuals created via mutation. This process is repeated until a stop criterion is reached. For more information, refer to [5], [6], [14].

[1] in comparison to Deep Learning models, as most of GP variants.

### B. Active Learning in CGP

Generally, Active Learning (AL) is used with Deep Learning, one of the most frequently used methods for processing biomedical data [7], [16] with most of the work on AL found in the literature focuses on finding the metric that leads the model to the highest performance [7]. Interestingly, some papers suggest random sampling as a strong baseline [17], [18].

In the domain of GP, the efficacy of Active Learning for symbolic regression tasks has been widely demonstrated in a diverse group of research, from works that focus on reducing the number of such evaluations [19] and on creating smaller, balanced datasets by recursively keeping the most 'meaningful' exemplars [20] to studies on improving the rate and consistency at which well-performing solutions are found while reducing the required number of training samples [8], [21]. For classification problems, Hamida et al. [22], [23] showed how different sampling methods studied across the years affect the performance of GP. Yet, we did not find works that combine AL and GP in the biomedical domain.

### C. CGP implementation

The CGP implementation we use is based on [3], [4], a modular Cartesian Genetic Programming system to generate

**Algorithm 1** Data sampling via Active learning in CGP

1: Initialize (1 + $\lambda$) CGP.
2: **while** *Stopping criterion is not met* **do**
3: **Get** new dataset $S$ via a selection method.
4: **Control** size of $S$.
5: **Evolve** CGP with the dataset.
6: **Update** elite solution.
7: **Update** the fitness values associated with selected images given the new elite.
8: **If** restart is true, save elite in external archive, restart CGP and dataset $S$.
9: **end while**

our programs for Computer Vision tasks. This system introduces the notion of non-evolvable nodes which are functions not subjected to optimization of the syntactic graph. Through this algorithm, the image processing stack optimizes the order of functions and their parameters resulting in an image processing algorithm similar to a human-designed one, since they are based on established functions for image processing.

Here, we use image processing functions mostly from OpenCV and scikit-image, which apply programs directly to images. "Active" nodes are the subset of nodes present in the final program, since they are connected to the output of the program graph. Other nodes without connections to the output are called "inactive" nodes. The outputs of the program can be taken from any node, which is defined during evolution.

Moreover, in this implementation, the end nodes of CGP are connected to a fixed endpoint. This endpoint is useful since it allows CGP to have insight given by a Computer Vision expert. We use Watershed Transform [24] as the endpoint. Thus, our CGP has 2 outputs, corresponding to the mask and markers necessary for the Watershed Transform endpoint.

## III. Data Sampling

The CGP with AL (AL-CGP) template we propose for instantiating and designing sampling method variants is shown in Algorithm 1. The main difference to standard CGP is that instead of using a fixed training dataset during evolution, our template uses a smaller dataset that is selected during evolution, given the fitness values of the highest performing solution on the images.

### A. Sampling method

We sample the subset of images to be part of the dataset used in evolution via a sampling method. At each step, two images are selected: one image with low fitness, and another with high fitness. The idea is to include (a) an image that challenges the current elite solution and (b) another image on which the elite solution performed well, ensuring valuable information is preserved. By addressing both challenges and successes, we ensure that the elite solution is exposed to diverse images, preventing the loss of valuable information and promoting generalizability.

**Algorithm 2** Roulette selection

1: Inputs: *fitness* values associated with each image.
2: **Sample** one image given a probability based on the *fitness* values.
3: **Sample** another image given a probability based on the opposite *fitness* values ($1 - fitness$).
4: **Return** Two images

*a) Roulette sampling:* Sampling is done based on weighted probabilities given by the fitness values associated with each image. These values change every time that an image is evaluated, when this image is introduced to the dataset, or if it is on the training set $S$. Algorithm 2 shows the pseudocode. As said above, two images are selected each time. Either via direct fitness values for the group (1) or the inverse fitness values for group (2).

*b) Random sampling:* We use random sampling as a baseline. Sampling is done using uniformly distributed values to select one image at a time.

To decide when to add new images to the dataset, we follow the work of Hamida et al. [22] and use a deterministic sampling frequency, $f$. This method follows this equation $f = int((C * generation)^{\alpha})$. Thus, AL-CGP adds images at the generation $gen$, when $gen$ mod $f = 0$.

### B. Controlling the size of the dataset

Adding images with Active Learning might increase the size of the dataset unreasonably. Here, we choose to control the size of the dataset by simply removing two images from the dataset randomly. This happens every time that the size of the dataset is above the maximum limit size.

### C. Evaluation metric - Average Precision (AP)

We follow the work in [25] that defines $AP = TP/(TP + FP + FN)$, where *TP* mean true positives, *FP* mean false positives and *FN* mean false negatives. We use AP as our fitness function with a threshold of 0.5 to determine the true positives of the predicted mask.

### D. Stopping criterion - Images processed

Our goal is to study the effects of sampling images from the dataset during the evolution of CGP. Thus, we have a different number of images processed by AL-CGP and standard CGP, at each generation. Given the different sizes of datasets used during evolution, we cannot use directly the number of evaluations or generations as our stop criterion. Thus, we use the number of images processed during evolution as the stopping criterion. This criterion does not discriminate if there is repetition of data points.

### E. CGP function library

Table I describes the basic function library used in this work and their related arity[2]. These operators are functions from the OpenCV Python package and are fixed to all CGP variants.

[2]The number of parameters needed by a given function.

TABLE I: Description of the function library used in CGP.

| Function | Arity | Function | Arity |
|---|---|---|---|
| Max | 2 | Min | 2 |
| Mean | 2 | Add | 2 |
| Subract | 2 | Bitwise_not | 1 |
| Bitwise_or | 2 | Bitwise_and | 2 |
| Bitwise_and_mask | 2 | Bitwise_xor | 2 |
| sqrt | 1 | pow2 | 1 |
| exp | 1 | log | 1 |
| median_blur | 1 | gaussian_blur | 1 |
| laplacian | 1 | sobel | 1 |
| robert_cross | 1 | canny | 1 |
| sharpen | 1 | gabor | 1 |
| abs_diff | 1 | abs_diff2 | 2 |
| fluo_tophat | 1 | ref_diff | 1 |
| erode | 1 | dilate | 1 |
| open | 1 | close | 1 |
| morph_gradient | 1 | morph_tophat | 1 |
| morph_blackhat | 1 | fill_holes | 1 |
| remove_small_objects | 1 | remove_small_holes | 1 |
| threshold | 1 | threshold_at_1 | 1 |
| distance_transform | 1 | distance_transform_and_thresh | 1 |
| inrange_bin | 1 | inrange | 1 |

TABLE II: Parameters used.

| Parameter | Value |
|---|---|
| Number of nodes | 30 nodes |
| Offspring size $\lambda$ | 5 |
| Inputs | 2, $\alpha$-tubulin and DAPI channels |
| Outputs | 2, mask and markers |
| Mutation of function nodes | 0.15 |
| Mutation of outputs nodes | 0.20 |
| Images processed | 60,000 |
| Generations - CGP | 1,000 |
| Generations - AL-CGP | $>$ 1,000 |
| Size dataset used in CGP | 10 unique images, sampled before the run |
| Size dataset used in AL-CGP | from 2 to 10, sampled during the run |
| When to add data (AL-CGP only) | $gen$ mod $int((C * gen)^{\alpha})$ == 0<br>C = 2<br>$\alpha$ = 0.5 |

## IV. Experimental Setup

To verify if AL can be used to identify samples from dataset to use during training and create efficient programs, we study how to use AL to sample a small, yet informative set of images samples to improve the performance of CGP. For that, we compare variations of AL-CGP with standard CGP.

### A. Parameter setting

AL-CGP builds the dataset to be used during evolution using Active Learning and standard CGP uses a sample of 10 images randomly, the minimal amount of data found for CGP to achieve high-performance values, see [3], while AL-CGP builds the training dataset using the methods in Section III. Since we want to use only a small amount of labeled data during evolution, we limit the maximum training dataset size to 10 images, as chosen in [3]. We run all CGP variants with the parameters shown in Table II. For the standard CGP parameters, we use the same values as in [3]. AL-CGP builds the dataset for training using the deterministic sampling frequency sampling (Section III-A) and we used the same parameters as used by [22]. More work is needed in tuning the parameters for the CGP variants. We run 30 independent runs of each variant for statistical purposes.

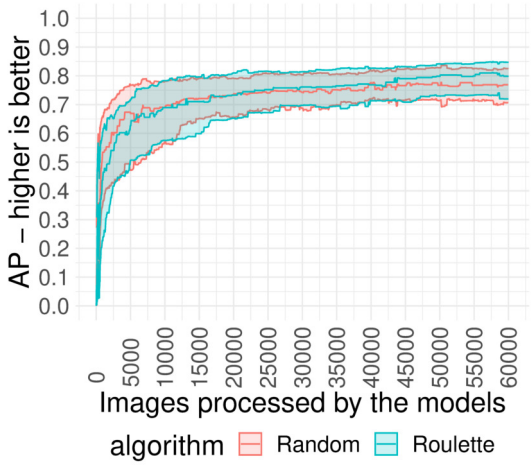



Fig. 2: Median AP values (shaded areas show the standard deviation) of elite solutions of AL-CGP with 2 sampling methods. Since the datasets vary in size, we show the number of images processed. The roulette sampling leads to higher performance at the end.

We test if using restarts to explore different areas of the search space leads to more efficient programs, complementing the faster convergence enhanced by AL [7]. Here, restarts are done at different steps of evolution by clearing the training dataset and randomly initializing solutions. Thus, we verify impacts in the performance of multiples starts of AL-CGP: 2, 5, 10, and 20 times.

### B. Performance comparison

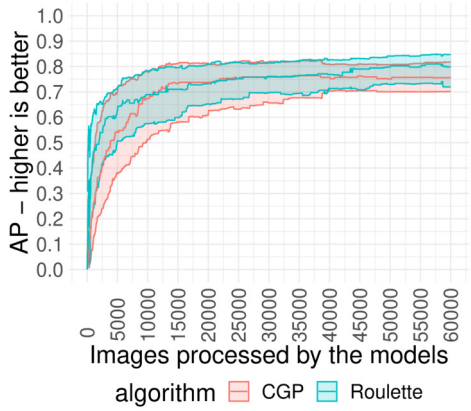
We compare the results of the different CGP variants based on convergence behavior and statistical analysis (Student t-test). The convergence behavior is used to analyze the AP performance at multiple points of the evolution to investigate the performance of different CGP variants. For the statistical analysis, the student t-test was used with a significance level of $\alpha = 0.05$. For fair comparison and facilitating rapid prototyping, we use the CELLPOSE dataset [25] consisting of 100 fluorescently labeled protein images of cultured neuroblastoma cells with phalloidn FITC and DAPI nuclear stain. For fair comparisons, we follow the work in [25], where the data is split into 89 images for training and 11 for testing.

### C. Reproducibility

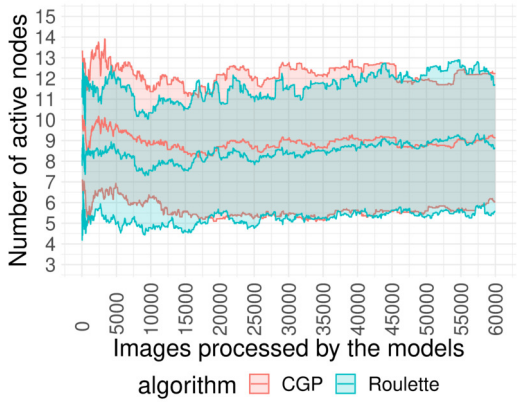
Relevant data and code are available at: https://zenodo.org/records/10869851. We run the experiments on HPC resources on the OLYMPE supercomputer, a SEQUANA (ATOS-BULL) computing cluster with a power of 1.365 Pflop/s Peak, equipped with Intel Skylake 6140 processors.

## V. Experimental Results

First, we validate the sampling mechanism for AL-CGP described in Section III. Figure 2 shows the median AP convergence results. Using our sampling mechanism has a marginal advantage over randomly choosing images, however

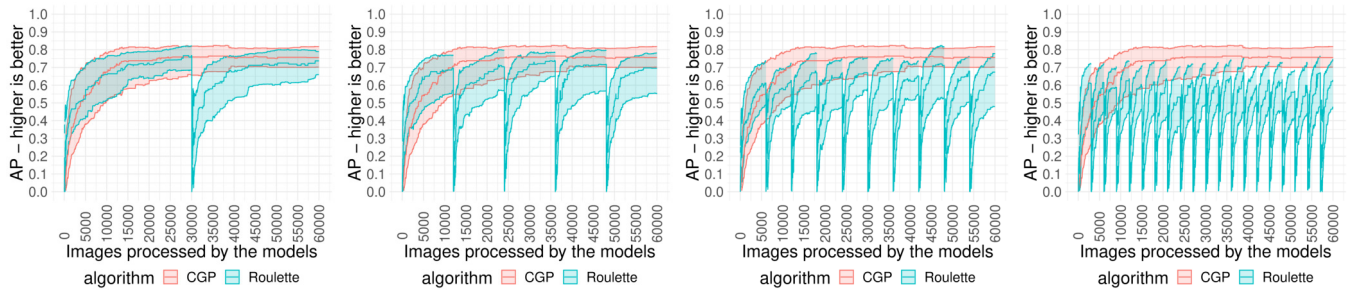



(a) Median convergence performance. AL-CGP is faster at the beginning and better performing at the end.




(b) Mean number of active nodes. AL-CGP finds smaller programs at the beginning of evolution.

Fig. 3: Metric values (shaded areas show the standard deviation) of the elite solutions of AL-CGP versus CGP. AL-CGP uses a dataset that changes in size, hence, we show the number of images processed to provide a fair comparison.




(a) 2 starts. (b) 5 starts. (c) 10 starts. (d) 20 starts.

Fig. 4: Median AP performance convergence (shaded areas show the standard deviation) of AL-CGP with different restart periods versus CGP. The size of the datasets used by the different AL-CGP restarting strategies varies, thus we show the number of images processed instead of generations. Most of the increments in performance are at the beginning of the search.

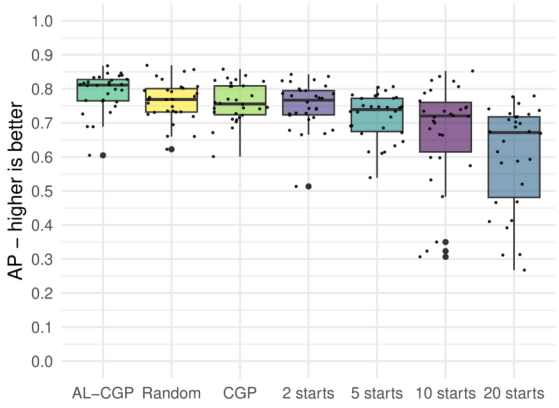



Fig. 5: Boxplots of AP values of the elite solutions found by different CGPs. AL-CGP performs better than CGP. Using 2 starts leads to higher median values than CGP.

this superiority does not reach statistical significance. This small improvement with our method suggests a subtle impact on dynamically modifying the training data based on information from the elite (best performing) solution, as opposed to random sampling. Given the higher performance of AL-CGP with roulette, we move on to comment on the differences in performance exclusively between AL-CGP with roulette sampling versus standard CGP.

### A. *Convergence*

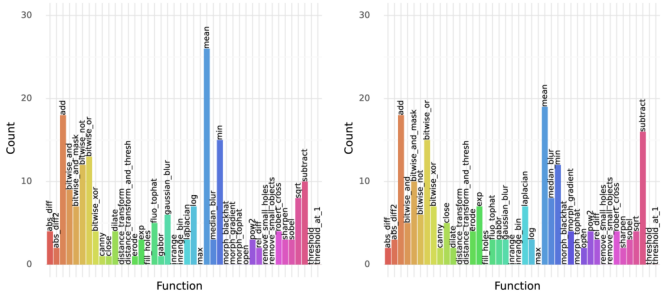
Figure 3a shows the median convergence behavior AL-CGP versus standard CGP (shaded areas show the standard deviation). Using our sampling method, AL-CGP converges faster than CGP to high values, but eventually, CGP achieves a performance slightly below that of AL-CGP. We see that using AL leads to faster convergence, but there is still a gap to be filled at the end of the execution when compared with the state-of-the-art Deep Learning performance with DL or CGP [3]. That said, the efficacy of AL-CGP lies in its ability to diminish the amount of required training data while delivering comparable results, as shown here.

### B. *Number of active nodes*

Figure 3b illustrates the mean number of active nodes (shaded areas show the standard deviation) over the iterations

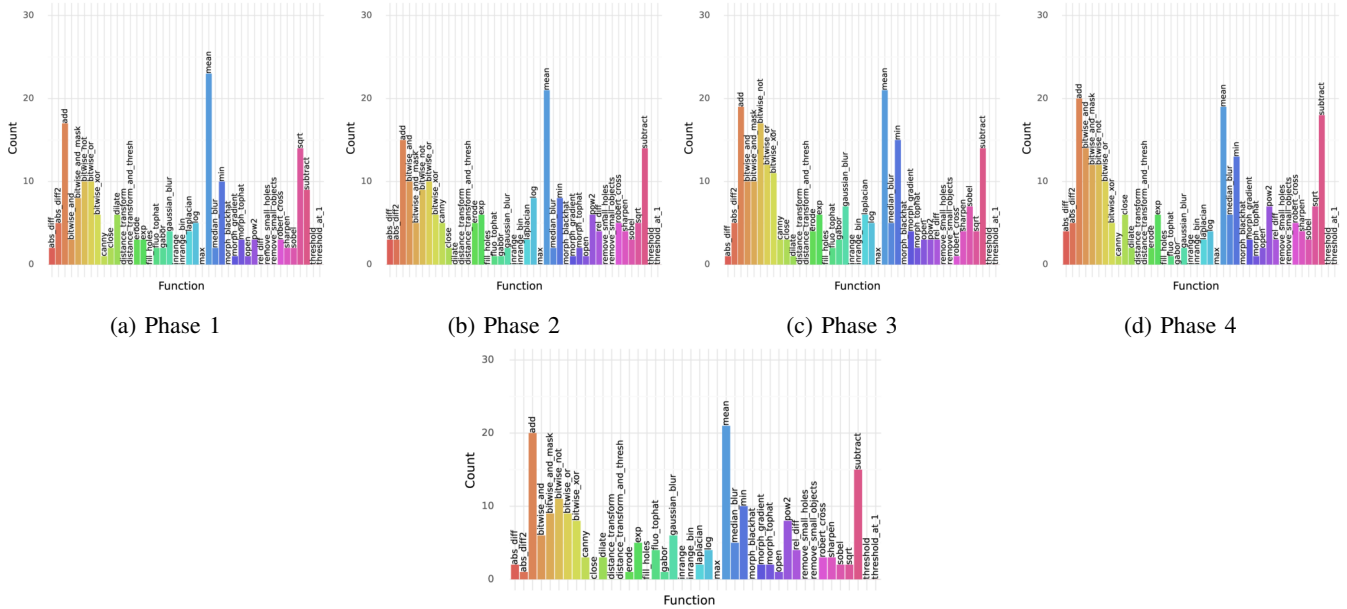

(a) 2 starts. (b) 5 starts. (c) 10 starts. (d) 20 starts.

Fig. 6: Mean number of nodes of AL-CGP with different restart periods versus standard CGP over the images processed, since the size of the datasets used in training is different. AL-CGP progresses the search with smaller programs.




(a) Phase 1 (b) Phase 2

Fig. 7: Number of functions used in AL-CGP with 2 start phases. The functions “mean”, “add” and “subtract” are frequently present. The distribution of functions is different depending on the phase, which suggests diverse elite solutions.




(a) Phase 1 (b) Phase 2 (c) Phase 3 (d) Phase 4

(e) Phase 5

Fig. 8: Functions used in AL-CGP with 5 start phases. The functions “mean”, “add” and “subtract” are present are used in many stages of the search process. The distribution of functions varies depending on the phase.

TABLE III: Statistical analysis between AL-CGP and CGP. We see statistically significant differences at the begining and the end of evolution.

| Images Used | p-value |
|---|---|
| 2,500 | 0.018 |
| 5,000 | 0.049 |
| 7,500 | 0.221 |
| 10,000 | 0.281 |
| 45,000 | 0.389 |
| 47,500 | 0.074 |
| 50,000 | 0.028 |
| 52,500 | 0.024 |
| 55,000 | 0.034 |
| 57,500 | 0.04 |
| 60,000 | 0.137 |

of the runs of CGP in comparison to AL-CGP. CGP consistently finds larger programs during the search progress, but around the first third of the process, the number of active nodes starts to decrease, aligning with the elite solution's performance convergence. On the other hand, AL-CGP, continually evolves smaller programs, suggesting that the inclusion of AL aids CGP to generate simpler and more general programs. Towards the end of the search, both methods converge to programs of similar and reduced sizes. Therefore, we argue that AL-CGP increases program interpretability, as evidenced by the discovery of more concise programs. Overall, our findings suggest that the AL-CGP contributes to performance convergence, in accord with the literature, and enhances one of the main features of genetic programming: the ability to generate interpretable programs.

### C. Statistical Analysis

The student t-test is conducted across various numbers of processed images, and the outcomes are illustrated in Table III. AL-CGP demonstrates statistically significant differences compared to CG during the initial one-third of the search progression until fewer than 7,500 images are processed. Subsequently, there are not any statistically significant differences among the results obtained by the CGP variants. This pattern changes again after processing more than 47,500 images, where a statistically significant difference appears. However, towards the conclusion of the search process, the performance is not statistically significant. This variability could be associated with a potential shift in how CGP processes images during evolution.

### D. Restart

Given the fast convergence of the solutions shown by AL-CGP, we conjecture that restarting the training data and randomly initializing the solutions could benefit the performance of AL-CGP even further. We argue that restarting the search before the end of the search progress could lead to a higher exploration of the search space to regions that might help the algorithm find high-performing solutions.

Figure 4 shows the convergence plots of AL-CGP when using restart variants and Figure 5 displays the performancee of elite solutions found by AL-CGP with restart. We select this elite from the start phase with the highest median value for each restarting frequency. We can see that using restarts reduces the overall performance as the frequency increases. That said, we can also see that the best-performing solution is found at different phases, which suggests that the initial solution from which evolution starts the search process impacts the final solution generated. In addition, the difference in performance between restarting halfway through the search progress or not restarting is small, indicating that most of the increments in performance happen during the first half of the search and that more sophisticated restart methods might be an interesting path to explore for enhancing the performance of AL-CGP.

Moving to Figure 6, we can confirm that using AL results in a CGP model with a lower number of active nodes. The number of nodes is only similar between the restart strategies and standard CGP at the beginning of each start phase. Of course, the impact of this effect varies given the restart frequency and is clearer in the early stages of the evolution. We think that the reason for such an effect is that frequent changes in the dataset force CGP to find more generalizing solutions, instead of finding solutions overfitted to the images processed.

Finally, we comment on the distribution of functions used by the elite solutions of the different runs, just before restarting. This is shown in Figures 7 and 8. We focus our analysis on AL-CGP with 2 and 5 starting phases to reduce the amount of information analyzed. We can see that the distribution varies according to the phase in consideration for both restart strategies. Some functions, especially aggregation functions, such as "mean", "add" and "subtract" are frequently present, independently of the phase in consideration. We understand that doing several restarts during the search can lead to a more diverse set of elite solutions, given the distribution of used functions.

## VI. Conclusions

We studied using ideas from Active Learning in GP and we focused on how sampling biomedical images from the available dataset affects the performance of CGP in comparison to using the whole dataset. We found that the convergence speed, measured by the average precision metric, considerably improves until it converges at around the first third of the search process, to similar values as standard CGP. Furthermore, we observed that using AL reduces the number of active nodes throughout evolution, but the effects and possible implications of such reduction are yet to be studied. We also saw that using restarts led to more diverse individuals, as shown by the distribution of functions used.

Future work could utilize diverse individuals from multiple starting points to generate an ensemble and then use disagreement or uncertainty of the ensemble as a metric for selecting new data to see if these ideas from Active Learning via such metrics sampling could boost performance. One limitation of our method is that it assumes that a ground truth is known for all the images. In scenarios with a collection of unlabeled

images, our method can only be applied to ask an expert to label these unlabeled data. Thus, we think that including unsupervised learning techniques, is a possible direction to increase the versatility and efficacy of our work in real-case scenarios.

This work represents a new stride towards achieving a broader objective in the construction of an interactive learning system within the domain of biomedical image analysis. We anticipate that the combination of CGP, along with efficient active learning, has the potential to catalyze the development of applications where a human expert annotates specific images or areas of large images upon request. The images or areas for annotation would be suggested by the learning algorithm based on its current requirement for annotated data, such as specific cell types, color diversity, shapes, or others. Our goal is to foster a collaborative effort between our CGP learner and a human expert, guiding the learning procedure through the complexities of the images to be analyzed.

## ACKNOWLEDGMENT

This work is funded by the Laboratoire d'Excellence Toulouse Cancer TOUCAN, contract ANR11-LABX. This work is supported by the AI Interdisciplinary Institute ANITI, funded by the French program "Investing for the Future – PIA3" under Grant agreement no. ANR-19-PI3A-0004.

## REFERENCES

[1] A. Esteva, K. Chou, S. Yeung, N. Naik, A. Madani, A. Mottaghi, Y. Liu, E. Topol, J. Dean, and R. Socher, "Deep learning-enabled medical computer vision," *NPJ digital medicine*, vol. 4, no. 1, p. 5, 2021.

[2] S. Deng, X. Zhang, W. Yan, E. I.-C. Chang, Y. Fan, M. Lai, and Y. Xu, "Deep learning in digital pathology image analysis: a survey," *Frontiers of medicine*, vol. 14, pp. 470–487, 2020.

[3] K. Cortacero, B. McKenzie, S. Müller, R. Khazen, F. Lafouresse, G. Corsaut, N. Van Acker, F.-X. Frenois, L. Lamant, N. Meyer *et al.*, "Evolutionary design of explainable algorithms for biomedical image segmentation," p. 7112, 2023.

[4] Y. Lavinas, K. Cortacero, and S. Cussat-Blanc, "Evolving graphs with cartesian genetic programming with lexicase selection," in *Proceedings of the Companion Conference on Genetic and Evolutionary Computation*, ser. GECCO '23 Companion. New York, NY, USA: Association for Computing Machinery, 2023, p. 1920–1924. [Online]. Available: https://doi.org/10.1145/3583133.3596402

[5] J. F. Miller, "An empirical study of the efficiency of learning boolean functions using a cartesian genetic programming approach," in *Proceedings of the 1st Annual Conference on Genetic and Evolutionary Computation - Volume 2*, ser. GECCO'99. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1999, p. 1135–1142.

[6] D. G. Wilson, S. Cussat-Blanc, H. Luga, and J. F. Miller, "Evolving simple programs for playing atari games," in *Proceedings of the Genetic and Evolutionary Computation Conference*, ser. GECCO '18. New York, NY, USA: Association for Computing Machinery, 2018, p. 229–236. [Online]. Available: https://doi.org/10.1145/3205455.3205578

[7] M. Gaillochet, C. Desrosiers, and H. Lombaert, "Active learning for medical image segmentation with stochastic batches," *Medical Image Analysis*, vol. 90, p. 102958, 2023. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1361841523002189

[8] N. Haut, W. Banzhaf, and B. Punch, "Active learning improves performance on symbolic regression tasks in stackgp," in *Proceedings of the Genetic and Evolutionary Computation Conference Companion*, ser. GECCO '22. New York, NY, USA: Association for Computing Machinery, 2022, p. 550–553. [Online]. Available: https://doi.org/10.1145/3520304.3528941

[9] D. A. Cohn, Z. Ghahramani, and M. I. Jordan, "Active learning with statistical models," *Journal of artificial intelligence research*, vol. 4, pp. 129–145, 1996.

[10] B. Settles, "Active learning literature survey," University of Wisconsin–Madison, Computer Sciences Technical Report 1648, 2009.

[11] C. Rudin, "Stop explaining black box machine learning models for high stakes decisions and use interpretable models instead," *Nature machine intelligence*, vol. 1, no. 5, pp. 206–215, 2019.

[12] A. M. Ahmad, G. M. Khan, S. A. Mahmud, and J. F. Miller, "Breast cancer detection using cartesian genetic programming evolved artificial neural networks," in *Proceedings of the 14th Annual Conference on Genetic and Evolutionary Computation*, ser. GECCO '12. New York, NY, USA: Association for Computing Machinery, 2012, p. 1031–1038. [Online]. Available: https://doi.org/10.1145/2330163.2330307

[13] M. Suganuma, S. Shirakawa, and T. Nagao, *Designing Convolutional Neural Network Architectures Using Cartesian Genetic Programming*. Singapore: Springer Singapore, 2020, pp. 185–208. [Online]. Available: https://doi.org/10.1007/978-981-15-3685-4_7

[14] J. Biau, D. Wilson, S. Cussat-Blanc, and H. Luga, "Improving image filters with cartesian genetic programming." in *IJCCI*, 2021, pp. 17–27.

[15] S. Harding, J. Leitner, and J. Schmidhuber, *Cartesian Genetic Programming for Image Processing*. New York, NY: Springer New York, 2013, pp. 31–44. [Online]. Available: https://doi.org/10.1007/978-1-4614-6846-2_3

[16] V. Nath, D. Yang, B. A. Landman, D. Xu, and H. R. Roth, "Diminishing uncertainty within the training pool: Active learning for medical image segmentation," *IEEE Transactions on Medical Imaging*, vol. 40, no. 10, pp. 2534–2547, 2021.

[17] A. Kirsch, J. van Amersfoort, and Y. Gal, "Batchbald: Efficient and diverse batch acquisition for deep bayesian active learning," in *Advances in Neural Information Processing Systems*, H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, Eds., vol. 32. Curran Associates, Inc., 2019. [Online]. Available: https://proceedings.neurips.cc/paper_files/paper/2019/file/95323660ed2124450caaac2c46b5ed90-Paper.pdf

[18] A. Casanova, P. O. Pinheiro, N. Rostamzadeh, and C. J. Pal, "Reinforced active learning for image segmentation," in *International Conference on Learning Representations*, 2020. [Online]. Available: https://openreview.net/forum?id=SkgC6TNFvr

[19] C. Gathercole and P. Ross, "Dynamic training subset selection for supervised learning in genetic programming," in *Parallel Problem Solving from Nature — PPSN III*, Y. Davidor, H.-P. Schwefel, and R. Männer, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 1994, pp. 312–321.

[20] E. Vladislavleva, G. Smits, and D. den Hertog, "On the importance of data balancing for symbolic regression," *IEEE Transactions on Evolutionary Computation*, vol. 14, no. 2, pp. 252–277, 2010.

[21] N. Haut, B. Punch, and W. Banzhaf, "Active learning informs symbolic regression model development in genetic programming," in *Proceedings of the Companion Conference on Genetic and Evolutionary Computation*, ser. GECCO '23 Companion. New York, NY, USA: Association for Computing Machinery, 2023, p. 587–590. [Online]. Available: https://doi.org/10.1145/3583133.3590577

[22] S. Ben Hamida, H. Hmida, A. Borgi, and M. Rukoz, "Adaptive sampling for active learning with genetic programming," *Cognitive Systems Research*, vol. 65, pp. 23–39, 2021. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1389041720300541

[23] H. Hmida, S. B. Hamida, A. Borgi, and M. Rukoz, "Sampling methods in genetic programming learners from large datasets: A comparative study," in *Advances in Big Data*, P. Angelov, Y. Manolopoulos, L. Iliadis, A. Roy, and M. Vellasco, Eds. Cham: Springer International Publishing, 2017, pp. 50–60.

[24] S. Beucher and F. Meyer, "The morphological approach to segmentation: the watershed transformation," in *Mathematical morphology in image processing*. CRC Press, 2018, pp. 433–481.

[25] C. Stringer, T. Wang, M. Michaelos, and M. Pachitariu, "Cellpose: a generalist algorithm for cellular segmentation," *Nature methods*, vol. 18, no. 1, pp. 100–106, 2021.