Dynamically Sampling biomedical Images For Genetic Programming

Yuri Lavinas
lavinas.yuri.xp@alumni.tsukuba.ac.jp
IRIT - CNRS UMR5505, University of Toulouse
Toulouse, France

Nathan Haut
hautnath@msu.edu
Michigan State University
Michigan, United States of America

William Punch
punch@msu.edu
Michigan State University
Michigan, United States of America

Wolfgang Banzhaf
banzhafw@msu.edu
Michigan State University
Michigan, United States of America

Sylvain Cussat-Blanc
sylvain.cussat-blanc@irit.fr
IRIT - CNRS UMR5505, University of Toulouse
Toulouse, France

ABSTRACT

In this contribution we study how to effectively evolve programs tailored for biomedical image segmentation by using an Active Learning approach in Cartesian Genetic Programming (CGP). Active Learning allows to dynamically select training data by identifying the most informative next image to add to the training set. We study how different metrics for selecting images under active learning impact the searchability of CGP. Our results show that datasets built during evolution with active learning improve the performance of Cartesian GP substantially. In addition, we found that the choice of the particular metric used for selecting which images to add heavily impacts convergence speed. Our work shows that the right choice of the image selection metric positively impacts the effectiveness of the evolutionary algorithm.

CCS CONCEPTS

• **Computing methodologies → Bio-inspired approaches**.

KEYWORDS

Cartesian genetic programming, image processing, active learning, data sampling

ACM Reference Format:

Yuri Lavinas, Nathan Haut, William Punch, Wolfgang Banzhaf, and Sylvain Cussat-Blanc. 2024. Dynamically Sampling biomedical Images For Genetic Programming. In *Genetic and Evolutionary Computation Conference (GECCO '24 Companion), July 14–18, 2024, Melbourne, VIC, Australia.* ACM, New York, NY, USA, 4 pages. https://doi.org/10.1145/3638530.3654202

1 INTRODUCTION

The field of biomedical image analysis has recently been revolutionized by Neural Networks and in particular Deep Learning (DL) approaches, as has happened in most of the Computer Vision domain. Deep Neural Networks perform very well in many image classification and/or segmentation tasks within the medical domain such as in dermatology, radiology and pathology [11, 12], and in some cases even beat human experts. However, DL approaches have two main drawbacks. First, they are considered "blackbox" approaches: the decisions taken by deep Neural Networks are often hard to explain and it might be close to impossible to describe to a human expert, an important obstacle in critical human-life applications such as medicine. Second, DL approaches require large amounts of annotated data, a time-consuming and expensive task that must be performed by experts who are often very busy, making it hard to obtain sufficient amounts of data available for use.

Recent work showed that Cartesian Genetic Programming (CGP) is an effective approach to address the above-mentioned limitations of DL [10, 21], with results competitive to DL. CGP evolves solutions based on a set of mathematical functions that process given inputs to produce an expected output. The phenotype of a CGP program can be represented as a graph which often is evolved using a $(1+\lambda)$ evolutionary strategy [23]. One of the main benefits of CGP is the use of a fixed-length integer-based genome to encode the functional graphs, reducing the bloat effect encountered in many tree GP approaches. Therefore, small programs can be evolved which facilitates interpretability [28]. 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.

Minimizing the amount of necessary data for evolving effective programs in such a task poses an important challenge. In this context, our goal in this work is to improve the evolution of CGP when processing biomedical images. In our recent work [22], we have shown that AL can help CGP to use a small amount of training data, without loss in performance. Here, we investigate whether the effects of AL are amplified in the absence of limitations on training data size, thereby potentially enhancing its impact on CGP.

It is now well established from a variety of studies that Active Learning (AL) [15] methods can provide performance improvements in image processing by using information from evolution to iteratively build a training dataset.

GECCO '24 Companion, July 14–18, 2024, Melbourne, VIC, Australia

ACM ISBN 979-8-4007-0495-6/24/07.
https://doi.org/10.1145/3638530.3654202

We here argue that we can incorporate ideas from AL methodologies and apply them to CGP to generate more data-efficient programs for biomedical image segmentation. We focus our efforts on developing a method to sample informative images, selected during evolution, resulting in a smaller training set than without these methods. We expect that CGP with AL can achieve higher performance with faster convergence than traditional CGP.

2 PRELIMINARIES

CGP is a Genetic Programming variant [23] specialized in evolving graph phenotypes. 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 function of each node.

CGP has been successfully applied in multiple domains [1, 8, 26]. Specifically, CGP has been applied in Computer Vision tasks such as for controlling agents to play ATARI games [28], and in image processing tasks, such as biomedical image segmentation and object detection in robotics [10, 14]. Solutions in CGP are generally optimized by using the (1+λ) Evolutionary Strategy, although any other evolutionary algorithm could be used. Initially, a population of λ 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 their performance on the task considered. The solution with the highest performance is maintained to the next generation step, influencing the next λ individuals created via mutation. This process is repeated until a termination criterion is reached. For more information, the reader is referred to [8, 23, 28].

GP has been widely used in the biomedical domain, and as Khan et al. stated, GP is "often applied for classifying cancerous and non-cancerous cells" [19]. In particular, one can find GP contributions that involve feature extraction [3, 4, 7, 18, 27] or that involve image classification [2, 5, 29].

2.1 Dynamic Data Sampling

Generally, dynamic data sampling is one of the most important components of Active Learning (AL) methods. AL is frequently used in Deep Learning in the context of processing biomedical data [13, 24] with most of the work on AL in the literature focusing on finding the metric that leads models to the highest performance [13]. Interestingly, there are papers that suggest random sampling as a strong baseline [9, 20].

In the domain of GP, the efficacy of AL for symbolic regression tasks was shown to improve the rate and consistency at which well-performing solutions are found, while reducing the number of required training instances [15, 16]. For classification problems, Hamida et al. [6, 17] has shown how different sampling methods affect the performance of GP. In our recent work [22], we have shown that AL can help CGP to use a small amount of training data, without loss in performance and that CGP with AL benefits from restarts, that lead to more diverse individuals.

3 DATA SAMPLING IN CARTESIAN GP

The CGP with **Dyn**amic data sampling (dyn-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 different metrics (explained below).

Algorithm 1 Dynamically sampling in CGP (dyn-CGP)

```
Initialize (1 + λ) CGP.
while Stopping criterion is not met do
    Evolve CGP with the training dataset.
    Update elite solution.
    Calculate uncertainty metric on the images not in use.
    Add image to the training dataset given metric.
end while
```

We sample the subset of images to be part of the training dataset used in evolution via a sampling mechanism and compare different methods. Two of these mechanisms sample data based on uncertainty-related information, while the other mechanism samples images based on values taken from a uniform distribution.

To calculate uncertainty in CGP, we use a group of parallel CGP runs, executed in parallel. Uncertainty-based AL utilizes this group of diverse runs of programs to search different areas of the search space, the diversity of the models then allows their disagreement to be used as an uncertainty measure to select new training data where uncertainty is maximized. The idea is that selecting data where uncertainty is high will lead to the selection of data that will be most informative to the current models in training. [16]. In this study, the parallel runs are only employed to estimate uncertainties, while for generating segmentation masks and all metrics related to the performance of dyn-CGP we choose from the parallel runs one program, one with the highest fitness value on the training data.

Uncertainty. Uncertainty is computed by counting the pairwise differences between the pixels of the predicted masks for each program on a given image. If two pixels have a different label between two masks, an uncertainty of 1 is assigned to that pixel. If the labels are the same, an uncertainty of 0 is applied. The overall uncertainty for a given image is the average uncertainty of all the pairwise uncertainties between each program. The metric ranges from 0 to 1.

Weighted Uncertainty. The weighted uncertainty works similarly to the basic uncertainty mentioned above with one key difference. Here, we double the uncertainty measurement if one pixel is labeled as 0 and the other pixel is labeled with a non-zero value. This assumes higher uncertainty should be assigned if models disagree on whether a pixel is foreground (non-zero) or background (0). Where models agree on a pixel being foreground but disagree on which label it should be assigned, an uncertainty of 1 is given. The weighted metric ranges from 0 to 2 instead of 0 to 1.

Uniform sampling. We use uniform random sampling as a baseline. Sampling is done using uniformly distributed values to select one image at each step. There is no information gathered from the parallel runs of CGP in terms of the images selected.

3.1 Termination Criterion

Images processed. Our goal is to study the effect of sampling of images from the dataset during evolution. Thus, we have a different

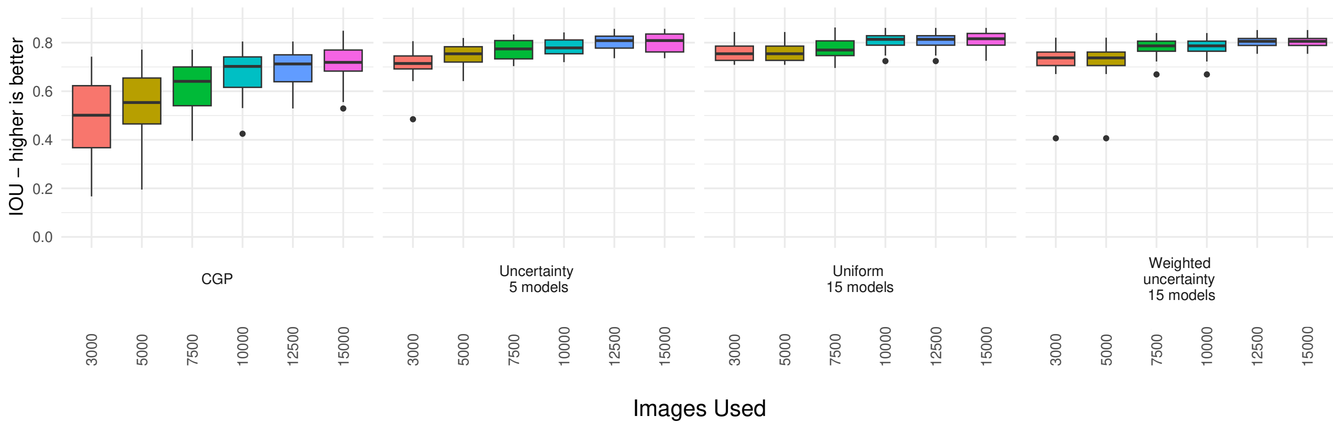

Figure 1: Dynamically sampling images for training helps CGP to achieve high performance faster.

Table 1: Parameters used.

Parameter	Value
Number of nodes	30 nodes
Offspring size, λ	5
Inputs	2, α-tubulin and DAPI channels
Outputs	2, mask and markers
Mutation of function nodes	0.15
Mutation of outputs nodes	0.20
Images processed	15,000
Training generations	100
independent executions	30

number of images processed by AL-CGP and standard CGP, at each generation. Given the different sizes of the datasets used during evolution, we cannot use the number of evaluations or generations as our termination criterion. Instead, we use the number of images processed during evolution as the termination criterion.

4 EXPERIMENTAL SETUP

dyn-CGP builds the dataset for training using the dynamic data sampling methods from Section 3 while standard CGP uses all 89 images available in the training set. We run all CGP variants with the parameters shown in Table 1. For standard CGP parameters, we use the same values as in [10]. dyn-CGP adds new data to the dataset for training every 100 generations, but more work is needed if we want to tune the parameters for the CGP variants. For statistical purposes, 30 independent runs were done for each variant.

We analyze the impact of different CGP parallel runs in terms of exploration of different areas of the search space and their ability to find different regions of the search space that can lead to more efficient programs. Here, we consider three different configuration on the number of parallel runs. For traditional CGP and dyn-CGP with uniform metric, the number of parallel run tested are: 1, 5, 10 and 15. For the uncertainty-based metrics, parallel runs in dyn-CGP are: 2, 5, 10 and 15.

For fast prototyping, we use the CELLPOSE dataset [25] which consists of 100 images of fluorescent-labeled protein of cultured neuroblastoma cells with phalloidin FITC and DAPI nuclear stain. For a fair comparison, we follow the work in [25], where the data is split into 89 images for training and 11 for testing.

Table 2: Mean AP performance of the parallel runs of dyn-CGP with different sampling mechanisms given 30 independent runs for each case. The $*$ shows statistical difference, p-value < 0.05, in comparison to GCP given the Student t-test.

Sampling mechanism	Parallel runs	Mean (standard deviation)
Uncertainty	2	0.836 (0.024)
Uncertainty	**5**	**0.843 (0.015)*****
Uncertainty	10	0.839 (0.016)
Uncertainty	15	0.833 0.016)
Uniform	1	0.842 (0.021)
Uniform	5	0.845 0.012)
Uniform	10	0.849 (0.017)
Uniform	**15**	**0.852 (0.015)*****
Weighted uncertainty	2	0.845 (0.027)
Weighted uncertainty	5	0.840 (0.022)
Weighted uncertainty	10	0.843 (0.016)
Weighted uncertainty	**15**	**0.847 (0.015)*****

5 EXPERIMENTAL COMPARISON

First, we look at the impact on Average Precision (AP) performance of using parallel runs in dyn-CGP on the performance on the test data. The results are shown in Table 2. The main result here is that dynamic data sampling positively impacts the performance of CGP, since all methods found better performance than traditional CGP that uses all data available for training. That is not a surprise, since sampling data during evolution allows the programs to explore more carefully individual features of images, without overfitting. Moreover, combining dynamic data sampling with parallel dyn-CGP runs increases the performance of CGP, since the highest performance of dyn-CGP given the different metrics is with 15 parallel dyn-CGP runs.

5.1 CGP vs dyn-CGP

We now look at the highest-performing CGP variants for CGP and dyn-CGP, given each sampling method. These highest performing configurations are highlighted in **bold** in Table 2. We can verify the speed of such algorithms in the sequential boxplots shown in Figure 1 (on the first page). The most striking observation is that dyn-CGP clearly converges faster than traditional CGP. While all dyn-CGP showed a relatively high performance from the very beginning.

6 CONCLUSION

We have studied how to effectively evolve programs for biomedical image segmentation using Cartesian GP (CGP). We utilized ideas from the Active Learning domain to create a training dataset that grows in size during evolution. We found that there are two main benefits of using growing training datasets: (i) It improves the performance of the algorithms in terms of AP, and (ii) it increases the convergence speed of CGP. That said, there is still a gap to be filled when compared with the state-of-the-art performance in this dataset of 0.93 by Deep Learning and 0.89 with CGP [10].

ACKNOWLEDGMENT

This work was is funded by the Laboratoire d'Excellence Toulouse Cancer TOUCAN, contract ANR11-LABX. This work was also 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] Arbab Masood Ahmad, Gul Muhammad Khan, Sahibzada Ali Mahmud, and Julian Francis Miller. 2012. Breast Cancer Detection Using Cartesian Genetic Programming Evolved Artificial Neural Networks. In *Proceedings of the 14th Annual Conference on Genetic and Evolutionary Computation* (Philadelphia, Pennsylvania, USA) *(GECCO '12)*. Association for Computing Machinery, New York, NY, USA, 1031–1038. https://doi.org/10.1145/2330163.2330307

[2] Qurrat Ul Ain, Harith Al-Sahaf, Bing Xue, and Mengjie Zhang. 2022. Genetic programming for automatic skin cancer image classification. *Expert Systems with Applications* 197 (2022), 116680. https://doi.org/10.1016/j.eswa.2022.116680

[3] Qurrat Ul Ain, Harith Al-Sahaf, Bing Xue, and Mengjie Zhang. 2023. Automatically Diagnosing Skin Cancers From Multimodality Images Using Two-Stage Genetic Programming. *IEEE Transactions on Cybernetics* 53, 5 (2023), 2727–2740. https://doi.org/10.1109/TCYB.2022.3182474

[4] Qurrat Ul Ain, Bing Xue, Harith Al-Sahaf, and Mengjie Zhang. 2019. Genetic Programming for Multiple Feature Construction in Skin Cancer Image Classification. In *2019 International Conference on Image and Vision Computing New Zealand (IVCNZ)*. Institute of Electrical and Electronics Engineers (IEEE), Piscataway, New Jersey, 1–6. https://doi.org/10.1109/IVCNZ48456.2019.8961001

[5] Eid Alkhaldi and Ezzatollah Salari. 2022. Ensemble Optimization for Invasive Ductal Carcinoma (IDC) Classification Using Differential Cartesian Genetic Programming. *IEEE Access* 10 (2022), 128790–128799. https://doi.org/10.1109/ACCESS.2022.3228176

[6] Sana Ben Hamida, Hmida Hmida, Amel Borgi, and Marta Rukoz. 2021. Adaptive sampling for active learning with genetic programming. *Cognitive Systems Research* 65 (2021), 23–39. https://doi.org/10.1016/j.cogsys.2020.08.008

[7] Ying Bi, Bing Xue, and Mengjie Zhang. 2022. Genetic Programming-Based Discriminative Feature Learning for Low-Quality Image Classification. *IEEE Transactions on Cybernetics* 52, 8 (2022), 8272–8285. https://doi.org/10.1109/TCYB.2021.3049778

[8] Julien Biau, Dennis Wilson, Sylvain Cussat-Blanc, and Hervé Luga. 2021. Improving Image Filters with Cartesian Genetic Programming.. In *IJCCI*. 17–27.

[9] Arantxa Casanova, Pedro O. Pinheiro, Negar Rostamzadeh, and Christopher J. Pal. 2020. Reinforced active learning for image segmentation. In *International Conference on Learning Representations*. https://openreview.net/forum?id=SkgC6TNFvr

[10] Kévin Cortacero, Brienne McKenzie, Sabina Müller, Roxana Khazen, Fanny Lafouresse, Gaëlle Corsaut, Nathalie Van Acker, François-Xavier Frenois, Laurence Lamant, Nicolas Meyer, Béatrice Vergier, Dennis G. Wilson, Hervé Luga, Oskar Staufer, Michael L. Dustin, Salvatore Valitutti, and Sylvain Cussat-Blanc. 2023. Kartezio: Evolutionary Design of Explainable Pipelines for Biomedical Image Analysis. arXiv:2302.14762 [cs.CV]

[11] Shujian Deng, Xin Zhang, Wen Yan, Eric I-Chao Chang, Yubo Fan, Maode Lai, and Yan Xu. 2020. Deep learning in digital pathology image analysis: a survey. *Frontiers of medicine* 14 (2020), 470–487.

[12] Andre Esteva, Katherine Chou, Serena Yeung, Nikhil Naik, Ali Madani, Ali Mottaghi, Yun Liu, Eric Topol, Jeff Dean, and Richard Socher. 2021. Deep learning-enabled medical computer vision. *NPJ digital medicine* 4, 1 (2021), 5.

[13] Mélanie Gaillochet, Christian Desrosiers, and Hervé Lombaert. 2023. Active learning for medical image segmentation with stochastic batches. *Medical Image Analysis* 90 (2023), 102958. https://doi.org/10.1016/j.media.2023.102958

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

[15] Nathan Haut, Wolfgang Banzhaf, and Bill Punch. 2022. Active Learning Improves Performance on Symbolic Regression Tasks in StackGP. In *Proceedings of the Genetic and Evolutionary Computation Conference Companion* (Boston, Massachusetts) *(GECCO '22)*. Association for Computing Machinery, New York, NY, USA, 550–553. https://doi.org/10.1145/3520304.3528941

[16] Nathan Haut, Bill Punch, and Wolfgang Banzhaf. 2023. Active Learning Informs Symbolic Regression Model Development in Genetic Programming. In *Proceedings of the Companion Conference on Genetic and Evolutionary Computation* (Lisbon, Portugal) *(GECCO '23 Companion)*. Association for Computing Machinery, New York, NY, USA, 587–590. https://doi.org/10.1145/3583133.3590577

[17] Hmida Hmida, Sana Ben Hamida, Amel Borgi, and Marta Rukoz. 2017. Sampling Methods in Genetic Programming Learners from Large Datasets: A Comparative Study. In *Advances in Big Data*, Plamen Angelov, Yannis Manolopoulos, Lazaros Iliadis, Asim Roy, and Marley Vellasco (Eds.). Springer International Publishing, Cham, 50–60.

[18] Marlene Huml, René Silye, Gerald Zauner, Stephan Hutterer, Kurt Schilcher, et al. 2013. Brain tumor classification using AFM in combination with data mining techniques. *BioMed research international* 2013 (2013).

[19] Asifullah Khan, Aqsa Saeed Qureshi, Noorul Wahab, Mutawarra Hussain, and Muhammad Yousaf Hamza. 2021. A recent survey on the applications of genetic programming in image processing. *Computational Intelligence* 37, 4 (2021), 1745–1778. https://doi.org/10.1111/coin.12459 arXiv:https://onlinelibrary.wiley.com/doi/pdf/10.1111/coin.12459

[20] Andreas Kirsch, Joost van Amersfoort, and Yarin Gal. 2019. 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. https://proceedings.neurips.cc/paper_files/paper/2019/file/95323660ed2124450caaac2c46b5ed90-Paper.pdf

[21] Yuri Lavinas, Kevin Cortacero, and Sylvain Cussat-Blanc. 2023. Evolving Graphs with Cartesian Genetic Programming with Lexicase Selection. In *Proceedings of the Companion Conference on Genetic and Evolutionary Computation* (Lisbon, Portugal) *(GECCO '23 Companion)*. Association for Computing Machinery, New York, NY, USA, 1920–1924. https://doi.org/10.1145/3583133.3596402

[22] Yuri Lavinas, Nathan Haut, William Punch, Wolfgang Banzhaf, and Sylvain Cussat-Blanc. 2024. Data sampling via Active Learning in Cartesian Genetic Programming for Biomedical Data. In *2024 IEEE Congress on Evolutionary Computation (CEC)*. To Appear.

[23] Julian F. Miller. 1999. 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* (Orlando, Florida) *(GECCO'99)*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1135–1142.

[24] Vishwesh Nath, Dong Yang, Bennett A. Landman, Daguang Xu, and Holger R. Roth. 2021. Diminishing Uncertainty Within the Training Pool: Active Learning for Medical Image Segmentation. *IEEE Transactions on Medical Imaging* 40, 10 (2021), 2534–2547. https://doi.org/10.1109/TMI.2020.3048055

[25] Carsen Stringer, Tim Wang, Michalis Michaelos, and Marius Pachitariu. 2021. Cellpose: a generalist algorithm for cellular segmentation. *Nature methods* 18, 1 (2021), 100–106.

[26] Masanori Suganuma, Shinichi Shirakawa, and Tomoharu Nagao. 2020. *Designing Convolutional Neural Network Architectures Using Cartesian Genetic Programming*. Springer Singapore, Singapore, 185–208. https://doi.org/10.1007/978-981-15-3685-4_7

[27] Leonardo Vanneschi, Antonella Farinaccio, Mario Giacobini, Giancarlo Mauri, Marco Antoniotti, and Paolo Provero. 2010. Identification of Individualized Feature Combinations for Survival Prediction in Breast Cancer: A Comparison of Machine Learning Techniques. In *Evolutionary Computation, Machine Learning and Data Mining in Bioinformatics*, Clara Pizzuti, Marylyn D. Ritchie, and Mario Giacobini (Eds.). Springer Berlin Heidelberg, Berlin, Heidelberg, 110–121.

[28] Dennis G Wilson, Sylvain Cussat-Blanc, Hervé Luga, and Julian F Miller. 2018. Evolving Simple Programs for Playing Atari Games. In *Proceedings of the Genetic and Evolutionary Computation Conference* (Kyoto, Japan) *(GECCO '18)*. Association for Computing Machinery, New York, NY, USA, 229–236. https://doi.org/10.1145/3205455.3205578

[29] Tian Xia, Jeremy Cosgrove, Jane Alty, Stuart Jamieson, and Stephen Smith. 2019. Application of Classification for Figure Copying Test in Parkinson's Disease Diagnosis by Using Cartesian Genetic Programming. In *Proceedings of the Genetic and Evolutionary Computation Conference Companion* (Prague, Czech Republic) *(GECCO '19)*. Association for Computing Machinery, New York, NY, USA, 1855–1863. https://doi.org/10.1145/3319619.3326822