Veni, Vidi, Evolvi (commentary on W. B. Langdon’s "Jaws 30")

Genetic Programming and Evolvable Machines (2023) 24:24
https://doi.org/10.1007/s10710-023-09472-0

COMMENTARY

Veni, Vidi, Evolvi commentary on W. B. Langdon's "Jaws 30"

Giovanni Squillero[1] · **Alberto Tonda**[2,3]

Accepted: 13 October 2023

Genetic programming (GP) has been pushing the boundaries of what a computer may achieve in an autonomous way since its introduction [1]. Over the years, John Koza himself tracked some one hundred results that are competitive with human-produced ones in a wide variety of fields,[1] and success stories have been steadily published by both scholars and practitioners in the specialized literature. However, we cannot help noticing that today GP is largely underutilized in the real-world domains where it was originally supposed to excel. Artificial intelligence is considered a core technology of the fourth industrial revolution (4IR, or Industry 4.0), but, while machine learning (ML) is explicitly mentioned, there is little doubt that the term refers to statistical models and neural networks, not GP nor other evolutionary algorithms.

Regression is a paradigmatic example of this trend. In the early 1990s, researchers excitedly demonstrated the GP's ability to evolve mathematical functions that could fit to a set of data, but after 20 years, deep neural networks are showing competitive performances [2]—the two winners of the GECCO22 SRBench competition on inter- pretable symbolic regression for data science[2] do not exploit GP, nor do

[1] https://www.human-competitive.org/awards.
[2] https://cavalab.org/srbench/competition-2022/.

Giovanni Squillero and Alberto Tonda have contributed equally to this work.

This comment refers to the article available at https://doi.org/10.1007/s10710-023-09467-x.

Special Issue: Thirtieth Anniversary of Genetic Programming: On the Programming of Computers by Means of Natural Selection.

✉ Giovanni Squillero
giovanni.squillero@polito.it

Alberto Tonda
alberto.tonda@inrae.fr

[1] Politecnico di Torino, Corso Duca degli Abruzzi 24, 10129 Turin, Italy

[2] UMR 518 MIA-PS, INRAE, AgroParisTech, University of Paris-Saclay, 22, Place de L'Agronomie, 91120 Palaiseau, France

[3] UAR 3611 CNRS, Institut des Systèmes Complexes de Paris Île-de-France (ISC-PIF), 113, Rue Nationale, 75013 Paris, France

Published online: 22 November 2023

they mention "evolutionary computation" in their descriptions. Nowadays, the most popular algo- rithms for classification are either ensembles of boosted trees, like XGBoost [3], or, again, deep neural networks [4]. A cursory analysis of the number of *likes* on GitHub repositories provides a rather clear overview of the situation: as of July 22, 2023, one may observe 176k stars for Tensorflow,[3] 68k for PyTorch,[4] and 55k for scikit-learn.[5] TPOT,[6] a GP-based optimizer for ML pipelines, scores nearly 10k stars, presumably thanks to the "ML" connection, DEAP,[7] a library for evolutionary optimization that also includes GP has 5.2k stars, but jenetics,[8] gplearn,[9] tiny-gp[10] and the other 7 projects listed when searching for "genetic programming" cumulatively got less than 2k stars.

Here, we would like to draw the readers' attention to an archetypal topic, although "less explored" as pointed out by prof. Langdon: the creation of *computer programs*. When the goal is to generate fragments frequently coded by humans, such as API calls or common algorithms, the task can be delegated almost safely to neural networks. Non-evolutionary, "AI-powered" tools like ChatGPT, Github Copilot, or Tabnine are practical because of their speed and the reduced amount of meta-parameters that need to be tweaked. Extremely complex models are often available out-of-the-box, already trained, and may be tweaked with reduced effort exploiting transfer-learning techniques; moreover, practitioners are finding clever workarounds, such as the down- casting of the weights for inference, to allow even large models to work on end-user PCs.

However, while deep learning (DL) methodologies have been shown able to efficiently learn from huge amounts of data and interpolate among existing results, GP displayed a unique ability to slowly unfold brand new solutions; and in the generation of never-before-written programs it could be thriving with little to no competition. For example, in the creation of assembly-language programs to test modern micropro- cessors [5] there are no libraries of already-written solutions and the goal is to create a unique program from scratch, targeting a new hardware design. More broadly, when- ever the goal is to devise a test, by definition, one cannot exploit already-existing material, and therefore neural-network models trained on available data are of little use. GP- and other evolutionary-based techniques have been and still are perfectly suited as fuzzer and feedback-based test generators [6, 7]. In another emblematic case study, GP was able to design a novel antenna [8], proving its effectiveness in creating a structure considerably different from human blueprints. A hypothetical generative DL system applied to the same task would be unlikely to uncover such a solution, as the final result fell well outside the distribution of

[3] https://github.com/tensorflow/tensorflow.
[4] https://github.com/pytorch/pytorch.
[5] https://github.com/scikit-learn/scikit-learn.
[6] https://github.com/EpistasisLab/tpot.
[7] https://github.com/DEAP/deap.
[8] https://github.com/jenetics/jenetics.
[9] https://github.com/trevorstephens/gplearn.
[10] https://github.com/moshesipper/tinygp.

samples it could have had observed. Apart from this niche, in some cases GP seems to have followed the old saying "if you cannot defeat them, join them": While GP cannot compete with ML and DL directly, its inherent characteristics might be used to support them. GP-based neuro- evolution is again at the forefront, with interesting results, close to or better than the state-of-the-art human-designed networks [9–11]; and even for boosted trees, recent attempts at using ensembles of GP trees evolved with a MAP-Elites [12] scheme were able to outperform classical strategies for boosting [13]—not to mention TPOT [14], seen above, by far the GP-based tool with more *stars* on GitHub. In general, evolutionary ML is a rapidly growing subfield, with dedicated workshops and tracks.

To conclude, we believe that so far GP failed to be adopted by the mainstream community, especially in industrial contexts. However, this is not just due to the relative effectiveness of the different techniques. The DL/ML scholars managed to advertise their successes and coalesce a large community of practitioners around their algorithms. Just like some researchers kept the fire going while the interest in neural networks waned during the 90s and the 2000s, the GP community should keep working to make the world aware of the potentiality of this approach. After 30 years, the future seems ripe for GP applications; moreover, the ML/DL rising wave is creating vast search spaces that need to be effectively explored (neuro-evolution, diversity of boosted trees); and with DL models being complete black boxes, there is a growing need for explainability, a call for symbolic or neuro-symbolic AI, where GP could provide good solutions, especially in areas where DL fails, like the Abstraction and Reasoning Corpus benchmark [15]. A new GP spring may very well be looming on the horizon.

Funding Open access funding provided by Politecnico di Torino within the CRUI-CARE Agreement.

References

1. J. Koza, Genetic programming: a paradigm for genetically breeding populations of computer programs to solve problems. Technical Report STAN-CS-90-1314 (Dept. of Computer Science, Stanford University, 1990)
2. P.-A. Kamienny, G. Lample, S. Lamprier, M. Virgolin, Deep generative symbolic regression with Monte-Carlo-tree-search (2023). 2302.11223
3. T. Chen, C. Guestrin, *XGBoost: A Scalable Tree Boosting System, KDD '16* (ACM, New York, 2016), pp.785–794. https://doi.org/10.1145/2939672.2939785
4. I. Goodfellow, Y. Bengio, A. Courville, *Deep Learning* (MIT Press, 2016)

5. G. Squillero, Artificial evolution in computer aided design: from the optimization of parameters to the creation of assembly programs. Computing **93**, 103–120 (2011)
6. N. Alshahwan et al., *Deploying Search Based Software Engineering with Sapienz at Facebook* (Springer, 2018), pp.3–45
7. S. Gandini, W. Ruzzarin, E. Sanchez, G. Squillero, A. Tonda, A framework for automated detection of power-related software errors in industrial verification processes. J. Electron. Test. **26**, 689–697 (2010). https://doi.org/10.1007/s10836-010-5184-5
8. J.D. Lohn, G.S. Hornby, D.S. Linden, An evolved antenna for deployment on NASA's space technology 5 mission. Genet. Program. Theory Pract. **2**, 301–315 (2005)
9. R. Miikkulainen et al., *Evaluating Medical Aesthetics Treatments Through Evolved Age-Estimation Models* (ACM, 2021). https://doi.org/10.1145/3449639.3459378
10. F. Assuncao, N. Lourenco, P. Machado, B. Ribeiro, DENSER: deep evolutionary network structured representation. Genet. Program. Evol. Mach. **20**, 5–35 (2018). https://doi.org/10.1007/s10710-018-9339-y
11. E. Real et al., in *Large-Scale Evolution of Image Classifiers*, eds. by D. Precup, Y.W. The. *Proceedings of the 34th International Conference on Machine Learning*, Vol. 70 of *Proceedings of Machine Learning Research* (PMLR, 2017), pp. 2902–2911. https://proceedings.mlr.press/v70/real17a.html.
12. J.-B. Mouret, J. Clune, Illuminating search spaces by mapping elites (2015). https://arxiv.org/abs/1504.04909
13. H. Zhang et al., *MAP-Elites with Cosine-Similarity for Evolutionary Ensemble Learning* (Springer, Switzerland, 2023), pp.84–100. https://doi.org/10.1007/978-3-031-29573-76
14. R.S. Olson, N. Bartley, R.J. Urbanowicz, J.H. Moore, *Evaluation of a tree-based pipeline optimization tool for automating data science*, GECCO '16 (ACM, New York, 2016), pp. 485–492. https://doi.org/10.1145/2908812.2908918
15. F. Chollet, On the measure of intelligence (2019). https://arxiv.org/abs/1911.01547