Diagnosing Autonomous Vehicle Driving Criteria with an Adversarial Evolutionary Algorithm

Mark A. Coletti
colettima@ornl.gov
Oak Ridge National Laboratory
Oak Ridge, Tennessee, USA

Shang Gao
gaos@ornl.gov
Oak Ridge National Laboratory
Oak Ridge, Tennessee, USA

Spencer Paulissen
paulissensr@ornl.gov
Oak Ridge National Laboratory
Oak Ridge, Tennessee, USA

Nicholas Quentin Haas
haasnq@ornl.gov
Oak Ridge National Laboratory
Oak Ridge, Tennessee, USA

Robert Patton
pattonrm@ornl.gov
Oak Ridge National Laboratory
Oak Ridge, Tennessee, USA

ABSTRACT

We repurposed an adversarial evolutionary algorithm, Gremlin, from finding driving scenarios where a model of an autonomous vehicle drove poorly to troubleshooting driving quality evaluation criteria. We evaluated the driving performance of a "perfect driver" robot in a virtual town environment using the same fitness criteria intended for a deep learner (DL) trained driver. We found that the fitness evaluation criteria poorly handled turns, and used Gremlin to iteratively improve that criteria. We were confident that the same criteria could then be applied to the DL-based models as originally intended, and that this approach could be used as a general means of troubleshooting autonomous vehicle driving criteria.

KEYWORDS

evolutionary algorithms, autonomous vehicles, adversarial algorithms

ACM Reference format:
Mark A. Coletti, Shang Gao, Spencer Paulissen, Nicholas Quentin Haas, and Robert Patton. 2021. Diagnosing Autonomous Vehicle Driving Criteria with an Adversarial Evolutionary Algorithm. In *Proceedings of 2021 Genetic and Evolutionary Computation Conference Companion, Lille, France, July 10–14, 2021 (GECCO '21 Companion),* 2 pages.
https://doi.org/10.1145/3449726.3459573

1 MOTIVATION

Sound driving evaluation criteria is paramount for training an effective Autonomous Vehicle (AV) model. If the criteria is too simple, then it is difficult to train a correspondingly high quality model. For example, some criteria focus mainly on the total distance driven by an AV without collision or intervention, which lacks differentiation between factors such as weaving, staying within lanes, having abrupt accelerations, or severe braking [1, 3, 5]. Moreover, as the number of driving criteria increases, the more challenging it is to tune the criteria to achieve desired driving behavior. This complexity creates a need for a tool to assist in designing and tuning the parameters of a given driving quality metric.

2 METHODOLOGY

We created such a tool by applying an adversarial evolutionary algorithm (EA), Gremlin, to evaluate specific driving scenarios using a "perfect driver". The expectation is that for a given fitness criteria a "perfect driver" would yield consistently high score values regardless the driving scenario. Evaluation scores that had high variance would signal a potential problem, and an analysis of the worst performing scenarios could yield clues as to how to improve the evaluation criteria. Gremlin could then be iteratively applied to gradually improve the fitness evaluation criteria until the criteria yielded consistently desirable results.

We tested this approach by representing scenarios as 37m stretches of road for the CARLA AV simulation framework's "Town01" map that used one of 14 preset times of day and weather [2] — alternatively these segments could be described as a 25m non-overlapping stretch of road with 6m that overlapped with the segments on either end, thus ensured that there was comprehensive coverage of available route features. There were a total of 64 such overlapping segments for Town01. What scenarios were selected and how they were scored was dictated by the EA, Gremlin, where each individual represented a single scenario. Each scenario was statically bound to one of those 14 presets, such as "Noon, Sunny" or "Hard Rain, Sunset" [2].

Gremlin would evaluate a scenario, or individual, by placing a virtual car driven by a CARLA "roaming agent" on a segment dictated by the individual's genome, assign the time of day and weather associated with that segment, and drive that segment yielding a score at the end of the run. It would repeat this trial three times and assign the average of the fitness scores as the overall final fitness. The "roaming agent" was a robot driver provided by the CARLA framework that would perfectly drive a virtual AV along a preset collection of waypoints.

Gremlin was implemented as an asynchronous steady-state evolutionary algorithm (ASEA) to ensure minimal idle time of associated HPC resources [4]. That is, a population of evaluated scenarios was continually updated as new offspring that represented scenarios had their fitnesses computed. A single, new offspring was then

GECCO '21 Companion, July 10–14, 2021, Lille, France
2021. ACM ISBN 978-1-4503-8351-6/21/07.
https://doi.org/10.1145/3449726.3459573

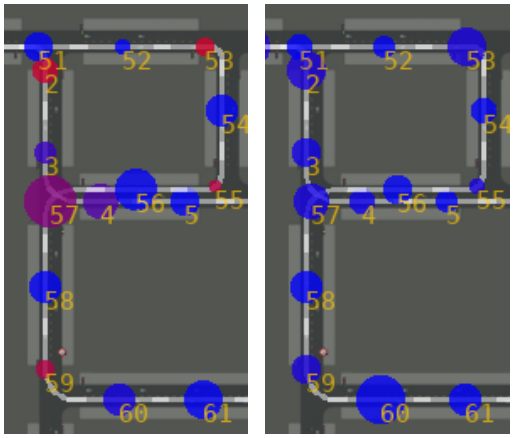

Figure 1: Town01 map of scenarios for both evaluation criteria. This shows a close-up of the Town01 test route that was subdivided into 64 overlapping 37m segments represented by numbered grey strips with overlaps as slightly brighter grey. The dots are shaded from red to blue, with red denoting the worst scenarios for a single run and blue the best. The dot radius is the number of times that scenario was exercised for that Gremlin run. The left map shows the original evaluation criteria results, and the right the newer, improved version.(Background map image courtesy of the CARLA project.)

immediately created and assigned to that recently freed computational resource. We ran this configuration on Oak Ridge National Laboratory (ORNL)'s Summit supercomputer with an initial population size of 25 individuals, a population of 25, a birth budget of 500, and ran this on two Summit nodes with six CARLA servers per node evaluating scenarios in parallel. We did two runs, one for a set of driving evaluation criteria that turned out to be flawed, and another for evaluation criteria that had been improved based on analysis of the first run.

3 RESULTS

Figure 1 shows a side-by-side subset of the map of Town01 with the left map corresponding to the first run with original driving quality criteria, and the right for the improved criteria. The scenarios are numbered from 0 to 63 in gold letters next to their respective segments. The dots are colored from red to blue in proportion to the relative fitnesses for that run, with red for the worst and blue the best.

Based on the left map, we observed that the roaming agent lower scores were for turns. The original evaluation criteria had only been tested on straight sections, and so was updated to better accommodate turns. The right map shows that these changes corrected that problem in that the roaming agent had the expected consistent higher scores.

4 CONCLUSIONS

Gremlin was originally intended to tune model training data by adding more examples of scenarios where a driving model performed poorly to model training data. However, we found while developing that system that it could also be used to find and troubleshoot problems with driving quality fitness criteria.

That is, a "perfect driver," which in this case was a CARLA AV roaming agent, did not exhibit perfect fitnesses using a particular fitness criteria of driving behavior, which signaled a problem with that fitness criteria; i.e., we would expect a "perfect driver" to have "perfect scoring." Analyzing the spatial patterns of the hot spots of poor performance revealed that the original critieria made certain assumptions that did not hold for turns. Once the criteria was updated to handle turns, we could use Gremlin with the "perfect driver" once more to verify that the updated criteria corrected the problem. This approach could be similarly applied to troubleshooting fitness criteria for other AV systems that have an available "perfect driver" to serve as a control.

ACKNOWLEDGMENTS

This research used resources of the Oak Ridge Leadership Computing Facility, which is a DOE Office of Science User Facility supported under Contract DE-AC05-00OR22725.

Also, we would like to thank the ORNL Leadership Computing Facility (OLCF) for their generous grant of 20,000 Summit node-hours via Director's Discretion grant LRN013 that made the Summit results possible. And, lastly, we would like to thank the U. S. Department of Energy (DOE)'s Vehicle Technologies Office (VTO) for their funding support.

This manuscript has been authored by UT-Battelle, LLC under Contract No. DE-AC05-00OR22725 with the U.S. Department of Energy (DOE). Funding provided by the U.S. Department of Energy Office of Energy Efficiency and Renewable Energy Vehicle Technologies Office. The views expressed in the article do not necessarily represent the views of the DOE or the U.S. Government. The United States Government retains and the publisher, by accepting the article for publication, acknowledges that the United States Government retains a non-exclusive, paid- up, irrevocable, worldwide license to publish or reproduce the published form of this manuscript, or allow others to do so, for United States Government purposes. The Department of Energy will provide public access to these results of federally sponsored research in accordance with the DOE Public Access Plan (http://energy.gov/downloads/doe-public-access-plan).

REFERENCES

[1] Mariusz Bojarski, Davide Del Testa, Daniel Dworakowski, et al. 2016. End to end learning for self-driving cars. arXiv preprint arXiv:1604.07316 (2016).
[2] Alexey Dosovitskiy, Germán Ros, Felipe Codevilla, et al. 2017. CARLA: An Open Urban Driving Simulator. CoRR abs/1711.03938 (2017). arXiv:1711.03938 http://arxiv.org/abs/1711.03938
[3] Yunpeng Pan, Ching-An Cheng, Kamil Saigol, et al. 2018. Agile Autonomous Driving using End-to-End Deep Imitation Learning. In Robotics: Science and Systems XIV, Vol. 14. https://academic.microsoft.com/paper/2806163172
[4] Eric O Scott and Kenneth A De Jong. 2015. Understanding simple asynchronous evolutionary algorithms. In Proceedings of the 2015 ACM Conference on Foundations of Genetic Algorithms XIII. 85–98.
[5] Dequan Wang, Coline Devin, Qi-Zhi Cai, Fisher Yu, and Trevor Darrell. 2019. Deep Object-Centric Policies for Autonomous Driving. In 2019 International Conference on Robotics and Automation (ICRA). 8853–8859. https://academic.microsoft.com/paper/2967895468