WOGAN at the SBST 2022 CPS Tool Competition

Jarkko Peltomäki
Åbo Akademi University
Turku, Finland
jarkko.peltomaki@abo.fi

Frankie Spencer
Åbo Akademi University
Turku, Finland
frankie.spencer@abo.fi

Ivan Porres
Åbo Akademi University
Turku, Finland
ivan.porres@abo.fi

ABSTRACT

WOGAN is an online test generation algorithm based on Wasserstein generative adversarial networks. In this note, we present how WOGAN works and summarize its performance in the SBST 2022 CPS tool competition concerning the AI of a self-driving car.

ACM Reference Format:
Jarkko Peltomäki, Frankie Spencer, and Ivan Porres. 2022. WOGAN at the SBST 2022 CPS Tool Competition. In *The 15th Search-Based Software Testing Workshop (SBST'22), May 9, 2022, Pittsburgh, PA, USA.* ACM, New York, NY, USA, 2 pages. https://doi.org/10.1145/3526072.3527535

1 SBST 2022 CPS TOOL COMPETITION

The SBST 2022 CPS tool competition is concerned with finding road scenarios that cause the AI of the BeamNG.tech driving simulator to drive a car out of its designated lane. The organizers of the competition performed two experiments in which the competition entries were compared according to three metrics: efficiency, effectiveness, and failure-inducing test diversity. The BeamNG.AI was used in the first experiment while the DAVE-2 AI was used in the second. Complete details on the simulator, road scenarios, the AIs, and experiment results are found in the competition report [3].

2 ABOUT WOGAN

Here we provide a brief explanation how WOGAN works. A more complete description is found in [4].

We consider an input road to the BeamNG.tech simulator as a *test*, and we call the output of the test its *fitness*. We chose the fitness to be the maximum percentage of the body of the car that is out of the boundaries of its lane during the simulation (in short BOLP). In the experiments of the competition report [3], tests were considered failed if BOLP was over 0.85 (BeamNG.AI) or 0.10 (DAVE-2).

The central idea is to use a generative machine learning model to generate tests with high fitness. We chose to train a Wasserstein generative adversarial network (WGAN) which is capable of producing diverse samples from its target distribution [1]. Since we lack training data in advance, we need to train the WGAN online. Our approach is to first do a random search to obtain an initial training data for the WGAN and then augment this training data by executing tests generated by the WGAN which are estimated to have high fitness.

The initial random search produces a training data (t_i, f_i), $i = 1, \ldots, N$, of tests t_i and fitnesses f_i. The WGAN, say G, is trained with tests of high fitness (what "high" means is explained in [4]). In order to find a new candidate test, the WGAN G is sampled for new tests. In order to choose the best candidate test, we employ an analyzer A which is a neural network trained to learn the map from tests to fitnesses (the collected training data enables us to do so). Thus A acts as a surrogate model for the simulator, and by using it we avoid costly executions on the simulator. We select the candidate test t with the highest estimated fitness and execute it on the simulator to find its true fitness f. Finally we add (t, f) to the training data, retrain the models, and repeat the preceding procedure until the test budget is exhausted. The tests of the final training data can be considered as a test suite for the simulator.

Intuitively our WOGAN algorithm should be able to find tests with high fitness. As more training data is available, the generator G should be more capable of generating high-fitness tests and the analyzer A should be more accurate in assessing test fitness. The experiments described in [4] and the experiments of the competition report [3] experimentally validate this intuition.

Using WOGAN requires choosing hyperparameters. For the tool competition, we used the same hyperparameters as described in [4, Sec. 3.1]. Most importantly, we used 20% of the execution budget for random search, and we always generate roads defined by exactly 6 points (this is an arbitrary decision). For more on the input representation and normalization, see [4, Sec. 2.1].

3 RESULTS AND THEIR INTERPRETATION

Here we summarize the results of the experiments of [3]. The metrics considered here are efficiency, effectiveness, and failure-inducing test diversity; see [3] for their more elaborate definitions.

Efficiency. As written in [3], test generation efficiency is measured as the time spent to generate tests. In the experiments, two time budgets are specified: generation budget (1 h) and simulation budget (2 h). The latter is the time used for running the simulator while the generation budget accounts for the remaining time used.

WOGAN is the most efficient tool in the above sense: it uses least time to propose a new test [3]. In fact, it uses one magnitude less time than the other tools; see Figure 1. We believe that this is mainly due to the fact that WOGAN is not a traditional SBSE algorithm. The actual search for a new test occurs when the weights of the neural networks are adjusted. After this, candidate tests are sampled and analyzed which amounts to few forward passes of neural networks. These operations run efficiently on modern hardware especially since we have very little training data and small networks.

Since WOGAN does not use most of the generation budget, it would make sense to use more time for generation. More complicated models or ensembles of models could be trained to improve

SBST'22, May 9, 2022, Pittsburgh, PA, USA
© 2022 Association for Computing Machinery.
ACM ISBN 978-1-4503-9318-8/22/05…$15.00
https://doi.org/10.1145/3526072.3527535

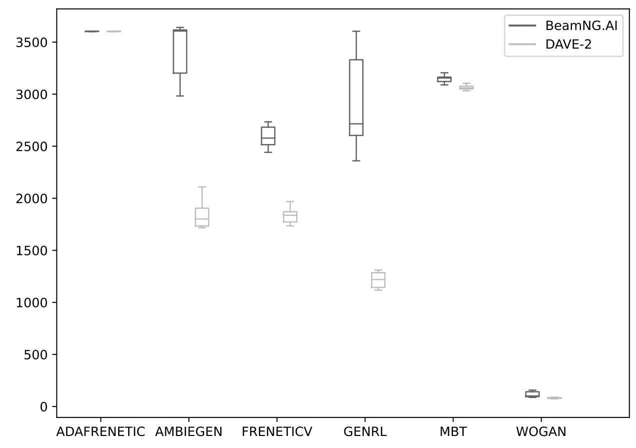

Figure 1: Efficiency (in seconds)

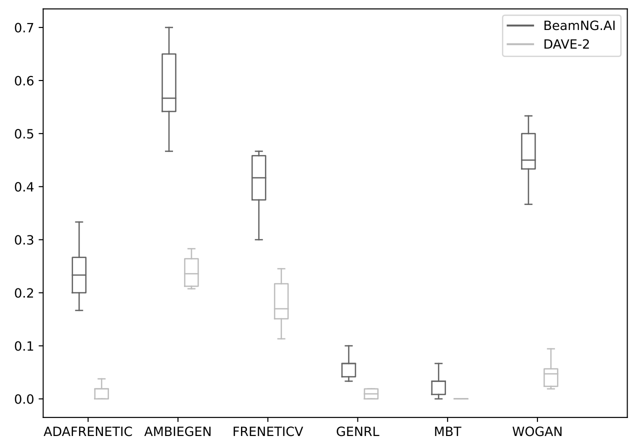

Figure 2: Relative map coverage

the metrics. In addition, more domain knowledge could be utilized.

Effectiveness. Not all input roads are valid, so a good tool needs to learn to avoid producing invalid roads. Effectiveness is defined as the ratio of valid tests over all generated tests [3].

WOGAN generates many invalid tests: roughly 50% of the tests could not be executed [3]. WOGAN has no built-in notion of a valid test, so the neural networks used need to learn this by trial and error. There is no smooth function measuring how close a road is being valid, so learning the notion validity accurately with a small training data can be challenging.

The situation could perhaps be improved by introducing an additional validator classifier model, but we did not experiment with this. We do not view this notion of effectiveness particularly important as an efficient validator is provided in the simulation suite meaning WOGAN could validate a candidate road internally before proposing it as a test being simulated. This way it could achieve a perfect effectiveness with 0 invalid tests. This was also noted in the competition note for Frenetic 2021 [2].

Failure-Inducing Test Diversity. In the BeamnNG.AI experiment WOGAN generated a large number of failed tests: on average 330.3 (SD 55.8) over 10 repetitions whereas the next best tool, AMBIEGEN, generated on average 90.4 (SD 12.0) failed tests. The tests generated by WOGAN were however not as diverse as those generated by AMBIEGEN (see Figure 2), so WOGAN was ranked second in the failure-inducing test diversity metric [3].

In the DAVE-2 experiment, WOGAN did not fare that well. It was ranked third by failure-inducing test diversity, but its performance was considerably lower than that of AMBIEGEN and FRENETICV; see Figure 2. WOGAN was able to generate on average 3.1 (SD 1.3) failed tests over 10 repetitions. AMBIEGEN and FRENETICV respectively achieved the means 15.3 (SD 6.3) and 11.1 (SD 4.5).

We believe the reason for WOGAN's worse performance in the DAVE-2 experiment is that the BOLP output of the simulator is not sensitive enough. We have observed that a randomly chosen road often has a very low BOLP value in the DAVE-2 setting, and the neural networks could have trouble learning with so homogeneous data. We believe we should have considered the distance of the center of the car to the edges of the lane. This distance yields information even when BOLP is 0. One additional factor is hyperparameter tuning: we mainly used the BeamNG.AI for development. Perhaps our hyperparameters for BeamNG.AI are subpar for DAVE-2.

We are satisfied with the diversity of roads generated by WOGAN. We relied solely on the ability of WGAN's to produce varied samples, and we did not utilize any domain-specific knowledge. Moreover, we worked under the assumption that the road diversity metric was translation and rotation invariant and generated only roads that initially point to north. This was not the case in the competition.

Frenetic 2021 [2] is a genetic algorithm with domain-specific mutations which are applied only to failing tests in order to improve solution diversity. Once the algorithm finds a failing test it can mutate it, e.g., by mirroring. This produces new tests that are likely to fail and are diverse by construction. It is possible to extend WOGAN to include such mutations using domain-specific rules. We conjecture this would increase WOGAN's failure-inducing test diversity metric. We also conjecture that a more varied initial random search could lead to more diverse failing tests.

ACKNOWLEDGMENTS

We thank the organizers of the SBST competition for creating and continuing the competition.

This research work has received funding from the ECSEL Joint Undertaking (JU) under grant agreement No 101007350.

REFERENCES

[1] M. Arjovsky, S. Chintala, and L. Bottou. 2017. Wasserstein generative adversarial networks. In *Proceedings of the 34th International Conference on Machine Learning (Proceedings of Machine Learning Research, Vol. 70).* PMLR, 214–223.

[2] E. Castellano et al. 2021. Frenetic at the SBST 2021 Tool Competition. In *2021 IEEE/ACM 14th International Workshop on Search-Based Software Testing (SBST).* 36–37. https://doi.org/10.1109/SBST52555.2021.00016

[3] A. Gambi, G. Jahangirova, V. Riccio, and F. Zampetti. 2022. SBST tool competition 2022. In *15th IEEE/ACM International Workshop on Search-Based Software Testing, SBST 2022.*

[4] J. Peltomäki, F. Spencer, and I. Porres. 2022. Wasserstein generative adversarial networks for online test generation for cyber physical systems. In *15th IEEE/ACM International Workshop on Search-Based Software Testing, SBST 2022.* https://doi.org/10.1145/3526072.3527522