DRVN at the ICST 2025 Tool Competition – Self-Driving Car Testing Track

Bartlett, A.; Liem, C.; Panichella, A.

DOI
10.1109/ICST62969.2025.10988997

Publication date
2025

Document Version
Final published version

Published in
Proceedings of the 2025 IEEE Conference on Software Testing, Verification and Validation (ICST)

DRVN at the ICST 2025 Tool Competition – Self-Driving Car Testing Track

Antony Bartlett
Delft University of Technology
Delft, The Netherlands
a.j.bartlett@tudelft.nl

Cynthia Liem
Delft University of Technology
Delft, The Netherlands
c.c.s.liem@tudelft.nl

Annibale Panichella
Delft University of Technology
Delft, The Netherlands
a.panichella@tudelft.nl

2025 IEEE Conference on Software Testing, Verification and Validation (ICST) | 979-8-3315-0814-2/25/$31.00 ©2025 IEEE | DOI: 10.1109/ICST62969.2025.10988997

***Abstract*—DRVN is a regression testing tool that aims to diversify the test scenarios (road maps) to execute for testing and validating self-driving cars. DRVN harnesses the power of convolutional neural networks to identify possible failing roads in a set of generated examples before applying a greedy algorithm that selects and prioritizes the most diverse roads during regression testing. Initial testing discovered that DRVN performed well against random-based test selection.**

***Index Terms*—image recognition, search-based testing, autonomous driving, test case diversity**

I. Introduction

Regression test optimization is an important software testing activity, particularly in systems where test cases are expensive to execute. This challenge is especially prominent in `self-driving car` (SDC) testing, where the cost to create and run an extensive suite of test cases can increase drastically over time. Due to the critical environments in which SDCs are executed, testing will often initially be executed in a simulation environment before the validation can be performed in a real-world system. Although simulation environments are cheaper to run than real-world applications, they are still expensive due to the graphical overhead and the time it takes to execute.

With this in mind, the SDC Tool competition was created and run for the first time in 2025 as part of the ICST conference [3]. This competition combines knowledge from existing regression test optimization work [1], [2], [5], bringing together teams to design and create tools that help establish diverse and optimized regression suites. These are suites in which possible 'failing' roads are identified and sorted for diversity through an algorithmic process rather than running the entire test suite in simulation. This process helps to reduce the cost of simulation by only executing the most diverse test cases that could cause a failure. The competition hopes to address the above-mentioned issue of test case selection for SDCs by creating a structured test environment with which teams can validate their tools. This environment allows teams to focus solely on their selection methods.

This paper introduces our submission for the competition, `DRVN`. `DRVN` utilizes the power of `Convolutional Neural Networks` (CNNs) to identify possible 'failing' roads in a test suite before applying a greedy optimization algorithm [7] that prioritizes the most diverse roads for the regression. A failing test case is a road map that an SDC fails to drive correctly on, normally involving the SDC driving out-of-bounds (too far over the side or middle line of a road).

`DRVN` is available at https://github.com/checkdgt/sdc-testing-competition.

II. Competition Setup

The competition tool uses the `gRPC` [1] interface to create a consistent environment where all tools implement/follow the same interfaces, allowing the competition evaluator to work smoothly with all submissions.

To begin with, the evaluation tool distributes the test cases to our tool. Here, each test case contains limited information, giving just the test case ID and the road points that delimit the road map (the control point for interpolation). Here, ID is a unique identifier of the test case, and road points are x and y coordinates on a 2-dimensional Cartesian plane.

From this information, each tool must then apply its approach to locate viable test cases, returning them to the evaluator. The evaluator then validates this selection, using withheld information on the test case's success or failure.

III. DRVN

`DRVN` is our entry to the ICST'2025 SDC Testing competition. We implemented a novel approach that uses the power of CNNs to help locate failing roads from images before executing a greedy algorithm to find diversity in the selected roads. This hybrid approach removes the need for complex algorithms and instead utilizes the feature extraction of a CNN to find patterns on roads where failures occur. The greedy algorithm is used to locate the most diverse roads in our subset, as it is an inherently fast algorithm for such scenarios.

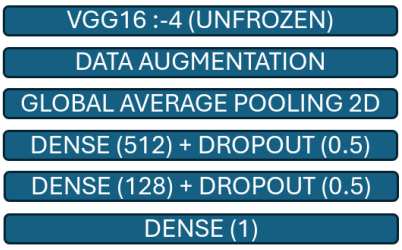

Fig. 1: Additional layers for the fine-tuned VGG16 model.

[1] https://grpc.io/

Fig. 2: Example of road image used for training VGG16.

A. CNN Training

We utilize a CNN to predict which test cases could fail, as CNNs are powerful tools for classifying images based on features learned during training. For this task in particular, we chose to fine-tune the existing, well-known CNN, `VGG16` [9]. VGG16 is classified as a 'very deep' convolutional neural network for large-scale image recognition, performing exceptionally well in the ImageNet benchmark challenges [2]. During our initial experimentation, VGG16 proved to be fast and accurate against larger models such as `VGG19` [9] and the `ResNet` [6] family of CNNs.

To create our training data, we utilized the `SensoDat` dataset, created by Birchler *et al.*, 2024 [4]; SensoDat is a dataset of more than 30, 000 simulation-based SDC test cases executed. This data set contains information on road points and whether a road had caused a failure, allowing us to accurately label each image as pass or fail for training.

We generate a road map image from each test case (road points) using the `matplotlib` [3] library in Python. These road images were created with a size of 224×224 pixels, which is the input size of the VGG16 model. This worked well for us, as the original roads were created for a driving space of 200×200 meters. Figure 2 contains an example of the road plot generated by matplotlib.

To accommodate our task, we unfroze the last four layers of the VGG16 model, adding extra layers for our specific use case as seen in Figure 1. By unfreezing the last four layers, we only affect the higher-level features of the original model, allowing it to better adapt to our specific feature set (roads). An augmentation step was also added to the model pipeline. This allows us to create metamorphic images by flipping and rotating the input images. Performing this additional step helps to build further robustness into our model.

The remaining layers help to round out our model. `Global Average Pooling` replaces the flattening layer in the original model. Next, the dense layers helps the model to capture new relationships and patterns from our dataset, whilst the dropout layers help to prevent overfitting during training.

Our final dense layer applies a `sigmoid` activation to scale the output to a value between 0 and 1, which serves as our prediction value during selection.

[2]https://image-net.org/challenges/LSVRC/index.php

[3]https://matplotlib.org/

B. Test Case Selection

Initial test case selection is performed by our CNN model by taking the entire test suite from the evaluator and batch processing them for predictions. To make predictions, we follow the same approach as for our training data in Section III-A, plotting the road points to create an image of 224×224, before feeding the image to the model for prediction. During this prediction stage, we store each test case with a predictive value > 0.6 in a pool, where a higher prediction should denote a higher likelihood of the test case to fail.

C. Greedy Selection

With a pool of predicted test case failures, we are then able to apply our greedy algorithm to find the most diverse cases.

Our greedy algorithm selects a random test case from the pool of likely-failing test cases as a starting point. Then, it iterates over the pool of likely-failing tests, calculating the diversity of each test case against the starting point. We use the pairwise Euclidean distance [8] between the curvature profiles of the test case roads, giving us a diversity distance to use as a fitness value. We chose an arbitrary diversity threshold > 0.9 for diversity selection, where test cases that met this selection were iteratively added to our final selection of test cases.

The greedy search continues incrementally to add select test cases to our final selection pool, until we no longer see an increase of this pool at the end of an iteration. At this point, the final pool of diverse test cases is returned to the evaluation tool for final validation.

This initial evaluation revealed that our approach was able to outperform the supplied random-based baseline, allowing us to release our tool to the competition for further evaluation.

References

[1] Birchler, C., Ganz, N., Khatiri, S., Gambi, A., Panichella, S.: Cost-effective simulation-based test selection in self-driving cars software with sdc-scissor. In: 2022 IEEE international conference on software analysis, evolution and reengineering (SANER). pp. 164–168. IEEE (2022)

[2] Birchler, C., Khatiri, S., Bosshard, B., Gambi, A., Panichella, S.: Machine learning-based test selection for simulation-based testing of self-driving cars software. Empirical Software Engineering **28**(3), 71 (2023)

[3] Birchler, C., Klikovits, S., Fazzini, M., Panichella, S.: ICST tool competition 2025 - self-driving car testing track. In: IEEE/ACM International Conference on Software Testing, Verification and Validation, ICST 2025 (2025)

[4] Birchler, C., Rohrbach, C., Kehrer, T., Panichella, S.: Sensodat: Simulation-based sensor dataset of self-driving cars. In: 21th IEEE/ACM International Conference on Mining Software Repositories, MSR 2024, Lisbon, Portugal, April 15-16, 2024 (2024). https://doi.org/to appear

[5] Birchler, C., Rohrbach, C., Kim, H., Gambi, A., Liu, T., Horneber, J., Kehrer, T., Panichella, S.: Teaser: Simulation-based can bus regression testing for self-driving cars software. In: 2023 38th IEEE/ACM International Conference on Automated Software Engineering (ASE). pp. 2058–2061. IEEE (2023)

[6] He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition (2015). https://doi.org/10.48550/ARXIV.1512.03385

[7] Jungnickel, D.: The Greedy Algorithm, pp. 129–153. Springer Berlin Heidelberg, Berlin, Heidelberg (1999). https://doi.org/10.1007/978-3-662-03822-2_5

[8] Liberti, L., Lavor, C., Maculan, N., Mucherino, A.: Euclidean distance geometry and applications. SIAM Review **56** (05 2012). https://doi.org/10.1137/120875909

[9] Simonyan, K., Zisserman, A.: Very deep convolutional networks for large-scale image recognition (2015), https://arxiv.org/abs/1409.1556