DESIGNATOR: a Toolset for Automated GAN-enhanced Search-based Testing and Retraining of DNNs in Martian Environments

Mohammed Oualid Attaoui
SnT Centre, University of Luxembourg
Luxembourg, Luxembourg
mohammed.attaoui@uni.lu

Fabrizio Pastore
SnT Centre, University of Luxembourg
Luxembourg, Luxembourg
fabrizio.pastore@uni.lu

***Abstract*—We present *DESIGNATOR*, a toolset for generating datasets for testing and retraining deep neural networks (DNNs) performing computer vision tasks in Martian-like environments. The toolset integrates Marsim, a simulator of the Mars environment, and DESIGNATE, a search-based approach combining simulation, generative adversarial networks (GANs), and search-based test input generation. The tool enables users to select a search strategy, launch simulations in MarsSim, and observe the evolution of simulated images, corresponding realistic images, ground truth labels, model predictions, and fitness values. Beyond supporting researchers and practitioners in generating datasets capable of identifying failures and retraining DNNs, *DESIGNATOR* can be used as a didactic tool to explain how image datasets can be generated using meta-heuristic search. Furthermore, MarsSim can be used standalone, through its API and GUI. MarsSim enables researchers to assess search-based approaches beyond the predominantly studied automotive context. A demo video of *DESIGNATOR* is available at https://youtu.be/fGxgpgViPEo.**

***Index Terms*—DNN testing, Functional Safety Analysis, Martian simulation**

I. Introduction

Deep Neural Networks (DNNs) have become integral to computer vision tasks in safety- and mission-critical applications, including autonomous robots, drones, planetary rovers, and vehicles. For instance, DNNs are employed in autonomous vehicles for object detection and lane recognition, enhancing driving safety and efficiency [1]. In the realm of planetary exploration, DNNs assist rovers in terrain classification and navigation, enabling them to make real-time decisions in unstructured environments [2], [3]. NASA's Perseverance rover utilizes machine learning-based perception modules to interpret Martian landscape. The European Space Agency (ESA), our project funder, supports the development of Martian exploration missions that utilize DNN-based systems for visual perception and autonomous navigation. However, despite their impressive capabilities, DNNs remain limited when confronted with out-of-distribution (OOD) inputs, rare scenarios, or subtle visual conditions that deviate from the training data distribution.

DNN failures in mission-critical contexts may result in mission failures or equipment damage, highlighting the need for rigorous testing and validation under realistic, hard-to-capture conditions. To address this, the DESIGNATE approach [4] combines generative adversarial networks (GANs) with search-based testing to systematically generate failure-inducing test inputs and guide retraining. A key enabler for this process is *MarsSim*, a standalone, high-fidelity simulator designed to generate realistic and diverse Martian imagery for DNN evaluation and dataset creation.

In this paper, we present *DESIGNATOR*, a GUI-based tool that integrates DESIGNATE and the MarsSim simulator. *DESIGNATOR* enables users to configure search strategies, launch simulations, visualize the evolution of generated test cases, and efficiently identify DNN weaknesses for targeted retraining. The significance of *DESIGNATOR* lies in its ability to close the loop between simulation, test generation, and model improvement. It operationalizes the core ideas of DESIGNATE and embeds them into a hands-on, exploratory tool. Notably, MarsSim can also be used independently of *DESIGNATOR*, either through its API or via a dedicated GUI, enabling dataset generation and experimentation beyond DESIGNATE.

This paper proceeds as follows. Section II reviews related work. Section III provides background on DESIGNATE and its components. Section IV describes the tool architecture. Section V details the standalone MarsSim simulator. Section VI outlines the envisioned usage workflow. Section VII presents empirical results. Finally, Section VIII concludes the paper.

II. Related Work

Recent surveys [5] provide a comprehensive overview of DNN testing approaches. Some approaches rely on the perturbation of given dataset images [6], [7], [8], even with advanced GAN-based approaches capable of changing part of the background or the foreground [9], but still being limited in their capability of generating new scenarios (i.e., they alter existing images but cannot add new elements).

Simulation-based approaches are instead a better choice

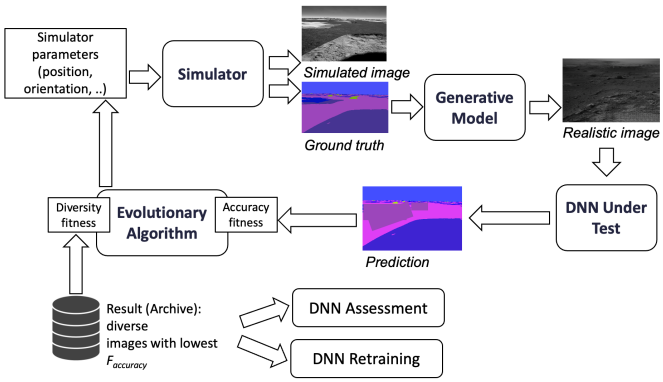

Fig. 1. The DESIGNATE Approach

for constructing situations that could have been missed or underrepresented in the training set despite being feasible. An example is DESIGNATE, which does not rely on existing data but generates novel, high-fidelity test inputs via GANs, enabling greater flexibility in testing. Compared to DeepHyperion [10] and DeepJanus [11], which also explore input space using simulation or symbolic constraints, DESIGNATE supports a realism-aware pipeline that integrates a GAN and simulator feedback loop.

Other works focus on the identification and characterization of failure scenarios but their support for testing is limited to the selection of images from existing datasets [12], [13]. Their integration with simulation approaches has been partially explored [14].

In the space domain, state-of-the-art simulators like PANGU [15]—used in ESA-funded projects for planetary surface simulation—are proprietary, require a specific license, and demand significant computer vision expertise and effort to adapt the environment or generate usable labels. Further, they may not provide all the required ground truth labels (e.g., segmentation is missing in PANGU). In contrast, MarsSim offers both ground-truth outputs and an accessible Python API, enabling automated testing workflows and seamless integration with other algorithms.

III. DESIGNATE Background

DESIGNATE is a search-based testing approach that uses GANs to create realistic test images. It extends the NSGA-II algorithm with an archive: in each iteration, candidate images are generated and evaluated according to a fitness functions that captures accuracy and diversity. Promising failure-inducing inputs are stored in the archive, which is used to ensure diversity. Figure 1 illustrates the DESIGNATE approach.

To test DNNs for semantic segmentation, two fitness components are used:

- Accuracy fitness: Measures the mean Intersection-over-Union (mIoU) between the DNN prediction and the ground-truth label. This score is minimized to uncover inputs that cause the DNN to fail.
- Diversity fitness: Encourages the generation of novel failure-inducing inputs by maximizing the distance in feature space between new candidates and existing ones in the archive.

The DESIGNATE framework supports several variants, each offering a different strategy for test input generation and evaluation:

- DESIGNATE: the approach described above.
- DESIGNATE$_{pix}$: Measures diversity directly in the image pixel space, which can be more sensitive to visual perturbations but less semantically meaningful.
- DESIGNATE$_{single}$: Relies only on the accuracy fitness (mIoU), focusing purely on generating inputs that degrade model performance without considering diversity.
- DESIGNATE$_{NoGAN}$: Disables the GAN refinement step, using raw simulator outputs as test inputs to assess the impact of realism on test effectiveness.
- Random: Serves as a baseline that samples test inputs randomly from the latent space, without guided search or fitness optimization.

IV. Tool Architecture

The *DESIGNATOR* tool provides a graphical user interface (GUI) that guides users through the setup, execution, and inspection of DNN testing results. Figure 2 illustrates the *DESIGNATOR* GUI showing the main window with controls for selecting search variant and parameters (top), the simulator display (middle), and visualizations of simulated, labeled, realistic, and predicted images (bottom).

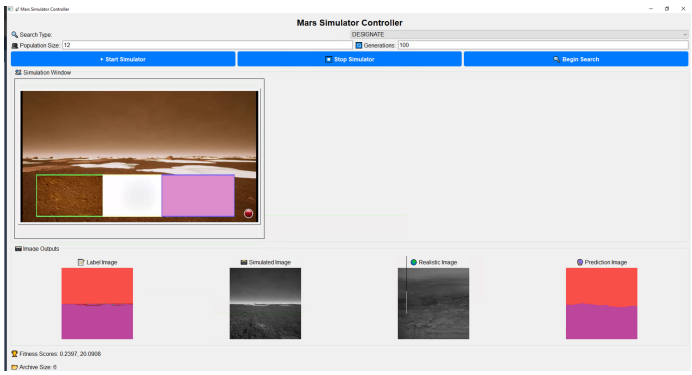

Fig. 2. *DESIGNATOR*'s Graphical User Interface

Figure 3 illustrates the *DESIGNATOR* components workflow. The numbered workflow steps are described below:

1) **Simulator Start:** MarsSim is launched automatically by *DESIGNATOR* and embedded within the GUI, enabling end-users to observe the inputs generated during testing.
2) **Parameter Settings:** The user selects the search variant and configures the parameters (i.e. number of generations, population size) on the GUI input page.

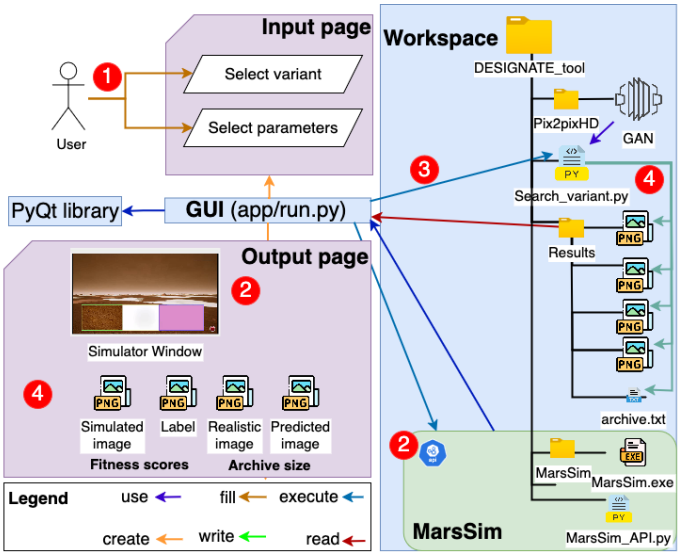

Fig. 3. *DESIGNATOR*'s architecture and outputs

3) **Search Start:** The start of the search process is triggered by the end-user; test generation algorithms generate candidate images, execute simulations, and evaluate DNN predictions.
4) **Results Saved Locally and Displayed:** The generated results—including simulated images, labels, realistic images, predicted images, and fitness values—are saved to the workspace. The last output added to the archive is displayed in the output page of the GUI for inspection and analysis. All results and metadata are stored at each search iteration for later inspection. The system also logs the outputs added to the archive, helping users track how search progresses across runs.

DESIGNATOR is implemented in Python, integrating PyTorch-based DESIGNATE components with a custom PyQt GUI. MarsSim is controlled via scripts triggered by the GUI. The GAN model used for image realism (i.e., Pix2PixHD [16]) operates on the synthetic outputs of MarsSim and transforms them into photorealistic Mars-like images; the GAN was trained on the subset of the AI4Mars [17] dataset used to train the DNN under test in our experiments. The tool architecture separates UI control logic from back-end modules for extensibility. The integrated simulator can be replaced by modifying the `search_variant.py` component.

V. Standalone MarsSim Simulator

To address the need for realistic, controllable Martian datasets, we developed MarsSim, a custom-built simulator constructed using Unreal Engine [18] and the AirSim plugins [19], the simulator requires Windows OS. MarsSim generates synthetic images that resemble those found in the AI4MARS dataset [17]: rocks, bedrock, sand, and soil, using textures, colors, and geometries derived from Mars imagery and consistent with the AI4MARS object classes.

MarsSim is designed as a standalone simulator and can be used independently from *DESIGNATOR*. This enables researchers to generate realistic Mars-like images independently of our search-based testing workflow.

We provide MarsSim as a GUI and a Python API, both can be used for Martian scenario generation, camera control, environmental manipulation, and automated image capture, making it suitable for integration with different search or learning algorithms. This extensibility positions MarsSim as a valuable resource for the wider machine learning and computer vision communities.

VI. Envisioned usage workflow

The user starts DESIGNATOR and selects a DESIGNATE variant. She configures key parameters like the number of generations and population size. She then clicks a button to launch MarsSim. Once the simulator is displayed and embedded in the GUI, the user clicks "Start Search" to begin the scenario generation process.

In each iteration, the simulator generates few simulated images, which are then passed through the GAN to produce realistic versions. The images are fed to the DNN, and the outputs (i.e., semantic segmentations) are evaluated. The fitness function, defined by the chosen DESIGNATE variant, scores the inputs. Promising inputs are archived, and the evolution continues.

For each iteration, one image from the archive is displayed in the GUI along with its ground truth label, the GAN-generated realistic image, and the DNN's predicted segmentation mask. The GUI also presents the accuracy fitness and diversity fitness values for this scenario, as well as the current size of the archive. The GUI updates in real-time, allowing users to visually inspect whether the DNN's prediction aligns with the ground truth and track the search progression.

The simulator can also be launched independently using a dedicated graphical user interface (GUI). Through this interface, users can interactively configure scene parameters such as camera position, orientation, and weather conditions, and visually explore the simulated Martian environment before capturing images or scenarios of interest. Furthermore, MarsSim is also provided as a Python-based API, enabling researchers to programmatically control the simulator and integrate it with custom algorithms or external pipelines.

VII. Empirical Evaluation

We used DESIGNATE to test DeepLabV3 DNNs trained to perform segmentation of Martian landscapes (when trained on the AI4MARS dataset) and urban landscape (when trained on CityScapes [20]); although, for copyright reasons, the released version of DESIGNATOR does not integrate the simulator used for the urban segmentation DNN (i.e., Airsim [21]), our results highlight the generalizability of the approach. Our goal was to assess whether the tool can generate realistic, failure-inducing

inputs and whether these inputs improve model robustness through retraining (details in ref. [3]). We summarize the findings by addressing the following research questions (RQs):

RQ1. Effectiveness for Test Generation. We compared DESIGNATE to variants and baselines, including random sampling, DeepJanus [11], and variants without GANs. Results show that DESIGNATE achieved the lowest median mIoU (0.49), outperforming random (0.64) and DeepJanusGAN (0.60). It also showed higher feature-based diversity (median = 14.75), indicating the ability to generate a broader range of failure-inducing scenes.

RQ2. Retraining Effectiveness. We used the generated images to retrain the DNN and evaluated its performance on the AI4MARS test set. The model retrained with DESIGNATE outputs achieved the highest mIoU (0.53), an improvement of +0.09 accuracy points over the original model. This confirms that the realistic and diverse inputs generated by *DESIGNATOR* contribute to more robust DNN performance.

These results demonstrate that *DESIGNATOR* can uncover critical failure scenarios in Martian environments and provide valuable data for retraining, validating its practical relevance.

VIII. Conclusion

We presented *DESIGNATOR*, a GUI tool implementing the DESIGNATE approach and integrating MarSim, a simulator of Martian landscape.

DESIGNATOR enables GAN-enhanced search-based testing of DNNs in Martian environments. It integrates realistic simulation, GAN-enhanced input generation, and fitness-guided search into a unified and interactive interface. Through its configurable design and real-time visualization, the tool facilitates the identification and analysis of failure-inducing inputs in object detection tasks on Mars-like terrains. Our evaluation demonstrated that our approach effectively generates realistic and diverse scenarios that challenge the target DNN, leading to tangible improvements in model accuracy when used for retraining.

The integrated MarsSim simulator can also be used as a standalone tool, through a provided API, thus supporting broader research and educational use of simulation-driven DNN testing beyond the simulators currently available for research, which focus on automotive.

IX. Data Availability

DESIGNATOR is available online at https://doi.org/10.5281/zenodo.15574895. MarsSim is available for standalone use at https://doi.org/10.5281/zenodo.16263373.

X. Acknowledgments

This project has received funding from the ESA discovery contract I-2022-02236 (TIA - Test Improve Assure, Activity No. 1000035943).

References

[1] NVIDIA Corporation, "Deep neural networks for autonomous vehicle landscape," http://bit.ly/41ZrkEz, 2017.

[2] J. Zhang, Y. Xia, and G. Shen, "A novel deep neural network architecture for mars visual navigation," *arXiv preprint arXiv:1808.08395*, 2018.

[3] M. Attaoui and F. Pastore, "Gan-enhanced simulation-driven dnn testing in absence of ground truth," *arXiv preprint arXiv:2503.15953*, 2025.

[4] M. O. Attaoui, F. Pastore, and L. C. Briand, "Search-based dnn testing and retraining with gan-enhanced simulations," *IEEE Transactions on Software Engineering*, 2025.

[5] Q. Hu, Y. Guo, X. Xie, M. Cordy, L. Ma, M. Papadakis, and Y. Le Traon, "Test optimization in dnn testing: a survey," *ACM TOSEM*, vol. 33, no. 4, pp. 1–42, 2024.

[6] K. Pei, Y. Cao, J. Yang, and S. Jana, "Deepxplore: Automated whitebox testing of deep learning systems," in *Proceedings of the 26th SOSP*, 2017, pp. 1–18.

[7] Y. Tian, K. Pei, S. Jana, and B. Ray, "Deeptest: Automated testing of deep-neural-network-driven autonomous cars," in *ICSE*, 2018.

[8] L. Ma, F. Juefei-Xu, F. Zhang, J. Sun, M. Xue, B. Li, Z. Su, Y. L. Wang, and J. Zhao, "Deepgauge: Multi-granularity testing criteria for deep learning systems," in *33rd ASE*, 2018.

[9] L. Baresi, D. Y. X. Hu, A. Stocco, and P. Tonella, "Efficient domain augmentation for autonomous driving testing using diffusion models," *arXiv preprint arXiv:2409.13661*, 2024.

[10] T. Zohdinasab, V. Riccio, A. Gambi, and P. Tonella, "Deephyperion: exploring the feature space of deep learning-based systems through illumination search," in *Proceedings of the 30th ISSTA*, 2021.

[11] V. Riccio and P. Tonella, "Model-based exploration of the frontier of behaviours for deep learning system testing," in *Proceedings of the 28th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*, 2020, pp. 876–888.

[12] M. O. Attaoui and F. Pastore, "Clusterxplain: a clustering-based tool for dnn components debugging," in *FSE Companion '25*. Trondheim, Norway: ACM, 2025.

[13] M. Attaoui, F. Pastore, and L. Briand, "Safe: Safety analysis and retraining of dnns," in *Proceedings of the 2024 IEEE/ACM 46th International Conference on Software Engineering: Companion Proceedings*, 2024, pp. 74–78.

[14] H. Fahmy, F. Pastore, L. Briand, and T. Stifter, "Simulator-based explanation and debugging of hazard-triggering events in dnn-based safety-critical systems," *ACM TOSEM*, vol. 32, no. 4, May 2023.

[15] Star-Dundee, "PANGU: Planet and asteroid natural scene generation utility," https://pangu.software/, 2024.

[16] T.-C. Wang, M.-Y. Liu, J.-Y. Zhu, A. Tao, J. Kautz, and B. Catanzaro, "High-resolution image synthesis and semantic manipulation with conditional gans," in *CVPR*, 2018, pp. 8798–8807.

[17] R. M. Swan, D. Atha, H. A. Leopold, M. Gildner, S. Oij, C. Chiu, and M. Ono, "Ai4mars: A dataset for terrain-aware autonomous driving on mars," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2021, pp. 1982–1991.

[18] Epic Games, "Unreal engine." [Online]. Available: https://www.unrealengine.com

[19] S. Shah, D. Dey, C. Lovett, and A. Kapoor, "Airsim: High-fidelity visual and physical simulation for autonomous vehicles," in *Field and Service Robotics: Results of the 11th International Conference*. Springer, 2018, pp. 621–635.

[20] M. Cordts, M. Omran, S. Ramos, T. Rehfeld, M. Enzweiler, R. Benenson, U. Franke, S. Roth, and B. Schiele, "The cityscapes dataset for semantic urban scene understanding," in *Proc. of CVPR*, 2016.

[21] Microsoft, "Airsim," https://github.com/microsoft/AirSim, 2024, version 1.8.1.