Project Number 101017258

D6.4 Recommendations for EDDI Repair and Hardening

Version 1.0
6 July 2023
Final

Public Distribution

University of York

Project Partners: Aero41, ATB, AVL, Bonn-Rhein-Sieg University, Cyprus Civil Defence, Domaine Kox, FORTH, Fraunhofer IESE, KIOS, KUKA Assembly & Test, Locomotec, Luxsense, The Open Group, Technology Transfer Systems, University of Hull, University of Luxembourg, University of York

Project Partner Contact Information

Aero41 Frédéric Hemmeler Chemin de Mornex 3 1003 Lausanne Switzerland E-mail: frederic.hemmeler@aero41.ch	**ATB** Sebastian Scholze Wiener Strasse 1 28359 Bremen Germany E-mail: scholze@atb-bremen.de
AVL Martin Weinzerl Hans-List-Platz 1 8020 Graz Austria E-mail: martin.weinzerl@avl.com	**Bonn-Rhein-Sieg University** Nico Hochgeschwender Grantham-Allee 20 53757 Sankt Augustin Germany E-mail: nico.hochgeschwender@h-brs.de
Cyprus Civil Defence Eftychia Stokkou Cyprus Ministry of Interior 1453 Lefkosia Cyprus E-mail: estokkou@cd.moi.gov.cy	**Domaine Kox** Corinne Kox 6 Rue des Prés 5561 Remich Luxembourg E-mail: corinne@domainekox.lu
FORTH Sotiris Ioannidis N Plastira Str 100 70013 Heraklion Greece E-mail: sotiris@ics.forth.gr	**Fraunhofer IESE** Daniel Schneider Fraunhofer-Platz 1 67663 Kaiserslautern Germany E-mail: daniel.schneider@iese.fraunhofer.de
KIOS Maria Michael 1 Panepistimiou Avenue 2109 Aglatzia, Nicosia Cyprus E-mail: mmichael@ucy.ac.cy	**KUKA Assembly & Test** Michael Laackmann Uhthoffstrasse 1 28757 Bremen Germany E-mail: michael.laackmann@kuka.com
Locomotec Sebastian Blumenthal Bergiusstrasse 15 86199 Augsburg Germany E-mail: blumenthal@locomotec.com	**Luxsense** Gilles Rock 85-87 Parc d'Activités 8303 Luxembourg Luxembourg E-mail: gilles.rock@luxsense.lu
The Open Group Scott Hansen Rond Point Schuman 6, 5th Floor 1040 Brussels Belgium E-mail: s.hansen@opengroup.org	**Technology Transfer Systems** Paolo Pedrazzoli Via Francesco d'Ovidio, 3 20131 Milano Italy E-mail: pedrazzoli@ttsnetwork.com
University of Hull Yiannis Papadopoulos Cottingham Road Hull HU6 7TQ United Kingdom E-mail: y.i.papadopoulos@hull.ac.uk	**University of Luxembourg** Miguel Olivares Mendez 2 Avenue de l'Universite 4365 Esch-sur-Alzette Luxembourg E-mail: miguel.olivaresmendez@uni.lu
University of York Simos Gerasimou & Nicholas Matragkas Deramore Lane York YO10 5GH United Kingdom E-mail: simos.gerasimou@york.ac.uk nicholas.matragkas@york.ac.uk	

Document Control

Version	Status	Date
0.1	Document outline	15 May 2023
0.2	First draft	05 June 2023
0.3	First full draft	20 June 2023
0.4	Further editing draft	27 June 2023
0.5	Version 1	28 June 2023
0.6	Revision after review	29 June 2023
0.7	Final revision after review	30 June 2023
1.0	Completed QA version	6 July 2023

Table of Contents

1 INTRODUCTION **1**

1.1 OVERVIEW . . . 1

1.2 REPAIRING AND HARDENING OF DEEP LEARNING COMPONENTS IN SESAME: GOALS, CHALLENGES, AND OUTCOMES . . . 1

1.2.1 Algorithmic and Computational Challenges . . . 2

1.2.2 Domain Specific Challenges . . . 3

1.3 INTEGRATION IN THE EDDIS . . . 4

1.4 BENEFITS TO THE PROJECT AND DESIRED INDUSTRIAL IMPROVEMENTS . . . 5

2 RELATED WORK **7**

2.1 Deep Learning-based Software Testing . . . 7

2.1.1 Test Adequacy . . . 8

2.1.2 Fuzzing Testing in DNNs . . . 8

2.2 DEEP LEARNING SOFTWARE REPAIRING . . . 9

2.3 DATA AUGMENTATION FOR DEEP LEARNING . . . 10

2.4 GENERATIVE AI: TEXT-GUIDED DIFFUSION MODELS . . . 10

3 SYSTEMATIC REPAIR AND HARDENING FOR EDDI DATA-DRIVEN COMPONENTS **13**

3.1 APPLICATION SCENARIO: AUTONOMOUS PEST MANAGEMENT IN VITICULTURE . . . 13

3.2 METHODOLOGY OVERVIEW . . . 14

3.3 GENERATIVEFUZZER: SEMANTIC DATA AUGMENTATION FOR SAFETY ASSURANCE . . . 16

3.3.1 Problem Formulation . . . 16

3.3.2 Seed Selection . . . 18

3.3.3 Semantic Data Augmentation Techniques . . . 18

3.3.4 Fidelity Estimation: Photo-realism of Synthetic Images . . . 23

3.4 SAFETYREPAIR : A COVERAGE-GUIDED REPAIRING METHODOLOGY . . . 23

3.4.1 Coverage Tracing . . . 25

3.4.2 Continual Learning-based Repairing . . . 25

3.4.3 Repairing Process . . . 27

4 EXPERIMENTAL STUDY **29**

4.1 RESEARCH QUESTIONS . . . 29

4.2 IMPLEMENTATION . . . 29

4.3 EXPERIMENTAL DESIGN AND SETTINGS . . . 29

4.3.1 Subject datasets and DNN models . . . 30

5 Results and Discussion 31

5.1 RQ1: Photo-realism . . . 31

5.2 RQ2: Effectiveness . . . 32

5.3 RQ3: DNN Repairing . . . 33

5.4 Threats to Validity . . . 38

6 Satisfaction of SESAME Requirements 39

7 Conclusion 40

List of Figures

1 The process of integrating GENERATIVEREPAIR quality assurance results into the EDDIs . . . 5
2 Illustration example of collaboration between SESAME WPs for safety assuring of MRS trajectory planning and tracking for fungicide spraying in DKOX's viticulture use case. 6
3 A zoom into individual neuron function in a fully connected DL system that receives inputs from MRS sensors (camera, LiDAR, pressure sensors) and outputs a decision for robot arm usage and navigation 7
4 Taxonomy of data augmentation approaches [4]. 11
5 A directed graphical model as a simplified illustration of the probabilistic diffusion model (Source: [52]) 12
6 Illustrative example of DNN output needed for fungicide spraying inspired by the YOLO architecture [5] with a CNN backbone and object localisation and classification heads. The input image is a high-resolution representation of a grape leaf with the Esca disease. 14
7 Unlabelled real-life high-resolution vineyard images dominated with the ESCA disease. 15
8 The overall process of the GENERATIVEREPAIR approach for hardening and repairing DNNs. 16
9 GENERATIVEFUZZER pipeline. 17
10 Semantic data augmentation workflow. In blue, we denote all processes implemented to guarantee the semantic augmentation process validity. 19
11 Overview of the proposed Semantic Occlusion method, which does include a semantic segmentation example (based on YOLO dataset) for illustration. 20
12 Images synthesised using GENERATIVEFUZZER with Stable Diffusion Inpainting as the augmentation technique. The invalid sample is identified using the Fidelity Estimation process. . . 21
13 Images synthesised using GENERATIVEFUZZER with Stable Diffusion Inpainting as the augmentation technique. The initial dataset used as a reference is COCO with superclass 'animal'. 24
14 Illustrative example of different continuous learning paradigms adapted to the reality data gap between disease detection tasks. 26
15 Performance of different augmentation strategies on the Grape Leaves dataset using the SSIM score. 31
16 CLIP vs Scaled FID scores on 512x512 augmented images using the Stable Diffusion Inpainting technique. As shown, FID (lower shows better quality images), and CLIP (higher shows better quality images) quality assessment matches for most augmented images with Inpainting techniques. 33
17 Improvement in accuracy of (a), (b) and (c) DNNs when the model is re-trained with augmented sets with the same number of inputs generated by different data augmentation strategies. 35

List of Acronyms

AI	Artificial Intelligence
DL	Deep Learning
DNN	Deep Neural Networks
EDDI	Executable Digital Dependability Identity
InDD	In-Domain data Distribution
ML	Machine Learning
GenerativeRepair	Repair and Hardening Platform for Deep Learning systems
GenerativeFuzzer	Coverage-Guided Fuzzing tool-enabled technology
SafetyRepair	Repairing tool-enabled technology
OOD	Out-Of-Domain dataset
TCC	TrKnw neurons cluster combinations

Executive Summary

Deep Learning models (DLs) are the most deployed data-driven and learning components across significantly diverse applications and Multi-Robot System (MRS) architectures in SESAME use cases (e.g., for person detection and image recognition in PAL and Locomotec use cases). In order to engineer a methodology that assures the quality of MRS at both design time and runtime, a holistic approach that supervises and repairs any faulty behaviour is required.

In this deliverable, the tool-supported methodology for hardening and repairing DLs is introduced. The proposed methodology is developed by leveraging the assurance results obtained from the techniques developed in Tasks 6.1 and 6.2.

Task 6.3 Recommendations for EDDI Repair and Hardening.

On the implementation front, we pay special attention to developing our tool in a simple multi-paradigm technology, i.e., using a set of Python components (scripts). This enables quick adaptation and code maintenance by our partners. The developed prototype tool is also compatible with most systems and architectures used by our industrial partners.

Structure of the Document.

This document is structured as follows:

- We provide a general overview of the software we developed and its connection with the SESAME use cases in Section 1;
- We give an overview of the used technology, including an overview of existing approaches and their relation to repairing Deep Learning-based software systems in Section 2
- We present the architecture and methodology of GenerativeRepair in Section 3;
- We detail the research questions we address in our experimental study and the experimental setup in Section 4.
- We present and discuss the obtained results in Section 5;
- We present how we satisfy the project requirements in Section 6
- We conclude the report in Section 7.

1 Introduction

1.1 Overview

This document summarises the activities concerning the Repair and Hardening of Data-driven and Learning Components aspect of the project. In particular, our focus is on Deep Learning (DL) models and algorithms used in any of the MRS components (e.g., classification, detection). The work has focused on the development of practical and useful recommendations, following the quality assurance results obtained from the techniques designed in Tasks 6.1 and 6.2, to produce a set of techniques and steps to improve and harden the Executable Digital Dependability Identities (EDDIs). Our recommendations are supported by a prototype tool for data augmentation and repairing DL components deployed as part of the SESAME project.

This deliverable reports the work carried out within Task 6.3, focusing on the description and integration of the DL hardening and repairing approaches developed in Work Package 6. The key idea underpinning the approach, called GenerativeRepair, is the use of Generative AI technology for semantic data augmentation to (i) improve the diversity of the DL test set and achieve higher testing coverage; and (ii) introduce unknown (likely failure) patterns into the training data set to repair DL models via fine-tuning and re-training strategies. Taken together, these steps enable the development of more robust DL models, and, consequently, more resilient and trustworthy EDDIs.

In the following sections, we first describe the motivation that guides the development of the proposed approach alongside the assumptions and hypotheses underpinning our recommendations discussed in more detail later on in this document.

1.2 Repairing and Hardening of Deep Learning Components in SESAME: Goals, Challenges, and Outcomes

This deliverable reports on the developed tool-supported methodology for the repairing and hardening of DL faulty behavior reported at run-time. More specifically, the methodology encompasses a set of approaches and techniques developed to tackle specific failure behaviours of DL components of the MRS exhibited at runtime. In addition, it focuses on improving and hardening the EDDI for fault diagnosis.

The goal of Task 6.3 is twofold: 1) to develop a set of recommendations to harden and repair the Deep Learning parts of EDDI components; and 2) to devise a methodology that turns these recommendations into an actionable set of practices for repairing Deep Learning models.

At a conceptual level, the recommendation takes the form of (i) a new algorithm that improves the robust generalization of DNNs by augmenting existing test suites using coverage-guided test generation; and (ii) a re-training approach to address the semantic data augmentation problem for the DNNs under realistic environmental variations.

Based on our findings from Tasks 6.1 and 6.2, our work targets robustness issues arising from natural, environmental perturbations affecting the DL-based systems prediction performance, resulting in hazardous situations. Due to the black-box nature of DL, it is rather challenging to interpret and properly repair these incorrect behaviours through amendment operations on the program source code level. Instead, our approach focuses on the data level and simulates the real-world novel/corner/or even noise patterns in the operational environment. The intuition behind our solution is that semantic augmentation operations could cover unknown data patterns, allowing to repair the DL progressively and removing the faulty behaviour that is responsible for the increased DL uncertainty at runtime, thus contributing to a more robust EDDI.

Inspired by the emergence of generative AI models for input synthesis using latent representations [22, 50], we posit that leveraging these models can inform the generation of diverse and realistic synthetic inputs that capture the underlying variability of the real-world patterns in the operational environment. This is the core principle underpinning our work.

1.2.1 Algorithmic and Computational Challenges

The revival of deep neural networks (DNNs) and the availability of high computational power laid the foundation for their massive recent success in highly complex recognition tasks like image classification [60], object detection [69] and natural language processing [55]. DNNs have seen exponential adoption in various application domains, including autonomous driving [6] and multi-robots system applications [38].

Regardless of the huge advances in DL capabilities, serious concerns about the perceived reliability, robustness, and explainability of DL, especially when used as key components within autonomous and robotic systems [1, 24]. These concerns arise hand in hand with the number of accidents of various severity levels that threaten the safety of human lives [10].

In fact, quality assurance of DL-based software is an unmistakable Software Engineering (SE) problem where automated repairing tools could be beneficial. Despite its potential, challenges to hardening and repairing DL faulty behaviour are numerous. We briefly present below the most representative challenges related to the task presented in this deliverable.

Challenge 1 – Identify DL Faulty Behaviour: A major DL challenge is that the reasoning underpinning DL models is not fully understood nor how its faulty behaviour arises. Test oracles in traditional SE testing assume that the software's output can be verified against the expected values designed by the developer. This is not applicable in the case of DNNs. DNNs are "untestable software", with complex logic that is hard to manually encode. As a consequence, DNNs behaviour is largely unexplainable, and their faulty (i.e., erroneous) behaviour cannot be easily predicted. Subsequently, testing engineering experts raised questions about common bugs, i.e., failure patterns, in DL-based software: *How can we identify these common bugs? Can we generate test oracles to identify them?*

Answering these questions has the potential to fuel Deep Learning research on bug detection, repair, and quality assurance for DNNs in general [70].

As described in deliverable 6.1 [43], in Tasks 6.1 and 6.2, we investigated situations where a robot's DL onboard system could incorrectly "interpret and use" the data sent from a sensor (i.e., camera), making the robot unable to detect an obstacle or to correctly navigate in the natural environment. This led to the delivery of a Test Adequacy criterion (termed DEEPKNOWLEDGE) that is used in Task 6.3 to identify common bugs in such software and guide the automatic generation of test inputs that reveal these bugs, i.e., failure cases, through a coverage-guided fuzzing methodology.

Challenge 2 – Generate Repair Patterns: A key ingredient in achieving impressive DL results is the availability of large volumes of high-quality annotated datasets that adequately encode the characteristics of the target domain [33]. Within a typical DL pipeline, this data enables the construction of high-dimensional representations during training, and instruments the robustness and generalisability assessment during testing [19]. Data scarcity poses a major challenge in devising robust and competitive DL models [51]. This is particularly important in domains such as security, where relevant data is not typically available because of privacy considerations or simply because such data does not exist. Data augmentation aims at alleviating this issue by automatically synthesising new data informed by the available dataset [58]. In the context of DNN repairing, data augmentation is mostly deployed to generate failure cases for DNN retraining. Existing data augmentation techniques focus on simple geometric and colour space transformations, like noise, flipping and resizing producing datasets with limited diversity. However, in real-world settings, failure cases are the results of mismatching between the training dataset and real-world data due to real-world factors, e.g., weather, blur, unseen corner cases, etc. The challenge here is: "How can we synthetically generate failure cases that are semantically representative of natural environmental variation?"

Challenge 3 – Repairing Strategy: DL repairing aims at fixing faulty DNN architectures (e.g., weights, layers) based on optimization criteria. In the traditional software repairing literature, research efforts have been restricted to the debugging process [25]. This process requires analyzing and understanding failed executions, identifying the causes of the failures, implementing fixes, and validating them [20].

Such a process can not be applicable in the case of DL because of the lack of transparency, the complexity of the computational mechanism, and the unpredictability of DNN decision logic [70]. DNN repairing strategies are distinctive compared to traditional debugging and bug-fixing techniques. Overall, the repairing process aims to harden the DL-based system, by improving the robust generalization of the DNN model underpinning it. The most common repairing strategies are (i) retraining; (ii) iterative weight adaptation; and (iii) pruning. These strategies have different potential benefits in DNN repairing for incorrect behaviors. However, the challenge here is: "How can they reduce the DNN vulnerabilities (i.e., improving accuracy for the identified failure cases) without harming the model performance on normal data?

In this deliverable, we review existing repairing methods in DNN and determine the strategy that could best achieve this balance.

To overcome the aforementioned challenges, in this deliverable, we propose two distinct but complementary approaches for DL-based software hardening and repairing that are detailed in Sections 3.3 and 3.4.

1.2.2 Domain Specific Challenges

In the context of SESAME project and its use cases, hardening and repairing Deep Learning components (and any Machine Learning algorithm) is fundamental to remain compliant with the safety and security requirements and allowing the MRS system to navigate dependably in the natural environment.

In particular, our industrial partners, when deploying Machine Learning models to production, face challenges in maintaining the model's accuracy level. This is partly due to:

- The need to process massive amounts of data in real-time;
- The difficulty to adapt to shifting data patterns (data or concept drift) in the deployment environment;
- The difficulty to collect relevant data from various data sources for continuous repairing.

Overall, the significant DL deployment problem is that business applications need to handle enormous amounts of varying real-time data. At run-time, DL components can be surprised [54] by a new data type that is semantically different from the training dataset. In other words, the DL component could encounter a reality gap issue that as a consequence results in a gradual decline in the model's accuracy. While, DL must meet adequate response times, along with supporting a large number of functionalities across MRS, the repairing process can slow down such a role. This is due to the hundreds or thousands of new data patterns that need to be used for the DL repairing and updating process.

Tackling the DL vulnerability issues in the deployment environment is an expensive, laborious and resource-intensive process. For instance, recent studies showed that repairing activities often account for about 50 percent of the overall development process. Even with the rise of MLOps technologies, DL repairing is still time-consuming and can not be performed efficiently during runtime [29]. Our methodology of hardening and repairing DL aims to tackle the problem related to data shift and prevent the reality gap issue through a novel coverage-guided augmentation strategy. We leverage a semantic data augmentation strategy that augments an existing test set by extending it with mutant inputs that increase its semantic diversity. Hardening and repairing a DL-based system using this synthetically generate data help prevent the aforementioned problems and, as shown in our experimental evaluation, reduce them to a certain extent.

1.3 Integration in the EDDIs

This section describes how GenerativeRepair can be integrated into an EDDI, also explaining how the technique will allow for more efficient and effective operation while minimizing the overall risks of deploying DNN components and assuring their safety.

As shown in Figure 1, GenerativeRepair is a design time activity, focusing on hardening and repairing DL components (step 4 in the figure) based on the results of safety assessment activities (step 2). To help address the challenge of DL robustness and correctness, the GenerativeRepair is developed to support MRS applications, e.g., in navigation or vision task (computer vision operations for object detection), by simulating the runtime configurations. Given that DL components are generally tested at design time using testing datasets (*in-distribution domain data* – InDD) and with configurations that cannot be predicted or analysed at runtime, their evaluation can be faulty. In fact, this evaluation may drift if data encountered at runtime does not match what was used at design time. This can result in runtime discrepancies and consequently significantly worse performance discrepancy than intended, leading to possible safety implications.

GenerativeRepair addresses this anomaly using semantic data augmentation principles to simulate the number of corner cases, where the model fails to make correct predictions. These corner cases can be caused by the noise factors in the operational environment, which causes hazardous situations that often result in system failure, i.e., erroneous behaviours. Expanding the training and testing datasets with samples of these corner cases and semantically meaningful inputs enables us to re-evaluate and fine-tune the DL model. These are key objectives of the GenerativeRepair approach.

The GenerativeRepair is part of a task manager that deploys the test adequacy estimation as the fault-revealing mechanism (output of step 2 in Figure 1), and uses it to guide the selection and generation of synthetic images that simulate the corner cases that can be encountered during runtime.

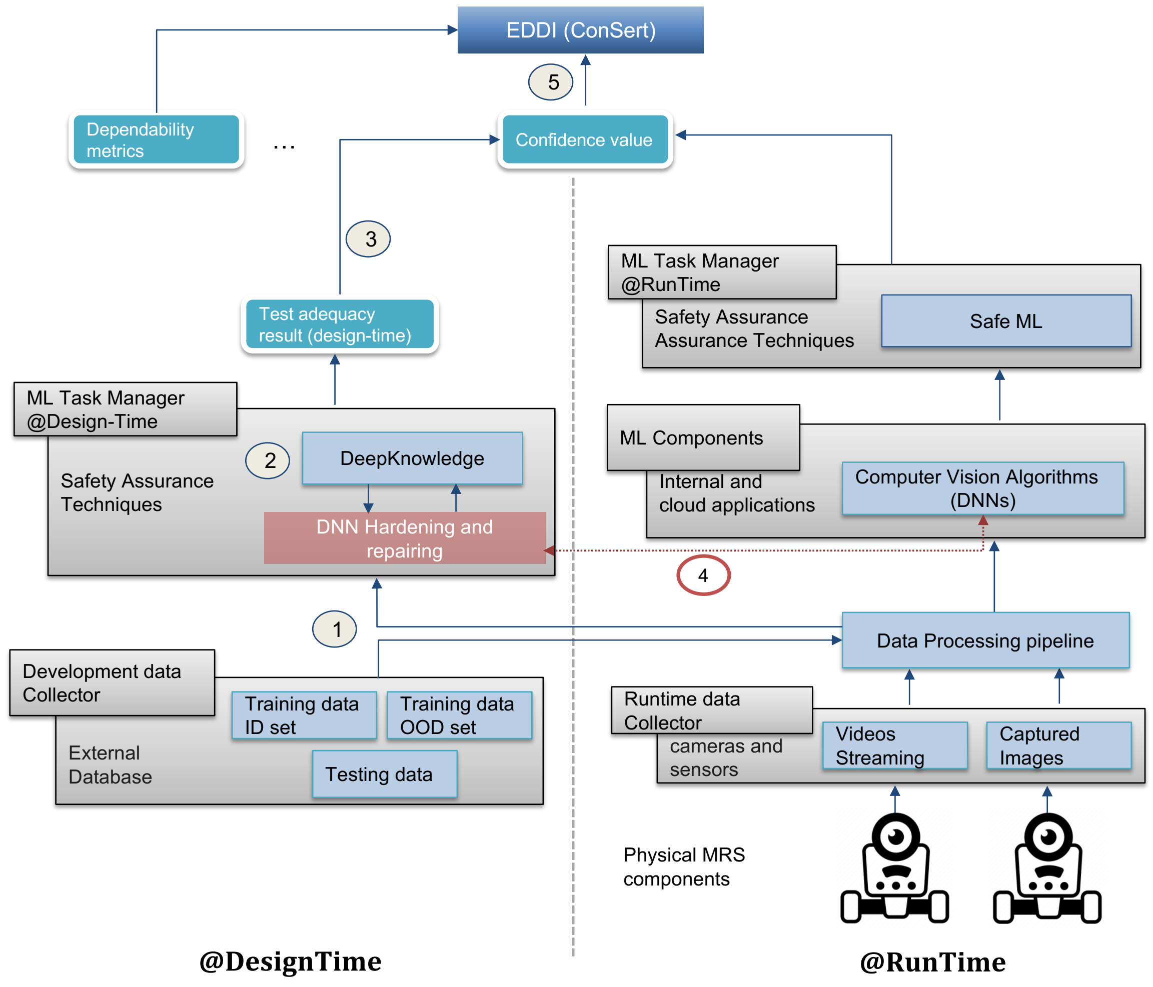
First, our methodology focuses on collecting instance-wise errors, which are single inputs that result in a DNN model's erroneous outputs, that are found to be related to many real-world errors without malicious attackers. As detailed in Figure 1, the estimated coverage, i.e, test adequacy, of the DNN component using testing datasets during design time (steps 1, 2, and 3), serves as a metric to guide the Hardening and Repairing process.

In particular, the proposed approach leverages multiple extensible coverage criteria (from Task 6.1 and 6.2) as feedback to guide the new data samples generation. It uses a semantic data augmentation methodology based on generative AI models (Stable Diffusion) to generate new semantically diverse tests which could simulate corner cases that cannot be predicted correctly, so that the DL model raises errors during usage. The augmented data is then used to retrain and finetune the DL model in a sequential-based strategy to expand its generalisability and enhance its performance on corner cases, which alleviates to a certain extent the system failure issue during runtime.

In fact, the repaired model is used to update the computer vision capability of MRS. Then, continuously a confidence metric is being estimated to assess the prediction performance of the new DL for each outcome at runtime (step 5 in Figure), using DeepKnowledge approach developed in Task6.1 alongside the SafeML estimation. This information can then be used to make runtime decisions about safety, e.g., to generate warnings for the operator (step 5).

Figure 2 illustrates a concrete example of the usability of GenerativeRepair assurance results in collaboration with other WPs to assure the safety of fungicide spraying tasks.

In DKOX's viticulture use case, a robotic team will spray chemical compounds in vineyard fields. This task needs to be guaranteed that no drone will get in contact with surfaces outside the area of interest, objects, or people that happen to be within the intended area. In this scenario, object detection algorithms are deployed in a way that a certain level of safety is guaranteed. The safety of trajectory for fungicide spraying is assured through a collaboration between WP2, WP3, and WP6. When obstacles are detected within the flight path (using ML components for object detection), the WP2 updates the trajectory and provides robots with an optimal object bypass. This prediction is evaluated and its correctness is assured using GenerativeRepair hardening results.



.

Figure 1: The process of integrating GENERATIVEREPAIR quality assurance results into the EDDIs

1.4 Benefits to the Project and Desired Industrial Improvements

To support the end-term goal of assuring the quality of MRS and the correct behaviour of their components, especially in safety-critical systems, we propose a hardening and repairing methodology that should be able to enable the development of more robust DL components for the target robotic systems.

The intention is to provide the user with confidence that DL components used for vision and navigation during run-time are conformant, in the sense that the model inference is correct and reliable, to provide a critical decision for the EDDI at runtime. By identifying DL-based system faults using quality and safety assuring techniques from Task 6.1 (DEEPKNOWLEDGE), 6.2, and Task 7.2 (safeML), we developed a holistic methodology that encompasses two distinct but complementary approaches easily integrable in any MRS platform. The methodology offers (i) a cost-effective way of generating semantically relevant data for DL repairing; and (2) a retraining strategy practical for continuous repairing effectively with small data samples.

While manually collecting and preparing data is inefficient and can lead to suboptimal results, our synthetic data augmentation approach (1) automatically generates high-fidelity data inputs allowing for saving time, and preventing human-induced errors. In this way, our partner software engineering teams can engage in more value-added efforts rather than repetitive tasks. Next, by incorporating our repairing approach (2) into the EDDI, we make DL inference during run-time more reliable, which offers a good trade-off between repaired network accuracy and time consumption, compared to traditional (data augmentation-based) repairing tech-

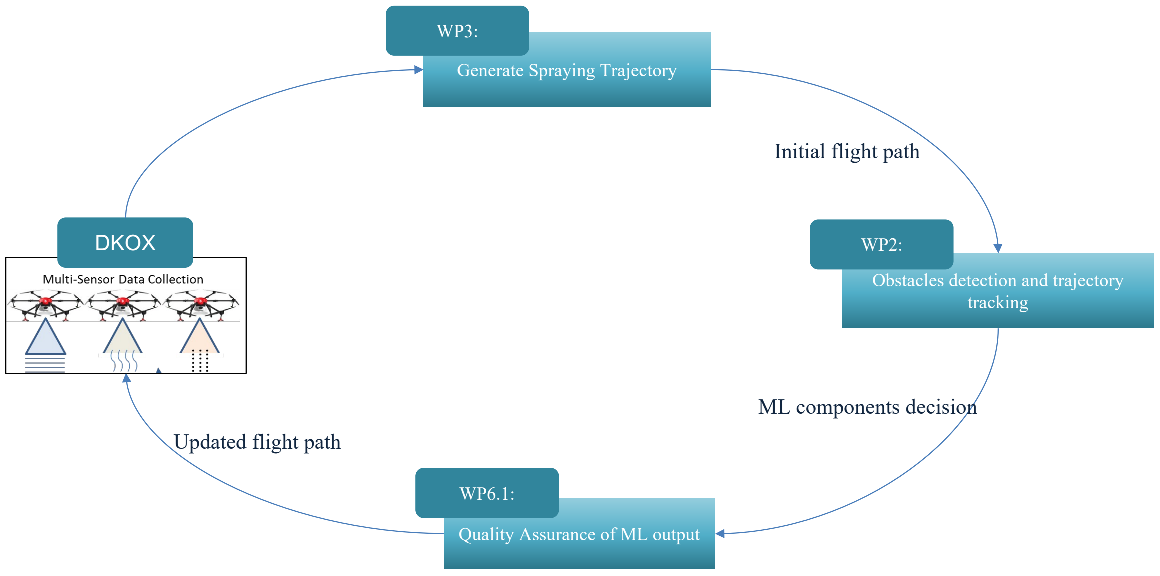

WP3:
Generate Spraying Trajectory
Initial flight path
WP2:
Obstacles detection and trajectory tracking
ML components decision
WP6.1:
Quality Assurance of ML output
Updated flight path
DKOX
Multi-Sensor Data Collection

Figure 2: Illustration example of collaboration between SESAME WPs for safety assuring of MRS trajectory planning and tracking for fungicide spraying in DKOX's viticulture use case.

niques. It is able to achieve a substantial improvement in repaired accuracy on each of the augmented datasets. The magnitude of this improvement is 26% average with no manual labelling effort for the newly generated inputs needed.

It's worth mentioning that our methodology is universal and could be applied to any DNN architecture-data combination. We provide a tool to support the easy integration of this methodology, which is developed as a set of Python scripts that can be easily deployed and does not require hands-on experience in DL development or testing. Concerning the domain-specific challenges, our tool satisfied our partners' requirements as described in Section 6. Finally, in Section 5, we provide empirical evidence for the effectiveness of the proposed methodology.

2 Related Work

In this task, we study data-driven components of MRS and/or EDDIs with a main focus on Deep-Learning (DL) based systems. These systems are based on deep neural networks (DNNs), and have multiple potential applications in MRS, such as dynamic task allocation [14], distributed tracking control [72], navigation control [62], etc.

While our approach is independent of the particular application scenario or the undertaken tasks by the computer vision component, to simplify the conceptual presentation of DNN, we restrict our description to the classification tasks as follows:

Formally, a deep neural network N comprises n layers sequentially composed such as $N = f_n \circ ... \circ f_1$. Each layer l is assigned an activation function $a_\sigma$and learnable parameters θ consisting of a weight matrix W and a bias vector b. Thus, the output of the ith layer $l^{[i]}$, is a function $f_i : \mathbb{R}^{i-1} \rightarrow \mathbb{R}^i$, with $f_i(Z) = a_\sigma(W_i Z + b_i)$. Each of these layers is composed of a finite set of neurons, i.e., computational units, that's as shown in Figure 3 is behind the final output of f_i.

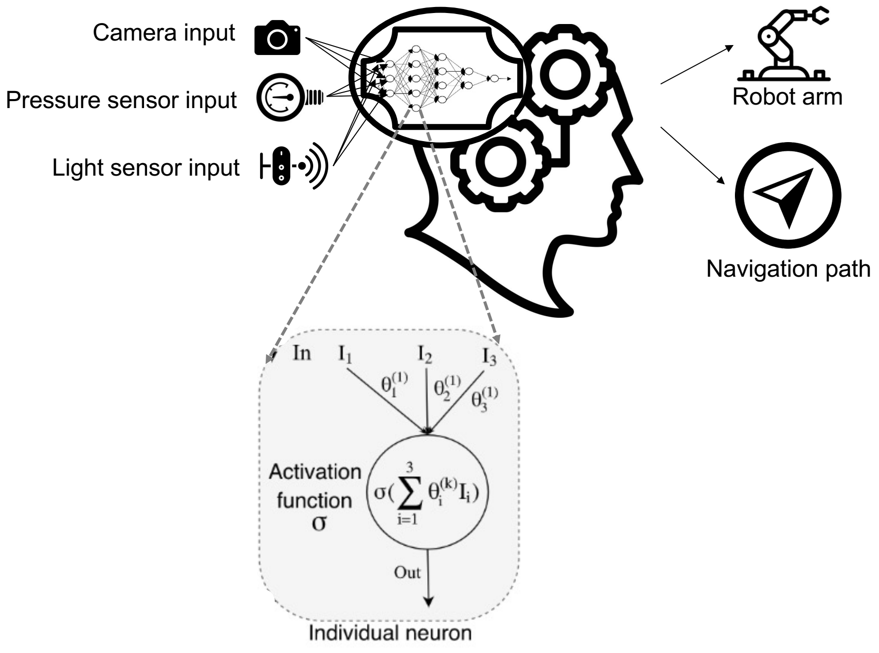


Figure 3: A zoom into individual neuron function in a fully connected DL system that receives inputs from MRS sensors (camera, LiDAR, pressure sensors) and outputs a decision for robot arm usage and navigation

The DNN training process relies on the dataset fed into the model to learn and update the θ parameters in a gradient-decent fashion. It uses unconstrained optimization of a loss function [23]. It is important to note that DNN is a data-driven programming paradigm, that highly depends on the data and relies on a non-convex function, i.e., the loss optimization function. This characteristic keeps the DNN artifact in the form of an encoded model that is particularly hard to debug. Such unique features of DNN models bring unique challenges to the quality assurance of DL-based software as we discuss in the upcoming section.

2.1 Deep Learning-based Software Testing

Deep Learning (DL) testing is a growing field of research. Unlike software testing [46], which is accompanied by a well-defined methodology, encompassing validation and verification steps, DL testing is more challenging, reflected in the difficulty of creating detailed system specifications against which the DL model behaviour can be checked. However, software testing such as code coverage, test oracle, and coverage-guided fuzzing (CGF) have widely inspired the DL community. We provide below a careful analysis of relevant work.

2.1.1 Test Adequacy

Testing is a major approach to DL-based software quality assurance. Thus, a growing core of research has focused on white box testing for DL, including coverage criteria like neuron coverage (NC) [45], DeepGauge [37], DeepGini [17], DeepImportance [21], and Surprise Adequacy [31]. This recent research has developed and pioneered research in DL testing.

In particular, Neuron Coverage (NC) [45] captures the neuron's state and classifies it into activated and non-activated. This state is determined based on the activation output value and the data input ability to activate this value beyond a certain predefined threshold. As a result, NC measures the ratio of activated neurons of a DNN, i.e., the ability of a suite of test cases to activate the DNN computational units.

The k-Multisection Neuron Coverage (KMNC) [37] has also adapted the same principle of activation values, but instead of the range of these values (based on training data), they are partitioned into k sections. As a result, KMNC measures the ratio of all covered sections of all neurons of a DNN model.

Similar to KMNC, Strong Neuron Activation Coverage (SNAC) and Neuron Boundary Coverage (NBC) analyses the value range of a neuron covered by training data [37]. They tailored the approach to measure to what extent the corner-case regions of neurons can be covered by a test set. The difference between both test adequacy criteria is that SNAC considers the value range that is above the maximum value seen during training.

Top-K Neuron Coverage (TKNC)is a layer-level testing criterion, which measures the ratio of neurons that have once been the most active k neurons of each layer on a given test set. While Importance Driven Coverage (IDC) also focuses on layer-level [21], it is an explanation-based criterion that only focuses on measuring the activation values of important neurons responsible for the prediction of the DNN model.

All these techniques have successfully tested the ability of the test set to capture the model's erroneous behavior by identifying, based on different criteria, the parts of DNN logic exercised by a test suite and thereafter used this understanding as a test adequacy criterion.

Coverage measures have become a reference in DNN testing because of the significant advances over manual ad-hoc testing and its effectiveness in detecting the diversity of test inputs. Also, all these techniques provide useful insights about the possible types of faults, i.e., the inputs responsible for the failures (erroneous behaviour).

2.1.2 Fuzzing Testing in DNNs

Coverage-Guided Fuzzing is a widely adopted principle in software engineering that has been employed as an effective testing technique to detect bugs in various software applications. The technique focuses on maximizing code coverage to reveal more bugs during fuzz testing [61].

Coverage-Guided Fuzzing principles have been applied successfully to DL testing aiming at identifying bugs, i.e., erroneous behaviour, in real DNN-enabled systems [11]. This mechanism has been adapted to DL testing where it mainly discovers defects in the model that maximize different coverage criteria such as Neuron Coverage [45] instead of code coverage. Another noteworthy set of related recent advances are DeepHunter [64], DeepTest [57] and TensorFuzz [42] which employ fuzzing to generate tests through metamorphic transformations with the intention that the new test and its original (seed) share the same semantics from a human perspective.

Given a DNN model, all these approaches select a set of initial seeds from the in-domain data as the input. Then, while maintaining a seed queue, they generate passed tests that maximize the coverage criteria, and failed tests that are incorrectly predicted by the DNN model.

In contrast to greybox fuzzing [18], coverage-guided fuzzing allows controlling the seed selection and mutation based on application-specific criteria. In this way, the generated test cases can effectively and efficiently achieve certain testing goals, i.e., achieve certain coverage thresholds. Depending on the scenario, the generated tests are metamorphically mutated using geometric and colour-space transformations, e.g., flipping, cropping. The resulting test cases are used as new test cases.

In our work within the context of the SESAME project, we study the advantage of deploying semantic mutation techniques to generate more effective and semantically diverse test cases. Our methodology adopts generative AI models (i.e., diffusion methods) to enable the integration of efficient manipulations in the data latent space using meaningful semantic transformations, e.g., changing the background, adding or replacing an object, etc. Specifically, these transformations result in a feature representation corresponding to a novel sample with the same class identity and different semantics, or completely novel class objects. The generated image samples effectively expand the dataset semantics, leading to more diversity for augmented samples.

2.2 Deep Learning Software Repairing

Despite the high performance among the DL-based systems, the underlying DNN models still make misclassification and output wrong predictions [44]. To fix the incorrect DNN behaviour, multiple methods have been proposed, mostly inspired by traditional software repairing paradigms, for instance (i) software healing; and (b) software repairing [29]. Software healing (i.e., a word similar to hardening) targets runtime failures on the deployed DNN applications by first detecting bugs or errors and making modifications to rectify them without interfering with the system code, i.e., the model architecture. The coverage-guided fuzzing techniques mentioned above fall under this category. The software repairing paradigm introduces amendment operations on the program source code level to remove the fault that causes failures at runtime. Conceptually most of the DNN re-training and fine-tuning approaches fall under this category.

One of the recent approaches is SENSEI [19] which focuses on employing mutation-based fuzzing for data augmentation of DNNs to improve the model generalization. The approach targets robustness issues arising from environmental perturbations. It leverages genetic algorithms to guide the selection of data samples from the set of natural environmental variants as inputs for the data augmentation process. SENSEI relies on metamorphic operations for optimal data augmentation by introducing natural perturbations.

Similar to SENSEI, the Arachne approach [53] deploys a search-based algorithm guided by a fitness function following the “Generate and Validate” automatic program repair paradigm. The advantage of this approach is its ability to update the network weights without retraining the model.

In the same spirit, DeepRepair [68] casts the problem of DNN repairing through a style transfer-guided data augmentation method. The method simulates failure noise patterns in the operational environment and introduces them into the training data for DNN repairing.

RNNRepair [63] is a completely different paradigm that formulates the problem of DNN repairing as a statistical analysis problem. Using an influence function, the proposed method can efficiently estimate the influence of each training sample on the model prediction at both sample level and segmentation level of the recurrent neural networks.

Despite their promising results, most of the existing DNN repairing approaches conceptually share the same fundamental principle of augmenting the dataset to improve the model’s generalisation capability in natural environmental settings. However, all of the developed approaches apply metamorphic data augmentation techniques to alleviate the difficulties in testing and retraining autonomous systems, yielding encouraging results. In the following, we discuss some of these augmentation techniques.

2.3 Data Augmentation for Deep Learning

A key ingredient in achieving these impressive results is the availability of large volumes of high-quality annotated datasets that adequately encode the characteristics of the target domain [33]. Within a typical DL pipeline, this data enables the construction of high-dimensional representations during training, and instruments the robustness and generalisability assessment during testing [19].

Data scarcity poses a major challenge in devising robust and competitive DL models [51]. This is particularly important in domains such as healthcare or security, where relevant data is not typically available because of privacy considerations or simply because such data does not exist. Data augmentation aims to alleviate this issue by intelligently synthesising new data informed by the available dataset. Compared to the manual creation of labelled datasets involving human experts, data augmentation is time-efficient and cost-effective [58]. In fact, data augmentation entails creating new synthetic data samples from existing ones to improve the effectiveness and robustness of DNN models. The augmented data serves for both (i) fuzzing testing [41, 65, 49] to identify potential DNN misbehaviour that can lead to hazardous situations; and (ii) model repairing to improve the DNN robustness and accuracy [68, 19].

Driven by traditional software engineering principles, data augmentation in DL testing is increasingly adopted to improve the diversity of the test set and achieve higher testing coverage [70].

In fields with restricted data, like medical or security, data augmentation serves to reduce problems such as sample inadequacy, and model overfitting.

As illustrated in Figure 4, various methods have been proposed to augment data, including GAN augmentation, i.e., adversarial-based approaches [13], style transfer informed [34], or by using tailored heuristic approaches [48]. Some of the proposed augmentation techniques are domain-specific [48, 8], and some are more generic and can be applied on different input domains [73]. Each of these methods can lead to very different stability performance results when applied to DNN training or testing.

Notwithstanding its merits, conventional augmentation techniques are limited to noise injection or to the application of content-preserving geometric and colour-space transformations, e.g., flipping, cropping, rotation, translation, random cropping or erasing, etc [42, 57, 19]. In general, these approaches stick to metamorphic techniques for data augmentation which results in a limited number of failure samples not fully representative of real-world corner cases. Furthermore, these techniques cannot perform semantic transformations, like altering the content of an input [12], thus producing testing suites that, while yielding higher coverage numerical results, are of low quality and lack semantic diversity [51].

Overall, according to the best of our knowledge, existing work on data augmentation for DL systems, whether for testing or repairing, lacks a comprehensive study of the utility and effectiveness of generative AI models for augmenting the data and discovering failure cases. Our GenerativeRepair leverages these aspects, yielding more semantically-meaningful augmentation methods.

A prominent advantage of generative AI is its semantic data augmentation ability, which compared to traditional metamorphic methods, provides the means so that the DNN can capitalize on the abundant set of semantically-informative real-world scenario inputs. These are highly relevant data samples that can improve the model's performance.

2.4 Generative AI: Text-Guided Diffusion Models

With the emergence of generative models such as Stable Diffusion (SD), GANs and deep belief networks (DBNs), data augmentation can benefit from their tremendous success and be significantly more effective. In this section, we discuss generative AI models for computer vision and their benefits for data augmentation. Our GenerativeRepair employs these principles.

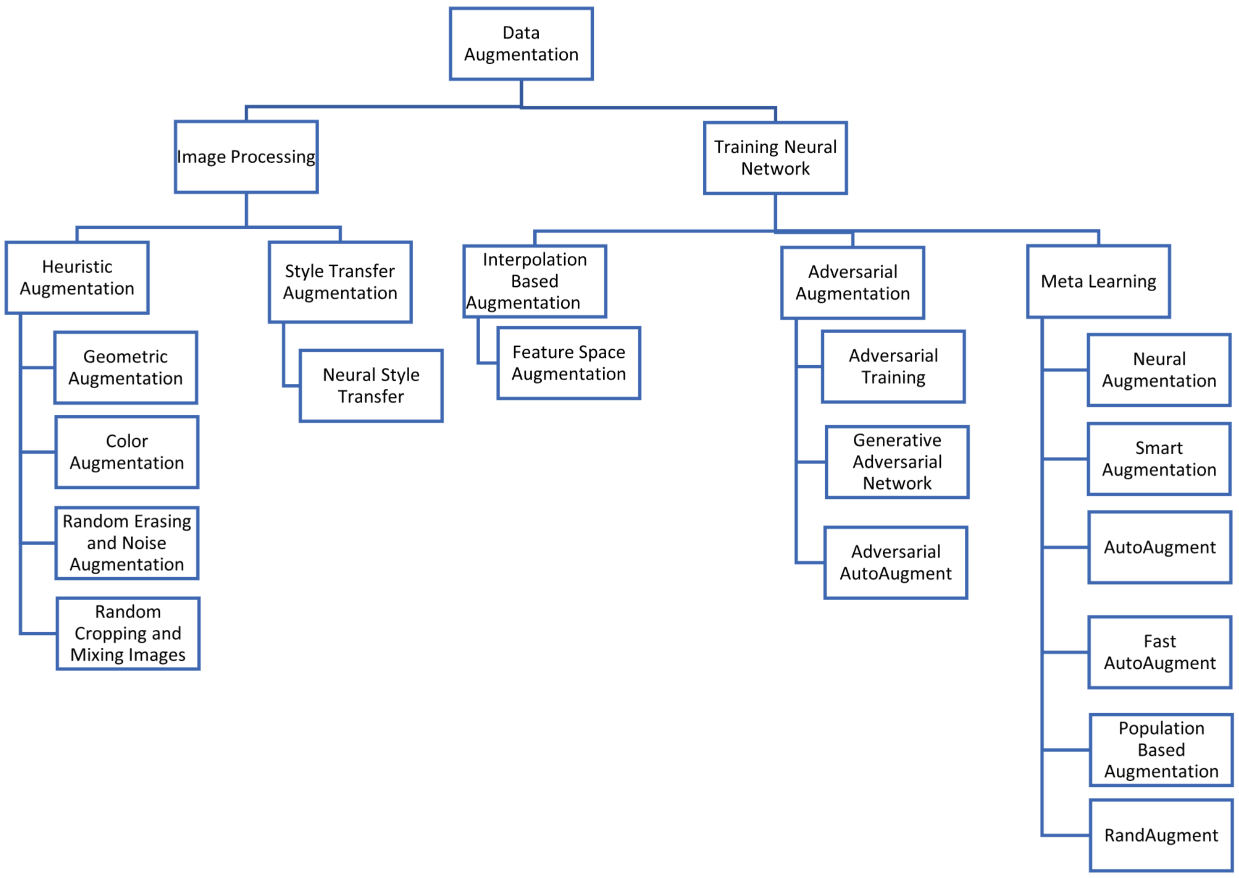


Figure 4: Taxonomy of data augmentation approaches [4].

Generative models have emerged as one of the most powerful content generation tools capable of learning complex and high-dimensional distributions of data, which allows them to generate diverse, and novel synthetic samples that capture the underlying variability of the data distribution.

In particular, Stable Diffusion [59] has demonstrated state-of-the-art performance in content generation without requiring training. Accordingly, the technique has the potential to automatically learn natural features from a dataset, and generate realistic images using natural language input as prompts to create rich and diverse visual content with unprecedented precision.

Stable Diffusion is a class of Gaussian models introduced by Sohl-Dickstein et al. [52] and has seen exponential adoption both for research and industrial purposes. The fundamental idea of the probabilistic diffusion model is to model a specific distribution from random noise. Formally, given an input sample from a natural data distribution $x_0 \sim q(x)$, the model produces a forward diffusion process where a small amount of Gaussian noise is progressively added to the samples. This results in a Markov chain of latent variables $x_1, ..., x_T$:

$$q(x_t \setminus x_{t-1}) = \mathcal{N}\left(x_t; \sqrt{1-\beta_t}x_{t-1}, \beta_t\mathcal{I}\right) \tag{1}$$

The magnitude β_t of the added noise at each step should be small enough, and controlled by a variance schedule so that $q(x_t \setminus x_{t-1})$ is approximated by a diagonal Gaussian.

The goal of training the diffusion model is to learn the reverse process, so that:

$$q(x_t \setminus x_{t-1}) = q(x_{1:T} \setminus x_0) = \prod_{t=1}^{T} q(x_t \setminus x_{t-1}) \tag{2}$$

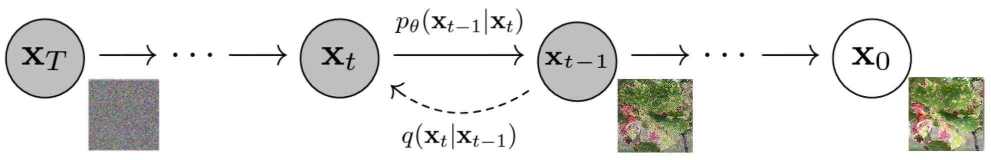

$\mathbf{x}_T$
$\mathbf{x}_t$
$p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t)$
$\mathbf{x}_{t-1}$
$q(\mathbf{x}_t|\mathbf{x}_{t-1})$
$\mathbf{x}_0$

Figure 5: A directed graphical model as a simplified illustration of the probabilistic diffusion model (Source: [52])

This process suggests learning a neural network to approximate the conditional probabilities in order to run the reverse diffusion process and approximate the true posterior $p_0(x_t \setminus x_{t-1})$ as shown in Figure 5.

Driven by the tremendous success of the probabilistic diffusion models, the research effort continues to grow. Latent Diffusion Models (LDM) have lowered the training costs and increased the inference speed by introducing the latent space rather than pixel space. This stream of work evolved with models such as the Feature Pyramid Latent Diffusion Model (Frido) [16]. Following a similar path, Guided Diffusion Models have become more prominent. For instance, Dhariwal and Nichol [15] improved diffusion models with classifier guidance. Then, Radford et al. [47] introduced another solution that casts the problem of guided diffusion by introducing CLIP as a scalable approach for learning joint representations between text and images. CLIP enables encoding the image $f(x)$ and the text $g(c)$ separately, then optimises a contrastive cross-entropy loss that encourages a high dot-product $f(x) \circ g(c)$ to pair the image with the given text and provide a score of how close they are. These works pioneered advances in text-guided Inpainting techniques, such as eDiffi [3] and Glide [39].

As can be easily understood, Stable Diffusion models can be used to increase the diversity of the training data and expose the model to semantic situations that it would not have otherwise seen. Accordingly, this strategy helps to improve the generalisation capability and reduce the overfitting problem of a DNN.

However, whether Stable diffusion-based data augmentation can be used to improve DNN testing and repairing in natural environmental applications is still an open question. In this work, we investigate this question through a comprehensive analysis of AI generative models for synthetic data augmentation purposes. To this end, we adopt a text-guided Stable Diffusion model, termed Inpainting, which is a text-conditional diffusion model capable of synthesizing images from textual information. In the next section, we provide more details about how our GENERATIVEREPAIR works.

3 Systematic Repair and Hardening for EDDI Data-Driven Components

EDDIs are runtime components that monitor the dependability aspects of the subject robotic system or robotic team. As described in Section 1.3 and illustrated in Figure 1, the use of DNN (and ML in general) for the execution of various robotic system actions (e.g., perception) enables using the quality assurance results about the capacity of the target DNN to support the dependability analysis executed by the EDDI. This is particularly important as DNNs have shown to be sensitive to small changes to the data input, such as occlusion (PAL use case) or new leaf spot diseases (DKOX use case). These changes to the initial data (i.e., *in-domain* data *distribution* (InDD)) could result in new inputs that we call *corner-cases*. These corner cases can radically change the DNN decision, reducing the system performance and leading to safety hazards during runtime.

The runtime evidence from DeepKnowledge (Task 6.1) and SafeML [2] execution is used in this regard by providing observations from previous responses of the system and the target deployment environment to detect corner cases where the system is not serving its purpose accurately.

We introduce GenerativeRepair, an approach for retraining the DNN using the original training dataset augmented with the newly identified corner case inputs. While most existing methods focus only on automatically encoding buggy inputs through simple geometric and colour space transformations (e.g., changing brightness, rotation, blurring, etc.), GenerativeRepair leverages semantic-based methods to generate inputs that have not been encountered during training, and which could yield potentially severe corner cases.

The synthetic generation of corner cases that encode semantically diverse inputs, however, is not a trivial task. It requires maintaining a balance so that the introduced transformation creates a semantic shift compared with InDD datasets, but without further reinforcing an obvious covariate *domain shift*. While state-of-art data augmentation techniques enable applying class identity-preserving manipulations (e.g., changing backgrounds of objects or varying visual angles), they fall short in introducing more advanced semantic transformations such as changing the object and creating new classes of data [48].

One natural alternative is the employment of generative AI algorithms that enable the generation of meaningful and photo-realistic data samples in a very cost-effective way without the need to collect thousands of real-world data inputs. This is particularly important for situations where the data available for a specific scenario is scarce or where its collection may be risky or require careful instrumentation.

Motivated by the concept of generative AI, fresh synthetic data can be created from existing data, allowing our partners to reduce their reliance on training data preparation (i.e., data collection and labelling) and to develop accurate DNN models faster and with lower effort. Major benefits are expected from applying semantic-based data augmentation methods, including:

- Mitigate data scarcity;
- Lower data overfitting;
- Increase data variability; and
- Resolve class imbalance issues in classification.

These benefits have direct effect on increasing the generalisation ability of DNNs, enabling the development of more accurate models. Accordingly, these models are being repaired using synthetically augmented data to resist data distribution shifts during runtime (as detailed in Section 3.4).

3.1 Application Scenario: Autonomous Pest Management in Viticulture

One of the main tasks for pest management in the viticulture use case is fungicide spraying. An optimal fungicide spraying mission can be achieved using LXSNS's DNN algorithms with the novel MRS capabilities

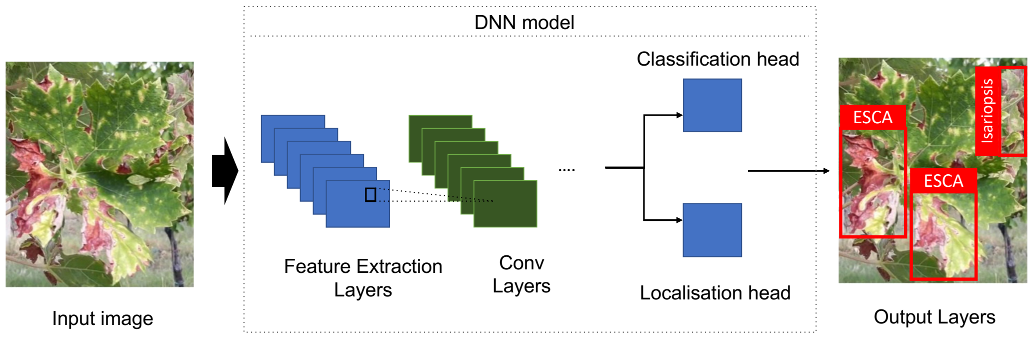


Figure 6: Illustrative example of DNN output needed for fungicide spraying inspired by the YOLO architecture [5] with a CNN backbone and object localisation and classification heads. The input image is a high-resolution representation of a grape leaf with the Esca disease.

for sensor fusion. To this end, high importance is set on repairing the DNN algorithms for better accuracy and safe behaviour. In fact, repairing DNNs, i.e., improving their performance, is a major requirement to decrease the average fungicide spraying by 30%, thus contributing to environmental preservation.

This scenario focuses on applying and evaluating the ability of the SESAME technologies to harden and repair the DNN components by strengthening the ability of the robotic team to successfully achieve the fungicide spraying task in a hazardous environment. Achieving this benefit entails the use of EDDIs to capture both safety and security risks related to DNN inference through the runtime execution of these dependability models, using technologies developed in Tasks 6.1 and 6.2.

At a conceptual level, the efficiency of the fungicide spraying task depends among other parameters on the precision and correctness of a DNN algorithm in plant physiology identification at the vine level, i.e., the detection of the infected plants, as well as precisely identifying the type of infection as shown in Figure 6.

The DNN is required to provide precise positioning of the infected leaves in order to maintain a low but constant distance to the vines to be sprayed. Furthermore, identifying the correct type of disease can inform the selection of the appropriate pesticide and amount needed. The performance of the DNN is extremely dependent on the quality of training data and its correlation to the real-life scenario at runtime. The data collected by our partners during their last exercise in vineyard plots as shown in Figure 7 is limited in terms of diversity as it is dominated by plants infected with the Esca disease. Therefore, when faced with a new type of disease the DNN prediction accuracy degrades leading to safety hazards during runtime. For example, given a convolutional DNN model N^j offline trained on T_{train} and evaluated on a testing dataset T_{test} for the disease classification task with $|C|$ data classes, we deploy it in the real world, using images collected from the vineyards (Figure 7), we find that it usually misclassifies images corrupted by noise or occluded by specific objects. Also, it fails with some kind of new pattern when the data class $c_n \notin C$. Thus, we simulate these patterns and collect novel case examples and obtain an augmented set $\mathcal{T}$ that is then used to repair the DNN (but, importantly, without deteriorating its current classification capabilities).

To overcome these challenges, we specify the safe behavior of the DNN system, and identify causes of hazardous situations, i.e., new disease input samples that are not available in the training set. The goal of our work using the DKOX example is to effectively generate novel cases that simulate real-life data and use them to harden and repair the DNN algorithms for improved and more accurate prediction.

3.2 Methodology Overview

In this section, we give a high level of our hardening and repair approach called GenerativeRepair. A more detailed explanation follows in later sections.

We introduce a DNN repair approach that aims at strengthening the prediction capabilities of DNNs, enabling them to operate more safely when facing (i) buggy inputs; and (i) data distribution shifts. The ultimate goal of

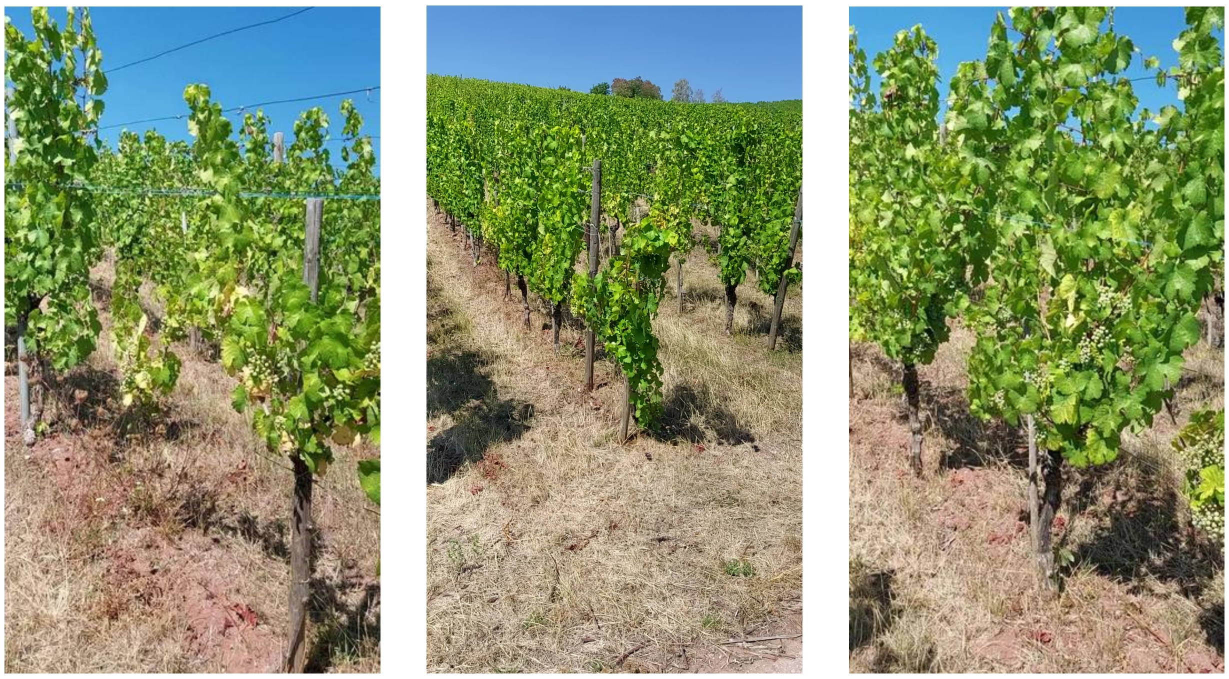


Figure 7: Unlabelled real-life high-resolution vineyard images dominated with the ESCA disease.

our approach is to repair DNNs such that we enhance the model's accuracy on novel data samples, i.e., corner cases, without harming its performance on other InDD data using data augmentation as the main strategy. The underlying data augmentation method balances between extending the dataset and avoiding over-fitting issues by providing efficient image generation from the InDD latent space which exhibits a balance between scaling to new features and maintaining the main semantics of the InDD dataset. data semantic diversity.

GENERATIVEREPAIR is an effective methodology for coverage-guided repair. It is a scalable and performance-preserving DNN repairing approach that iteratively retrains the model with a semantically augmented dataset, verifying its performance until a safe version, capable of correctly predicting novel corner cases[1], is eventually obtained.

The general structure of our GENERATIVEREPAIR approach is fully detailed in Figure 8. GENERATIVEREPAIR iterates back and forth between two main components: a corner-cases generation component (GENERATIVEFUZZER) and a coverage-guided repair (SAFETYREPAIR) component. Our approach works as follows. First, we define a satisfaction function, which is an objective function (in the form of a test adequacy criterion [70]) that informs the selection of input data with the highest effect on this specific coverage criterion. This facilitates the search for input seeds, i.e., inputs that may result in corner cases that resemble previously unseen data samples, potentially leading to unsafe DNN behaviour due to input transformations (e.g., corruption, perturbation).

Second, we introduce an augmentation method to synthetically generate these corner cases using the selected input seeds. For this purpose, we leverage generative AI models, e.g., Inpainting using Stable Diffusion [59]. Then, we iteratively execute a semantic-based data augmentation loop.

Third, we introduce an automated fidelity estimation technique that filters out non-photo-realistic augmented seeds. The generated inputs must not only yield a higher coverage and be semantically-diverse, they should also resemble realistic inputs. Then, additional iterations are performed and extensible coverage criteria are deployed as feedback to guide the selection of augmented seeds from the previous step. Crucially, a set of

[1]In conventional software testing nomenclature, corner cases typically refer to interesting and fault-revealing test cases, so we will alternate between both terms in the later description.

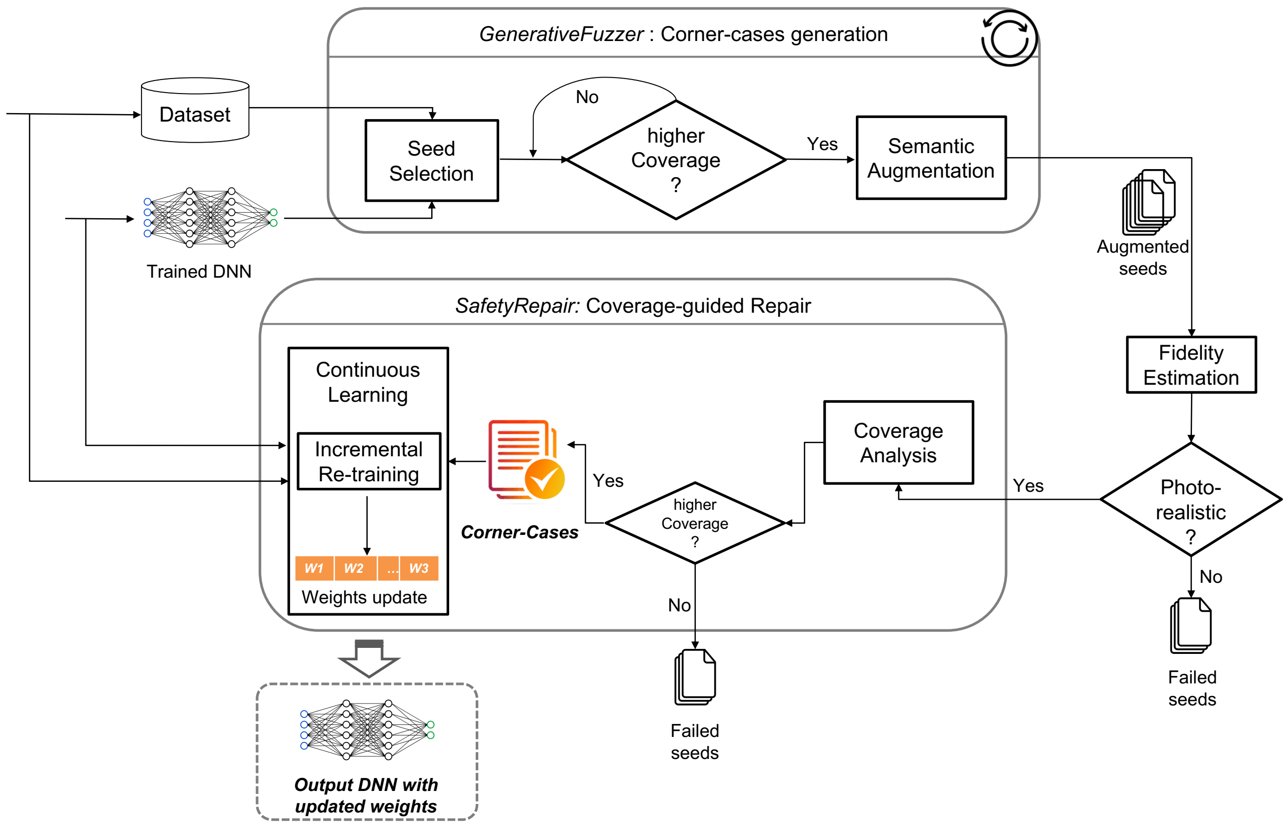


Figure 8: The overall process of the GENERATIVEREPAIR approach for hardening and repairing DNNs.

novel corner cases that is important for the performance of the DNN in its target operational environment, i.e., solve the oracle problem and increase the coverage score, is created.

Finally, the InDD dataset is augmented with corner cases from previous steps to increase its diversity. An iterative re-training mechanism is introduced, termed *Coverage-Guided DNN Repairing*, for which the collected corner cases are used as sources that enable the rectification of these failure patterns. After many iterations and verification steps are carried out, the output of the procedure is a DNN model with repaired weights. We demonstrate the performance of GENERATIVEREPAIR compared to several state-of-the-art approaches in Section 5. The experimental results show that GENERATIVEFUZZER efficiently generates photo-realistic corner cases, and SAFETYREPAIR successfully repairs DNNs achieving high accuracy without harming their initial performance on InDD data.

3.3 GENERATIVEFUZZER: SEMANTIC DATA AUGMENTATION FOR SAFETY ASSURANCE

3.3.1 Problem Formulation

Following the automated test case generation principles, GENERATIVEFUZZER can be performed to enhance the size, quality, and semantic diversity of datasets such that extensible DNN testing can be performed [70]. Unlike existing methodologies, GENERATIVEFUZZER implements a semantic data augmentation technique fueled by generative AI models to generate new test cases, i.e., corner cases, with high fidelity.

As shown in Figure 9,GENERATIVEFUZZER is a coverage-guided fuzzing (CGF) approach that can be used separately as a hardening mechanism for DL testing and quality assurance. Using a dataset X, and a trained DNN model N on X_{train}, our approach produces synthetically augmented images using Inpainting diffusion models introduced by [52] and improved by [28], as well as semantic occlusion technique.

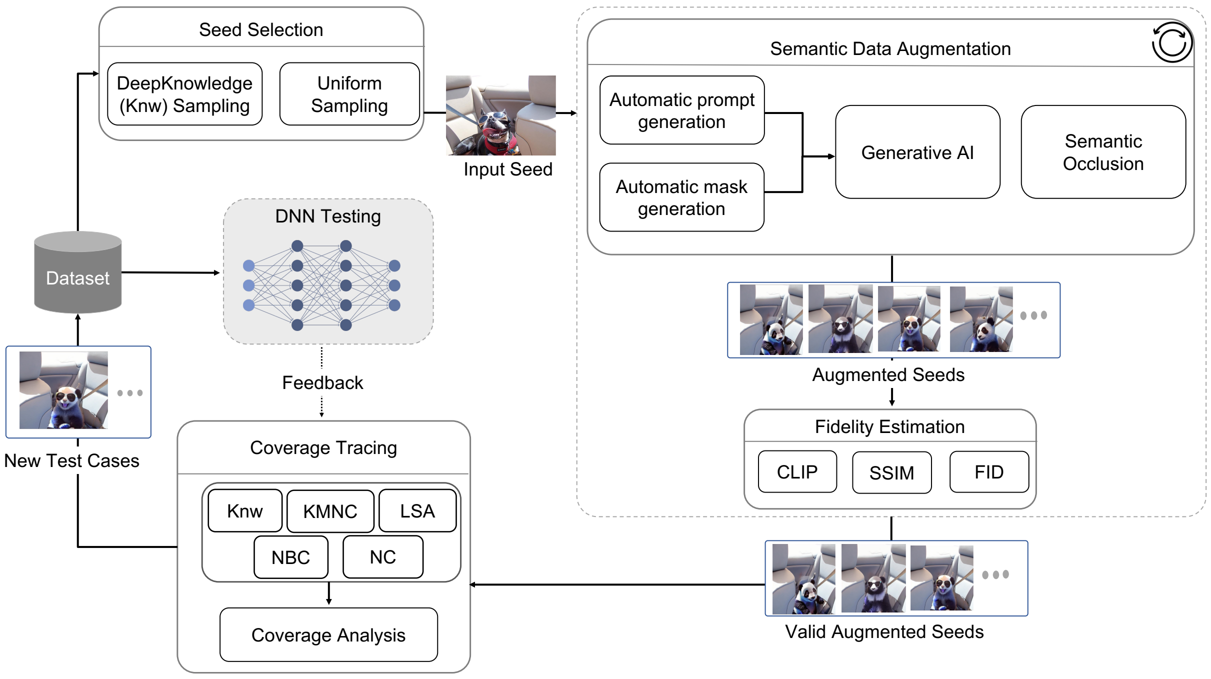


Figure 9: GENERATIVEFUZZER pipeline.

Unlike existing methodologies, a fidelity analysis step is carried out to select images that are photo-realistic (based on a predefined hyperparameter of the approach) and effectively augment the dataset's feature space. Finally, GENERATIVEFUZZER deploys extensible testing criteria as feedback to guide the selection of augmented images of increased importance for DNN testing. We run the trained DNN against the newly generated tests; test cases that increase coverage are kept as denoted corner cases. These corner cases simulate real-world scenarios with high fidelity and are deployed for DNN testing to assess their robustness.

The intuition behind this solution is that the initial test set X_{test} is not representative of diverse failure patterns that the model can encounter during runtime. The diverse augmentation operations adopted in our methodology (i.e., transformations that simulate the real-world noise and semantically-diverse patterns in the operational environment) could cover the unknown failure patterns through specific semantically-diverse transformations.

Our GENERATIVEFUZZER component starts by selecting a DNN model trained in a specific scenario of interest, i.e., data distribution domain $\mathcal{X}_{InDD}$, and observe the system's erroneous behaviour given this scenario using the tool-supported technique DEEPKNOWLEDGE (developed in Task 6.1).

Formally, the problem of semantic-based test case generation for DNN can be formulated as the problem of generating synthetic examples x', semantically rich and diverse within the data domain $\mathcal{X}_{InDD}$, but with enough perturbation to enhance its feature space. This can be understood as the ability of an augmentation technique $\mathcal{A}_\kappa$ to generate an augmented seed x' that enhances the initial coverage and achieves a high-fidelity score. The objective is twofold: maximize the coverage adequacy criterion Knw, i.e., the DEEPKNOWLEDGE coverage score, and maximize the fidelity score $f(x')$. This problem can be encoded in the following objective function:

$$E_{x' \sim \mathcal{X}_{InDD}} \left[\max_{\varepsilon} f(x', \epsilon), \max \mathcal{C}ov(x') \right] \qquad (3)$$

Maximizing the DEEPKNOWLEDGE coverage score, $\mathcal{C}ov(x')$, is done through the maximization of the covered cluster combinations (TCC). These are regions within the important neurons value domain that is central to the DNN model execution, i.e., clusters of activation values identified using the k-means clustering algorithm [35]. Please refer to deliverable D6.1 [43] for more details on how these cluster combinations are identified.

$$\max \mathcal{C}ov(x') = max(\text{Knw} \leqslant \lambda) = \frac{\left|\left\{TCC(i) | \exists x' \in X : \forall V_n^j \in TCC(i) \bullet min\ d\left(\phi\left(x,n\right), V_n^j\right)\right\}\right|}{|TCC|} \quad (4)$$

$TCC(i)$ is covered if the augmented input x' able to minimize the Euclidean distance $d\left(\phi\left(x',n\right), V_n^j\right)$ between the activation values of all important neurons and the i-th neuron's clusters centroids. λ is a threshold that is experimentally identified to guarantee that the augmented input x' effect on total coverage increase is different from the adversarial samples effect so that the synthetic sample x' belongs to the in-domain distribution of $\mathcal{X}_{inDD}$. We denote also that our methodology is generic. The framework implementation allows the usage of a set of different coverage criteria for the purpose of satisfying this objective function; for instance, the surprise adequacy criteria LSA and DSA [31], neuron coverage [45] and the criteria from DeepGauge [37].

3.3.2 Seed Selection

As shown in Figure 9, we use two seed selection strategies $SS(.)$: (1) the random uniform sampling strategy;and (2) coverage-based seed selection, where this strategy selects the input images that satisfy the objective function (4) as input seed.

For random sampling $SS(.)$ is the seed selection strategy that samples and selects input seeds based on a random uniform sampling strategy with sampling probability $P_i \in \mathcal{P}$. $\mathcal{P}$ defines the sampling probability of all samples $x_i \in X$, defined as :

$$\mathcal{P} = \left\{P_i = \frac{1}{|X|}\right\} \quad (5)$$

Note that we can use X_{train} or X_{test} for seed sampling with $\forall x_i \sim \mathcal{X}_{inDD}$.

For coverage-guided sampling, $SS(.)$ selects $x_i \in X_{test}$ such that $\max \mathcal{C}ov(x_i) = max(\text{Knw} \leqslant \lambda)$

3.3.3 Semantic Data Augmentation Techniques

One main limitation in the literature related to using data augmentation techniques for DNN repairing is the focus on metamorphic operations that guarantee to generate semantically-preserved test cases. The number of mutated samples we can collect in practice created by these techniques is often limited, and not fully representative of the failures or corner cases that a DNN model can encounter in the operational environment.

To alleviate this issue, we deploy semantic augmentation techniques including generative models, i.e., text-guided Stable Diffusion. These techniques have been selected to generate mutated images that capture the variability of the data distribution and allow us to increase it by introducing new patterns and features that were not present in the original dataset. The mutated images are new variations of the original input seed that does not necessarily preserve the original labels. This is a novel aspect of GENERATIVEREPAIR and distinct compared to the state-of-the-art research in the domain.

More specifically, in order to solve the optimization problem defined in equation (3) effectively and efficiently, we carefully select the data augmentation technique used to create augmented seeds. In fact, this research provides novel insights into:

- The effectiveness of diffusion models as a semantic data augmentation technique, demonstrating its ability to generate high-fidelity synthetic failure cases; and
- The effectiveness of semantic occlusion as a semantic data augmentation technique.

In Task 6.3 and within the context of the GENERATIVEREPAIR approach, we leverage the capabilities of diffusion models to alleviate data scarcity and (ii) deal with sample inadequacy problems when testing/training

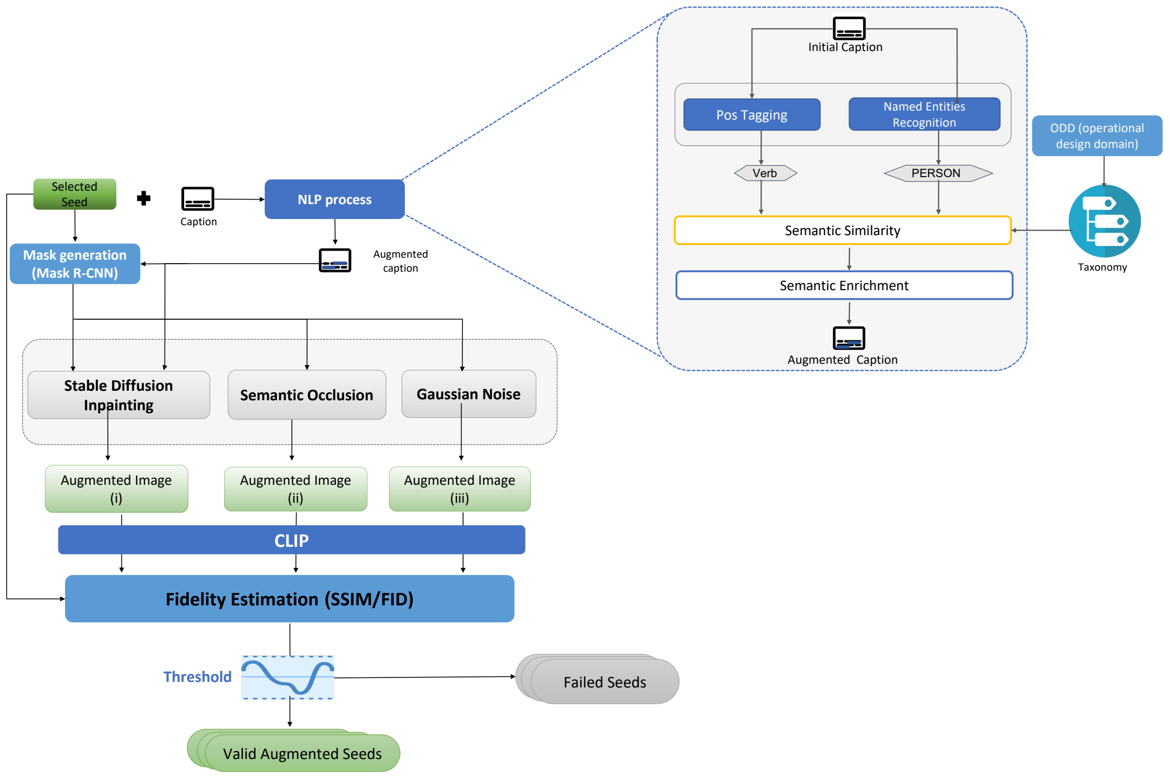


Figure 10: Semantic data augmentation workflow. In blue, we denote all processes implemented to guarantee the semantic augmentation process validity.

DNNs for different real-life applications. In particular, the diffusion models have recently shown high effectiveness in generating high-quality images, especially when paired with a guidance technique to help trade off diversity for fidelity. However, their ability as a data augmentation technique to semantically augment a dataset through new classes, or new instances of existing classes, has not been studied yet. In addition, diffusion models require a lot of manual pre-processing to provide textual guidance (i.e., text prompt) and fitted masks to enable the efficient generation of new synthetic images.

Our work introduces a scalable approach that automatically guides the generation of joint representations between text and images using Stable Diffusion models. As shown in Figure 10, we add an NLP layer (NLP process) to the Stable Diffusion Inpainting model that allows us to automatically create textual guidance that preserves the InDD dataset (i.e., training data) semantics while introducing new features. This process is based on a separate *operational design domain* representation of each partner use case. The output of this NLP process also guides the selection of the right mask to fit the generated textual guidance (i.e., augmented caption in Figure 10).

Concerning the proposed semantic occlusion technique, the aim is to harden the model against real-life occlusion (i.e., corner cases) which are very common in the wild leading to degraded performance and robustness of DNNs and challenges during deployment.

The occlusion of objects is one of the inevitable problems in computer vision. Much effort has been devoted to removing occlusions from data; however, in this work we devote our attention to augmenting the dataset with occluded objects. Our work aims to augment the data with varying shapes and forms of occlusions so that we generate repair patterns for the studied DNN model.

Our work is inspired by work on De-Occlusion [67]. As shown in Figure 11, semantic segmentation, which is the process of extracting objects/regions from images, is a main valuable step to addressing the semantic occlusion problem. In this example (Figure 11), real-time instance segmentation (step 1) is made using YOLOv4 [5], which allows us to identify the bounding box of potential regions of interest (i.e., the person in this example).

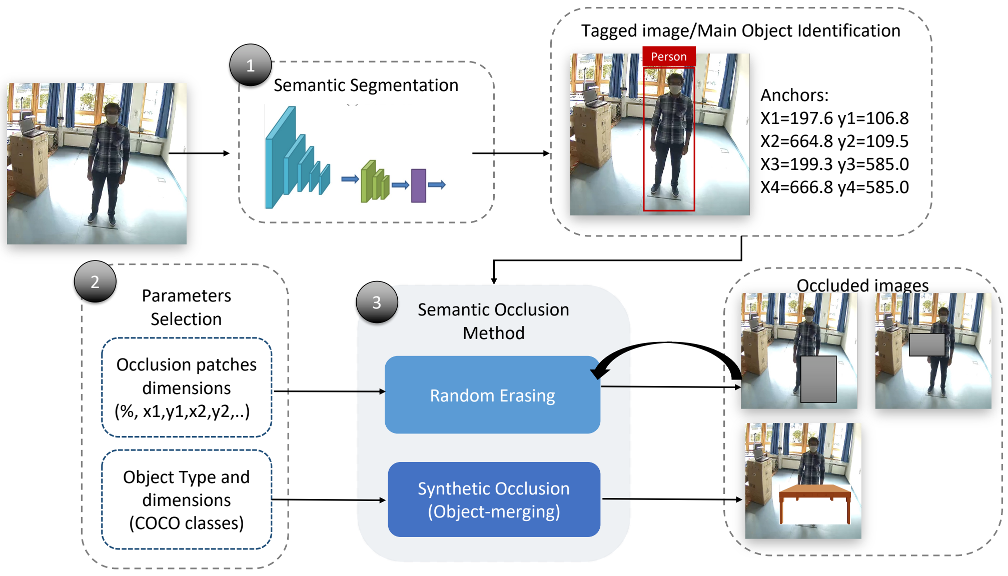

1
Semantic Segmentation
Tagged image/Main Object Identification
Person
Anchors:
X1=197.6 y1=106.8
X2=664.8 y2=109.5
X3=199.3 y3=585.0
X4=666.8 y4=585.0
2
Parameters Selection
Occlusion patches dimensions (%, x1,y1,x2,y2,..)
Object Type and dimensions (COCO classes)
3
Semantic Occlusion Method
Random Erasing
Synthetic Occlusion (Object-merging)
Occluded images

Figure 11: Overview of the proposed Semantic Occlusion method, which does include a semantic segmentation example (based on YOLO dataset) for illustration.

This example is tailored to the COCO dataset [36], but Mask R-CNN is also used for other datasets, i.e., grape leaves for the DKOK use case. The identified bounding box is used to specify the geometric characteristics of the added occlusion (i.e., step 2, parameters selection in Figure 11). In the final step (step 3 in Figure 11) our method alternates between adding *random erasing* and *synthetic object merging*.

Each of these techniques is tailored to a specific SESAME use case. For instance, *random erasing* is used for the grape leaves dataset to cover certain disease spots. Thus, we generate novel corner cases that occlude key semantic features for the disease detection task. On the other hand, *synthetic object merging* enables to cover parts of the main object in the image to classify. This is useful to simulate hazardous cases where the **TIAGo** robot fails to detect persons hidden behind objects (i.e., desk, door, etc.).

Formally, we define the augmentation technique as part of an augmentation operator. Let N^j be our original trained DNN model on a training dataset T_{train}, T_{val} be the validation set, and T_{test} be the testing set, which belong to the same data distribution domain $\mathcal{X}_{inDD}$.

Let $\mathcal{A}_\kappa \in \mathbb{A}$ be our augmentation method for N^j with κ hyper-parameters. $\mathcal{A}_\kappa$ is a data-level augmentation operator that targets the testing/training data T, by augmenting it with a set of augmented input seeds to obtain the initial augmented set T^κ.

Each of the augmentation operators in $\mathbb{A}$ is configured using a set of hyper-parameters that determine the extent of the change/mutation introduced into the original dataset. These will be detailed later in this section. Specifically, considering each image in the initial set as an input seed that serves as a potential reference for the augmentation process, we represent the augmentation operator as:

$$\mathcal{A}_\kappa(x_i) = Aug\left(x_i, \eta\right), x_i = SS\left(T, \mathcal{P}\right) \tag{6}$$

where x_i is the input seed for the augmentation technique $Aug(.)$, and $\mathcal{P}$ defines the sampling probability of all samples $x_i \in T$.

As per equation (6), $\mathcal{A}_\kappa$ has four components: (1) seed selection strategy SS, i.e., the sampling strategy out of all input seeds database; (2) the data augmentation technique Aug(.), i.e., semantic occlusion or Stable Diffusion, to mutate the selected seeds; (3) the dataset T used as the database of initial seeds[2]; and (4) the set of hyper-parameters η that are specific to the augmentation operator, i.e., size of the selected seed, number of iterations, validation measure.

[2]As stated above, as we are generative semantically-diverse inputs, both T_{train}, and T_{test} can be used.

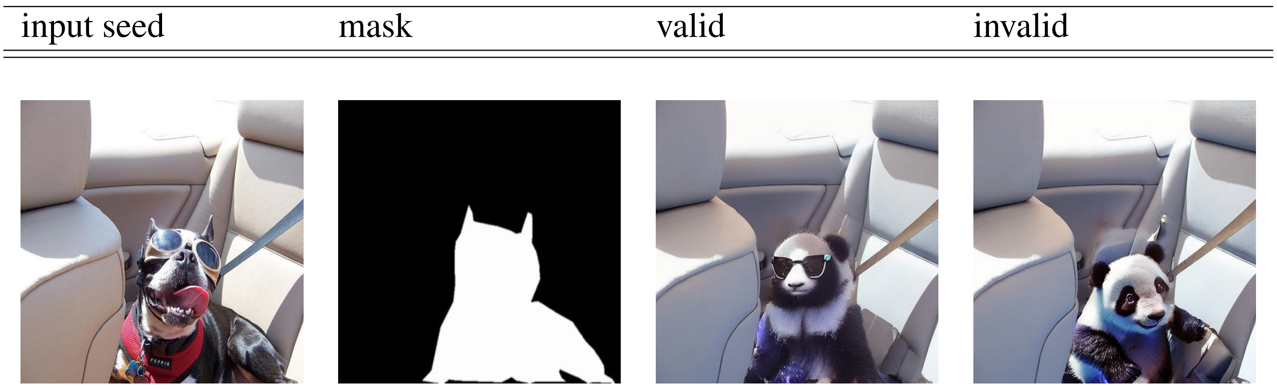


Figure 12: Images synthesised using GENERATIVEFUZZER with Stable Diffusion Inpainting as the augmentation technique. The invalid sample is identified using the Fidelity Estimation process.

It is important to note that our approach is generic and can support different generative models as the data augmentation technique $Aug(.)$, including Glide [39], eDiffi [3] and Imagic[30]. We also emphasize the importance of using a text-conditional generative model as they have the advantage of providing a control mechanism through the textual prompt to adjust the mutations introduced to the original image and sampled using the seed selection strategy $SS(.)$ selected based on $\mathcal{P}$.

Below we detail the technical configuration of each of the augmentation techniques used to configure the supported augmentation operators, as shown in Figure 9:

(i) ***Text-Guided Stable Diffusion***: We use the *txt2img-inpainting* technique [15] that belongs to the family of generative AI models. This model has the ability to edit and generate photorealistic image samples based on free-form textual prompts. Originally, the model allows editing specific parts of an image by providing a mask and a text prompt. In our case, we needed to automatically generate a mask and prompt for each input seed image $x_i \in T$. To alleviate this issue, we apply Mask R-CNN [27] for instance segmentation and we use the image corresponding caption y_i to select a mask m_i for the main object in x_i. For the text prompt, we deploy a natural language-based model (SpaCy) to generate a sentence with the target object name. The txt2img-inpainting replaces the known region of the image (i.e., masked) with the target object. We make sure that the target object belongs to the set of classes $C = \{c_j, ..., c_f\}$ of the original dataset. For *txt2img-inpainting* $\eta = (m_i, y_i, t_i)$. Figure 12 gives a full example of how the process is performed, with the first image on the left being the input seed x_i with corresponding caption y_i =“*a **dog** sitting in the back seat of a car with sunglasses on*”, the class $c_i = dog$ and the generated prompt t_i=“*a high-fidelity image of a **panda** sitting in the back seat of a car with sunglasses on*”, where we replaced the main object dog (for an object classification problem) with a panda. The second image represents the used mask, and the two last images represent the output of different iterations of the *txt2img-inpainting* model.

(ii) ***Semantic Occlusion*** (Occ): To add occlusion to the input seed, we have used masks generated by the Mask R-CNN [27] model, but this time we selected the mask of a secondary object in the image. Then, we simply deployed the erasing technique to replace this object with random pixels in the output image x'. Although this process introduces a semantic change to the original image, it keeps the same caption/label y_i valid for the mutated seed. For semantic occlusion, $\eta = (m_i, anchors_i)$.

(iii) ***Gaussian noise***: This is a metamorphic technique that has been widely used in the literature [74]. We add white Gaussian noise to x_i by adding a random number to every colour channel of each pixel. The η parameters for Gaussian noise include the mean μ and standard deviation σ of the random noise, i.e., $\eta = (\mu, \sigma)$.

The main benefit of this process is augmenting the semantic diversity of the dataset. To this end, we extend the dataset feature space without surpassing the bounded range of the in-domain distribution $\mathcal{T} \sim \mathcal{X}_{InDD}$, or by only creating a near out-of-domain dataset $\mathcal{T} \sim \mathcal{X}_{nearDD}$. In practise, an augmented image that can

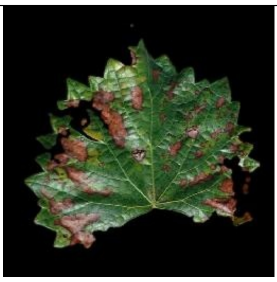
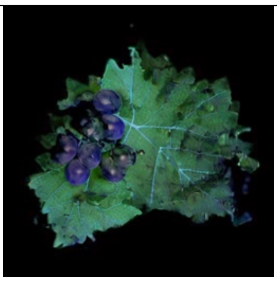
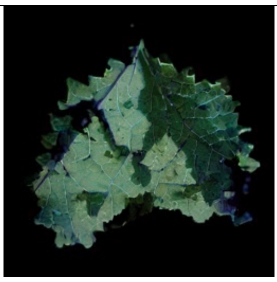
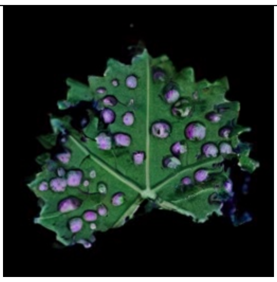
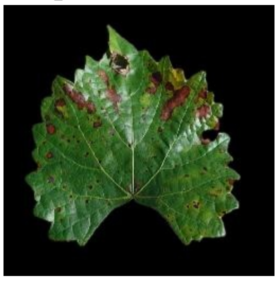
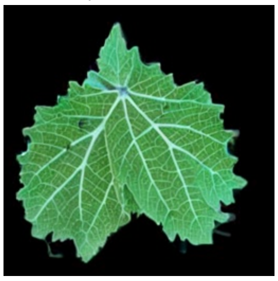
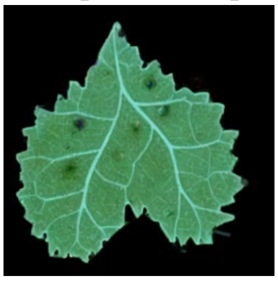
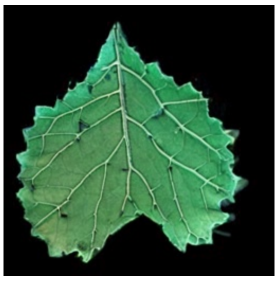
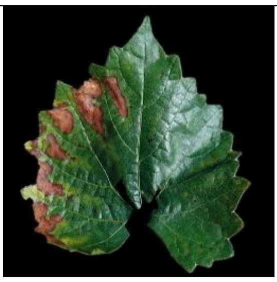
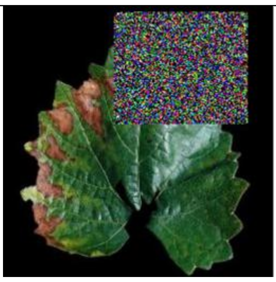
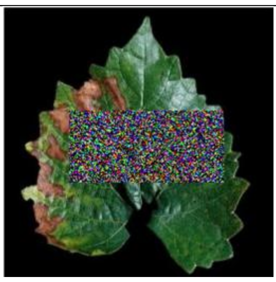
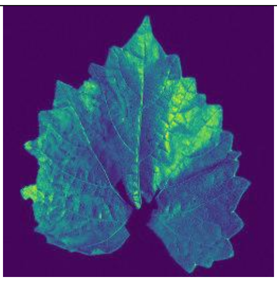
Original seed	Transformations with Inpainting		
			
Grape leaf	healthy	Isariopsis Leaf Spot	Black Measle (invalid)
			
Grape leaf	healthy	Isariopsis Leaf Spot	Black Measles (invalid)
Original seed	**Transformations with Semantic Occlusion**		
			
ESCA disease	Occluded ESCA (i)	Occluded ESCA (ii)	Gaussian Noise- ESCA

Table 1: Images synthesised using GENERATIVEFUZZER with Stable Diffusion Inpainting and Semantic Occlusion as augmentation techniques. The dataset is the Grape Leaves disease.

be classified as one of the dataset classes is an InDD sample. On the other hand, the *near out-of-distribution* (nearOOD) encloses all the samples that have semantic shift compared with InDD datasets [66]; in our case, such image belongs to the new data class. The augmented set can be used in two different ways:

1. Hardening the DL testing process (GENERATIVEFUZZER): Finding corner cases that can lead to unsafe behaviour, i.e., misclassification, and exercise them to create the set of corner cases $\mathcal{T}$. Then, the augmented test set with these new test cases $\mathcal{T}$ can serve as a vulnerability discovery technology to assess the safety of the model at design-time (before deployment), thus reducing the risk of potential hazardous behaviour at runtime;

2. Repairing the DL model (SAFETYREPAIR): Tackling the problem of correcting the DNN model once unsafe behaviour is found. We augment the training set with the new test cases $\mathcal{T}$, and retrain the model. This helps the DNN model learn more robust and generalisable representations of the data distribution, leading to better performance on unseen data samples, which could incur failure cases (i.e., missclassifications or imprecise detection) in the operational environment.

3.3.4 Fidelity Estimation: Photo-realism of Synthetic Images

To obtain the augmented set T^{κ} generated by augmentation operator $\mathcal{A}_{\kappa}$, we first gather valid augmented seeds, i.e., inputs seeds on which the data augmentation technique $Aug(.)$ has executed correctly. We evaluate $Aug(.)$ performance, i.e., how good is the model in approximating the data distribution $\mathcal{X}_{InDD}$, and generating photorealistic output images x' using a fidelity function.

Fidelity. We aim to maximize the photo-realism of the augmented seeds by maximizing the function f that represents a mutated seed fidelity with respect to the input seed. Since not all augmented seeds correspond to realistic images, our approach estimates the visual or textual fidelity of each augmented seed x', keeping only those whose score exceeds a predefined fidelity threshold given by:

$$\text{isValid}(x_i^{\kappa}, Q) = \begin{cases} True & \text{if } f(x_i^{\kappa}, Q) \geq \delta \\ False & otherwise \end{cases} \tag{7}$$

where $f(.)$ denotes the employed fidelity assessment technique. For visual fidelity, $Q = x_i$, i.e., the input seed in which case we use FID or SSIM measures. For textual fidelity assessment, $Q = t_i$, i.e., the text prompt and we use CLIP.

In the literature, the problem of image fidelity is addressed by introducing quantitative measures that identify the acceptable deviation between the original and mutated images. We use the most widely deployed quality assessment measures from the literature for evaluating generative models and image fidelity. SSIM [9] and FID [40] both rely on quantifying errors between a reference and a sample image, while CLIP [47] is widely used to guide image generation through textual input. More specifically, CLIP is used to assess the fidelity of a generated image (i.e., mutated seed) with respect to the textual prompt. The choice of CLIP is an important element in the creation of a new dataset. In fact, this technique provides a score of how close an image is to a caption, which makes it ideal to steer generative models (i.e., GANs, Glide) toward a user-defined text caption. In our work, CLIP provides a grounded evaluation of the synthetic image labelling process. In this case, the high score of CLIP validates the prompt as a ground-truth caption for the generated image.

As shown in Figure 13 and Table 1, the augmented seeds fidelity varies a lot and in most cases, the measures FID and CLIP used to select valid seeds (tagged with Valid in the figures) are in line with the human judgment based on simple observations of the evolved inputs.

Overall, each of these measures serves to evaluate and validate the output of one of the above-mentioned data augmentation techniques. We consider as valid the seeds samples that achieved a fidelity score above a certain threshold (i.e., a domain-specific threshold experimentally defined for each measure) and use it to select the most photo-realistic seeds to form the set of valid augmented seeds T^{κ}. More details about the configurations of each of these measures and the defined thresholds will be given in Section 4.

3.4 SAFETYREPAIR : A COVERAGE-GUIDED REPAIRING METHODOLOGY

In GENERATIVEFUZZER, Coverage-guided Fuzzing has been successfully used for DNN testing [71] as a vulnerability discovery technology to help establish the quality assurance threshold of the target DNN. More specifically, this technique selects a set of test cases that achieve the highest possible coverage for the selected test quality criteria. Inspired by the success of this technique in detecting and revealing bugs during fuzzing, we introduce the *coverage-guided repairing* methodology named SAFETYREPAIR.

As explained in Section 4, in previous steps the synthetic images have been generated using an augmentation operator that validates the augmented seeds according to their photo-realism. In this step, a coverage-guided fuzzer is used to select failure patterns out of the set T_{κ} generated by the augmentation output. The selection is based on their effect on the targeted coverage criteria. Only mutated seeds that bring new desired information, i.e., reveal corner cases will be kept.

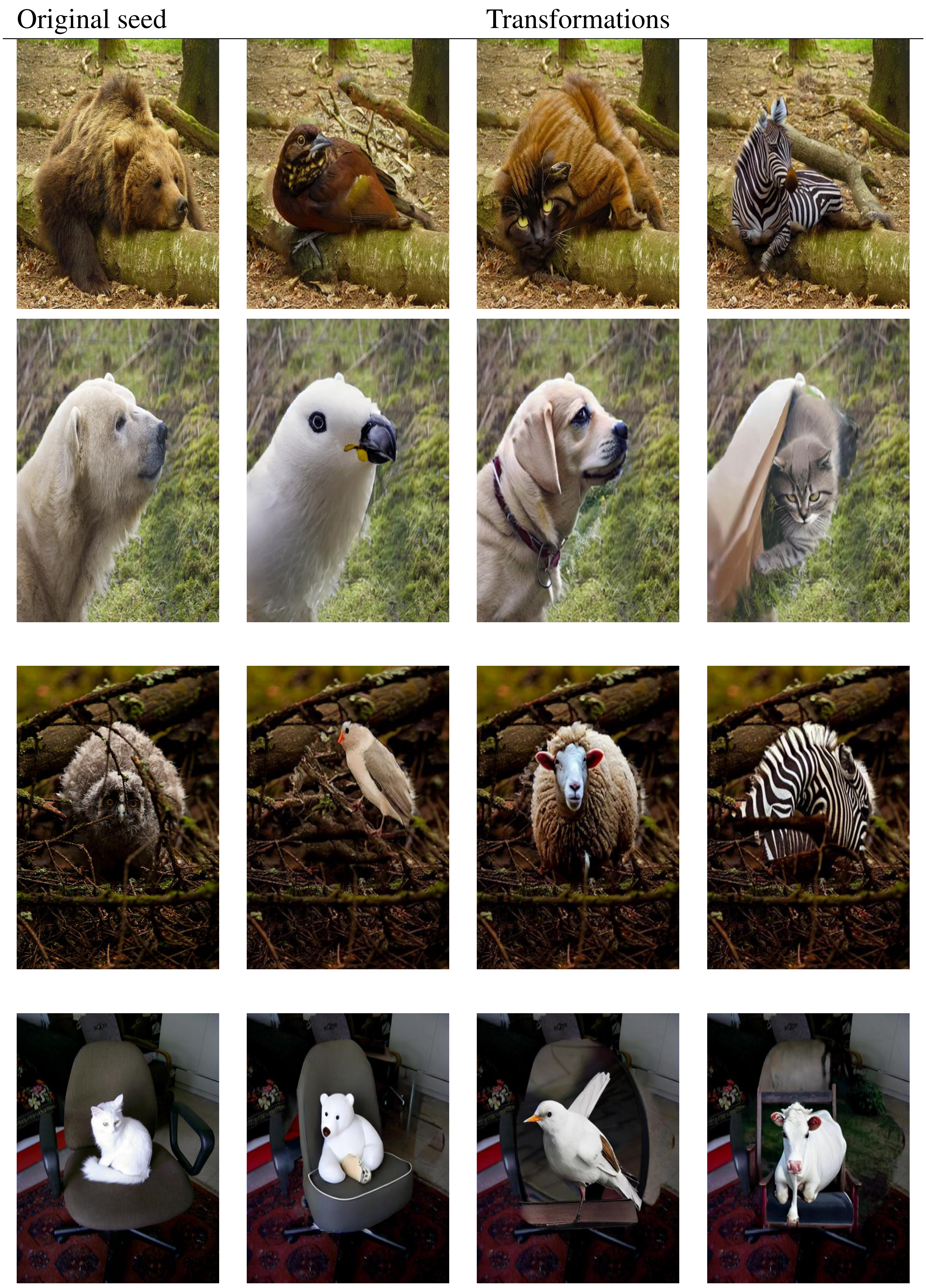


Figure 13: Images synthesised using GENERATIVEFUZZER with Stable Diffusion Inpainting as the augmentation technique. The initial dataset used as a reference is COCO with superclass 'animal'.

The proposed method focuses on selecting augmented images that maximize the dataset's coverage and can be used to repair the DNN model through a re-training process so that the DNN model decision boundaries expand and new knowledge can be learned. We next explain the coverage tracing process alongside continual learning and repairing.

3.4.1 Coverage Tracing

Our approach uses coverage guidance to filter valid augmented seeds T^{κ} and identify new test cases $\mathcal{T}$. SAFETYREPAIR keeps seeds that bring new semantic information to the testing set and adds them to the fuzzing queue. The semantic diversity of these seeds is quantified based on their ability to maximize selected coverage criteria when added to the test set T_{test}. $\mathcal{T}$ largely increases the fuzzing effectiveness of T_{train} and enables adding informative semantic information into the initial set. The rationale for this selection is that we want to augment the training set with inputs that will yield diverse behaviour to those already available in the dataset. Adding inputs similar to those existing in the dataset could be beneficial, but, certainly, to a lesser extent as the DNN would already have the ability to learn by using input with similar features.

Recently, many coverage criteria have been proposed for measuring the inner behaviours of DNN models ranging from neuron coverage to surprise adequacy criteria. Our approach is flexible and supports most of them. We select a wide range of test adequacy criteria that serve to analyse the inner behaviours of DNN models and provide feedback metrics to guide the selection of relevant augmented seeds that should be kept for repair. Each metric uses a specific test adequacy criterion that identifies the parts of DL logic exercised by a given test set.

SAFETYREPAIR currently supports neuron coverage (NC) [45], DeepImportance (IDC) [21], likelihood-based surprise adequacy (LSA) [31], and the neuron-level criteria k-Multisection Neuron Coverage (KMNC) and Neuron Boundary Coverage (NBC) from DeepGauge [37].

3.4.2 Continual Learning-based Repairing

One major challenge in Deep Learning is that the real-world application domain keeps evolving and even if it shares the same characteristics (e.g., input features, distributions) as the training/test data, new features are continuously introduced to the data space. To ensure that models generalize well to new unseen data, our repairing strategy needs to achieve a twofold goal: (i) acquire knowledge for new tasks; and (ii) retain knowledge from previously trained tasks (i.e., avoid catastrophic forgetting).

Regarding our application scenario (cf. Section 3.1), the new task can be viewed as new diseases that can be encountered in the vineyards during the fungicide spraying mission. The DNN model should be repaired so that it recognises this disease variant, thus informing the effective spraying of the affected vine plants.

In these cases, Continual Learning (CL) [56] can tackle this problem, as it accommodates the continuous change in the data-generating distribution. CL enables the model to be trained incrementally by adding new knowledge step by step. As a result, the DNN model becomes able to handle different types of data distribution shifts, i.e., detect changes in data and react accordingly.

Learning Paradigms for Continual Learning. Given an augmented dataset $T^{aug} = \mathcal{T} \cup T_{train}$, the goal is to learn the network (N^{j}) parameters θ_j for each time window $t \geq 0$ with data split T_t^{aug}, where T^{aug} is divided into a series of learning datasets $T_t^{aug} \subset T^{aug}$.

Depending on the overlap between the augmented set T_t^{aug} and the training set T_{train} distributions, i.e., $T_t^{aug} \sim \mathcal{X}_{nearDD}$, $T_{train} \sim \mathcal{X}_{inDD}$, we can differentiate between three CL scenarios, i.e., tasks as shown in Figure 14:

I) Continuous learning of known classes. In this case, with the augmented set T_t^{aug} we get additional examples for existing classes $C = \{c_1, ..., c_m\}$, the model N^j is trained on from the initial learning phase. This case represents a continuously growing application scenario, where both the training and augmented data distributions are identical.

II) Class incremental learning (i.e., new classes). In this case, the DNN model encounters unknown classes during runtime. The augmented set T^{aug} mitigates this problem by simulating examples of the new classes. Hence, the new data classes $\left|C^{'}\right| > |C|$, with $C^{'} = C \cup C^{aug}$. These new classes C^{aug} must be incorporated into the training set in a sequential process. Accordingly, we can opt for the addition of one single class $c_i \in C^{aug}$ to the learning process at each time step t, or all C^{aug} at time step t with fewer samples x' added at each time step.

III) Task incremental learning. This is a classical fine-tuning approach for new tasks. This case happens when two disjoint datasets are given. Thus, the deployment environment presents data characteristics considerably different from the training data, i.e., $\mathcal{X}_{inDD} \not\equiv \mathcal{X}_{nearDD}$. This distributional shift is essentially caused due to the dynamic nature of the real world, e.g., new problems (diseases) affect the vineyards. Accordingly, at time step t to achieve correct prediction y^t, the model is fine-tuned by re-learning the last layer only or by performing optimization steps for all layers using the new data distribution $\mathcal{X}_{nearDD}$.

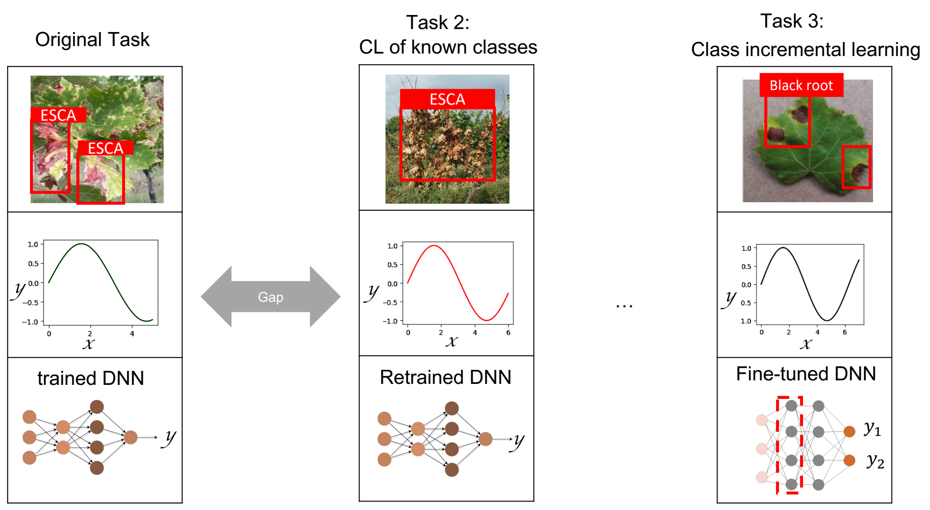


Figure 14: Illustrative example of different continuous learning paradigms adapted to the reality data gap between disease detection tasks.

Real-World Continual Learning Requirements. Real-world requirements for CL scenarios can be critical to the success of DL repairing. In this section, we summarise the requirements derived from the application scenario (Section 3.1). These requirements are manifold:

1. **Performance Stability.** Enhance the model's capability without harming its performance on ESCA data. In response to this requirement, our CL approach has to transfer knowledge from the initial task to the new one using the set of augmented data, while maintaining the model performance and prediction capability on $\mathcal{X}_{inDD}$ data.

2. **Constant Memory Consumption.** We need to establish the right computation-memory-accuracy trade-off in order to make the approach suitable for the hardware available from our use case partners. In fact, any learning failures during real-world CL will affect the hardware infrastructure, causing long delays in experiments. MRS are embedded platforms, thus, their limited memory and computation resources should be accounted for when planning the CL process to allow effective execution without failure.

3. **Incremental Training.** CL is performed sequentially over a sequence of timesteps t. This sequential learning process lacks stability; if one step is corrupted, that leads to memory degradation, and then to deterioration of the whole learning process.

4. **Ground Truth Availability.** Autonomous agents, i.e., drones, must have the ability to gather high-quality and useful data information while interacting with a new/unknown environment. These data samples serve as the ground truth base for the augmentation process and feed directly into the CL scenario.

3.4.3 Repairing Process

The intuition behind this approach is to simulate real-world situations under which DNN models usually fail to make correct predictions. In other words, simulating specific failure patterns (e.g., occlusion), then re-train a DNN on these patterns to enhance its accuracy without harming its performance on the original input dataset, i.e., in domain data. Our definition of failure patterns focuses on unseen data samples that the model could encounter in the operational environment. Our simulation of these failures is limited to synthetically generated inputs that are photo-realistic but considered near-distribution compared to the training dataset for the DNN system.

As described in Algorithm 1 (lines 3-15), to ensure that the selected augmented image x' is effective, the coverage criterion is computed. Then, the augmented input image is added to the corpus of effective augmented seeds T^{κ} if it exercises a new coverage score. As a result, only augmented images that satisfy the objective function $isGained(cov)$ are selected.

Following our main objective of mitigating DNN erroneous behaviour such that the model can address corner cases under real-world environments with high performance, we use T^{κ} for DNN repairing through retraining.

Algorithm 1 (lines 17-23) presents the whole process of our proposed method, following the continuous learning paradigm, which can be decomposed into two stages:

(i) Extending the original training set using a split of the selected augmented images set ($\mathcal{T}$ is split in a 80% - 20% ratio) to obtain T^{aug}_{train} which can be described as the training set augmented with corner cases generated using one or more augmentation operators $\mathcal{A}_{\kappa}$ (line 18). The remaining set is used to extend the test set and obtain T^{aug}_{test} that is be used to evaluate the DNN performance on the new task. T^{aug}_{test} is a combination of the original test set T_{test} and $\mathcal{T}$, which allows assessing the stability of DNN on the old task and ensures its ability to achieve reasonable performance on the new task.

(ii) Conducting the DNN repairing through the coverage-guided approach (lines 19-20), which includes calculating the model loss function and updating its weights and parameters.

Repairing the DNN misclassification involves inserting T^{aug}_{train} into the influential training samples such that the missing knowledge is learned and the DNN generalisation capability expands, and new parameters learnt. Formally, repairing the DNN is satisfying the objective function of optimizing the accuracy acc, by learning the model new parameters such that we optimize the loss function L:

$$\underset{\theta_j'}{\arg\min}\ \mathbb{E}_{(x,y)\sim\mathcal{X}_{nearDD}}\left[\mathcal{L}\left(\theta_j', x, y\right)\right] \tag{8}$$

where θ_j' denotes the model's new parameters.$\mathcal{L}$ denotes the task-related loss function; we use the cross-entropy function for the image classification task.

By retraining the model with these synthesized images, the decision boundary is fine-tuned such that the DNN erroneous behaviour, i.e., misclassified input is corrected.

Algorithm 1 Coverage Guided DNN Repairing

```
Input: Training dataset $T_{train}$, Testing dataset $T_{test}$, Augmented input seeds $T^{\kappa}$ pre-trained $N^j$, Coverage Criterion $S_i$,
ratio of data split $ratio$
Output: Repaired $N^{j'}$
 1: $\forall x' \in T^{\kappa}$ :
 2: if $x'$ is photo-realistic then
 3:     $T^{aug} \leftarrow x'$
 4: end if
        # Coverage Guided Fuzzing
 5: function FUZZING($N^j, T^{aug}, T_{test}, S_i$)
 6:     $cov_j,\ acc_j \leftarrow run(N^j, T_{test}, S_i)$
 7:     $\mathcal{T} \leftarrow \emptyset$
 8:     for $x' \in T^{aug}$ do
 9:         $\hat{y} = \boldsymbol{predict}(N^j, x')$
10:
11:         if $\hat{y} \neq y'$ then                          ▷ $y'$ is the label of $x'$
12:
13:             $F_{\kappa} \leftarrow x'$                     ▷ set of irrelevant augmented seeds
14:         else
15:             $Update(cov_j) \leftarrow run(N^j, x', S_i)$
16:             if $isGained(cov_j) \leftarrow True$ then
17:
18:                 $\mathcal{T} \leftarrow x'$                ▷ set of new/corner test cases
19:             else
20:                 $F_{\kappa} \leftarrow x'$
21:
22:             end if
23:         end if
24:     end for
25:     return $\mathcal{T}$
26: end function
27: # DNN repairing
28: function RETRAIN($N^j, \mathcal{T}$)
29:     $T^{aug}_{test},\ T^{aug}_{train} \leftarrow Split\&Merge(\mathcal{T}, T_{train}, T_{test}, ratio)$
30:     Calculate the loss function $\mathcal{L}\left(\theta_j{}', x, y\right)$ using $T^{aug}_{train}$
31:     Updating model weights and parameters $\theta_j{}'\ of\ \phi_{\theta_j{}'}$
32:     $N^{j'} \leftarrow \phi_{\theta_j{}'} \Leftrightarrow N^j \leftarrow N_j{}'$
33:     $cov_{\kappa}, acc_{\kappa} \leftarrow run\left(N_j{}', T^{aug}_{test}, S_i\right)$
34:     return $N^{j'}$
35: end function
```

4 Experimental Study

4.1 Research Questions

The goal of this experimental evaluation is to examine whether generative DNN models can be used to augment data that are diverse, photorealistic and semantically meaningful. Another goal is to showcase how practitioners could achieve the expected quality assurance targets for the augmented data using the developed coverage-guided fuzzing technique to perform testing, debugging, and repairing tasks for DNN-based software. We aim to achieve these goals by investigating the following research questions.

RQ1: Photo-realism *How effective is the proposed technique in generating synthetic images that are semantically meaningful?*

RQ2: (Effectiveness) *How effective is Stable Diffusion for coverage guided-fuzzing compared to metamorphic mutation?* i.e., can the generated test cases improve a given set in terms of its testing capability?

In particular, we assess if different synthetic image augmentation techniques activate different neuron activation combinations under the DeepKnowledge test adequacy criterion.

RQ3: DNN Repairing *How effective are different mutation techniques (semantic vs metamorphic) in improving DNN's accuracy?*

In particular, when combining data augmentation and retraining techniques for repairing DNN models we are interested in investigating the effectiveness of different augmentation strategies in improving the overall model performance.

4.2 Implementation

All experiments in our study were conducted on a high-performance computer running a cluster GPU with NVIDIA 510.39. We implement the proposed GenerativeRepair approach based on the state-of-the-art framework, open-source machine learning framework Keras (v2.2.2) [26] with Tensorflow (v2.6) backend. At a high level, we provide a prototype tool implemented as a set of Python scripts to enable quick integration by our partners and easy adaptation as a part of the EDDI ConSert.

To allow for reproducibility, our full implementation and evaluation subjects are available on Github[3]. In the following section, we summarise the experimental results.

4.3 Experimental Design and Settings

We perform a large-scale comparative study to answer the four research questions defined in the above section. For each dataset, we followed a specific set of principles to ensure adequate application of the adopted augmentation technique.

For RQ1, we carried out hyper-parameter analysis to select a suitable threshold δ for each fidelity metric for the fidelity assessment part of GenerativeRepair. To this end, δ was set to 0.8 and 0.6 for CLIP and SSIM, respectively (higher is better). After normalization between $[0, 1]$, the FID threshold was set to 0.2 (lower is better).

[3]https://github.com/sesame-project/MLTesting

To answer RQ2 and RQ3, we perform a large-scale controlled study on the effects of testing criteria, seed selection strategies, and data augmentation techniques. The controlled experiments are designed by considering two aspects:

1. whether the used testing criterion DEEPKNOWLEDGE is helpful for guiding seed selection.
2. the impact of data augmentation on the repairing process when equipped with the seed selection strategy.

We have intensively evaluated the following four strategies, with the combination of different seed selection and testing criteria as guidance:

- *Random + Noise* : We use this as a baseline to compare with the other coverage-guided strategies. We employed Gaussian noise to create this fuzzer for the comparative study. Gaussian noise is a metamorphic technique that adds white noise to the x_i input by adding a random number to every colour channel of each pixel.
- *Random + Inpainting*: This strategy guides CL using random seed selection and Stable Diffusion Inpainting with different testing criteria as feedback.
- *DeepKnw + Inpainting*: Similar to the previous, this strategy guides CL using Stable Diffusion Inpainting but with the DEEPKNOWLEDGE test adequacy criterion as feedback for seed selection.
- *Random + Semantic Occlusion*: In each iteration, this strategy selects an input seed randomly, then applies both random erasing and synthetic occlusion, then after the fidelity filtering step, different coverage criteria are used based on which new inputs are generated.

To reduce the effect of the randomness during fuzzing, the execution has been repeated 10 times for each strategy and the results are averaged.

Fine-tuning/Continuous Learning hyper-parameters. Regarding the training setup, we employ the stochastic gradient descent (SGD) optimizer with a batch size of 32 for each training window. The learning rate is set at 0.1 and the decay level at 0.0005.

Concerning the coverage criteria, we used for each approach the hyper-parameters recommended in its original research paper. We set the threshold for NC to 0.7. For KMNC, the k value, i.e., the number of multisections is set to 10. For LSA, we manually choose the layer in each DNN model. We analyze the fidelity of uncompressed synthetic images in the JPG format, as produced by each model (Table 2).

4.3.1 Subject datasets and DNN models

We select two popular publicly available datasets, i.e., COCO [36], CIFAR-10 [32], and we construct the third dataset, called “Grape Leaves”, as a mixture between online samples of coloured and grey grape leaves (JPG images) collected from Github and a selection of high-quality data provided by our partners.

For the COCO dataset, we applied a filtering step to select only images from the superclass ‘animal’. As a result, we get a dataset of 10 classes of animals. For each class, we selected 500 samples that were randomly split. We began by splitting the dataset in a 60% - 20% - 20% ratio for training, validation, and testing (respectively). These splits were used respectively for fine-tuning, estimating the GENERATIVEREPAIR coverage, and the last split for data augmentation purposes. More details about the number of augmented seeds and the ratio of valid images will be discussed in the next sections. We also have resized the COCO images and re-scaled them to (32, 32, 3) for computational reasons.

As shown in Table 2, for the subject datasets, we study popular pre-trained DNN models, i.e., LeNet5, AllConvNet, and VGG19. Each of these domains has been fine-tuned on the corresponding dataset and achieved competitive test accuracy as stated in Table 2.

Dataset	Description	DNN model	#Layers	Accuracy (%)
COCO	subset of 10 animal categories	Vgg19	19	82.04
CIFAR -10	colour images of 10 classes	LeNet5	5	77.20
Grape Leaves	coloured and gray images of 4 classes	AllConvNet	18	97.73

Table 2: Datasets and DNN models used in our experiments

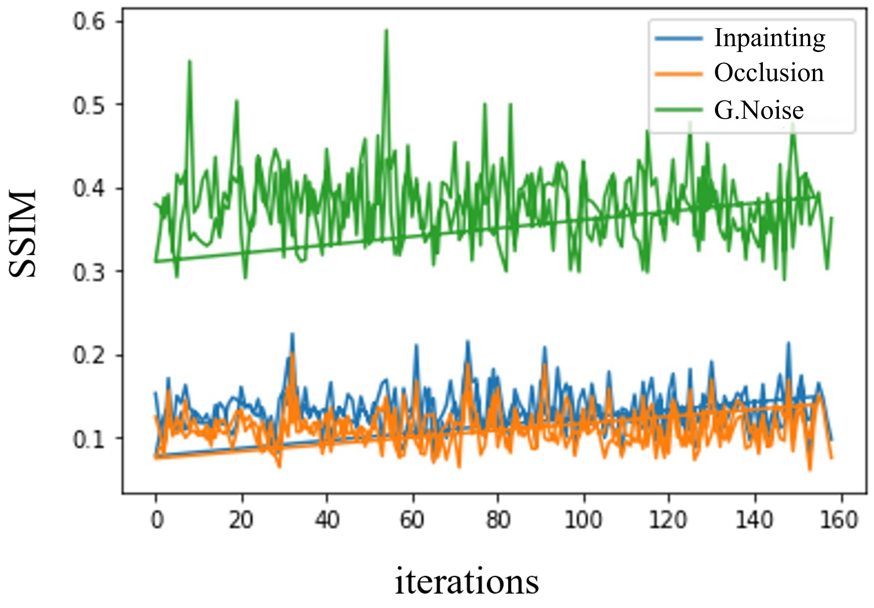


Figure 15: Performance of different augmentation strategies on the Grape Leaves dataset using the SSIM score.

5 Results and Discussion

5.1 RQ1: Photo-realism

To answer RQ1, we perform a large-scale controlled study using three image datasets (cf. Table 2). This experiment is designed to quantitatively assess the quality of the augmented images. For each selected input seed, the strategy augments $n = |C| - 1$ times (with C representing the data classes) and we deduct 1 to count for the input seed class. Each augmented seed is validated against the input seed and text prompt. Only those maximising the fidelity scores are kept.

Figure 15 shows an overview of how each augmentation strategy performs in terms of the fidelity of its generated images. In this figure, the three augmentation operators, i.e., Stable Diffusion Inpainting (Inpainting), Semantic Occlusion (Occlusion), and Gaussian Noise (G.Noise), are equipped with random seed selection to evaluate their performance independently of the test criteria guidance. The figure illustrates that Gaussian Noise (i.e., green chart) outperforms the semantic augmentation strategies based on the SSIM score (higher is better). To investigate these results more in-depth and assess the validity of SSIM as a fidelity score, we conduct a comparison between the SSIM, FID and CLIP scores.

Table 3 shows the ratio of valid synthetic images generated by different strategies through the Fidelity Estimation step using SSIM and FID. Overall, according to SSIM, the fidelity rates of COCO and Leaves datasets are generally higher than CIFAR-10 for both augmentation techniques, e.g., 49.44% of augmented seeds are evaluated as valid for COCO, while only 28.88% are valid augmented seeds for CIFAR-10. The FID metric provides different results, with 92.0% of augmented seeds by SD being valid for COCO and 8.11% for CIFAR-10.

	SSIM	FID	# augmented seeds
COCO	49.44%	92.0%	2750
Leaves	58.54%	89.04 %	1260
CIFAR 10	28.88%	8.11%	1800

Table 3: (%) Valid images generated by Stable Diffusion, according to fidelity scores FID and SSIM

Strategies	Coco	Leaves	CIFAR 10
Stable Diffusion	1596 (±58.0%)	1232 (97.78%)	226 (20.56%)
Gaussian Noise	450 (50%)	277 (22%)	7 (0.67%)
Semantic Occlusion	260 (58.22%)	62(5%)	13(1.14%)
# Initial seeds	50	460	150

Table 4: The ratio of valid images generated by different mutation strategies, using 0.8 threshold for CLIP score.

These results indicate that the fidelity metrics SSIM and FID are sensitive to the subject dataset and employed the augmentation strategy, complementing the findings in other studies like [7]. More specifically, FID and SSIM use different properties to evaluate the quality of images, which leads to incompatibility in their assessment results.

Another observation is that SSIM similarity scores are unsurprisingly low, e.g., only 28.88% of CIFAR augmented seeds are valid. This is not surprising as SSIM quantifies the similarity between input and augmented seed based on three key features (luminance, contrast, and structure). With Stable Diffusion Inpainting, the augmentation happens by introducing substantial changes into the input's image latent space, i.e., changes to these features, which consequently result in a high level of dissimilarity, hence, low SSIM values.

Thus, a deeper analysis using the CLIP score is performed and reported in Table 4. CLIP compares the augmented seed to the text prompt used to generate it. Different to SSIM, CLIP scores are highly comparable to FID. Figure 16 illustrates the significant match between CLIP and FID results when both are used to assess the fidelity of 512x512 augmented images using Stable Diffusion Inpainting technique.

Table 4 shows the results for augmented seed fidelity according to the CLIP score with threshold $\delta = 0.8$ after normalisation. The CLIP results are in line with FID and demonstrate the effectiveness of the SD Inpainting technique in generating photo-realistic images, especially for the Leaves and COCO datasets, with up to 97.78% of augmented seeds being evaluated as high fidelity for the Leaves dataset and 58% for the COCO dataset.

The fidelity rate of CIFAR-10 is generally lower than other datasets, with only 20.56%. Intuitively, the reason behind the low quality of CIFAR-10 augmented seeds is due to the resolution of CIFAR-10 which is relatively low.

Answer to RQ1: Results show that with proper constraint design and parameter tuning, Stable Diffusion is effective in generating high-quality synthetic images, deemed photorealistic, and reflect a wide breadth of in-domain data knowledge. It also achieved remarkable results with the Grape Leaves dataset with 97.78%. In addition, the Semantic Occlusion can be also effective when the occlusion is added carefully.

5.2 RQ2: Effectiveness

To answer RQ2, the experiments are designed to evaluate the effect of semantic data augmentation output, i.e., valid augmented seeds, (see Figure 9) on improving coverage in DNN testing under different criteria. We intensively evaluated two fuzzing strategies: (1)*Random-Inpainting*: adopts the uniform sampling seed

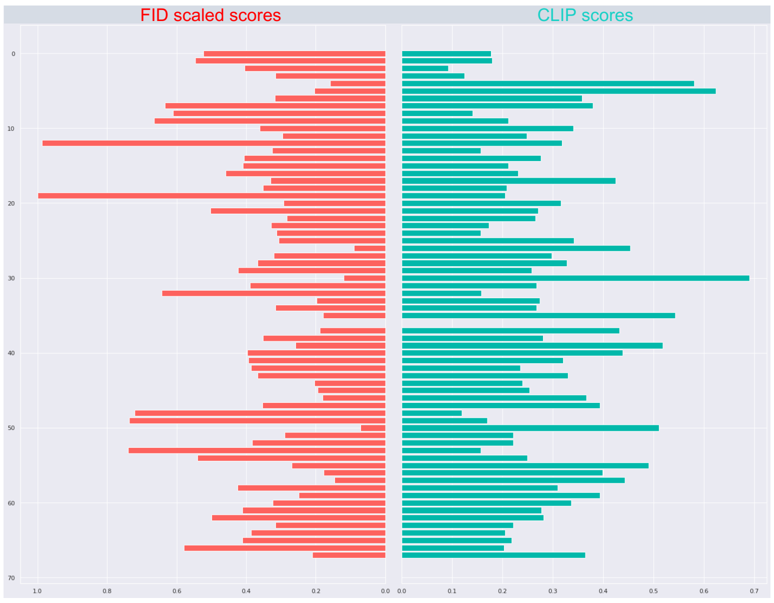


Figure 16: CLIP vs Scaled FID scores on 512x512 augmented images using the Stable Diffusion Inpainting technique. As shown, FID (lower shows better quality images), and CLIP (higher shows better quality images) quality assessment matches for most augmented images with Inpainting techniques.

prioritization strategy with Stable Diffusion Inpainting for data augmentation; and (2)*Random-Noise*: this is used as a baseline strategy.

Table 5 illustrates the results, where the row "Init." represents the coverage achieved by the initial test set T^t. In bold, we show the number that denotes the best coverage score for a specific adequacy criterion.

Compared to the initial test set (row Init.), we notice that test cases generated by Stable Diffusion $\mathcal{T}$, i.e., (i.e., *Random-Inpainting* line in Table. 6) improve the coverage scores across all the criteria when added to the test set by up to 23% and 26%. For instance, there was no difficulty in enhancing IDC, KMNC, and LSA criteria, when the test set was augmented with the new test cases $\mathcal{T}$, as they went respectively from 40.29%, 2.61%, 22.21% to 43.77% 16.71% 38.94% for the COCO dataset. On the other hand, with the Gaussian Noise, there was no significant increase, and in some cases, we noticed even a drop in the initial coverage scores.

Comparing the results between coverage-guide fuzzing strategies with semantic data augmentation (i.e., Stable Diffusion) and with metamorphic mutation (i.e., Gaussian noise), in most cases, the semantic data augmentation achieves higher coverage scores.

Answer to RQ2: CGF with semantic data augmentation, i.e., GENERATIVEREPAIR, is more effective to maximize coverage than random Gaussian noise ($Init.$ row in Table 6), and traditional CGF metamorphic technique (i.e., Gaussian Noise), especially for those criteria that are difficult to cover, i.e., IDC, LSA, and $KMNC$.

5.3 RQ3: DNN Repairing

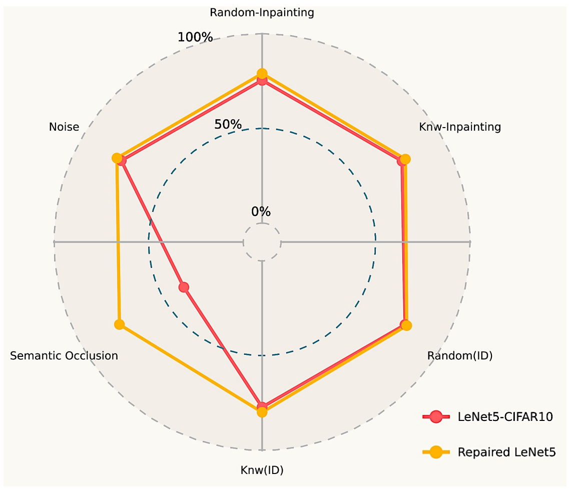
As detailed above, the used data augmentation techniques are leveraged to create different augmented datasets, thus resulting in 3 synthetically augmented sets for each DNN model alongside a Gaussian Noise set.

To answer this research question, we compare and evaluate these different augmented sets in both single-label and multi-label DNN-based classifications, including three DNN architectures for three initial datasets. As reference results, we use random-based selection and GENERATIVEREPAIR-based selection to select subsets from the In-domain distribution initial dataset.

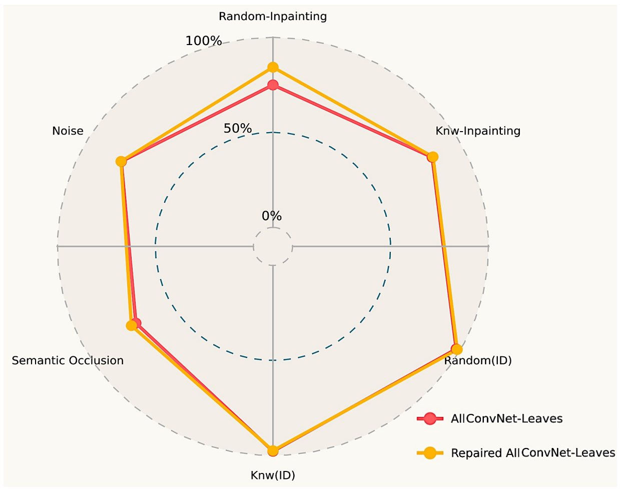
The results show that these data augmentation techniques all have their usefulness and scenarios in which they contribute the most. In particular, we retrain the DNN models (described in Table 2) with our new augmented set of images and assess if the detected bugs can be fixed after retraining.

Model	Strategies	$Knw_{0.3}$	KMNC	NBC	SNAC	NC
Vgg 19 +coco (animal 250)	Init.	40.29%	2.61%	13.66%	22.21%	5.93%
	Random + Inpainting	43.77%	**16.71**%	31.79%	38.94%	13.21%
	DeepKnw + Inpainting	**47.06**%	11.08%	**40.78**%	**42.09**%	**21.97**%
	Random + Noise	29.01%	14.60%	13.26%	18.12%	4.86%
	Random + Semantic Occlusion	30.08%	14.36%	13.19%	18.04%	4.79%
LeNet5 + CIFAR 10	Init.	**45.13**%	**21.24**%	0.73%	1.46%	**67.20**%
	Random + Inpainting	21.06%	13.78%	1.61%	1.31%	54.90%
	DeepKnw + Inpainting	22.30%	18.29%	**4.11**%	1.72%	63.08%
	Random + Noise	5.13%	14.27%	0.0 %	0.0 %	46.99 %
	Random + Semantic Occlusion	29.73%	19.76 %	1.97%	**3.80**%	43.77%
AllConvnet + Grape Leaves dataset	Init.	42.86%	19.62%	11.67%	49.59%	25.44%
	Random + Inpainting	68.75%	35.23%	42.55%	**69.17**%	32.90%
	DeepKnw + Inpainting	**72**%	**44.89**%	**46.06**%	68.61%	**36.86**%
	Random + Noise	37.50%	19.89%	21.69%	35.75%	8.29%
	Random + Semantic Occlusion	28.01%	36.23%	26.23%	51.42%	10.10%

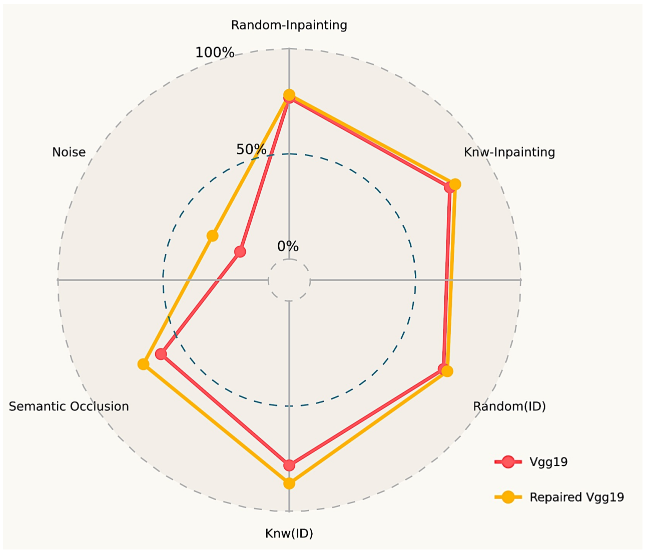
Table 5: Average improvement (%) of different coverage criteria over 10 runs by fuzzers with different configurations, i.e., seed selection and augmentation technique.



(a)



(b)



(c)

Figure 17: Improvement in accuracy of (a), (b) and (c) DNNs when the model is re-trained with augmented sets with the same number of inputs generated by different data augmentation strategies.

In our case, multiple executions using the same augmented dataset produce slightly different results each time we re-train the same model. Therefore, obtained quality metrics have been compared with the reference results (i.e., the Initial Accuracy column in Table 6) by averaging 5 runs for each model/training set combination.

Although we were able to generate a large number of test cases (corner cases) using the Stable Diffusion Inpainting technique, and these were capable of augmenting the coverage criteria, it is worth mentioning that these test cases are not fully representative of the real world. Thus continuous data augmentation and model repairing are needed to cover any failure patterns that can occur in the deployed operational environment.

These results highlight an important challenge which is how to repair the DNN while preserving its original performance and accuracy when handling in-distribution data that have been predicted correctly before. In other words, the key question is how to achieve both repairing and retaining performance simultaneously.

To answer this reason, we employ *Continual learning* (CL) which is a sequential learning technique that aims to extend the DNN abilities to new tasks as they come while preserving its ability to perform previous tasks.

For each augmented dataset, we start by evaluating the trained model respectively VGG19, LeNet5, and All-ConvNet on the augmented set (augset line in Table 6). Then, we proceed by repairing the models via our proposed method. We evaluate the re-trained models' performance by calculating their accuracy separately on: the augmented set and original test set (line **CL Accuracy (Xtest+ augset)** 6), the augmented set alone (line CL Accuracy (augset) 6), and finally on the original test set alone (CL Accuracy (X_test) 6).

In general, different learned models are known to be experts in specific subsets of data, i.e., data distribution domain, and when being fine-tuned they transfer the learned specification to the new subset data. The results in Table 6) confirm this principle. In fact, the fine-tuned models using Stable Diffusion, both Random-guided and GENERATIVEREPAIR-guided Inpainting generated subsets, become "more expert" (highly specialised) in detecting images generated by the corresponding techniques. For instance, the LeNet5 model trained in CIFAR10 has an initial accuracy of 75.39% when evaluated on the original Test set augmented with samples generated by Random-guided Inpainting (XTest +augset). Then, after retraining on 1000 samples generated by the same technique, the repaired LeNet5 achieved an accuracy of 78.91% on the same set (XTest +augset).

A similar behaviour is shown after repairing the model with Semantic Occlusion, i.e., the LeNet5 model accuracy went from 37.77% to 77.06% after being repaired using 1000 images generated by Semantic Occlusion. For AllConvnet, repairing the model with Semantic Occlusion samples helped to increase the model accuracy by 2% going from 70.73% initially to 73.36% after repairing.

In general, our repairing method shows significant advantages over baseline methods (Random-based repairing) on the three CNN architectures under 4 different techniques, including a failure data type, i.e., Gaussian Noise, demonstrating the effectiveness and generalization of the proposed method. Specifically, these results prove that using generative models to augment the dataset for model repair and hardening is highly beneficial.

The accuracy reduction on Xtest. Although the accuracy results can be unsatisfactory for subsets belonging only to the augmented dataset, i.e., rows CL Accuracy (augset) on Table 6, the repaired model still performs better than the initial accuracy. This behaviour can be explained by the fact that the CL has a twofold purpose: one is about ensuring that the model is performing well on the new tasks (in our case, the augmented dataset) while, secondly, not harming the model's capability of handling original task, i.e., clean data (In-DD).

To conclude, our repairing method enabled the DNN model to use its previous experience to generalize and improve its capabilities with the presence of synthetic augmented data, while also being resilient to catastrophic failure in which new information can be detrimental to its performance with previous tasks. This is in contrast with the transfer learning-based repair strategy in which the DNN model would lose its ability to perform the original task because it adapted its performance totally to the new augmented dataset.

Overall, we first identify that the data augmentation (guided by coverage criteria) DNN repairing methodology can help to address the problem of DNN erroneous behaviour and failures during runtime. In particular, Continuous Learning paradigms serve as a successful repairing approach when dealing with a limited number of new test cases, i.e., augmented data (comparatively to the initial training set). However, it raises new challenges related to improving the DNN generalisability without harming its performance on the in-domain data.

Answer to RQ3: Both semantic augmentation techniques and in particular, the generative AI model, i.e., Stable Diffusion Inpainting, enhance the accuracy on all novel/previously unseen data patterns significantly, demonstrating that the proposed coverage-guided semantic data augmentation indeed can repair the DNN under some specific patterns effectively.

		Random-Inpainting	Knw-Inpainting	Random(ID)	Knw(ID)	Semantic Occlusion	Noise
Leaves + AllConvNet	Init. Accuracy (Xtest+ augset)	**74.94%**	**83.77%**	97.92%	97.79%	**70.73%**	79.30%
	Init. Accuracy (augset)	13.84%	4.70%	100.00%	100.00%	17.82%	23.53%
	CL Accuracy (Xtest+ augset)	**84.26%**	**84.26%**	98.50%	97.46%	**73.36%**	79.47%
	CL Accuracy (augset)	27.34%	12.75%	99.38%	96%	24.72%	23.88%
	CL Accuracy (X_test)	90.57%	95.80%	98.18%	97.50%	98.18%	96.36%
CIFAR -10 + LeNet5	Init. Accuracy (Xtest+ augset)	**75.39%**	**75.40%**	77.22%	77.19%	**37.77%**	75.94%
	Init. Accuracy (augset)	9.90%	8.52%	78.50%	76.15%	6.67%	9.44%
	CL Accuracy (Xtest+ augset)	**78.91%**	**77.46%**	78.24%	79.88%	**77.06%**	78.59%
	CL Accuracy (augset)	12.87%	10.00%	77.80%	77.06%	10.56%	6.11%
	CL Accuracy (X_test)	80.24%	79.32%	78.26%	79.91%	78.26%	79.91%
COCO + VGG19	Init. Accuracy (Xtest+ augset)	**76.67%**	**78.04%**	74.62%	78.08%	**60.38%**	16.96%
	Init. Accuracy (augset)	5.74%	–%	71.08%	72.00%	10.00%	9.02%
	CL Accuracy (Xtest+ augset)	**78.00%**	**81.00%**	76.67%	86.67%	**70.00%**	32.08%
	CL Accuracy (augset)	12.00%	–%	75.62%	77.50%	20.00%	14.15%
	CL Accuracy (X_test)	82.03%	–%	89.87%	88.00%	82.34%	32.08%

Table 6: Accuracy of the original DNNs on the synthetic test cases (i.e., augset) and on the augmented testing datasets (Xtest+augset) before and after the repairing process using different techniques.

5.4 Threats to Validity

Construct Validity. The selection of the subject datasets and DNN models could be a threat to the validity of our GenerativeRepair approach. To counter this issue and maintain the effectiveness of the proposed approach, we used 3 well-studied datasets with diverse complexity and popular pre-trained DNN models that have competitive performance. It is worth noting that our methodology is generic and can be applied to any object classification dataset/model combination. However, it is also important to highlight some of the findings, where the experimental results demonstrated that high-resolution datasets are better suited for the proposed augmentation strategies and especially for semantic occlusion.

Another threat is the semantic mutation using the Inpainting technique that could not guarantee the validity of generated images in all cases. We counter this issue by designing and evaluating a whole filtering process that we called Fidelity Estimation. The use of multiple methods for predicting the perceived quality of digital images, i.e., SSIM, FID, CLIP, helps select semantically valid images and further mitigate any bias in the individual assessment of each of the used measures.

Internal Validity. The primary internal threat to the validity of our methodology is the limited data size of the generated corner cases, i.e., synthetic images, compared to the initial training set. To counter this issue we adopted the Continuous Learning paradigm. One of its main advantages is dynamically expanding the DNN capacity with only the necessary number of data samples, and preventing semantic drift by splitting/duplicating input samples in specific time windows.

A further threat is related to the randomness factor during model repair. We counter this issue by repeating multiple times each Continuous Learning configuration, generating multiple re-trained models, and averaging the results.

External Validity. Our proposed approach is generic and can be easily expanded using different: (i) fidelity measures, (ii) coverage criteria; and, more importantly, (iii) datasets and DL-based systems. To mitigate the external threats to validity and ensure the portability of this approach, we developed GenerativeRepair on top of the open-source frameworks Keras and TensorFlow, and make the open-source implementation publicly available on GitHub.

6 Satisfaction of SESAME Requirements

ID	Requirement	Priority	Status
D113	Support white-box analysis of the ML model given the training and testing sets.	SHALL	Full: The proposed tool enables testing the Deep neural networks by tracing the influence of each training input on the individual computation units of the model and tracking its fault-revealing ability
D114	Quantify the adequacy of the test set for the target ML model.	SHALL	Full: the tool provides facilities to improve assurance of deployed DNN models, through a novel test adequacy criterion based on the model generalisation capability.
D115	Synthesise new inputs using the training/testing sets as source.	SHALL	Full: The tool encompasses three different techniques for data augmentation and enable both semantic and metamorphic synthesis of new data inputs
D116	Select scenarios to be used in the online verification mode.	SHALL	Full: The tool has enabled the generation of new test cases that can be used for online verification
U99	Enable assessing whether classification/prediction limitations exhibited in offline mode are present in online mode (during simulation).	SHALL	Full: The tool has been tested against adversarial attacks and proven to be sensitive to adversarial inputs and is effective in detecting misbehaviours in test sets.
U100	Improve the capacity of the ML component given the testing adequacy result.	SHALL	Full: The tool provides a continuous repairing mechanism to improve the model's performance based on the coverage estimated during runtime.

Table 7: Satisfaction of SESAME Requirements from deliverable D.1.1: Assurance of Machine Learning

7 Conclusion

This report details the repairing and hardening technique for Deep learning components developed in Task 6.3. The provided approaches are built upon results from Tasks 6.1 and 6.2.

The GenerativeRepair framework has a twofold objective: (a) monitor the Data-Driven and Learning components (Deep Learning models) that are part of the MRS (e.g. person detection or object recognition tasks); and (b) improving the Deep Learning model's robustness in the MRS system. Our tool is a design-time artifact, it focuses on improving and hardening the EDDI for fault diagnosis by increasing the capability of SafeML to detect faulty behaviour.

To this end, the report gives a comprehensive presentation of the developed GenerativeFuzzer and SafetyRepair approaches, which both efficiently improve DL testing and repairing in an offline manner by augmenting the dataset with novel/corner test cases, then, performing sequential retraining for DNN model. This repair process dynamically expands the DL system capacity upon any changes detected in the data distribution and/or model's performance during the runtime task.

The overall methodology focuses on improving the robustness of DNN generalisation capability using a coverage-guided generation technique. The proposed data augmentation method is cast as an optimization problem that simulates the natural environmental patterns, by optimizing the photo-realism of the synthetic images. In order to identify the novel/corner cases, each augmented input data goes through a coverage-guided fuzzing stage based on DeepKnowledge as a test adequacy criterion. This step serves as a discovery approach to select novel patterns that the model has not seen before, i.e., maximise the used coverage criterion. Then, the continuous repairing phase gradually and effectively expands the model's generalisability with the new data samples, while also preventing any semantic drift by splitting data inputs and timestamping them as per the established Continuous Learning process.

We find that novel/corner cases from our model generated with Stable Diffusion models are both photorealistic and reflect a wide breadth of the real-world, when evaluated by different fidelity measures. They also help improve the DNN model response to corner cases in the operational environments.

We validate GenerativeRepair on multiple use cases with different datasets under lifelong learning scenarios (task changes, see Figure 6), on which it significantly outperforms traditional learning methods for DNNs. The approach achieves a good level of performance as the batch counterparts with substantially fewer synthetic images.

The proposed approach has been applied to the SESAME use case for Autonomous Pest Management in Viticulture (DKOX). The used methodology for DNN repairing and generative AI-based augmentation technique empirically validated in this deliverable has enhanced the accuracy of the disease detection model on all the simulated failure patterns significantly. Thus, demonstrating the effectiveness of our approach for the DKOX use case. The work is also being repurposed for PAL's use case with a few adaptations to support pose detection tasks. Then, GenerativeRepair will be applied to other suitable SESAME uses cases, as applicable and once their real-world data is available.

References

[1] Dario Amodei, Chris Olah, Jacob Steinhardt, Paul Christiano, John Schulman, and Dan Mané. Concrete problems in ai safety. *arXiv preprint arXiv:1606.06565*, 2016.

[2] Koorosh Aslansefat, Ioannis Sorokos, Declan Whiting, Ramin Tavakoli Kolagari, and Yiannis Papadopoulos. Safeml: safety monitoring of machine learning classifiers through statistical difference measures. In *International Symposium on Model-Based Safety and Assessment*, pages 197–211. Springer, 2020.

[3] Yogesh Balaji, Seungjun Nah, Xun Huang, Arash Vahdat, Jiaming Song, Karsten Kreis, Miika Aittala, Timo Aila, Samuli Laine, Bryan Catanzaro, et al. ediffi: Text-to-image diffusion models with an ensemble of expert denoisers. *arXiv preprint arXiv:2211.01324*, 2022.

[4] Ms Aayushi Bansal, Dr Rewa Sharma, and Dr Mamta Kathuria. A systematic review on data scarcity problem in deep learning: solution and applications. *ACM Computing Surveys (CSUR)*, 54(10s):1–29, 2022.

[5] Alexey Bochkovskiy, Chien-Yao Wang, and Hong-Yuan Mark Liao. Yolov4: Optimal speed and accuracy of object detection. *arXiv preprint arXiv:2004.10934*, 2020.

[6] Mariusz Bojarski, Davide Del Testa, Daniel Dworakowski, Bernhard Firner, et al. End to end learning for self-driving cars, 2016.

[7] Ali Borji. Pros and cons of gan evaluation measures: New developments. *Computer Vision and Image Understanding*, 215:103329, 2022.

[8] Efstathios Branikas, Paul Murray, and Graeme West. A novel data augmentation method for improved visual crack detection using generative adversarial networks. *IEEE Access*, 11:22051–22059, 2023.

[9] Dominique Brunet, Edward R Vrscay, and Zhou Wang. On the mathematical properties of the structural similarity index. *IEEE Transactions on Image Processing*, 21(4):1488–1499, 2011.

[10] Simon Burton, Lydia Gauerhof, and Christian Heinzemann. Making the case for safety of machine learning in highly automated driving. In *International Conference on Computer Safety, Reliability, and Security*, pages 5–16. Springer, 2017.

[11] Oliver Chang, Jonathan Metzman, Max Moroz, Martin Barbella, and Abhishek Arya. Oss-fuzz: Continuous fuzzing for open source software. *URL: https://github. com/google/ossfuzz*, 2016.

[12] Tsong Yueh Chen, Fei-Ching Kuo, Huai Liu, Pak-Lok Poon, Dave Towey, TH Tse, and Zhi Quan Zhou. Metamorphic testing: A review of challenges and opportunities. *ACM Computing Surveys (CSUR)*, 51(1):1–27, 2018.

[13] Antonia Creswell, Tom White, Vincent Dumoulin, Kai Arulkumaran, Biswa Sengupta, and Anil A Bharath. Generative adversarial networks: An overview. *IEEE Signal Processing Magazine*, 35(1):53–65, 2018.

[14] Wei Dai, Huimin Lu, Junhao Xiao, Zhiwen Zeng, and Zhiqiang Zheng. Multi-robot dynamic task allocation for exploration and destruction. *Journal of Intelligent & Robotic Systems*, 98:455–479, 2020.

[15] Prafulla Dhariwal and Alexander Nichol. Diffusion models beat gans on image synthesis. *Advances in Neural Information Processing Systems*, 34:8780–8794, 2021.

[16] Wan-Cyuan Fan, Yen-Chun Chen, DongDong Chen, Yu Cheng, Lu Yuan, and Yu-Chiang Frank Wang. Frido: Feature pyramid diffusion for complex scene image synthesis. *arXiv preprint arXiv:2208.13753*, 2022.

[17] Yang Feng, Qingkai Shi, Xinyu Gao, Jun Wan, Chunrong Fang, and Zhenyu Chen. Deepgini: prioritizing massive tests to enhance the robustness of deep neural networks. In *Proceedings of the 29th ACM SIGSOFT International Symposium on Software Testing and Analysis*, pages 177–188, 2020.

[18] Andrea Fioraldi, Alessandro Mantovani, Dominik Maier, and Davide Balzarotti. Dissecting american fuzzy lop–a fuzzbench evaluation. *ACM Transactions on Software Engineering and Methodology*, 2023.

[19] Xiang Gao, Ripon K Saha, Mukul R Prasad, and Abhik Roychoudhury. Fuzz testing based data augmentation to improve robustness of deep neural networks. In *Proceedings of the acm/ieee 42nd international conference on software engineering*, pages 1147–1158, 2020.

[20] Luca Gazzola, Daniela Micucci, and Leonardo Mariani. Automatic software repair: A survey. In *Proceedings of the 40th International Conference on Software Engineering*, pages 1219–1219, 2018.

[21] Simos Gerasimou, Hasan Ferit Eniser, Alper Sen, and Alper Cakan. Importance-driven deep learning system testing. In *2020 IEEE/ACM 42nd International Conference on Software Engineering (ICSE)*, pages 702–713. IEEE, 2020.

[22] Harshvardhan GM, Mahendra Kumar Gourisaria, Manjusha Pandey, and Siddharth Swarup Rautaray. A comprehensive survey and analysis of generative models in machine learning. *Computer Science Review*, 38:100285, 2020.

[23] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep learning*. MIT press, 2016.

[24] Ian Goodfellow and Nicolas Papernot. The challenge of verification and testing of machine learning, 2017.

[25] C. Le Goues, T. Nguyen, S. Forrest, and W. Weimer. GenProg: A generic method for automatic software repair. *IEEE Transactions on Software Engineering*, 38(1):54–72, 2012.

[26] Antonio Gulli and Sujit Pal. *Deep learning with Keras*. Packt Publishing Ltd, 2017.

[27] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask r-cnn. In *Proceedings of the IEEE international conference on computer vision*, pages 2961–2969, 2017.

[28] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. *Advances in Neural Information Processing Systems*, 33:6840–6851, 2020.

[29] Md Johirul Islam, Rangeet Pan, Giang Nguyen, and Hridesh Rajan. Repairing deep neural networks: Fix patterns and challenges. In *Proceedings of the ACM/IEEE 42nd International Conference on Software Engineering*, pages 1135–1146, 2020.

[30] Bahjat Kawar, Shiran Zada, Oran Lang, Omer Tov, Huiwen Chang, Tali Dekel, Inbar Mosseri, and Michal Irani. Imagic: Text-based real image editing with diffusion models. *arXiv preprint arXiv:2210.09276*, 2022.

[31] Jinhan Kim, Robert Feldt, and Shin Yoo. Guiding deep learning system testing using surprise adequacy. In *2019 IEEE/ACM 41st International Conference on Software Engineering (ICSE)*, pages 1039–1049. IEEE, 2019.

[32] Alex Krizhevsky and Geoff Hinton. Convolutional deep belief networks on cifar-10. *Unpublished manuscript*, 40(7):1–9, 2010.

[33] Yann Lecun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *Nature*, 521(7553):436–444, 2015.

[34] Hanbang Liang, Xianxu Hou, and Linlin Shen. Ssflow: Style-guided neural spline flows for face image manipulation. In *Proceedings of the 29th ACM International Conference on Multimedia*, pages 79–87, 2021.

[35] Aristidis Likas, Nikos Vlassis, and Jakob J Verbeek. The global k-means clustering algorithm. *Pattern recognition*, 36(2):451–461, 2003.

[36] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft coco: Common objects in context. In *Computer Vision–ECCV 2014: 13th European Conference, Zurich, Switzerland, September 6-12, 2014, Proceedings, Part V 13*, pages 740–755. Springer, 2014.

[37] Lei Ma, Felix Juefei-Xu, Fuyuan Zhang, Jiyuan Sun, Minhui Xue, Bo Li, Chunyang Chen, Ting Su, Li Li, Yang Liu, et al. Deepgauge: Multi-granularity testing criteria for deep learning systems. In *Proceedings of the 33rd ACM/IEEE International Conference on Automated Software Engineering*, pages 120–131, 2018.

[38] Zakariae Machkour, Daniel Ortiz-Arroyo, and Petar Durdevic. Classical and deep learning based visual servoing systems: a survey on state of the art. *Journal of Intelligent & Robotic Systems*, 104(1):11, 2022.

[39] Alex Nichol, Prafulla Dhariwal, Aditya Ramesh, Pranav Shyam, Pamela Mishkin, Bob McGrew, Ilya Sutskever, and Mark Chen. Glide: Towards photorealistic image generation and editing with text-guided diffusion models. *arXiv preprint arXiv:2112.10741*, 2021.

[40] Artem Obukhov and Mikhail Krasnyanskiy. Quality assessment method for gan based on modified metrics inception score and fréchet inception distance. In *Software Engineering Perspectives in Intelligent Systems: Proceedings of 4th Computational Methods in Systems and Software 2020, Vol. 1 4*, pages 102–114. Springer, 2020.

[41] A Odena, C Olsson, D Andersen, and others. Tensorfuzz: Debugging neural networks with coverage-guided fuzzing. *on Machine Learning*, 2019.

[42] Augustus Odena, Catherine Olsson, David Andersen, and Ian Goodfellow. Tensorfuzz: Debugging neural networks with coverage-guided fuzzing. In *International Conference on Machine Learning*, pages 4901–4911. PMLR, 2019.

[43] University of York. D6.1 assurance of data-driven and learning components of eddis., 2022.

[44] Nicolas Papernot, Patrick McDaniel, Somesh Jha, Matt Fredrikson, et al. The limitations of deep learning in adversarial settings. In *International Symposium on Security and Privacy (S&P)*, pages 372–387, 2016.

[45] Kexin Pei, Yinzhi Cao, Junfeng Yang, and Suman Jana. Deepxplore: Automated whitebox testing of deep learning systems. In *proceedings of the 26th Symposium on Operating Systems Principles*, pages 1–18, 2017.

[46] Mauro Pezze and Michal Young. *Software testing and analysis: process, principles, and techniques*. John Wiley & Sons, 2008.

[47] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. Learning transferable visual models from natural language supervision. In *International conference on machine learning*, pages 8748–8763. PMLR, 2021.

[48] Alexander J Ratner, Henry Ehrenberg, Zeshan Hussain, Jared Dunnmon, and Christopher Ré. Learning to compose domain-specific transformations for data augmentation. *Advances in neural information processing systems*, 30, 2017.

[49] Vincenzo Riccio, Nargiz Humbatova, Gunel Jahangirova, and Paolo Tonella. DeepMetis: Augmenting a deep learning test set to increase its mutation score. September 2021.

[50] Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10684–10695, 2022.

[51] Connor Shorten and Taghi M Khoshgoftaar. A survey on image data augmentation for deep learning. *Journal of big data*, 6(1):1–48, 2019.

[52] Jascha Sohl-Dickstein, Eric Weiss, Niru Maheswaranathan, and Surya Ganguli. Deep unsupervised learning using nonequilibrium thermodynamics. In *International Conference on Machine Learning*, pages 2256–2265. PMLR, 2015.

[53] Jeongju Sohn, Sungmin Kang, and Shin Yoo. Search based repair of deep neural networks. *arXiv preprint arXiv:1912.12463*, 2019.

[54] Youcheng Sun, Xiaowei Huang, Daniel Kroening, James Sharp, Matthew Hill, and Rob Ashmore. Testing deep neural networks. *arXiv preprint arXiv:1803.04792*, 2018.

[55] Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. Sequence to sequence learning with neural networks. In *International Conference on Neural Information Processing Systems*, pages 3104–3112, 2014.

[56] Hasan Tercan, Philipp Deibert, and Tobias Meisen. Continual learning of neural networks for quality prediction in production using memory aware synapses and weight transfer. *Journal of Intelligent Manufacturing*, 33(1):283–292, 2022.

[57] Yuchi Tian, Kexin Pei, Suman Jana, and Baishakhi Ray. Deeptest: Automated testing of deep-neural-network-driven autonomous cars. In *Proceedings of the 40th international conference on software engineering*, pages 303–314, 2018.

[58] Toan Tran, Trung Pham, Gustavo Carneiro, Lyle Palmer, and Ian Reid. A bayesian data augmentation approach for learning deep models. *Advances in neural information processing systems*, 30, 2017.

[59] Anwaar Ulhaq, Naveed Akhtar, and Ganna Pogrebna. Efficient diffusion models for vision: A survey. *arXiv preprint arXiv:2210.09292*, 2022.

[60] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

[61] Yan Wang, Peng Jia, Luping Liu, Cheng Huang, and Zhonglin Liu. A systematic review of fuzzing based on machine learning techniques. *PloS one*, 15(8):e0237749, 2020.

[62] Xuesu Xiao, Bo Liu, Garrett Warnell, and Peter Stone. Motion planning and control for mobile robot navigation using machine learning: a survey. *Autonomous Robots*, 46(5):569–597, 2022.

[63] Xiaofei Xie, Wenbo Guo, Lei Ma, Wei Le, Jian Wang, Lingjun Zhou, Yang Liu, and Xinyu Xing. Rnnrepair: Automatic rnn repair via model-based analysis. In *International Conference on Machine Learning*, pages 11383–11392. PMLR, 2021.

[64] Xiaofei Xie, Lei Ma, Felix Juefei-Xu, Minhui Xue, Hongxu Chen, Yang Liu, Jianjun Zhao, Bo Li, Jianxiong Yin, and Simon See. Deephunter: a coverage-guided fuzz testing framework for deep neural networks. In *Proceedings of the 28th ACM SIGSOFT International Symposium on Software Testing and Analysis*, pages 146–157, 2019.

[65] Xiaofei Xie, Lei Ma, Felix Juefei-Xu, Minhui Xue, Hongxu Chen, Yang Liu, Jianjun Zhao, Bo Li, Jianxiong Yin, and Simon See. DeepHunter: a coverage-guided fuzz testing framework for deep neural networks. In *Proceedings of the 28th ACM SIGSOFT International Symposium on Software Testing and Analysis*, ISSTA 2019, pages 146–157, New York, NY, USA, July 2019. Association for Computing Machinery.

[66] Jingkang Yang, Pengyun Wang, Dejian Zou, Zitang Zhou, Kunyuan Ding, Wenxuan Peng, Haoqi Wang, Guangyao Chen, Bo Li, Yiyou Sun, et al. Openood: Benchmarking generalized out-of-distribution detection. *arXiv preprint arXiv:2210.07242*, 2022.

[67] Xiangnan Yin, Di Huang, Zehua Fu, Yunhong Wang, and Liming Chen. Segmentation-reconstruction-guided facial image de-occlusion. In *2023 IEEE 17th International Conference on Automatic Face and Gesture Recognition (FG)*, pages 1–8. IEEE, 2023.

[68] Bing Yu, Hua Qi, Qing Guo, Felix Juefei-Xu, Xiaofei Xie, Lei Ma, and Jianjun Zhao. Deeprepair: Style-guided repairing for deep neural networks in the real-world operational environment. *IEEE Transactions on Reliability*, 71(4):1401–1416, 2021.

[69] Chuan-Wei Zhang, Meng-Yue Yang, Hong-Jun Zeng, and Jian-Ping Wen. Pedestrian detection based on improved lenet-5 convolutional neural network. *Journal of Algorithms & Computational Technology*, 13:1748302619873601, 2019.

[70] Jie M Zhang, Mark Harman, Lei Ma, and Yang Liu. Machine learning testing: Survey, landscapes and horizons. *arXiv preprint arXiv:1906.10742*, 2019.

[71] Jie M Zhang, Mark Harman, Lei Ma, and Yang Liu. Machine learning testing: Survey, landscapes and horizons. *IEEE Transactions on Software Engineering*, 2020.

[72] Shansi Zhang, Chao Sun, Zhi Feng, and Guoqiang Hu. Trajectory-tracking control of robotic systems via deep reinforcement learning. In *2019 IEEE International Conference on Cybernetics and Intelligent Systems (CIS) and IEEE Conference on Robotics, Automation and Mechatronics (RAM)*, pages 386–391. IEEE, 2019.

[73] Zhun Zhong, Liang Zheng, Guoliang Kang, Shaozi Li, and Yi Yang. Random erasing data augmentation. August 2017.

[74] Zhi Quan Zhou and Liqun Sun. Metamorphic testing of driverless cars. *Communications of the ACM*, 62(3):61–67, 2019.