D2.4 Safety Guidelines

Version 1.0

Documentation Information

Contract Number	101069595
Project Website	www.safexplain.eu
Contratual Deadline	31.09.2025
Dissemination Level	PU
Nature	R
Authors	Giuseppe Nicosia (EXI), Stefano Lodico (EXI), Carlo Donzella (EXI), Javier Fernández (IKR), Xuban Ceccon (IKR)
Contributors	Irune Yarza (IKR), Irune Agirre (IKR), Francesca Guerrini (EXI), Jaume Abella (BSC), Martí Caro (BSC)
Reviewed by	Jaume Abella (BSC) Frank Geujen (NAV)
Keywords	AI, Functional Safety, V&V, Catalog Scenarios, Safety Concept

This project has received funding from the European Union's Horizon Europe programme under grant agreement number 101069595.

Change Log

Version	Description Change
V0.1	First draft
V0.2	Internal Review
V1.0	Final Version

Table of Contents

Executive Summary

The main objective of this document is to present the key conclusions drawn from the reviews conducted by certification authorities and technical standards assessment experts of the previously consolidated safety guidelines generated as a result of WP2 activities applied to specific use case domains.

In addition, this document provides an update of Deliverable D2.3, incorporating enhancements and extensions to the previously established Deep Learning (DL) software Verification and Validation (V&V) strategy initially proposed. It includes a more detailed Systems Theoretic Process Analysis (STPA) Analysis, particularly focusing on the safety measures required to mitigate the identified hazardous situations at element level and finally, it also refines the previously defined strategies and specific solutions for implementing safety patterns and its associated safety mechanisms in safety-critical software systems that incorporate Artificial Intelligence (AI) components.

Finally, a dedicated section is included to present the applicable standards and normative references relevant to the project's safety assurance framework related to this project, which can be found in 4.4 Applied Normative:

- A summary for the ISO 26262, just a brief summary, since it was already described in the previous deliverables (linked in the section)
- A summary for ISO 21448, just a brief summary, since it was already described in the previous deliverables (linked in the section)
- A summary of the ISO/PAS 8800, not actually applied in this project as published in December 2024, but summarized here as it applies to the project itself (described in the dedicated section)
- ASPICE, applied to some use cases (described in the dedicated section)

1 Introduction

This document compiles the results from all tasks developed in Work Package (WP) 2: T2.1, T2.2, T2.3, T2.4, and T2.5. The objective of these tasks is to define and evaluate functionalities based on DL components that are safety-related or whose use may have implications for safety functions.

This document is organized as follows:

- Section 2 describes the feedback from who analysed each use case, performed by external resources from the domain owner
- Section 3 refines specific solutions for implementing safety patterns and its associated safety mechanisms, describes the evaluation methodology applied to their assessment, and presents the results obtained.
- Section 4 describes the analyses done for each use case, in particular for the safety measures as output of the STPA analysis including some examples for each domain and a description of each normative applicable in WP2.

This deliverable includes a list of acronyms and abbreviations as well as a list of references in all Sections. This deliverable ends with Annex A, which includes the PowerPoint presented to ESA and additional space experts.

2 Safety Reviews for the specific use cases domains

This section is divided into three subsections, each corresponding to a specific use case domain for which the reviews have been conducted:

- For the railway domain, the review was carried out by TÜV Rheinland, along with an internal review conducted by our colleagues from EXIDA.
- For the aerospace domain, meetings were scheduled, and email exchanges took place with ESA.
- For the automotive domain, the review was conducted by EXIDA personnel.

2.1 TÜV Railway

In the previous deliverable (D2.3), we presented the results of a safety concept applied to the railway domain. This safety concept proposed a set of architectural patterns adapted to different usage levels of AI constituent and the associated safety mechanisms for each pattern. At the time of writing that deliverable, we documented the full assessment process conducted by IKERLAN in collaboration with the certification authority TÜV Rheinland. However, although we had a comprehensive understanding of the evaluation, the final assessment report had not yet been received, and thus the safety review remained preliminary.

Following the reception of the official report from TÜV Rheinland, we can now confirm the conclusions anticipated in D2.3. According to TÜV Rheinland, the safety concept of the railway use case—centered on an AI-based collision avoidance system—has been positively assessed against the requirements and

recommendations of ISO 5469, while also demonstrating compatibility with conventional functional safety standards such as IEC 61508 and the railway-specific CENELEC standards.

The report includes a detailed account of the technical discussions held during the evaluation meetings, covering key topics such as:

- The pseudo-stochastic nature of AI-related faults.
- Safety patterns in the railway domain depending on AI usage levels and AI technology classification as defined in ISO 5469.
- Associated diagnostics and monitoring mechanisms for each safety pattern.
- Handling of AI-specific risk factors.

As a result, the assessment concludes that all critical aspects required for the safe integration of AI constituents into safety-related software systems have been appropriately addressed. The strategies and solutions proposed are considered a sound and reasonable architectural approach, well tailored to the specific demands of the railway use case.

2.2 EXIDA Automotive

The Exida considerations about SAFEXPLAIN for the automotive use case, based on formal and informal communication received from Automotive use case responsible (NAVINFO)in the last 6 months, is described in this section.

The contribution of NAVINFO (leader of the Automotive use case) in the project is not much related to the production of the so-called “items” (e.g. the vehicle systems or sub-system) as typically provided by OEMs and Tier 1 suppliers but rather related to the provision of add-on solutions.

The participating NAVINFO teams have undergone a considerable strategic evolution in terms of structure and focus.

One of the contributions of NAVINFO is related to the development of a model of the simulation environment partially based on ODDs and the catalogue of the scenarios provided by Exida: Exida provided 21 scenarios, NAVINFO focused on one of them.

The result of this development is very interesting from the DEMO’s point of view: even though it is limited in terms of completeness, it shows that the model is implementable with appropriate effort in an industrial project.

The DEMO developed is provided as Open Source and it will showcase all the developments contributed by NAVINFO.

In terms of Functional Safety scope (that is the viewpoint of WP2), these are the limitations to consider:

- From the scenarios proposed by Exida, one of them has been implemented;
    - The implemented scenario is not fully consistent with the proposed scenario, as it has some limitations regarding some parameters and conditions.
- Specific recommendations from Exida regarding scenario-based test cases, triggering condition-based test cases and safety measures have been partially considered.

- NAVINFO demonstrates a commendable commitment to achieve compliance with the ISO 26262 and ISO 21448 SOTIF, however the compliance with the normative cannot be fully achieved with just the adopted approach (see below “Example of V&V strategy application: Automotive Use Case”), for the detailed reasons explained there. Some critical hazardous behaviours were not considered (e.g., override abortion).

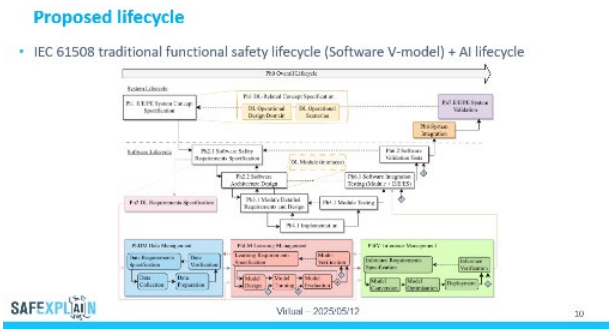
2.3 ESA Space

On 12 May 2025, the SAFEXPLAIN consortium presented its latest results to representatives of several aerospace and embedded system industries including Airbus DS, BrainChip, the European Space Agency (ESA), Gaisler, and Klepsydra, showcasing major strides in making AI safe and certifiable in critical autonomous embedded systems.

The meeting was hosted by project partner, AIKO, space case leader and a company in the space sector providing scalable, up-to-date and user-friendly AI-powered solutions for the industry. AIKO opened the meeting by introducing the project’s role in closing the gap between AI solutions and safety culture. AIKO representative Gabriele Giordana highlighted that, “the project is paving the way for something new by bridging the gap of different industrial domains that share the same challenge of safe AI integration.” In the following subsections, we present the agenda followed during the meeting, the reference materials shared with the ESA attendees to facilitate a clearer understanding of the content presented, and a summary of the main discussion points and feedback received—both during the meeting and in subsequent email exchanges. The PowerPoint presentation is provided in Annex A.

2.3.1 ESA meeting agenda

- IKERLAN presented its AI-Functional Safety Management (AI-FSM) methodology developed in the project. This methodology guides the development process, maps the traditional lifecycle of safety-critical systems with the AI lifecycle, and addresses their interactions.

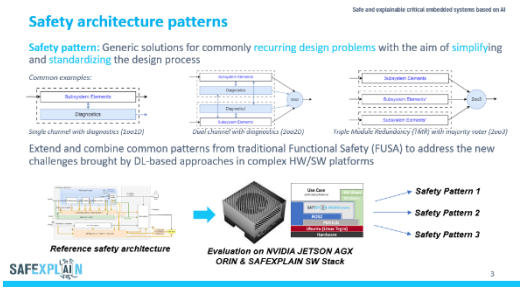


More information on the AI-FSM can be found here https://safexplain.eu/integrating-ai-into-functional-safety-management/

- Exida development followed with an overview of the project’s V&V strategy, which defines a valid method for proving the intended functionalities at the vehicle level and at the component level. The strategy considers ISO 26262 standard and ISO 21448 testing techniques. For the space case the V&V strategy has been applied to a guidance, navigation and control (GNC) technology implanted in a space vehicle.

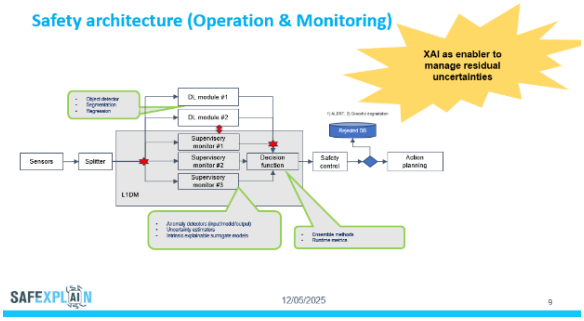
More information on Exida development's scenario catalogues as part of the V&V strategy can be found here.

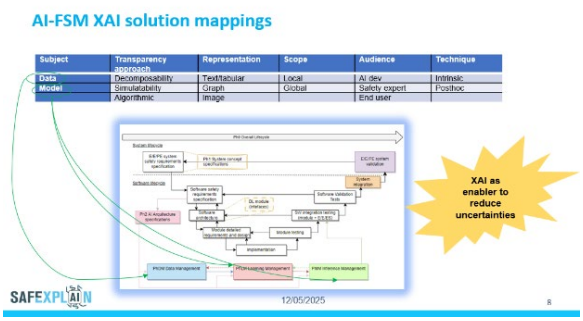
- IKERLAN introduced the reference safety architecture patterns for the adoption of DL in safety-critical systems with varying safety requirements. This topic addressed the importance of runtime safety mechanisms for dealing with random and residual systematic faults, HW/SW complexity, DL model insufficiencies and to support DL explainability.



More information on IKERLAN´s work on "Safety patterns for AI-based systems" can be found here. https://safexplain.eu/safety-patterns-for-ai-based-systems/

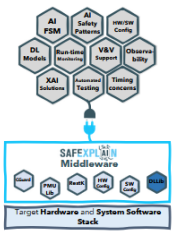
- RISE discussed challenges, strategies and examples of including AI in safety critical systems and making it explainable. Specifically, RISE discussed strategies and examples of using explainable AI (XAI) to reduce uncertainties and as an enabler to manage residual uncertainties.





More information on XAI can be found here: https://safexplain.eu/rise-webinar-highlights-xai-for-systems-with-functional-safety-requirements/

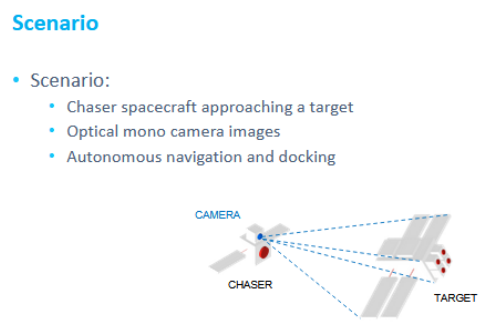
- The Barcelona Supercomputing Center highlighted the project's platform level support and tools, including support for HW integration, a common layer for SW V&V and functional safety mechanisms, an abstraction layer for system confirmation and a reference common stack.



More information on the SAFEXPLAIN platform can be found here: https://safexplain.eu/bsc-webinar-demonstrates-interoperability-of-safexplain-platform-tech-tools/

- AIKO then went into detail of the implementation of the space case study, specifically describing a docking scenario, an example that proved to be especially interesting for ESA representatives. The functions under evaluation, safety needs,

technology used and system integration were presented to show the value of the SAFEXPLAIN results.



More information on the space case study can be found here https://safexplain.eu/space-case-study/

- The presentation wrapped up with the unveiling of the core demo instantiated to the space domain.

2.3.2 Key meeting highlights

In this section it will be explained the ESA's considerations about SAFEXPLAIN project for the aerospace use case (led by AIKO) and its potential application on their projects, based on feedback received both during the meeting and through subsequent email exchanges.

Adopting AI for ESA missions represents a difficult challenge for the near future, since there are no ECSS standard addressing the certification and/or qualification of AI on-board.

To use AI on board, inference time robustness and uncertainty quantification improvements are the keys, but formal methods are here preferred.

Formal methods analytically assess the correctness of an output, while safety cages or qualitative methods give this guarantee only under a certain probability, which can pose issues to the safety compliance of the system.

This can be considered just for lower categories (not safety relevant), since just a COTS SW (e.g., linux-based system) cannot be used in a high-critical application such as the space use case developed in the SAFEXPLAIN project, but in a medium/long term future (3 ... 5 years) AI will be surely used in ESA missions, if the ESA-ECSS standards evolve to reach this goal. Currently it is present with an ESA - ECSS handbook for ML, but it does not address all the safety aspects.

Explainable AI could be useful in the process to support the humans work in the V&V strategy, even though it is less relevant during inference, unless a practical method which applies it in real time is developed and can be applied to quantitatively assess and monitor the AI function.

In the space field, the challenge of introducing AI is even more complex than in other sectors, due to the high level of criticality and the harsh environmental conditions, distance and visibility of the assets, impacting communication and control, and the many degrees of freedom.

Indeed, an important point depends on whether a human can interfere in the loop of the XAI execution, because absence of humans makes its utility, currently, limited.

Some concerns are:

- The V&V of AI components, the documentation starting from the ODD and the verification of the ODD against the impossibility of covering the entire input space.
- Whether methods beyond the formal ones are considered more feasible for XAI, since the formal methods are limited by architecture and scalability.

The focus of the future AI applications is to maximize accuracy while mitigating the latency in various use cases.

Techniques such as redundancy of critical components, inference on tiles of the original input image or temporal consistency verification are renowned and interesting, but some constraints need to be considered such as memory restrictions on an embedded platform or multitasking management of the models, or repeating the inference on porting of the same image, since they can represent a cost to trade off. SAFEXPLAIN challenged this aspect by testing theoretical solutions in an embedded device.

Another challenging aspect is to certify systems with multiple agents and manage their interaction. For instance, in SAFEXPLAIN, a standalone critical system is assessed thanks to its modularity. The whole system is considered feasible also for the functional safety aspects.

ESA is working on AI application for navigation and docking manoeuvres even it is more difficult than automotive use cases, and this confirms the relevance of SAFEXPLAIN works in this direction. Indeed, SAFEXPLAIN is a great initiative to take inspiration for the update of ESA - ECSS standards, even though SAFEXPLAIN can be improved on HW features to ease integration in safety critical systems.

In conclusion, the presentation was well received and complimented for its holistic approach that considers all aspects of integrating AI into safety critical systems. Industry representatives mentioned the relevance of the presentation given their interest in integrating the right AI technologies for activities like docking, computer vision and more, and for developing robust certification processes that can be developed without disrupting the current V&V cycle.

This feedback on the SAFEXPLAIN's results is especially valuable given the important roles these companies play in the space sector and their expert perspective on advancements for safety critical applications in the space domain.

3 Evaluation of Technical Solution

This section evaluates the suitability of a variety of diagnostics mechanisms defined in deliverable D2.2. This section is decomposed into the following subsections:

- Subsection 3.1 introduces some concepts related to the assessment of the results.
- Subsection 3.2 outlines the set-up under which the assessment has been carried out.
- Subsection 3.3 collects the results of assessment and includes conclusions.

3.1 Basic concepts

This subsection collects information about concepts and metrics employed for the assessment.

3.1.1 Bounding boxes

Bounding boxes are rectangular coordinates that enclose an object in an image to indicate its location and size. They are widely used in computer vision, specifically in the domain of object detection.

3.1.2 Intersection over Union

Intersection over Union (IoU) quantifies the overlap between two bounding boxes by dividing the area of their intersection (∩) by the area of their union (∪). The IoU value ranges from 0 to 1, where a value of 1 indicates a perfect overlap between a specific bounding box and another one, and a value of 0 indicates no overlap at all. Therefore, higher IoU values indicate more precise localization. The equation for IoU is:

$$IoU = \frac{Area\ (B_{\text{pred1}} \cap B_{\text{pred2}})}{Area(B_{\text{pred1}} \cup B_{\text{pred2}})}$$

Here, B_{pred1} represents the bounding box produced by one of the detectors, and B_{pred2} corresponds to the one generated by a second object detector. This metric is widely used in the field of computer vision, typically for evaluating model accuracy by comparing predictions against the ground truth. Although it can also be employed to assess the consistency between two separate detectors. For that case, if the computed IoU exceeds a predefined threshold, the detection is considered consistent.

Based on the IoU metric and the definition of an IoU threshold, and assuming that one of the bounding boxes is designated as the ground truth, the metrics True Positive (TP), False Positive (FP), and False Negative (FN) are defined as follows:

3.1.3 True Positive, False Positive, False Negative and True Negatives

In object detection evaluation, each predicted bounding box is matched to a ground-truth annotation of the same class when their IoU is equal to or exceeds a specified threshold. Based on this matching and the presence or absence of predictions each case is classified as one of the following:

- TPs occur when a predicted box both correctly identifies an object's class and meets the IoU threshold with a corresponding ground-truth box.
- FPs arise when a predicted box either fails to match any ground-truth annotation at the required IoU or is assigned the wrong class.
- FNs represent ground-truth objects for which no prediction of the correct class reaches the IoU threshold.
- Additionally, TNs correspond to the correct absence of any prediction in areas where no ground-truth object exists, although in dense detection benchmarks like COCO True Negatives are not explicitly counted because the background comprises an infinite number of potential detections.

3.1.4 Non-Maximum Suppression

Non-maximum suppression (NMS) is not a metric by itself but a post-processing technique that filters out redundant detections by retaining, for each object class, only the highest confidence bounding box among those that overlap beyond a specified IoU threshold and discarding the rest. It's applied by default in YOLO

models during inference output post-processing and it ensures that overlapping boxes of the same class are suppressed in favor of the most reliable prediction.

3.1.5 Precision

Precision is a performance metric that measures the proportion of correctly predicted positive instances among all instances predicted as positive by a model. It reflects how accurate the model's positive predictions are, the ratio being between the number of TP predictions and the total number of positive predictions made by the model:

$$Precision = \frac{TP}{\text{TP} + \text{FP}}$$

3.1.6 Recall

Recall quantifies the fraction of ground-truth objects that the model successfully detects. It is defined as the ratio of TP to the sum of TP and FN:

$$Recall = \frac{TP}{\text{TP} + \text{FN}}$$

where TP denotes the number of correctly detected objects and FN the number of missed objects. A higher recall indicates that the model locates a greater share of all objects, though without regard for precision it can also allow more incorrect detections.

3.1.7 Accuracy

Accuracy (ACC) is a performance metric that quantifies the proportion of correct predictions made by a model compared to the total number of predictions. It is commonly used in classification tasks, including object detection:

$$ACC = \frac{TP}{TP + FP + FN}$$

3.1.8 F1 Score

F1 Score is a metric which is calculated from Precision and Recall and combines them into a single number. It uses the harmonic mean, which penalizes extreme values more heavily than the arithmetic mean. This metric is particularly relevant in those cases in which there is an uneven class distribution or when both FP and FN carry important consequences. The equation of F1 Score is defined as follows:

$$F_1 = \frac{2 \times \text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}$$

3.1.9 COCO Eval

COCO eval is an evaluation protocol used in the COCO dataset, a widely used benchmark for object detection as well as other computer vision tasks. COCO eval has the feature of giving performance metrics about different IoU overlaps, and different area sizes. This gives us a detailed understanding of a model's

strength and weaknesses, as well as allowing fair comparison between models. The following paragraphs explain basic concepts (Area, maxDets) and metrics (AR, AP):

Area

The ground truth contains bounding boxes of objects of different types. COCO eval groups them into three categories: small, medium and large. The metrics are calculated for each of these object sizes which gives insights into how the model performs at different sizes. Small objects tend to be the hardest to detect, while large objects are generally easier to localize.

maxDets

MaxDets defines the maximum number of bounding box predictions per image that are retained for evaluation. In the COCO protocol, common settings for maxDets are one, ten and one hundred predictions per image to simulate different detection budgets. A low maxDets forces the model to output only its highest confidence detections, usually increasing precision at the expense of recall. By contrast, high maxDets allows more predictions, which raises recall but also increases the risk of false positives. Evaluating performance across these settings reveals the tradeoff between detection coverage and error rate.

AR

The COCO evaluation extends single threshold recall into Average Recall (AR) by computing recall at ten equally spaced IoU thresholds from 0.50 to 0.95 and then taking the mean. By averaging recall over these overlap criteria, AR captures the model's ability to find objects under varying localization strictness, offering a complete perspective on missed detections alongside Average Precision (AP), which is explained below.

AP

To deliver a more comprehensive evaluation, the COCO eval methodology replaces single-threshold precision with AP, which aggregates performance across multiple IoU criteria. Formally, AP is defined as the mean area under the precision recall curve over ten evenly spaced thresholds from 0.50 to 0.95:

$$AP_{COCO} = \frac{1}{10}\sum_{t=1}^{10}\int_{0}^{1} p_t(r)\ dr$$

Where $p_t(r)$ denotes the precision at a recall level r.

3.2 Setup

This subsection presents the experimental framework used to conduct the evaluation. It includes the prediction combination strategy, fault injection methodology, models, hardware platforms, validation dataset, and the COCO evaluation protocol used for validation.

3.2.1 Aim of the experiments

The aim of the experiments is to evaluate the impact of implementing Diverse Redundancy Schemes (DRS), as outlined in SAFEXPLAIN Deliverable D2.2 (Section 3.1 and Table 2), on the quality of object detection outputs. This is measured through COCO evaluation metrics explainedin subsection 3.1.9. In addition, the

experiments assess the the schemes' ability to maintain overall system performance during runtime in the presence of failures. For that, we have implemented a fault injection campaign, adapting the bit-flip methodology from D2.2 (Section 7.1.2.1) as explained in 3.2.2 to mimic random hardware faults, systematic errors, or model insufficiencies.

The tested DRS configurations align with D2.2 framework for varying Diagnostic Coverage (DC) levels:

- DRS_1 (Low coverage): Lockstep execution with replicated models on the same or diverse platforms (e.g., GPU-GPU or GPU-CPU), addressing traditional functional safety risks and making the system fault tolerant with theoretically identical computations.
- DRS_2/DRS_3 (Low to medium coverage): Inference platform and development diversity, such as combining PyTorch and TensorRT frameworks to mitigate framework-specific systematic faults and variations introduced during model export/optimization.
- DRS_4 (Medium to high): Combination of DRS2 and DRS3, which address AI and traditional FUSA risk factors together with performance insuffiencies.
- DRS_5 (Medium to high coverage): Concept diversity pairing standard object detection (e.g., pairing standard object detection with part-based detection (e.g., body parts like head, torso, limbs)) to handle AI insufficiencies like occlusions or noise.

3.2.2 Combining Predictions with a Decision Function

The rationale for combining predictions with a decision function is detailed in the “3.1.1 Decision Function” section of deliverable D2.2. It presents three decision functions: averaging, voting and NMS. The following experiments use NMS (Subsection 3.1.4) which tends to improve recall by ensuring more predicted objects are retained, but it does not necessarily improve precision, as valid detections with lower confidence might be suppressed.

3.2.3 Fault injection methodology

All experiments were carried out under controlled stress conditions by injecting small perturbations into the weights of the first convolutional layer of YOLO Convolutional Neural Network (CNN). This layer, which contains 32 kernels of size 3×3×3 (a total of 864 weights), was selected because it forms the basis of all subsequent feature extraction. By altering these early weights, we simulate hardware faults, systematic errors or model insufficiencies.

Adapting D2.2's bit-flip methodology (Section 7.1.2.1), we use seeded random noise addition to simulate transient faults in early layers. To generate the perturbations, a noise tensor N with the same shape as the weight tensor W was created via the PytTorch’s “*torch.randn_like(W)”* function as follows:

$$N = \text{torch.randn\_like(W)} * 0.01$$

This produces samples from a normal distribution N(0, 0.01), so that roughly 99.7% of noise values lie within ±0.03. Before each inference passes, we add N to W, yielding modified weights W’ = W + N, and then perform a forward pass. A fixed random seed guarantees that each run is reproducible, while regenerating N per image ensures that every inference experiences a distinct perturbation.

3.2.4 Underlying Platforms

All benchmarks were conducted on the NVIDIA Jetson AGX Orin platform. The NVIDIA Jetson AGX Orin is designed for embedded real-time systems, featuring 32GB of unified memory and efficient power consumption. It operates at a maximum of 50W, with reduced power usage in lower-performance modes. Unless otherwise specified, benchmarks utilized the GPU exclusively, except for the diverse platform benchmark, which leveraged both GPU and CPU

3.2.5 Validation dataset

The dataset used is the Common Objects in Context (COCO) -2017 validation split. It contains 5000 images and 80 objects. Regarding classes, coco dataset has 80 classes which are imbalanced, meaning that there are more objects of certain classes, like person which is 30% of the validation dataset. experiments do not focus on specific classes, but on the entirety of the dataset classes.

3.2.6 Coco Eval Validation Methodology

COCO eval has been used for experiments which are explained in depth section 3.1.7. In the results, we present the following metrics:

Table 1. Metrics and associated identifier employed for the assessment

ID	Metric
AP1	AP @IoU=0.50:0.95 (all areas, maxDets=100)
AP2	AP @IoU=0.50 (all areas, maxDets=100)
AP3	AP @IoU=0.75 (all areas, maxDets=100)
AP4	AP @IoU=0.50:0.95 (small areas)
AP5	AP @IoU=0.50:0.95 (medium areas)
AP6	AP @IoU=0.50:0.95 (large areas)

3.2.7 Models

The baseline object detection model for this study is YOLOv8x which is a convolutional neural network comprised of 68.2 million parameters. YOLOv8x processes input images at 640 × 640 pixels by default, with non-standard resolutions handled through upsampling or downsampling.

For the Diverse Concept experiment a custom body part detection model was trained based on the YOLOv11x architecture. This model detects whole person's and specific body parts[1]. It has been trained on a relabeled COCO 2014 dataset augmented with DensePose annotations and background images, containing 13,483 training images and 2,215 validation images. The model, trained on an NVIDIA RTX 4090 for 70 epochs with a batch size of 22 and 640px image size, achieves an overall mAP50 of 0.926 and mAP50-95 of 0.624 across all classes, with strong performance on person (mAP50: 0.993) and head (mAP50: 0.993) detection.

[1] Body parts: Person, Head, Torso, Upper Arm, Lower Arm, Upper Leg, Lower Leg, Hand, and Foot

3.3 Results

This section presents results from three experiments testing Diverse Redundancy Schemes (DRS_1: diverse platforms (Sections 3.3.2.1 and 3.3.2.2), DRS_2/DRS_3/DRS_4: framework diversity (Sections 3.3.3 and 3.3.3.2), DRS_5: concept diversity (Sections 3.3.4.1) from Section 3.2.1. We evaluate their impact on object detection quality using COCO metrics, checking performance in fault-free and fault-injected scenarios. Before presenting the experimental results, we conduct a preliminary analysis of the dataset used, which helps contextualize and interpret some of the observations discussed later in this section.

3.3.1 Preliminary Dataset Analysis

We provide an overview of the classes' distribution of the COCO-2017 training and validation datasets inFigure 1 and Figure 2. The blue bars depict the number of instances per class in the training dataset, while the orange bars correspond to the validation set. As shown, the distribution is clearly non-uniform. In fact, the class "person", is the by far the most presented, with 262,465 appearances in the training set and 11,004 in the validation set, followed by car and chair. At the tail end of the distribution, classes like "hair dryer" are much less frequent, appearing only 198 times in the training set and 11 times in the validation set.

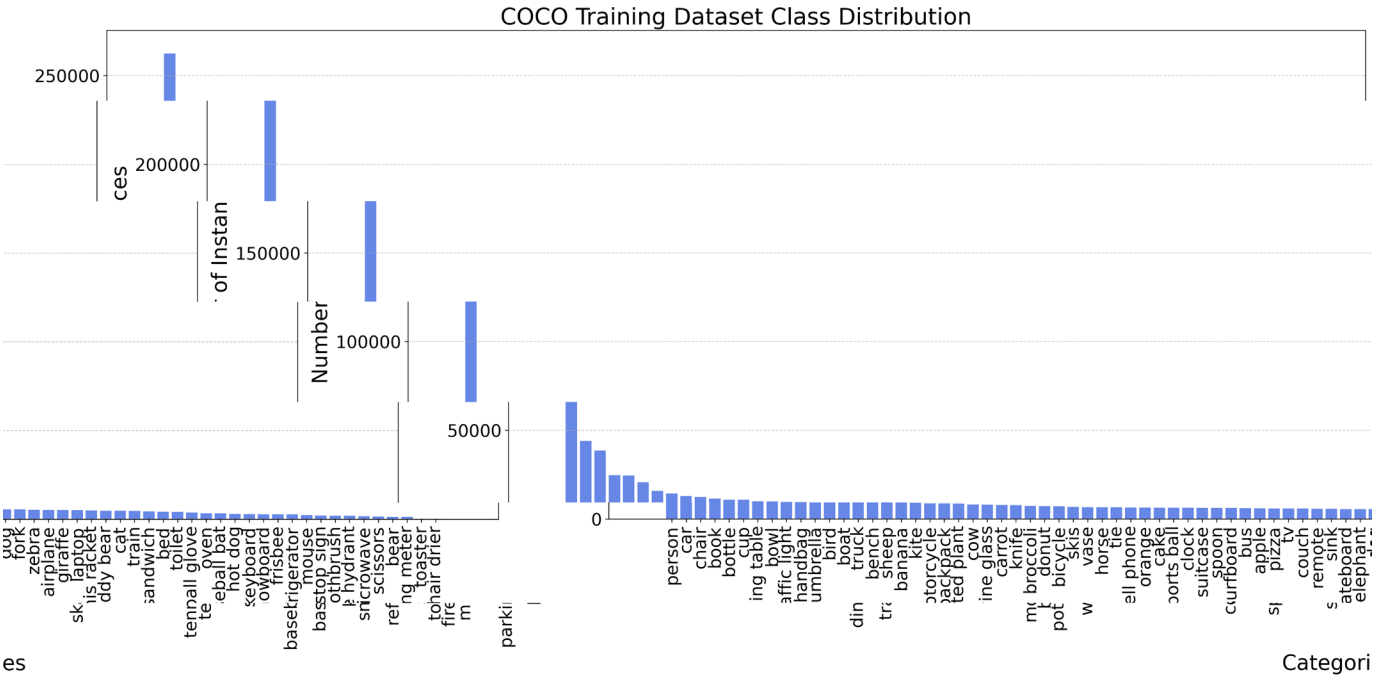


Figure 1. Distribution of classes in the COCO training dataset

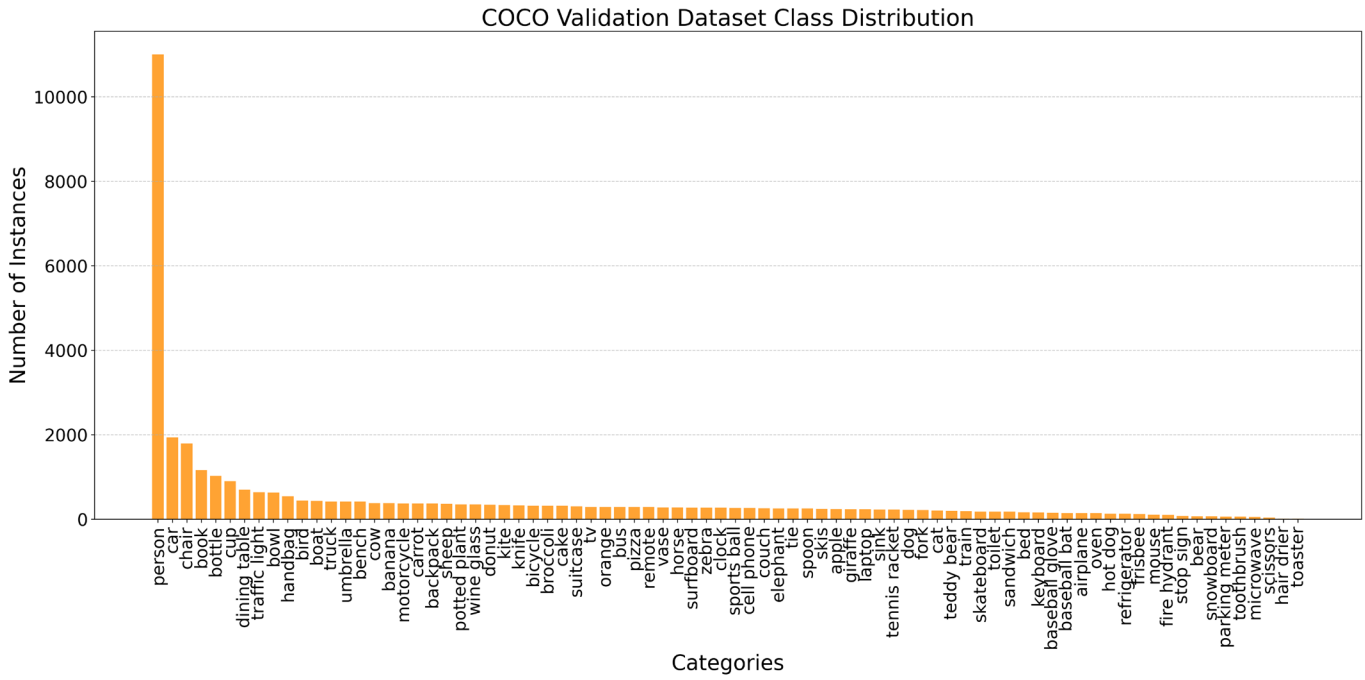


Figure 2. Distribution of classes in the COCO validation dataset

3.3.2 Diverse Redundancy Techniques

3.3.2.1 Diverse Input Image Transformations

In this subsection, we introduce the evaluation (YOLO implementation, dataset, evaluation metrics, and fault injection framework), we discuss the results obtained with each image transformation when executing a single model, and we discuss the results obtained with our scheme, considering the different image transformations and results merging schemes in fault-free and faulty scenarios.

The evaluation and the set-up of this this particular technique differs from the rests. Instead of the evaluation devised for the different domains, we have explored and tailored the technique to work in the automotive domain, and hence, the setup used for this technique differs to that of the others.

3.3.2.1.1 Setup

We use a 32-bit floating point Tensorflow Keras implementation of YOLOv4[2] [1] with the publicly available pre-trained YOLOv4 parameters with the training subset of the COCO dataset [2]. We evaluate our proposal using the default 608x608 (image width x image height) network size of YOLOv4. We evaluate our proposal with COCO, using the validation subset, and keeping only those images that contain objects such as vehicles and pedestrians, which leaves 870 images for evaluation.

However, the COCO dataset is not specific to AD, hence we have also evaluated our scheme with the KITTI dataset, designed specifically for AD applications. The KITTI dataset contains 7481 images captured from

[2] Different to the other techniques since this one allowed us to explore a number of fine-grain optimizations such as running redundant layers at once reusing weights to improve data locality without affecting the overall concept

onboard vehicle cameras, and also includes three temporally preceding frames of each image, but captured at a very low FPS, which makes detections across images highly independent. Hence, we resorted to evaluating each image independently. The main limitation when using the KITTI dataset with the pretrained YOLOv4 model is that some classes (e.g., bus) are not labelled, and some classes (e.g., person, and motorbike) have significantly different labelling policies to those of COCO. Therefore, we have restricted our evaluation with the KITTI dataset to those classes with fewer discrepancies to obtain accurate results (i.e., car, van, and truck object classes). Another difference w.r.t. the COCO dataset is that the KITTI dataset includes some regions of the image – mostly background regions – that have not been labelled. Therefore, predictions made within these unlabelled zones are ignored during the evaluation process.

As evaluation metrics, we use the Accuracy (ACC) and Mean Average Precision (mAP). To obtain those, first we have to compute the Intersection over Union (IoU), i.e., the fraction of the intersection of the bounding boxes w.r.t. the union of those bounding boxes. In particular, we do so for the final objects detected by our proposal and the ground truths. Since the detection process is stochastic and subject to some variation, whether an object is regarded as detected or not is done with a threshold, which in our case is 0.5, as set in other works [3]. This leads to TP if the IoU is above the threshold, FP if the IoU is below the threshold for a predicted object, and FN if the IoU of a groundtruth is below the threshold.

The mAP is a more complex metric that is often the reference for object detection evaluation. It builds on the AP metric presented before, and averages it across the different object classes. We refer the interested reader to other publications for detailed explanations of this metric [4].

Regarding our fault injection campaign, we focus on random hardware faults impacting the computation (i.e., transient faults). We have injected a single transient fault per image in the result generated by multiplication or addition operations of the convolutional layers, as these layers account for over 99.5% of the total number of operations to process an image. Furthermore, random faults have only been considered to affect the sign or exponent of the floating-point number representation (i.e., 9 highest order bits), since random bit flips in the mantissa are mostly masked and do not lead to semantic changes in the outputs of the model. This level of control in the fault injection is possible due to the use of YOLOv4, since newer versions of YOLO build on frameworks imposing additional limitations in the way faults can be injected.

In fact, Tensorflow Keras (used in YOLOv4) poses difficulties to inject faults in the intermediate results of a layer, since layers operate as black boxes. To overcome this limitation, we have implemented a custom convolutional layer (same functionality but less efficient implementation). We choose an operation (addition or multiplication) randomly across all those operations in all convolutional layers. To inject a fault in a specific result of a specific layer, we execute the Keras model until the previous layer, replace the target layer with our custom one, inject the fault as needed (flipping a random bit in the sign or exponent), and resume the execution of the Keras implementation from the following layer onwards passing the result of our custom layer as needed.

The image transformations and the corresponding fine-tuned parameters that we have used are as follows:

- Applying histogram equalization (EQ). No parameters needed.
- Image sharpening (SH). Parameters: kernel = ((0, -1, 0), (-1, 5, -1), (0, -1, 0)

- Dropout (DR) by setting some pixels to zero. Parameters: percentage = 0.05
- Applying Gamma correction (GC). Parameters: gamma = 1.5
- Blurring the image applying a Gaussian filter (GB). Parameters: kernel_size = 5
- Applying Gaussian noise (GN). Parameters: mean = 0, sigma = 10
- Blurring the image applying the median filter (MB). Parameters: kernel_size = 5
- Adding salt and pepper pixels (SP). Parameters: low = 0, up = 25
- Raising pixel values by a set amount (RV). Parameters: power = 0.9
- Shifting the image some pixels to the right (RS)[3]2. Parameters: pixel_shift = 5
- Shifting the image some pixels to the left (LS). Parameters: pixel_shift = 5
- Shifting the image some pixels to the top (TS). Parameters: pixel_shift = 5
- Shifting the image some pixels to the bottom (BS). Parameters: pixel_shift = 5
- Rotating the image clockwise at a certain angle (CR). Parameters: angle = 0.8
- Rotating the image anticlockwise at a certain angle (AR). Parameters: angle = -0.8
- Horizontally flipping the image (HF). No parameters needed.
- Vertically flipping the image (VF). No parameters needed.

3.3.2.1.2 Results of Independent Configurations

Table 2 shows the TP, FP, and FN counts, as well as the ACC and mAP for each image transformation with the COCO and KITTI datasets.

First, note that the VF image transformation produces very low ACC and mAP results for both datasets, since the pretrained model has not been trained with vertically flipped objects. However, we keep this transformation for completeness.

In the case of COCO there are 4 image transformations (TS, RV, HF, LS) providing up to 0.25% higher mAP than the baseline. For KITTI there are 3 image transformations (EQ, GC, BS) providing up to 0.73% higher mAP than the baseline.

Note that the pretrained YOLO model used the COCO training subset. Hence, the baseline model was expected to obtain high accuracy when evaluating similar images (e.g., COCO validation subset), but for different datasets, such as KITTI, it was indeed expected that some image transformations could surpass the accuracy of the baseline model by a greater margin.

Table 2. Results of each Image Transformation analysed independently with the COCO and KITTI datasets. Definitions of the configuration abbreviations can be found before in this section.

[3] Note that in the case of shifting transformations, certain pixel rows/columns will be set to 0 as they correspond to parts of the image that need to be filled in. For example, when applying an RS of 5 pixels, the leftmost 5 pixel columns will be set to 0.

	COCO					KITTI				
Config.	TP	FP	FN	ACC	mAP	TP	FP	FN	ACC	mAP
TS	3887	473	2304	58,33%	61,12%	25822	4165	6928	69,95%	76,50%
RV	3868	467	2323	58,10%	60,94%	25949	4142	6801	70,34%	76,85%
HF	3866	456	2325	58,16%	60,88%	25894	3998	6856	70,46%	76,75%
LS	3856	435	2335	58,19%	60,88%	25890	4132	6860	70,20%	76,73%
BL	3865	466	2326	58,06%	60,87%	25935	4093	6815	70,39%	76,86%
BS	3860	439	2331	58,22%	60,82%	25930	4047	6820	70,47%	76,86%
RS	3856	446	2335	58,10%	60,76%	25811	4135	6939	69,98%	76,46%
GC	3859	470	2332	57,93%	60,74%	26093	4401	6657	70,23%	77,14%
GN	3776	405	2415	57,25%	59,61%	25548	3880	7202	69,75%	75,64%
SP	3708	398	2483	56,28%	58,55%	25229	3763	7521	69,10%	74,65%
EQ	3699	450	2492	55,70%	58,24%	26193	4435	6557	70,44%	77,59%
CR	3676	466	2515	55,22%	57,83%	25533	3542	7217	70,35%	75,81%
AR	3675	504	2516	54,89%	57,72%	25607	3488	7143	70,66%	76,03%
SH	3240	324	2951	49,73%	51,15%	24765	3301	7985	68,69%	73,21%
GB	3213	267	2978	49,75%	51,05%	24796	2900	7954	69,55%	73,93%
MB	2894	295	3297	44,62%	45,71%	22545	2172	10205	64,56%	67,36%
DR	2335	298	3856	35,98%	36,40%	13191	1239	19559	38,81%	38,83%
VF	366	34	5825	5,88%	5,60%	568	86	32182	1,73%	1,56%

3.3.2.1.3 Results of the Merging Algorithms

We have obtained the TP, FP, and FN counts, as well as the ACC and mAP values for all TMR and DMR configurations with the relevant merging algorithms, namely Voting, Averaging, and Maximum in the case of TMR, and only Averaging and Maximum in the case of DMR. Since the number of combinations is too large to allow reporting data for all those configurations (817 per merging algorithm with TMR, and 154 per merging algorithm with DMR), we show the mAP for all the possible combinations of the image transformations that we have evaluated in Figure 1, Figure 2, Figure 3, and Figure 4, and the detailed results for the top-5 (in terms of mAP) for each algorithm in Table 3, Table 4, Table 5, and Table 6.

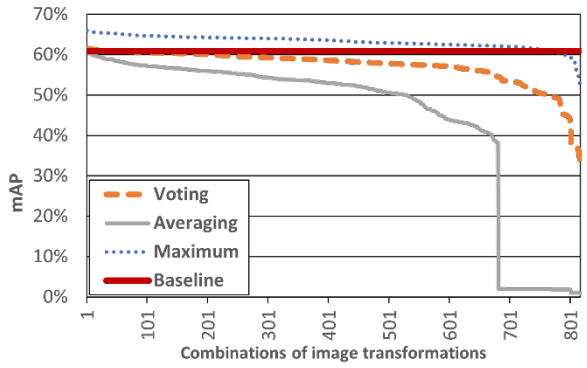


Figure 4. Sorted mAP for all TMR F(x), G(X), and H(x) permutations for all merging algorithms using COCO.

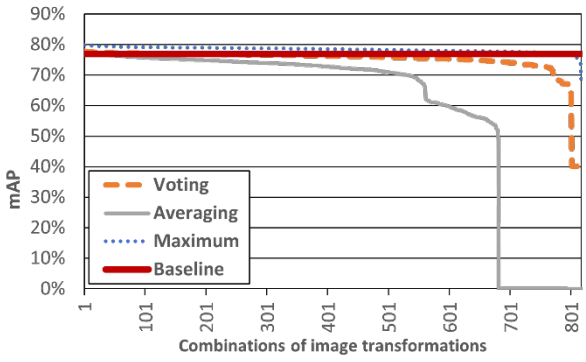


Figure 5. Sorted mAP for all TMR F(x), G(X), and H(x) permutations for all merging algorithms using KITTI.

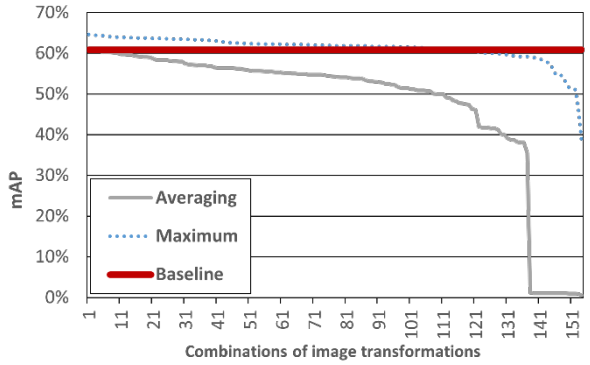


Figure 6. Sorted mAP for all DMR F(x) and G(X) for Averaging and Maximum using COCO.

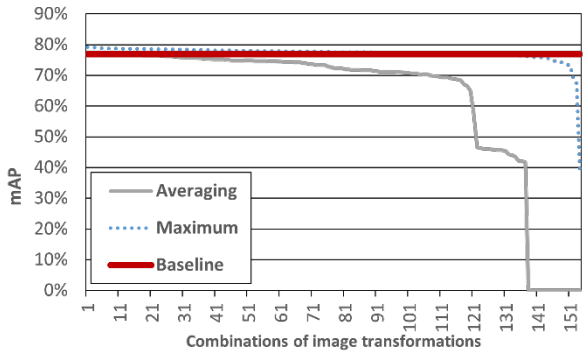


Figure 7. Sorted mAP for all DMR F(x) and G(X) permutations for Averaging and Maximum using KITTI.

Results in these figures are sorted independently for each merging algorithm, meaning that the nth best combination of image transformations for one algorithm may differ for the other algorithms despite being aligned w.r.t. the X-axis. For example, the combination labelled as 1 in Figure 1 corresponds to "HF, LS, TS" with Voting, "BL, BL, BL" with Averaging, and "HF, RS, BS" with Maximum. Hence, the X-axis is not labelled with the specific name of the image transformations used for that particular combination. However, showing the data this way allows us to get several conclusions for both TMR and DMR scenarios: (1) Maximum provides consistently the best results in terms of mAP across all merging algorithms; (2) Averaging is consistently worse, in terms of mAP, than the baseline when using the COCO dataset, but it is slightly better than the baseline in very few cases when evaluating the KITTI dataset; (3) Voting, for TMR scenarios, is slightly better than the baseline in some cases; and (4) all merging algorithms provide a smooth degradation of the mAP across image transformation combinations, hence meaning that, while some transformations may delve better results than others, there is not a strong dependence on very few combinations, so the approach provides robust results.

We have analysed the commonalities between the worst-performing set of transformations under each merging algorithm (i.e., Average, Voting, and Maximum) to determine if any patterns emerge or if each merging algorithm is unique in these terms. Upon reviewing the results, it is evident that the accuracy of each merging algorithm is influenced by the presence of poorly performing transformations. However, the impact varies based on the merging algorithm used:

(1) Average: This merging algorithm is particularly sensitive to poorly performing transformations. If one of the transformations yields poor results, it significantly degrades the overall accuracy. The averaging process diminishes the effect of the better-performing transformations by averaging them with the poor one. This implies that in the Average merging algorithm, consistency in accuracy across all selected transformations is crucial, and the presence of even a single weak transformation can lead to poor results.

(2) Voting: This merging algorithm shows a moderate level of robustness against poorly performing transformations. It generally requires at least two out of three transformations to perform well to achieve higher accuracy than the baseline. This implies that the occasional low accuracy of a single transformation does not critically impact the outcome, as long as the other two transformations have high accuracy.

(3) Maximum: This merging algorithm is the most robust against poorly performing transformations. In this algorithm, the presence of at least one good-performing transformation can yield results better

than the baseline. Even if two transformations perform poorly, the good performance of one transformation can still dominate and improve the overall outcome w.r.t. the baseline.

Our results indicate that the best combinations are achieved using transformations from the TOP-10 best accuracy independent configurations (Table 2). Therefore, we could safely ignore combinations involving transformations with lower independent accuracy, as they are less likely to contribute positively to the detection accuracy across the different merging algorithms. However, we included all the combinations for completeness. In summary, while each merging algorithm exhibits different levels of sensitivity to poorly performing transformations, the key to achieving high accuracy lies in selecting transformations that perform well independently.

Table 3. TMR results for the top-5 configurations of each merging algorithm using COCO.

BASELINE					
Config.	TP	FP	FN	ACC	mAP
BL	3865	466	2326	58,06%	60,87%
VOTING					
Config.	TP	FP	FN	ACC	mAP
HF, LS, TS	3901	403	2290	59,16%	61,60%
RV, LS, TS	3900	416	2291	59,03%	61,54%
GC, LS, TS	3897	410	2294	59,04%	61,49%
HF, RS, TS	3895	422	2296	58,90%	61,48%
HF, RV, BS	3893	417	2298	58,91%	61,43%
AVERAGING					
Config.	TP	FP	FN	ACC	mAP
BL, BL, BL	3865	466	2326	58,06%	60,87%
BL, GC, RV	3862	460	2329	58,07%	60,84%
BL, HF, GC	3829	423	2362	57,89%	60,42%
BL, GN, RV	3827	425	2364	57,84%	60,37%
HF, GC, RV	3825	421	2366	57,85%	60,35%
MAXIMUM					
Config.	TP	FP	FN	ACC	mAP
HF, RS, BS	4214	691	1977	61,23%	66,01%
HF, LS, TS	4207	713	1984	60,94%	65,87%
HF, LS, BS	4201	697	1990	60,99%	65,84%
HF, RS, TS	4204	715	1987	60,87%	65,82%
RS, LS, TS	4195	710	1996	60,79%	65,72%

Table 4. TMR results for the top-5 configurations of each merging algorithm using KITTI.

BASELINE					
Config.	TP	FP	FN	ACC	mAP
BL	25935	4093	6815	70,39%	76,86%
VOTING					
Config.	TP	FP	FN	ACC	mAP
HF, EQ, GC	26199	4101	6551	71,09%	77,76%
EQ, GC, LS	26196	4105	6554	71,08%	77,74%
EQ, GC, RS	26184	4083	6566	71,09%	77,70%
EQ, GC, BS	26170	4122	6580	70,98%	77,66%
EQ, LS, BS	26118	3862	6632	71,34%	77,61%
AVERAGING					
Config.	TP	FP	FN	ACC	mAP
HF, EQ, GC	26057	3879	6693	71,14%	77,32%
BL, HF, EQ	26011	3748	6739	71,27%	77,24%
EQ, GC, RV	26064	4152	6686	70,63%	77,21%
HF, EQ, RV	26007	3781	6743	71,19%	77,20%
BL, EQ, GC	26052	4124	6698	70,65%	77,20%
MAXIMUM					
Config.	TP	FP	FN	ACC	mAP
HF, EQ, LS	26974	5877	5776	69,83%	79,80%
HF, EQ, BS	26963	5717	5787	70,09%	79,79%
HF, EQ, TS	26949	5772	5801	69,96%	79,76%
EQ, LS, BS	26947	5863	5803	69,79%	79,73%
EQ, LS, TS	26947	5901	5803	69,72%	79,72%

Table 5. DMR results for the top-5 configurations of each merging algorithm using COCO.

BASELINE					
Config.	TP	FP	FN	ACC	mAP
BL	3865	466	2326	58,06%	60,87%
AVERAGING					
Config.	TP	FP	FN	ACC	mAP
BL, BL	3865	466	2326	58,06%	60,87%
BL, RV	3862	466	2329	58,01%	60,84%
BL, GC	3861	453	2330	58,11%	60,82%
GC, RV	3861	461	2330	58,04%	60,80%
HF, GC	3824	420	2367	57,84%	60,34%
MAXIMUM					
Config.	TP	FP	FN	ACC	mAP
HF, TS	4120	617	2071	60,52%	64,59%
HF, BS	4110	593	2081	60,58%	64,51%
LS, TS	4105	619	2086	60,28%	64,41%
RS, BS	4103	594	2088	60,47%	64,38%
RS, TS	4102	628	2089	60,16%	64,34%

Table 6. DMR results for the top-5 configurations of each merging algorithm using KITTI.

BASELINE					
Config.	TP	FP	FN	ACC	mAP
BL	25935	4093	6815	70,39%	76,86%
AVERAGING					
Config.	TP	FP	FN	ACC	mAP
EQ, GC	26135	4258	6615	70,62%	77,41%
EQ, RV	26051	4094	6699	70,71%	77,24%
BL, EQ	26038	4047	6712	70,76%	77,22%
HF, EQ	26000	3819	6750	71,10%	77,19%
GC, RV	26030	4246	6720	70,36%	77,03%
MAXIMUM					
Config.	TP	FP	FN	ACC	mAP
HF, EQ	26750	5250	6000	70,39%	79,21%
EQ, LS	26759	5367	5991	70,20%	79,19%
EQ, RS	26718	5365	6032	70,10%	79,08%
EQ, BS	26678	5179	6072	70,34%	78,99%
EQ, TS	26644	5215	6106	70,18%	78,87%

Results in the tables offer a different angle to our analysis. For the COCO TMR configurations, in the case of Voting, we note that slightly shifting the original image in one direction is a frequent choice since 9 out of the 15 choices correspond to top, bottom, left or right shift, HF appears 3 times, RV appears twice, and GC appears once. In the case of Averaging, the best combination, despite not providing diversity, consists of using 3 times the BL. RV and GC appear three times, HF appears twice, and GN appears once. Finally, in the case of Maximum merging, we observe that slightly shifting the original image in one direction is a frequent choice since 11 out of the 15 choices correspond to top, bottom, left or right shift. HF is the second most frequent choice with 4 appearances. The best image transformations in the TMR configurations also work particularly well in DMR scenarios. In particular, in the case of Averaging, the best combination is also using two baselines, but GC, RV and HF also appear in the top-5 mAP configurations. Similarly, in the case of Maximum, we observe that slightly shifting the original image in one direction is also the most frequent choice, and HF is the second most frequent choice. Therefore, the image transformations that appear in the top-5 configurations work particularly well regardless of the number of redundant components (i.e., DMR or TMR).

For the KITTI TMR configurations, in the case of Voting, EQ and shifting appear 5 times each, GC appears 4 times, and HF appears once. In the case of Averaging, EQ appears 5 times, HF and GC appear 3 times, RV and BL appear twice. In the case of Averaging, EQ appears 5 times, and HF and GC appear 3 times. Finally, in the case of Maximum merging, shifting appears 7 times, EQ appears 5 times, and HF appears 3 times. Similarly to the COCO dataset, the best image transformations in the TMR configurations also work particularly well in DMR scenarios. In particular, in the case of Averaging, EQ appears 4 times, RV and GC twice, and BL and HF appear once. In the case of Maximum merging, EQ appears in all top-5 configurations, shifting transformations appear 4 times, and HF appears once.

In summary, for the KITTI dataset, EQ is a particularly good option since it appears in all the TOP-5 configurations for all merging algorithms, and for both TMR and DMR scenarios, except for one configuration in the DMR Averaging case. The shifting image transformations works particularly well for both COCO and KITTI datasets when using a Voting with TMR or Maximum merging algorithm with both TMR and DMR. The BL works particularly well for the COCO dataset when performing an Average merging, and the HF and GC are also noteworthy configurations for the KITTI dataset with an Average merging.

For the COCO dataset we note that Voting may increase a bit TPs and decrease a bit FPs. Hence, despite not providing the best mAP values, it provides an interesting tradeoff since it outperforms the baseline in all fronts. Averaging, instead, tends to decrease TPs while failing to decrease FPs sufficiently, and it is systematically worse than the baseline case in both TMR and DMR scenarios. Finally, Maximum tends to increase TPs and FPs, which is expected since any object being detected by at least one of the three redundant inferences is regarded as a detection, and hence, a TP or a FP. Still, the combined effect clearly increases ACC and mAP values w.r.t. the baseline in both TMR and DMR scenarios. Overall, looking at the results from the COCO dataset, Maximum is clearly the best choice in DMR scenarios since Averaging produces worse results than the baseline. In TMR scenarios, if we care only about mAP or ACC, Maximum is clearly the best choice. If, instead, FPs are particularly problematic, Voting is the best solution.

For the KITTI dataset, looking at the mAP values, Maximum is clearly the best choice in DMR scenarios, providing up to ≈2% more mAP than Averaging, but in this case, all TOP-5 mAP Averaging configurations have higher mAP than the baseline. However, looking at the ACC, some Averaging configurations have slightly more ACC than Maximum merging. For instance, the top-mAP configuration with Averaging (i.e., EQ, GC) provides a 70,62% ACC while the top-mAP configuration with Maximum (i.e., HF, EQ) provides a lower ACC of 70,39%. Note that the ACC of some configurations is slightly lower than the baseline, specially for the KITTI dataset, where only one configuration (i.e., HF, EQ) has an ACC greater than the baseline. We ascribe this effect to the fact that the KITTI dataset does not properly label all background objects, which are mostly small, causing more FPs than it should if all objects were properly labelled, which causes a higher decrease in the ACC whereas the mAP seems not to be as impacted by this effect. We validated this observation in a subset of images by means of visual inspection, yet could not properly label the full dataset manually to fully fix this issue.

In the case of TMR, Voting is clearly superior than Averaging for all configurations. However, looking at the ACC values, these approaches have closer results. Finally, Maximum tends to increase TPs and FPs, but the combined effect provides the highest mAP values. However, the ACC of the Maximum merging algorithm is slightly lower than the baseline, due to the same limitations of the KITTI dataset, as previously discussed.

Overall, if we care only about mAP, Maximum merging is clearly the best choice for all the datasets analyzed. If, instead, FPs are particularly problematic, Voting is the best solution in the case of TMR.

It is worth mentioning that profiling every possible combination of transformations offline and ranking them, as shown in Figure 1, Figure 2, Figure 3, and Figure 4, is not mandatory each time a new dataset/model pair is used. We have evaluated the TOP-5 configurations from one dataset on the other dataset and found out that the differences were not very significant (i.e., 0.28-0.54% lower mAP when using the TOP-5 configurations of the COCO dataset on the KITTI dataset, and 0.49-1.16% mAP when using the TOP-5 configurations of the KITTI dataset on the COCO dataset). Our results suggest that a reasonable approach would be to use the TOP-5 configurations obtained from the COCO dataset when working with other datasets/models.

3.3.2.1.4 Results of Fault Injection Configurations

We have selected the top-mAP and top-ACC configurations for each merging algorithm, network size, and dataset, and analyzed the impact of random fault injections on this subset of configurations. In the case of a single configuration providing both the top-mAP and top-ACC, this configuration with both the top-mAP and top-ACC, and the second highest mAP configuration have been selected instead.

First, we analyze the individual image transformations used in any of the top-mAP and top-ACC combinations indicated above. Table 7 and Table 8, present the fault-free results as well as the results after fault injection for those image transformations for the COCO and KITTI datasets, respectively. Note that, as previously discussed, these configurations correspond to the execution of their respective image transformation with a single transient fault injected per image. The tables provide the FP, FN, and TP count, as well as the ACC and mAP. Since some combinations use an ensemble of up to three baselines, we perform 3 different fault injections in such baseline, which we refer to as BL_1 AF, BL_2 AF, and BL_3 AF. For the remaining cases we report the fault-free (e.g., LS) and fault-injected cases (e.g., LS AF).

Table 7. Results of the Fault Injection of the individual image transformations part of the combinations with the best mAP and ACC for the COCO dataset.

BASELINE					
Config.	TP	FP	FN	ACC	mAP
BL	3865	466	2326	58,06%	60,87%
INDEPENDENT CONFIGURATIONS					
Config.	TP	FP	FN	ACC	mAP
BL_1 AF	3667	487	2524	54,91%	56,89%
BL_2 AF	3661	463	2530	55,02%	57,01%
BL_3 AF	3633	468	2558	54,56%	56,47%
HF	3866	456	2325	58,16%	60,88%
HF AF	3708	450	2483	55,83%	57,79%
TS	3887	473	2304	58,33%	61,12%
TS AF	3703	471	2488	55,58%	57,74%
BS	3860	439	2331	58,22%	60,82%
BS AF	3681	408	2510	55,78%	57,80%
RS	3856	446	2335	58,10%	60,76%
RS AF	3653	439	2538	55,10%	56,90%
LS	3856	435	2335	58,19%	60,88%
LS AF	3667	418	2524	55,48%	57,41%
RV	3868	467	2323	58,10%	60,94%
RV AF	3664	440	2527	55,26%	57,41%
GC	3859	470	2332	57,93%	60,74%
GC AF	3710	461	2481	55,77%	58,11%

Table 8. Results of the Fault Injection of the individual image transformations part of the combinations with the best mAP and ACC for the KITTI dataset.

BASELINE					
Config.	TP	FP	FN	ACC	mAP
BL	25935	4093	6815	70,39%	76,86%
INDEPENDENT CONFIGURATIONS					
Config.	TP	FP	FN	ACC	mAP
HF	25894	3998	6856	70,46%	76,75%
HF AF	24774	3832	7976	67,72%	73,30%
EQ	26193	4435	6557	70,44%	77,59%
EQ AF	24914	4218	7836	67,39%	73,66%
GB	24796	2900	7954	69,55%	73,93%
GB AF	23548	2767	9202	66,30%	69,96%
AR	25607	3488	7143	70,66%	76,03%
AR AF	24431	3348	8319	67,68%	72,37%
GC	26093	4401	6657	70,23%	77,14%
GC AF	24925	4221	7825	67,42%	73,50%
LS	25890	4132	6860	70,20%	76,73%
LS AF	24671	3998	8079	67,14%	72,90%
CR	25533	3542	7217	70,35%	75,81%
CR AF	24278	3399	8472	67,16%	71,91%

We observe that, in general, fault-injected configurations have lower TP counts, and also lower FP counts. ACC and mAP drop similarly for all configurations, with a 2.88% and 3.62% drop on average, respectively.

Table 9, Table 10, Table 11, and Table 12 show the results of our proposal in the non-faulty TMR/DMR case, as well as in two faulty TMR/DMR cases: (i) with independent faults where, for a given image, only one of the three configurations is affected by a fault, and (ii) with faults in the same image for all three configurations. The latter case corresponds to faults that affect the exact same operation across all components of the system. However, while the faults affect the same operation, the specific bit that is flipped is chosen randomly for each component. The reason is that each component processes a different image transformation, and hence, values operated differ, so neither the impact of a fault can be fully cloned, nor it would have the same effect in practice since, at physical level the current flows differ, and therefore, the effects of electrical disturbances will also differ. In the fault-free scenarios, which would correspond to virtually 100% of the time given that faults occur seldom, we obtain the following conclusion: (1) Voting (in the TMR scenario) and Maximum (in both TMR and DMR) merging algorithms provide higher mAP and ACC than the Baseline, except for the HF, EQ, LS configuration for the KITTI dataset in the TMR scenario, where we observe a slight ACC drop w.r.t. the baseline (for reasons discussed before), (2) Averaging, in both TMR and DMR, only produces better results than the baseline with the KITTI dataset.

Table 9. TMR results for the best mAP and ACC configurations of each merging algorithm using the COCO dataset.

BASELINE					
Configuration	**TP**	**FP**	**FN**	**ACC**	**mAP**
BL	3865	466	2326	58,06%	60,87%
VOTING					
Configuration	**TP**	**FP**	**FN**	**ACC**	**mAP**
HF, LS, TS	3901	403	2290	59,16%	61,60%
HF, LS, TS (indep.faults)	3896	401	2295	59,10%	61,51%
HF, LS, TS (same frame)	3822	398	2369	58,01%	60,36%
RV, LS, TS	3900	416	2291	59,03%	61,54%
RV, LS, TS (indep.faults)	3880	418	2311	58,71%	61,23%
RV, LS, TS (same frame)	3831	410	2360	58,04%	60,48%
AVERAGING					
Configuration	**TP**	**FP**	**FN**	**ACC**	**mAP**
BL_1, BL_2, BL_3	3865	466	2326	58,06%	60,87%
BL_1, BL_2, BL_3 (indep.faults)	3697	490	2494	55,34%	58,17%
BL_1, BL_2, BL_3 (same frame)	3392	542	2799	50,38%	53,15%
BL_1, GC, RV	3862	460	2329	58,07%	60,84%
BL_1, GC, RV (indep.faults)	3751	472	2440	56,30%	59,05%
BL_1, GC, RV (same frame)	3449	510	2742	51,47%	54,05%
MAXIMUM					
Configuration	**TP**	**FP**	**FN**	**ACC**	**mAP**
HF, RS, BS	4214	691	1977	61,23%	66,01%
HF, RS, BS (indep.faults)	4204	743	1987	60,63%	64,32%
HF, RS, BS (same frame)	4186	980	2005	58,37%	58,75%
HF, LS, TS	4207	713	1984	60,94%	65,87%
HF, LS, TS (indep.faults)	4201	823	1990	59,89%	62,69%
HF, LS, TS (same frame)	4188	1071	2003	57,67%	57,59%

When faults occur independently, we obtain the following conclusions: (1) Voting, in the TMR scenario, provides higher mAP and ACC than the Baseline with both datasets. (2) Maximum, in both TMR and DMR, provides higher mAP and ACC than the Baseline for the COCO dataset. However, for the KITTI dataset, we observe that the HF, EQ, LS configuration provides higher mAP but lower ACC (for reasons

discussed before). (3) Averaging, corresponds to the top-ACC configuration and does not necessarily have to produce also higher mAP due to the differences between these metrics.

Table 10. TMR results for the best mAP and ACC configurations of each merging algorithm using the KITTI dataset.

BASELINE					
Configuration	TP	FP	FN	ACC	mAP
BL	25935	4093	6815	70,39%	76,86%
VOTING					
Configuration	TP	FP	FN	ACC	mAP
HF, EQ, GC	26199	4101	6551	71,09%	77,76%
HF, EQ, GC (indep.faults)	26168	4057	6582	71,10%	77,68%
HF, EQ, GC (same frame)	25910	3957	6840	70,59%	76,92%
EQ, GB, LS	25862	3426	6888	71,49%	77,01%
EQ, GB, LS (indep.faults)	25811	3387	6939	71,43%	76,87%
EQ, GB, LS (same frame)	25502	3293	7248	70,75%	75,95%
AVERAGING					
Configuration	TP	FP	FN	ACC	mAP
HF, EQ, GC	26057	3879	6693	71,14%	77,32%
HF, EQ, GC (indep.faults)	25135	3846	7615	68,68%	74,54%
HF, EQ, GC (same frame)	23178	3776	9572	63,46%	68,70%
HF, EQ, AR	25853	3468	6897	71,38%	76,85%
HF, EQ, AR (indep.faults)	24946	3449	7804	68,91%	74,13%
HF, EQ, AR (same frame)	22901	3425	9849	63,31%	67,99%
MAXIMUM					
Configuration	TP	FP	FN	ACC	mAP
HF, EQ, LS	26974	5877	5776	69,83%	79,80%
HF, EQ, LS (indep.faults)	26956	5988	5794	69,59%	78,79%
HF, EQ, LS (same frame)	26921	6183	5829	69,15%	77,33%
GB, CR, AR	26182	4318	6568	70,63%	77,79%
GB, CR, AR (indep.faults)	26171	4449	6579	70,35%	76,77%
GB, CR, AR (same frame)	26142	4694	6608	69,82%	75,31%

Table 11. DMR results for the best mAP and ACC configurations of each merging algorithm using the COCO dataset.

BASELINE					
Configuration	TP	FP	FN	ACC	mAP
BL	3865	466	2326	58,06%	60,87%
AVERAGING					
Configuration	TP	FP	FN	ACC	mAP
BL_1, BL_2	3865	466	2326	58,06%	60,87%
BL_1, BL_2 (indep. faults)	3679	456	2512	55,35%	57,93%
BL_1, BL_2 (same frame)	3543	448	2648	53,37%	55,69%
BL_1, RV	3862	466	2329	58,01%	60,84%
BL_1, RV (indep. faults)	3689	443	2502	55,61%	58,10%
BL_1, RV (same frame)	3543	438	2648	53,45%	55,72%
MAXIMUM					
Configuration	TP	FP	FN	ACC	mAP
HF, TS	4120	617	2071	60,52%	64,59%
HF, TS (indep. faults)	4105	613	2086	60,33%	64,09%
HF, TS (same frame)	4086	635	2105	59,86%	63,02%
HF, BS	4110	593	2081	60,58%	64,51%
HF, BS (indep. faults)	4102	601	2089	60,39%	64,05%
HF, BS (same frame)	4080	603	2111	60,05%	63,22%

Table 12. DMR results for the best mAP and ACC configurations of each merging algorithm using the KITTI dataset.

BASELINE					
Configuration	**TP**	**FP**	**FN**	**ACC**	**mAP**
BL	25935	4093	6815	70,39%	76,86%
AVERAGING					
Configuration	**TP**	**FP**	**FN**	**ACC**	**mAP**
EQ, GC	26135	4258	6615	70,62%	77,41%
EQ, GC (indep. faults)	25117	4070	7633	68,22%	74,39%
EQ, GC (same frame)	24101	3927	8649	65,71%	71,37%
HF, AR	25691	3370	7059	71,13%	76,37%
HF, AR (indep. faults)	24721	3252	8029	68,67%	73,47%
HF, AR (same frame)	23695	3111	9055	66,07%	70,41%
MAXIMUM					
Configuration	**TP**	**FP**	**FN**	**ACC**	**mAP**
HF, EQ	26750	5250	6000	70,39%	79,21%
HF, EQ (indep. faults)	26706	5217	6044	70,34%	78,92%
HF, EQ (same frame)	26610	5159	6140	70,19%	78,55%
GB, AR	25941	3941	6809	70,70%	77,15%
GB, AR (indep. faults)	25913	3932	6837	70,64%	76,79%
GB, AR (same frame)	25826	3915	6924	70,44%	76,41%

When faults occur in the same image for all the redundant components, we obtain the following conclusions: (1) for the COCO dataset, in the TMR case, all merging algorithms provide lower mAP than the baseline, (2) Voting produces the closest accuracy to the baseline, both in terms of ACC and mAP, (3) However, in the DMR case, Maximum merging still provides higher accuracy than the baseline even when faults are present in both images. (4) for the KITTI dataset, both Voting and Maximum have at least one configuration producing higher accuracy (either both ACC and mAP or only one of the metrics) than the baseline. For example, the HF, EQ, GC (same frame) configuration with Voting has higher ACC and mAP, the EQ, GB, LS (same frame) configuration with Voting has lower mAP but higher ACC, and the HF, EQ, LS (same frame) configuration with Maximum has higher mAP but lower ACC.

We can conclude the following: (1) Averaging is consistently worse than Voting , in the TMR scenario, and worse than Maximum in all cases. (2) Maximum produces the highest ACC and mAP results in the fault-free scenarios, as well as when faults occur independently, but for the KITTI dataset, in a TMR scenario, the ACC of Voting is slightly higher than that of Maximum with all configurations. (3) In a TMR scenario, Voting produces the highest mAP results with the COCO dataset when faults occur simultaneously. For instance, the highest mAP obtained with Voting is 60,48% with the RV, LS, TS (same frame) configuration, whereas the highest mAP obtained with Maximum is 58,75% with the HF, RS, BS (same frame) configuration. (4) When faults occur simultaneously for the KITTI dataset, Maximum provides the highest mAP but in a TMR scenario, Voting provides the highest ACC in all cases.

Overall, we can conclude that the Maximum merging algorithm is clearly the best choice in a DMR scenario. However, when considering a TMR scenario, Maximum merging produces the highest mAP results, but it provides significantly higher FPs than Voting. Therefore, if FPs are a major concern, Voting is the best solution, while if the objective is to obtain the maximum mAP, the Maximum algorithm is the best solution since fault-free operations occurs virtually 100% of the time, and typically faults will not affect all redundant components at the same time.

For completeness, we have also conducted an additional experiment to analyze the mAP trend when multiple faults are injected within the execution of an image. However, note that this case is very unlikely to occur since the probability of having a transient fault is extremely low, and hence, accumulating several of them within a short timeframe occurs with even much lower probabilities. Figure 5 and Figure 6 show this analysis when faults are injected independently, and when faults are injected simultaneously, respectively. We only show the results of one configuration (i.e., HF,LS,TS) since this configuration is among the best ones for both Voting and Maximum merging algorithms. We do not analyze the impact of multiple fault injections with the Average merging algorithm since this algorithm produces poor results even in non-faulty scenarios. Note that these figures only show one configuration for the COCO dataset. However, the mAP trends are analogous with other configurations/datasets.

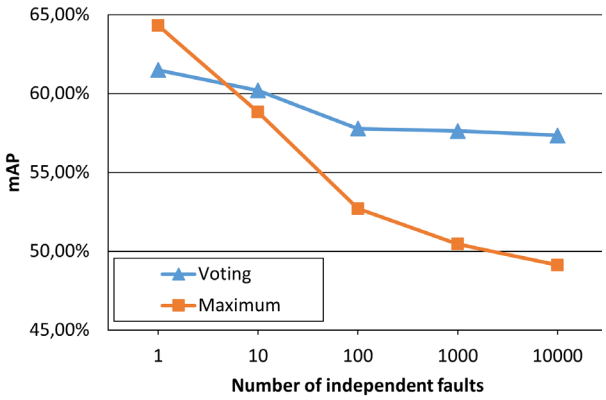


Figure 8. Analysis of multiple fault injections (Independent faults, "HF,LS,RS" configuration).

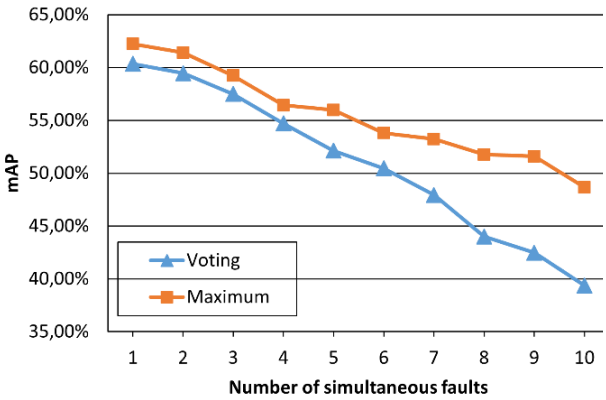


Figure 9. Analysis of multiple fault injections (Simultaneous faults, "HF,LS,RS" configuration).

When independent faults are injected, we observe that the Voting merging algorithm is highly resistant since even when 10000 faults are introduced, the mAP only decreases by 4.16%. In contrast, the Maximum algorithm performs significantly worse under these conditions. For instance, Maximum with 100 faults already performs worse than Voting with 10,000 faults. This difference is mainly due to the fact that introducing many faults decreases TPs but increases FPs. These added FPs can be discarded using the Voting merging algorithm, since only one configuration is affected by the independent faults. However, the Maximum algorithm cannot effectively discard or mitigate these numerous FPs.

When simultaneous faults are injected, the mAP decreases rapidly. The merging algorithms perform worse with 10 simultaneous faults than with 10000 independent faults. We observe that the Maximum merging algorithm always performs better than Voting, and the difference in mAP becomes larger as the number of faults increases. This difference is mainly due to the fact that now faults are being injected into all components at once, and hence, Voting has difficulties to filter out these faults since all components are faulty. On the other hand, Maximum does not filter out these faults, but still the number of added TPs supersede the added FPs w.r.t. Voting.

3.3.2.2 Lockstep Execution

As explained in D2.2 [Section 3.1.2.1, DRS_1], lockstep redundancy (low DC) involves replicating the same DL model and running identical operations in parallel on the same or different hardware platforms (GPU or CPU). This approach introduces a fixed delay in clock cycles to avoid common-cause failures.

In the Lockstep execution, the models (head and tail),are delayed by 5 cycles. The fault injection is applied constantly in the head or both head and tail. The fault injection has always the same value.

The following experiments were performed:

- EX1: Model on GPU - baseline model
- EX2: Model on GPU with fault injection
- EX3: Two models in GPU combined
- EX4: Two models in GPU combined, one with fault injection
- EX5: Two models in GPU combined, both with fault injection
- EX6: A model in GPU and a model in CPU combined

AP Table:

Metrics	EX1	EX2	EX3	EX4	EX5	EX6
AP1	0.4740	0.1801	0.4707	0.4661	0.1789	0.4707
AP2	0.6052	0.2471	0.6026	0.5982	0.2465	0.6026
AP3	0.5176	0.1954	0.5133	0.5073	0.1931	0.5132
AP4	0.2719	0.0766	0.2709	0.2696	0.0761	0.2709
AP5	0.5280	0.2101	0.5234	0.5216	0.2083	0.5235
AP6	0.6681	0.2573	0.6641	0.6590	0.2556	0.6641

AR Table:

Metrics	EX1	EX2	EX3	EX4	EX5	EX6
AR1	0.3595	0.1576	0.3595	0.3592	0.1576	0.3595
AR2	0.5267	0.2073	0.5217	0.5242	0.2051	0.5218
AR3	0.5331	0.2082	0.5279	0.5307	0.2058	0.5279
AR4	0.3018	0.0836	0.3002	0.3017	0.0830	0.3002
AR5	0.5862	0.2363	0.5795	0.5827	0.2337	0.5796
AR6	0.7389	0.2983	0.7327	0.7366	0.2953	0.7327

Results show consistent performance, with EX3 (dual GPU, no faults) and EX6 (GPU-CPU, no faults) achieving nearly identical Average Precision (AP1@0.50:0.95 = 0.4707 for both, difference ≤0.0001) and Average Recall (e.g., AR3@maxDets=100 = 0.5279 for both), confirming deterministic behavior and GPU-CPU equivalence. Fault injection in EX2 reduced performance significantly, dropping AP1 from 0.4740 (EX1, baseline) to 0.1801 (~62%) and AR3 from 0.5331 to 0.2082 (~60.9%), due to lower detection confidence. Redundancy in EX4, using Non-Maximum Suppression (NMS) to combine outputs, mitigated single-fault effects, reaching AP1 = 0.4661 (~98.3% of EX1) and AR3 = 0.5307 (~99.5% of EX1). However, EX5 (both models faulted) showed persistent degradation (AP1 = 0.1789, ~012% worse than EX2; AR3 = 0.2058), indicating limited recovery for correlated faults. These findings validate DRS_1's low diagnostic coverage (~62% performance loss for single faults, largely recovered by redundancy) and support reliable GPU-CPU integration for safety properties SP1/SP2.

3.3.3 Diverse Framework

This section evaluates the main modifications made to adapt a C-based implementation of YOLO in order to comply with the MISRA C coding guidelines. The primary objective is to enable the simultaneous execution of two object detection frameworks, ensuring that the MISRA C-compliant YOLO is executed at least once every N executions of the other object detector. The adaptation of YOLO aims to mitigate systematic errors by applying MISRA C recommendations, which is highly encouraged for applications that must meet high Safety Integrity Levels (SILs).

3.3.3.1 MISRA C-Compliant YOLO for Object Detection

In this subsection we have focused on adapting an AI-based system. Our work has targeted one of its most relevant functionalities, object detection. To this end, we target YOLOv4, which to the best of our knowledge, is the most recent Darknet-based version written in C and CUDA. Our efforts are primarily directed at mitigating potential programming errors in its C implementation to improve reliability and ensure alignment with functional safety standards. The contributions of this subsection are the following:

1. Analysis of MISRA C compliance within a C-based YOLO object detector using Polyspace Bug Finder Static Analysis Tool (SAT).
2. Adaptations of YOLO to comply with MISRA C recommended programming practices in order to comply with the requirements of safety standards.
3. Integration analysis of our adapted YOLO implementation within a safety architectural pattern, exploring its role as a diagnostic mechanism, and validating the behavioral consistency of the compliant framework by comparing its results against those of the original implementation using the COCO dataset.

For that, we have started analyzing the MISRA C compliance of the original C-based YOLOv4, and subsequently we have presented the modification made to the original YOLO v4.

3.3.3.1.1 Systematic Error Avoidance In AI-Based Object Detection

Polyspace generates a report listing all MISRA C violations after analyzing the 89 source files used by YOLO, which consist of 41,477 lines of code (33,231 after extracting comments). From this report, we extracted the number of violations by file and by violation type and depicted them in Figure 8. This figure shows advisory[4] violations in grey, mandatory[5] violations in pink, and required[6] violations in green. In the top-right corner, a pie chart shows the distribution of violations by type, revealing that most violations are required (65%), almost none are mandatory (5%), and the remaining are advisory (30%). The x-axis labels in blue correspond to files with required layer functionalities. The upper graph displays the 41 files with

[4] Guidelines that should be followed when reasonably practicable, but they do not require a formal deviation process.

[5] Guidelines for which deviation is not allowed.

[6] Guidelines that must generally be followed, but deviation is allowed if it is properly justified, documented, and authorized and a formal deviation process is followed.

the highest number of violations (16,820), accounting for 93.56% of the total violations (17,976), while the lower graph shows the remaining 48 files (note that the y-axis scale differs between the two bar graphs).

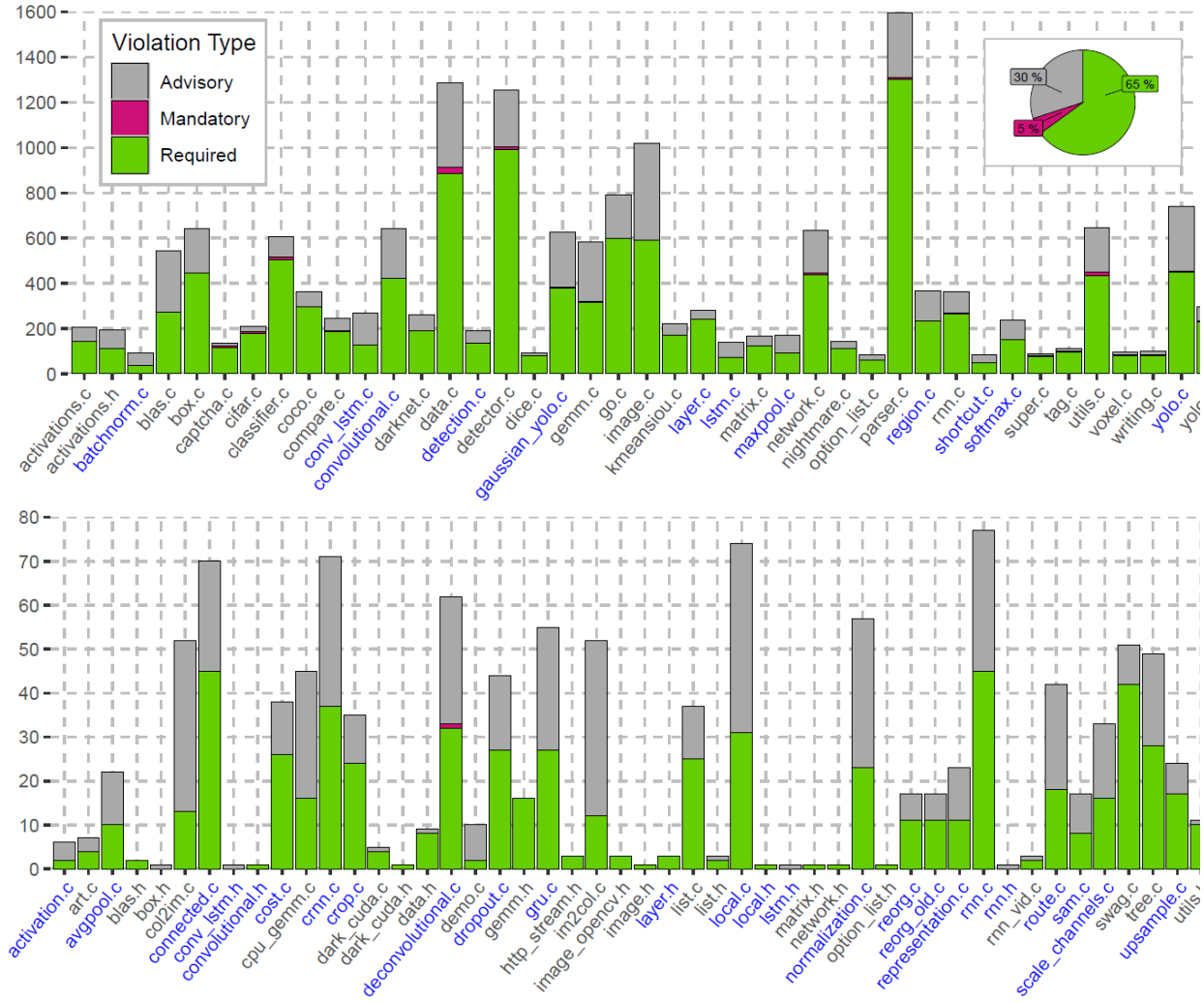


Figure 10. YOLOv4 violations decomposed by files

Additionally, we have collected in Table 13 the 15 most frequently violated rules in the entire YOLO from the results provided by Polyspace, limiting the displayed files to those with the highest number of violations. From this table, it can be observed that violations of these 15 rules in the 24 selected files account for 12,239 violations (68.07% of the total), increasing to 14,742 (82%) when considering all files. More specifically, the 28.69% of violations are concentrated in *data.c*, *detector.c*, *image.c*, and *parser.c* files. These are the largest files, where the network and layer configurations are performed—an area that is prone to MISRA C violations, as will be explained.

Table 13. Summary of main rules and files violations

Name	Rule ID															Sum
	12.1	10.3	10.4	15.6	17.7	21.6	7.4	14.4	11.5	10.1	8.7	8.13	21.3	8.4	5.8	
blas.c	162	42	75	62	8	8	-	11	1	4	3	73	1	7	20	477
box.c	162	169	142	13	7	7	-	-	1	3	14	4	1	13	4	540
cifar.c	14	26	14	11	21	21	21	6	-	-	8	1	2	9	4	158
classifier.c	42	50	31	66	57	50	69	36	11	11	12	15	13	14	12	489
coco.c	37	24	11	23	29	24	97	16	14	1	8	5	13	9	4	315
compare.c	25	23	15	9	25	23	5	4	5	2	10	1	3	11	8	169
conv_lstm.c	35	41	6	41	2	2	-	26	52	1	5	-	-	-	1	212
convolutional.c	109	135	71	60	13	13	-	37	45	13	12	12	16	10	8	554
darknet.c	36	20	10	2	28	27	5	11	3	17	13	1	-	13	3	189
data.c	211	217	210	84	62	58	17	52	37	32	37	19	34	28	31	1129
detector.c	163	103	99	118	154	149	93	67	28	16	10	27	37	10	13	1087
gaussian_yolo.c	197	123	104	31	4	3	-	15	10	10	8	6	12	5	4	532
gemm.c	178	47	38	35	12	9	-	6	2	79	14	36	4	17	3	480
go.c	126	60	36	85	82	95	4	32	17	64	24	9	21	25	17	697
image.c	316	117	134	87	34	35	2	10	10	13	30	13	9	25	30	865
layer.c	-	-	-	76	-	-	-	76	-	5	1	-	79	1	-	238
network.c	68	44	29	85	26	23	27	26	35	13	18	15	37	12	14	472
parser.c	159	189	13	173	128	139	349	49	25	56	55	8	20	57	23	1443
region.c	87	58	69	12	4	3	-	16	7	9	6	6	-	5	13	295
rnn.c	48	61	21	19	35	40	11	13	17	1	12	9	2	13	12	314
softmax.c	49	41	13	27	9	9	-	11	18	7	4	2	5	1	-	196
utils.c	65	64	48	64	28	33	44	13	18	28	11	33	18	7	11	485
yolo.c	35	24	11	22	25	20	36	16	15	1	6	5	12	7	6	241
yolo.c	231	125	119	54	17	14	-	28	17	25	8	7	5	7	5	662
Total	2555	1803	1319	1259	810	805	780	577	388	411	329	307	344	306	246	12239

3.3.3.1.2 Corrective actions

This subsection addresses the main corrective actions performed to adhere to MISRA C guidelines. Since our work focuses on the inference phase, where the model features remain fixed, it is possible to avoid dynamic memory allocation and properly store the configuration parameters. These are the first corrective actions discussed in this section, followed by general corrective actions to address the most frequently violated guidelines.

Memory Allocation Strategy: The original darknet-based YOLO initially focuses on configuring the network and parsing the layers from information extracted from external files. These include “yolov4.cfg”, for network and layers configuration, “coco.names”, for class labels for which the model has been trained and “yolov4.weights”, for the model’s learned parameters, among other configuration files involved in the setup.

Accessing this information from files relies on input/output functions, which violate rule 21.6 (1013 violations throughout the code). Once opened, some data needs to be stored as integers or floating-point values, while the provided information is in string format. The original YOLO uses conversion functions like “atoi” and “atof”, which are forbidden according to rule 21.7 due to their undefined behavior when conversion fails (34 violations). Additionally, it frequently allocates memory dynamically using macros expanded into functions such as “malloc” from <stdlib.h> library, violating rule 21.3 (386 violations) and, by extension, directive 4.12, which is explicitly enforced throughout this rule.

Since our work focuses on adapting the inference phase of YOLO, where the model and its associated features remain fixed and do not require further tuning, safeYOLO eliminates the previously mentioned issues by adopting a static memory allocation strategy. It embeds preloaded and preconverted configuration data directly into the final configuration structure and embeds the model’s learned parameters—biases, rolling mean, scales, activations and weights—and the outputs in source and header files using the raw IEEE 754 binary format to ensure floating-point precision and reproducibility.

Network Parameters Storage: Regarding the configuration structs, we use the same structs in which network and layer configuration are stored at runtime in the original YOLO but generate a template with predefined default values for each entity. This ensures that all values are initialized before being used and that structs are not left partially initialized, aligning with rule 9.3 for arrays and rule 9.1 for variable initialization. Although rule 9.1 does not explicitly extend to structs, we consider it good practice to apply the same principle.

Our design allows AI developers to modify this configuration structs for their specific applications in an intuitive way. However, care must be taken to avoid multiple initializations of the same entity, as this would violate rule 9.4. Additionally, the original defensive programming mechanisms are preserved but adapted to ensure all configuration values remain within the expected range.

General Corrective Actions: The rest of this subsection collects the general corrective actions applied to comply with MISRA C, grouping them according to the categories defined in MISRA C and indicating, in brackets, the total number of violations for each explained category.

- **Identifiers (914).** This category states that identifiers (macros, typedefs, tags, functions, etc.) must be unique regardless of scope, namespace, or linkage (internal/external). The most frequently violated rule is rule 5.8, with 318 violations, which specifically forbids using the same name for objects or functions with external linkage. To address this, we have renamed them accordingly.
- **Literals and constants (874).** Its main aspects involve enforcing proper suffixes for data types, such as “u” for unsigned integers and “f” for floating-point numbers and preventing non-const assignments of string literals to pointers. The latter is the most violated (rule 7.4), with 849 occurrences. This rule prevents attempts to modify string literals, which can result in undefined behavior. For example, C99 does not explicitly specify whether string literals that share a common termination must be stored in separate memory allocations. Consequently, modifying a string literal could inadvertently alter another one. To prevent this, we have explicitly defined them as const-qualified.
- **Declarations and definitions (1392).** It focuses on avoiding ambiguities and potential conflicts across translation units by providing guidelines for properly declaring and defining functions and variables (appropriate linkage, scope, visibility, etc.). The most frequently violated rule is rule 8.4 (379 violations), which states that objects or functions with external linkage must have a compatible declaration. Additionally, rule 8.7 (423 violations), an advisory rule, specifies that objects or functions referenced in only one translation unit should not be defined with external linkage. To address these violations, we have placed the declarations of external functions and variables in header files accessible by the required translation units. For functions or variables used in only one translation unit, we have restricted their visibility by employing internal linkage using the static keyword.
- **The essential type model (4303).** This category aims to enforce a robust type-checking system, reducing the risk of issues such as loss of value, sign, or precision due to improper type conversions. It requires that both implicit and explicit type conversions be carefully checked and controlled to ensure type compatibility and prevent potential problems during execution. The rules in this category emphasize using well-defined, compatible data types and avoiding complex or unsafe type casting, ensuring that type mismatches are minimized. The most frequently violated rules in this category are rule 10.1 (480 violations), rule 10.3 (2098 violations), and rule 10.4 (1527 violations). To address these violations, we have specifically employed appropriate

essential types, avoided assignments to narrower essential types, and ensured the use of compatible essential types in arithmetic operations, respectively.

- **Pointer type conversions (959).** It defines the conversions permitted and those that must be avoided due to potential unexpected behaviour, such as conversions between pointers and any arithmetic types other than integers, which can result in incorrect memory alignment. In this category, rule 11.5 (597 violations) is the most frequently violated. As an advisory rule, compliance has been ensured as much as possible; for any exceptions, it has been verified that pointer conversions do not lead to undefined behavior.
- **Expressions (3037).** It centers on avoiding comma operator, limiting the right-hand operands of a shift operator with a top range lower than the width in bits of the essential datatype or explicitly defining the precedence of operators within expressions, among others. Rule 12.1 is the most violated (3003 violations) and we have explicitly defined the precedence using parentheses to make the order of execution clear, as MISRA C recommends.

3.3.3.1.3 SafeYOLO Integration

This subsection describes the integration of the previously introduced safeYOLO into a real-time safety-critical system. One of the main limitations of the proposed safeYOLO approach is its sequential execution. The strategy adopted to minimize systematic faults—relying on the widely used C programming language—restricts the performance achievable by safeYOLO. To address this limitation, the integration strategy leverages safeYOLO as a periodic diagnostic mechanism for detecting runtime faults. Specifically, we propose a diverse redundancy scheme, as is depicted in Figure 9, as outlined in IEC 61508, to enhance fault detection capabilities while maintaining system reliability.

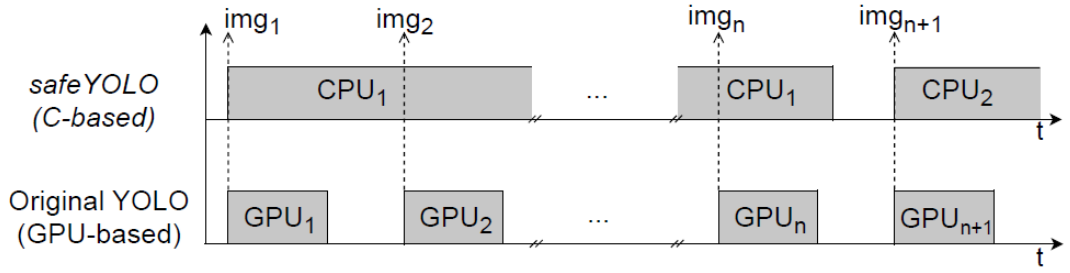


Figure 11. SafeYOLO integration with a GPU-based implementation

Since the execution time of the CPU-based safeYOLO implementation is higher than that of accelerator-based implementations such as GPUs, safeYOLO shall be synchronized with original YOLO to periodically evaluate the same image. A semantic comparison of the results explicitly verifies the proper behavior of all the components and built-in mechanisms, such as cache coherency, employed in the underlying platform. In Figure 9the subindex of GPU and CPU refers to the processed image index, numbered progressively based on their appearance. We conducted timing performance experiments to assess the feasibility of our proposal by measuring the average execution time over 5,000 images from the COCOeval validation dataset2. The safeYOLO implementation was deployed on the ARM Cortex-A78 cores, while the GPU-based implementation ran on the CUDA cores of the Ampere architecture in the NVIDIA Orin AGX GPU. The results indicate that the execution time of safeYOLO is 72 times higher than that of the original GPU-based implementation. Then, the applicability of our solution depends on whether this execution time remains acceptable within the Process Safety Time (PST) of the specific safety-related application. We employed COCOeval3, the official evaluation tool of the COCO dataset, to quantitatively compare the

object detection performance of our MISRA C-compliant framework against the original YOLO. The evaluation was conducted over the COCOeval validation subset using standard metrics such as Average Precision (AP) and Average Recall (AR) across different object sizes and IoU thresholds. The results show that both implementations yield identical evaluation scores, confirming that our modifications—primarily focused on achieving MISRA C compliance and eliminating dynamic behaviors—do not alter the detection performance. This outcome is consistent with expectations, as the inference logic and model parameters remain unchanged. Thus, we validate the introduced changes preserve functional equivalence while enabling a step towards a safe end-to-end safe AI-framework.

3.3.3.1.4 Conclusions

The adaptation and the experiments regarding the adaptation of YOLO according to MISRA C recommendations has led to safeYOLO, a statically memory-allocated version of the YOLO object detector that complies with MISRA C guidelines. We analyze the key corrective actions applied to address the 17,976 violations detected by Polyspace Bug Finder SAT across YOLO's 41,477 lines of code. The Polyspace analysis reveals that 15 rules account for 68.07% of the total violations, with 28.69% of all violations concentrated in just 4 out of the 89 files used by YOLO. We have described the main corrective actions undertaken to comply with MISRA C, resulting in a YOLO version with no required or mandatory violations, emphasizing the transition from dynamic to static memory allocation and our proposed configuration storage strategy. We also discuss general mitigation actions applied to address the most frequently violated rules.

3.3.3.2 Diverse Framework – TensorRT and Pytorch

As explained in D2.2 [Section 3.1.2.4, DRS_2], diverse framework redundancy (low to medium diagnostic coverage) involves replicating the same DL model and running it on different inference frameworks (e.g., PyTorch and TensorRT). This approach combines outputs to detect differences and mitigate faults.

The following experiments were performed:

- EX1: Single TensorRT model (baseline)
- EX2: PyTorch GPU and TensorRT models combined
- EX3: PyTorch GPU model with fault injection and TensorRT model combined

Fused AP Table:

Metrics	EX1	EX2	EX3
AP1	0.4740	0.4750	0.4659
AP2	0.6053	0.6092	0.5983
AP3	0.5174	0.5172	0.5078
AP4	0.2723	0.2750	0.2699
AP5	0.5300	0.5302	0.5238
AP6	0.6689	0.6689	0.6589

Fused AR Table:

Metrics	EX1	EX2	EX3
AR1	0.3611	0.3633	0.3609
AR2	0.5286	0.5298	0.5255
AR3	0.5350	0.5363	0.5321
AR4	0.3029	0.3070	0.3027
AR5	0.5901	0.5899	0.5871
AR6	0.3611	0.3633	0.3609

Diverse framework redundancy effectively enhances fault tolerance by replicating a deep learning model across PyTorch and TensorRT. Experiments demonstrate that EX2 (PyTorch GPU and TensorRT combined) achieves slightly improved performance over EX1 (single TensorRT baseline), with Average Precision (AP1@0.50:0.95 = 0.4750, +0.2%) and Average Recall (AR3@maxDets=100 = 0.5363, +0.2%), maintaining consistency across runs and stable recall for specific conditions (e.g., AR6 = 0.3633). Fault injection on PyTorch in EX3 reduces performance (AP1 = 0.4659, ~98.2% of EX2; AR3 = 0.5321, ~99.2% of EX2), but combining with TensorRT mitigates single-fault effects, recovering performance close to the baseline. These results confirm DRS_2's low to medium diagnostic coverage, validating its ability to detect and mitigate random and systematic faults for safety patterns SP1/SP2.

3.3.4 Diverse Concept

As explained in D2.2 [Section 3.1.2.5, DRS_5], diverse concept redundancy involves developing two DL models based on complementary but distinct concepts, aiming at the same goal. The first Diverse Concept implementation is Part Detector where a model detects the whole object while the diverse redundant model detects specific parts of the object.

3.3.4.1 Part Detector

In this case, we evaluate a diverse concept redundant implementation of an application based on one model for holistic object detection (e.g., identifying a "person") and another for object part detection (e.g., detecting sub-components such as head, torso, arms, legs, hands, feet). This scheme employs DL model diversity on the same inference platform, addressing both AI-related and traditional functional safety risk factors, while also mitigating AI performance limitations.

Outputs from these two models are fused using a decision function, to identify inconsistencies. For example, a detected person without supporting parts might indicate a FP, whereas detected parts without an associated person might reveal a FN, potentially enabling error detection. This "part voter" mechanism enhances error detection and recovery by leveraging semantic complementarity, as object parts provide redundant evidence for robust detection, especially under faults such as noise or occlusions.

The following experiments were performed:

- EX1: Baseline: Single YOLOv8x detecting the person class.
- EX2: Baseline with fault injection: YOLOv8x detecting people under fault injection.
- EX3: Part detector: Custom part detection model which detects the class person as well as parts (evaluated for person detections only).

- EX4: Part detector with fault injection: Custom YOLOv11x detecting persons and parts under fault injection (evaluated only for person detections).
- EX5: Part Voter: Combined baseline detector and parts
- EX6: Part Voter with fault injection in the baseline model
- EX7: Part Voter with fault injection in the part detector
- EX8: Part Voter with fault injection in both models

AP Table:

Metrics	EX1	EX2	EX3	EX4	EX5	EX6	EX7	EX8
AP1	0.6485	0.4786	0.3287	0.3259	0.5637	0.4624	0.5627	0.4591
AP2	0.8620	0.7030	0.4410	0.4400	0.7580	0.6630	0.7570	0.6620
AP3	0.7090	0.5150	0.3440	0.3420	0.6040	0.4840	0.6020	0.4800
AP4	0.4420	0.2850	0.0140	0.0150	0.3280	0.2060	0.3320	0.2090
AP5	0.7320	0.5720	0.3980	0.3930	0.6360	0.5380	0.6280	0.5320
AP6	0.8480	0.6690	0.7670	0.7580	0.8270	0.7690	0.8240	0.7570

AR Table:

Metric	EX1	EX2	EX3	EX4	EX5	EX6	EX7	EX8
AR1	0.2130	0.1770	0.1740	0.1740	0.2080	0.1910	0.2070	0.1890
AR2	0.6400	0.5010	0.3550	0.3530	0.5760	0.4850	0.5730	0.4810
AR3	0.7393	0.6074	0.3707	0.3688	0.6297	0.5381	0.6274	0.5345
AR4	0.5720	0.4180	0.0090	0.0100	0.4320	0.3060	0.4360	0.3090
AR5	0.8130	0.6890	0.4490	0.4470	0.6770	0.5910	0.6690	0.5850
AR6	0.9020	0.7950	0.8340	0.8270	0.8760	0.8310	0.8710	0.8220

These results clearly illustrate both the strengths and limitations of diverse concept redundancy. The Part Voter without fault injection reduces AP by approximately 13.1%, primarily due to strict voting criteria that inadvertently discard valid detections lacking sufficient parts evidence. Fault injection transparently degrades mono performance: Baseline AP drops significantly (~26.2%) due to spurious detections caused by injected noise, while Part detector AP exhibits greater resilience (~0.8% drop). Under single faults, the Part Voter shows mixed recovery effectiveness: with fault injection in the baseline model, performance is reduced by ~3.5%, whereas with fault injection in the part detector, performance nearly restores the original Part Voter result (within 0.2%). With fault injection in both models, AP drops strongly (~18.6%), indicating limited robustness against correlated noise.

4 DL Safety Lifecycle for DL-software V&V

The figure 10, has the goal to show the component-level V&V strategy objectives, such as:

1. the verification of the intended functionality; and
2. the systematic improvement of the system, where such improvement is ensured.

The process starts through an examination of the identified triggering conditions, which will be assessed to validate whether it is adequately mitigated by an existing safety measure.

When a triggering condition it is not covered by a safety measure, it is necessary to update the system specifications and therefore their implementation, whereas, when a condition is already covered by a safety measure, a test case is derived, which is useful to evaluate the intended functionality behaviour.

Once all test cases are created and executed, resulting data shall be analysed to:

- Create evidence that the intended functionality behaves as expected, in case of passed tests.
- Implement needed corrective action by updating system specification and implementation according to root cause analysis results, in case of failed tests.

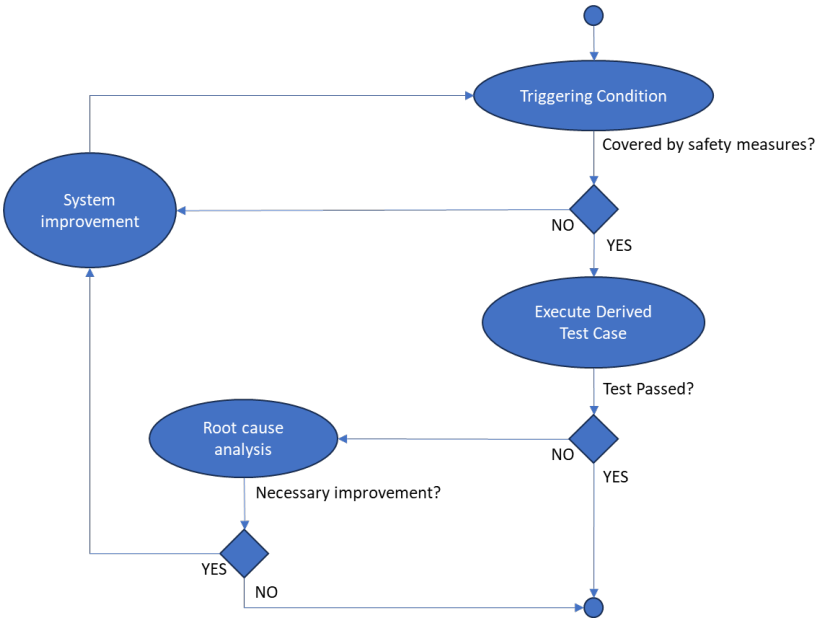


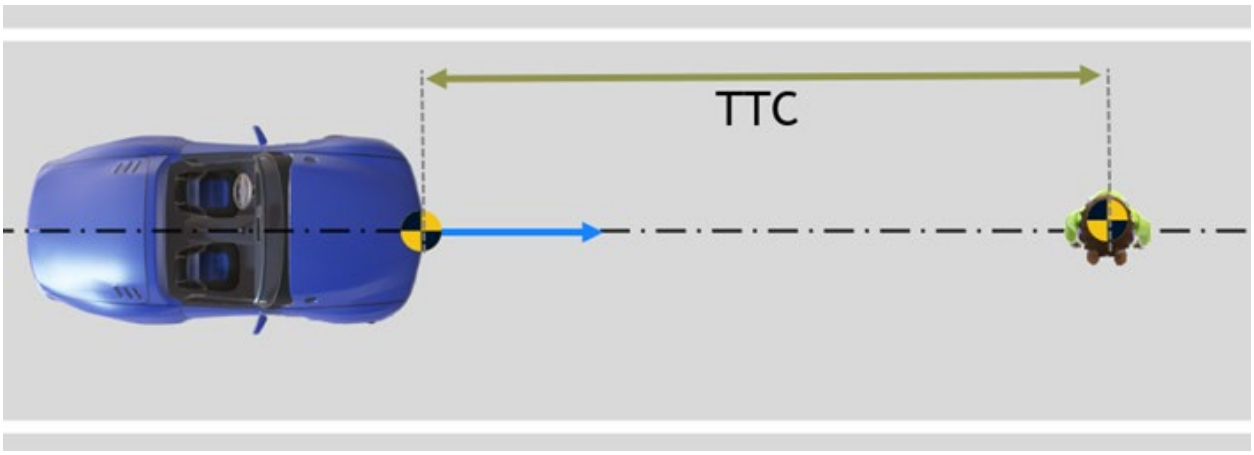
Figure 12. component-level V&V strategyScenario catalogue and related test matrix

The V&V strategy is initiated from the scenarios, which are created in accordance with the related ODD (where is defined every specific condition).This facilitates the analysis of the hazardous situation, which has the potential to impact on the safety of the vehicle.

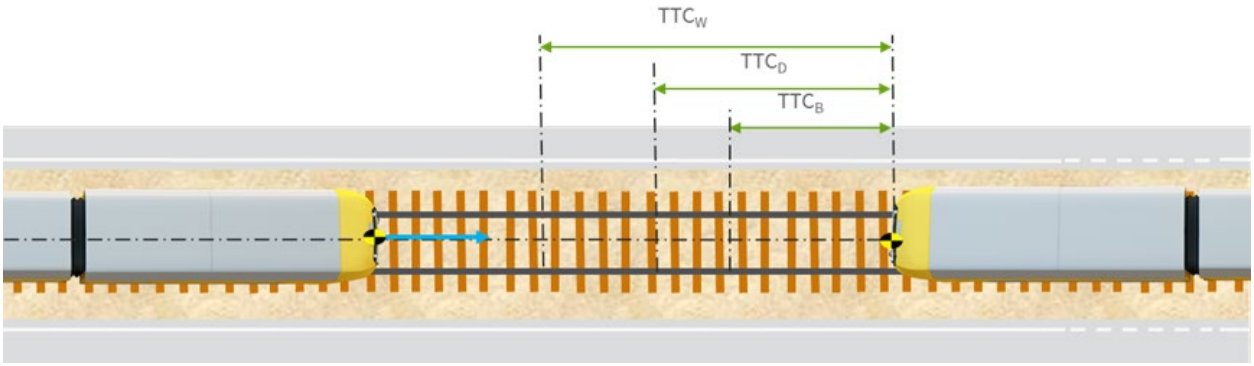
The validation of the created scenarios is to be conducted by means of tests assisted by test case specification derived from the scenarios themselves, which will be useful for the tester as evidence for the result of the test executions.

Below some descriptive pictures of the derived scenarios for each use case we worked on:

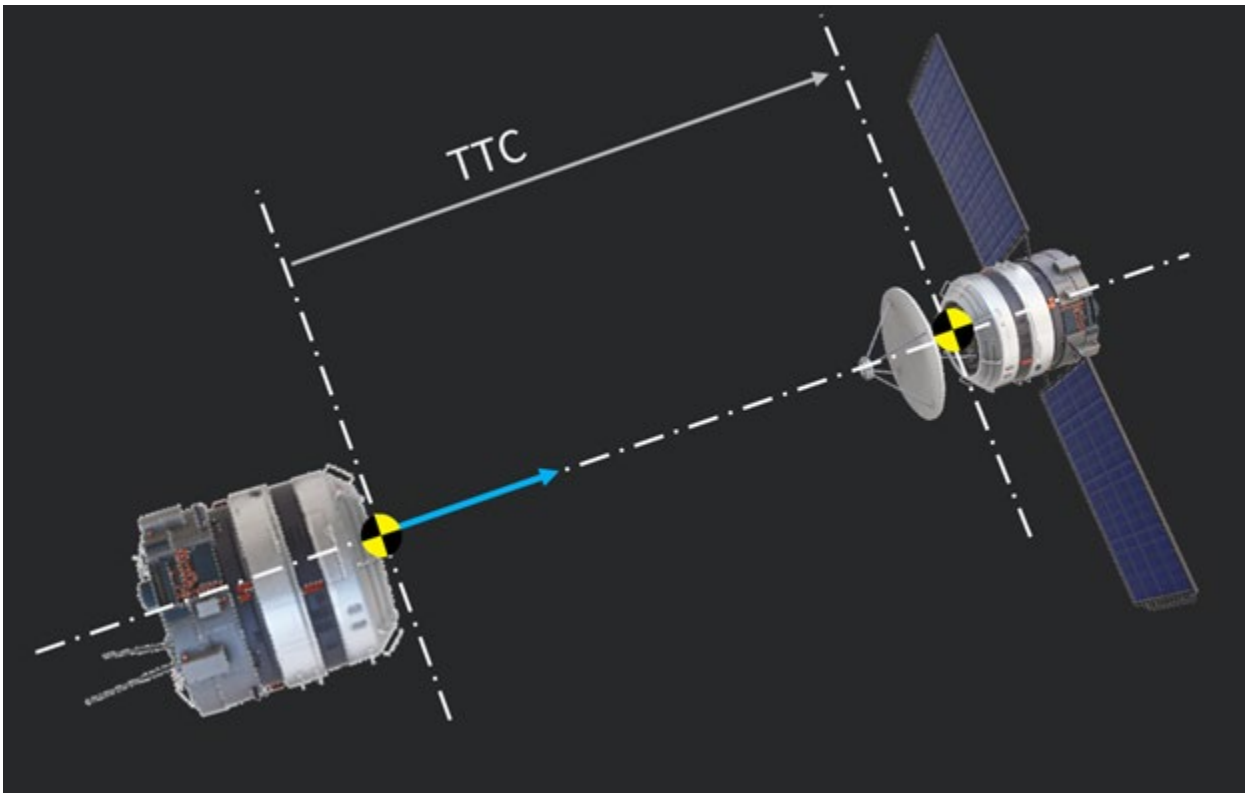
Automotive use case:



Railway use case:



Aerospace use case:



For a more comprehensive overview of the part of V&V strategy which concerns the scenarios catalogue and the related test matrix, please refer to the deliverable D2.3(see clause "Scenario catalogue and related test matrix").

4.1 System-Theoretic Process Analysis (STPA)

STPA is a hazard analysis technique that has been developed for the purpose of evaluating the safety of complex systems and identifying potential specification insufficiencies.

According to the established specifications, including the Item Definition, potential losses of functionality, that may lead to hazardous behaviour, have been identified. Appropriate improvements have been implemented to mitigate these risks.

A set of requirements are created from them, which are necessary to prevent or manage the hazards in question.The STPA method is divided into two sides:

- the analysis of the hazardous behaviours, see 4.2.1 Triggering Condition and related Test Matrix
- The potential improvements to mitigate the found hazardous behaviour, see 4.2.2 STPA: Safety Measures to mitigate the hazardous behaviour

For a more comprehensive overview of the part of V&V strategy which concerns the STPA, please refer to the deliverable D2.3(clause 3.2. Focus STPA (triggering condition) and related test matrix)

4.1.1 Triggering Condition and related Test Matrix

Triggering conditions have been shown to act as the initiator of a chain reaction which can bring to hazardous behaviours.These conditions, which may occur during normal use, could cause a safety risk due to the limitations of the system.

The functionality of the system is dependent on various environmental and situational factors, including but not limited to:

- sensor limitations, such as blind spots, which may affect the range over which the system can detect environmental stimuli
- ambiguous situations, where the system may misinterpret the environment
- other factors

The fundamental characteristic of triggering conditions is that they represent scenarios that are foreseeable, where it is the system's design limitations, rather than malfunctions, that create potential safety risks. These risks must be identified and addressed during the development process.These found triggering conditions will be validated through dedicated test execution aided by dedicated test case specifications.

The part of the V&V strategy related to the analysis of the hazardous behaviours which contains the analysis of the triggering condition related to them is explained in the deliverable D2.3(3.3. Example of V&V strategy application).

4.1.2 STPA: Safety Measures to mitigate the hazardous behaviour

A comprehensive analysis of hazardous behaviours is useless unless a method of preventing them is identified. The final step of the STPA method focuses, in fact, on improving the system to eliminate the risk (with the option to accomplish S=0 or C=0).

The hazardous causes are exposed by the found triggering condition which causes the chain reaction till the analysed hazardous behaviour.

The scope of STPA is mitigating the risk of occurrence related to the found triggering condition by implementing some safety measures, such as:

- **System modification** as measure which have the goalto maintain the intended functionality as much as possible, such as:
    - Increase the sensor performance and/or accuracy, which can be improved by:
        - Sensor technology improvements
        - Sensor calibration and installation improvements
    - Increase the actuator performance and/or accuracy by improving the actuator technology
        - Speed up image processing with enhanced computing power
        - using a machine learning accelerator or operation-efficient hardware

- Increase the performance and/or accuracy of the recognition and decision algorithms by algorithmic modifications
- Increase the visibility of the ego vehicle to enhance the controllability of other traffic participants in case of hazardous behaviour of the ego vehicle (e.g., installation of retro-reflectors, turn indicators, active sounds, etc.)

- **Functional restrictions** as measures which have the goalto partial maintain the intended functionality by degrading it:
  - Handing over authority as measure which have the goal to increase the controllability at lower levels of driving automation, by communicating to the driver to take over the driving task, modifying the Human-Machine Interface (HMI) or the user notification and DDT fallback strategy
- Addressing **reasonably foreseeable misuse** as measure which have the goal to address the misuses:
  - Customer education (information and training)
  - Improving the HMI to support the driver by providing the correct operation
  - Implementation of a driver monitoring and warning system
  - When the safety measures are implemented, they could produce unintended consequences, hence monitoring and review activities are relevant to ensure their reliability.

After the implementation of the safety measures, these must be validated by starting again the V&V strategy from the beginning (with the new safety measures implemented in the system), to ensure the reliability of the system, the correct implementation and integration in the system and the mitigation of the found hazardous behaviour.

4.2 Example of V&V strategy application

Most of the activities related to the V&V strategy were already explained in the deliverables D2.3 (3.3. Example of V&V strategy application and D2.1(4.3 Examples in the automotive domain), except for the safety measures foreseen, which covers the defined triggering conditions.

4.2.1 Automotive Use Case

All standards in the ISO/IEC 61508 tradition (including the automotive-oriented ISO 26262, 21448 SOTIF and the brand-new ISO/PAS 8800) share a two-pronged common approach to functional safety:

- On one side, they recommend/mandate progressive ‘rigor’ compared to basic engineering practices for requirements, design, implementation and testing;
- On the other side, they prescribe risk-based safety analyses to identify specific safety goals and to satisfy them with trusted safety measures.

While increased ‘rigor’, especially if ’safety-oriented’, is always welcome and commendable even in a FuSa-specific perspective, full compliance to FuSa standards cannot be achieved by just an “increased rigor" strategy.

The two cornerstone analyses in automotive-oriented FuSa are the HARA (introduced by ISO 26262) and the STPA (introduced by 21448 SOTIF, confirmed by ISO/PAS 8800 and complementary - not alternative - to HARA).

As mentioned above, the activities related to the V&V strategy were already explained in the deliverables D2.3 and D2.1.

The topic which will be reported here is related to the safety measures to be implemented at element level (triggering condition based).

For the automotive use case, we assumed some safety measures in the defined test cases specification based on the triggering condition, an example which describes all the path from the hazardous behaviour to the safety measure is reported below:

Hazard Vehicle-Level	AEB doesn't provide emergency braking when needed
Control Action	Perform emergency braking
Control Action Failure Mode	Emergency braking is not performed when a collision is imminent
Functional Insufficiency	The AEB does not provide the requested braking intervention
Triggering condition	Missing object detection due to camera low performances (e.g. damaged lens or undetected offline sensor)
Test Description	The test aims to verify whether the AEB will be deactivated when a damaged camera lens is detected at system boot state
Test Preconditions	• Kl.15 = off • No warning message available • A Collision Relevant Object is present • Camera lens is damaged • Intended functionality state: active
Test Steps (Operating)	Step 1: • AEB system is started Step 2: • At [x] ms, verify whether the AEB detects inaccurate camera position
Acceptance criteria (related to the last step)	• AEB is suppressed within [x] ms • Collision Relevant Object not detected • Intended functionality state: deactivated
Foreseen Safety measure	When a damaged camera is detected at boot state, the intended functionality will be suppressed and deactivated to reach the safe state.

These safety measures are supposed to be tested through a simulation to determine whether it is sufficient to mitigate the hazardous behavior that has been identified, or whether additional safety measures are required.

As indicated by the expected results of the test cases based on the triggering conditions, the designated safety measures have been implicitly delineated and explicated in this document.

The domain owner (NAVINFO) adopted a different approach from the one indicated by Exida, and it is the following one:

In order to demonstrate and guarantee the practical value of the project, the domain owner (NAVINFO) implemented 24 concrete variants, covering day/night, dry/wet, child/adult pedestrian, and multiple

approach speeds (10/30/50 km/h). It should be noted that these variants are based on a single scenario at vehicle-level from the 21 proposed by exida (coverage of the proposed scenarios: 4.7%).

NAVINFO implemented a limited number of the proposed triggering conditions (based on their applicability to the concept), and the related tests were not based on the proposed test matrix for the triggering conditions (one of which is illustrated above).

NAVINFO exhibited a high level of attention to how the software works with the specific hardware platform and the entire process is founded upon the construction of authentic working code with proper testing.

It is evident that meticulous documentation of each component is in place, thus allowing for comprehensive technical evaluation but not sufficient to provide all the evidence needed for a fully-fledged FuSa standard assessment.

Full compliance to automotive FuSa standards was never claimed as a direct requirement for the automotive use case, *that was rather oriented to adapt, port, and run the case study on the target platforms*.

The partial implementation of the expected work-products for full compliance is therefore not a shortcoming; the direction is the right one and the full road to compliance and 'certifiability' is clearly indicated, simply beyond the scope of the current R&D project and rather intended for full industrial projects, with quite different objectives and budgets.

As indicated by the expected results of the test cases based on the triggering conditions, the designated safety measures have been implicitly delineated in the related test cases specification and explicated in the example present in this document.

4.2.2 Railway Use Case

As mentioned above, the activities related to the V&V strategy were already explained in the deliverable D2.3.

The topic which will be reported here is related to the safety measures to be implemented at element level (triggering condition based).

For the railway use case, we assumed some safety measures in the defined test cases specification based on the triggering condition, an example which describes all the path from the hazardous behaviour to the safety measure is reported below:

Hazard Vehicle-Level	ATO doesn't provide service braking when needed
Control Action	Perform service braking
Control Action Failure Mode	Service braking is not performed when a collision is imminent
Functional Insufficiency	The ATO provides the intervention request too late.
Triggering condition	Too late detection of sensor decreases in performance (e.g. black screen)
Test Description	The test verifies whether the ATO system correctly detects repeated black frames (indicative of damage, obstruction, or misalignment) and transitions to safe state.

	NOTE: we may need to define some terms, like: • Black Frame Definition: a frame where average luminance (Y channel) < threshold (e.g. <10). • Temporal Condition: configurable number of black frames in last N frames (e.g. 5 of 10)
Test Preconditions	• Kl.15 = off • No warning message available • Camera field of view is covered • Intended functionality state: active
Test Steps (Operating)	Step 1: • Start ATO System Step 2: • Monitor image reception Count number of black frames • Check the number of the acquired black frames
Acceptance criteria (related to the last step)	• More than 5 of 10 frame received, are black frame • Safe state is triggered
Foreseen Safety measure	When a black screen issue in the camera is detected at boot state, the intended functionality will be suppressed and deactivated to reach the safe state.

These safety measures are supposed to be tested through a simulation (according to the defined test case) to determine whether it is sufficient to mitigate the hazardous behavior that has been identified, or whether additional safety measures are required.

As indicated by the expected results of the test cases based on the triggering conditions, the designated safety measures have been implicitly delineated in the related test cases specification and explicated in the example present in this document.

The showed example represents a complete STPA method execution from Hazard Vehicle-Level to the related safety measure to mitigate it.

Furthermore, some of the proposed safety measures were also implemented.

4.2.3 Aerospace Use Case

As mentioned above, the activities related to the V&V strategy were already explained in the deliverable D2.3.

The topic which will be reported here is related to the safety measures to be implemented at element level (triggering condition based).

For the aerospace use case, we assumed some safety measures in the defined test cases specification based on the triggering condition, an example which describes all the path from the hazardous behaviour to the safety measure is reported below:

Hazard Vehicle-Level	The GNC System wrongly provide information related to the docking operation
Control Action	Information needed to docking operation is provided to the driver
Control Action Failure Mode	Wrong information related to the docking operation is provided to the driver

Functional Insufficiency	The GNC algorithm wrongly processes the received data
Triggering condition	Agent parameters are wrongly estimated due to low computational power (e.g. overload, ...).
Test Description	The test aims to verify whether the GNC is deactivated when low computational power (which causes wrong agent parameters estimation) is detected
Test Preconditions	• Kl.15 = on • No warning message available • Intended functionality state: active
Test Steps (Operating)	Step 1: • Docking target is detected • No warning message provided • Intended functionality state: active Step 2: • Input signals from sensors are received by GNC • At [x] ms, before the Agent parameters estimation, the system has been overloaded
Acceptance criteria (related to the last step)	• GNC is suppressed within [x] ms • Intended functionality state: deactivated
Foreseen Safety measure	When a black screen issue in the camera is detected at boot state, the intended functionality will be suppressed and deactivated to reach the safe state.

These safety measures are supposed to be tested through a simulation to determine whether it is sufficient to mitigate the hazardous behavior that has been identified, or whether additional safety measures are required.

As indicated by the expected results of the test cases based on the triggering conditions, the designated safety measures have been implicitly delineated and explicated in this document.

Most of the safety measures we assumed, the domain owner (AIKO) covered them by testing the scenarios at vehicle-level.

4.3 Applied Normative

The V&V strategy applied in SAFEXPLAIN is created in compliance with regulation: Functional Safety and SOTIF, with the related standards.

The 'overarching' Functional Safety standard is the ISO/IEC 61508, that is not sector-specific and is the basis of almost all sector-specific standards. However, the current edition is quite old (2010) and does not cover aspects like Intended Functions and AI.

In the automotive sector, the quest for autonomous vehicles has spurred more specialized coverage, currently unparalleled by any other sectors. The standard which regulates the Functional Safety side is the ISO 26262, whereas the standard which regulates SOTIF (Safety of the Intended Functionality) is the ISO 21448.

The brand-new normative ISO/PAS 8800, specifically on AI aspects, was published in the last months. It appears that in the specific scope of this project most of its requirements have already been met thanks to rigorous adherence to what already required by ISO 26262 and SOTIF.

Finally, during the course of this project, an AI-extension of the popular ASPICE standard was published and officially integrated in the ASPICE PRM/PAM 4.0. Although not directly anticipated, compliance to this new standard has been also targeted and one of the three demos (railway domain use case) has already undergone an official ASPICE assessment, receiving a positive assessment.

4.3.1 ISO 26262

ISO 26262 is a functional safety standard for E/E (Electrical/Electronic) systems in road vehicles that provides a comprehensive framework to address hazards caused by malfunctioning behaviour, focusing on managing random hardware failures and systematic failures through a structured development process.

This approach presupposes the sufficiency and reliability of the intended functionality, focusing exclusively on failures and faults within the E/E systems.

The scope of the project encompasses the establishment of safety requirements, the implementation of safety measures derived from architectural-level analyses, and the provision of evidence to demonstrate that integrated system elements fulfil their safety requirements.

In the context of configurable systems comprising a multitude of elements or calibration data, ISO 26262 guarantees compliance across all potential configurations. However, it does not address hazards that may arise from the limitations inherent in the system itself when it is functioning correctly.

How the ISO 26262is applied in this Work Product was already explained in the deliverable D2.3(3.1 clause DL Safety Lifecycle for DL-software V&V).

4.3.2 ISO 21448

The objective of the Safety of the Intended Functionality (SOTIF) approach is to address a critical gap left by ISO 26262. The latter focuses on hazards that can occur even when systems function correctly according to their specifications.

SOTIF recognizes that electronic and electrical systems, dependent on environmental sensing, which may exhibit hazardous behaviour under certain conditions.

Such behaviour may be a consequence of limitations in their functionality.

SOTIF provides a framework for the identification, evaluation and mitigation of such risks through the implementation of comprehensive verification and validation strategies.

How the ISO 21448 is applied in this Work Product was already explained in the deliverable D2.3(3.1 clause DL Safety Lifecycle for DL-software V&V).

4.3.3 ISO/PAS 8800

This project was based on the two standards ISO 26262ì and ISO 21448, but, in the meantime, in December 2024, a new standard related to functionalities AI-based was published, which basically is an extension of the ISO 26262 and the ISO 21448 that specifically addresses the aspects related to AI.

At a first review, most of the results of this Work Package are already compliant with the new ISO/PAS 8800, as the specific narrow scope of the project was mostly covered by requirements of ISO 26262 and ISO 21448. In particular, we can mention:

- The analysis of the triggering condition related to the AI-based functionality, covered by 6.7.1 "cause-effect chain"6.7.1 "cause-effect chain“ of ISO/PAS 8800, aims to trace the progression from the initial cause of a functional insufficiency to the occurrence the related hazard.
- In the design it is also described the dataset source used to train the AI model with all the related considerations.
- The STPA (*System Theoretic Process Analysis*), introduced by SOTIF, is adopted and described in detail in the ISO/PAS 8800 (ANNEX E), and it is the method adopted in our V&V to analyse the triggering condition related to the found hazards in consistence with the ODD. ASPICE

Originally SPICE was a project funded by the European Union. The main goal was to develop a counterpart to CMMI. SPICE is an abbreviation and stands for "**S**oftware **P**rocess **I**mprovement and **C**apability d**E**termination". Even if software development was the main driver of the initiative at that time, the "S" in SPICE is nowadays used for the entire system development.

The project was very successful and finally, ISO issued the standard series ISO 15504-x to support the future development of specific standards for all kinds of domains. In the meantime, most parts of this standard have been replaced by the ISO 330xx series, and we have a choice from various Process Assessment Models. The focus of SPICE has widened to all Software-based Systems and to Operation and Maintenance processes. [source: https://intacs.info/spice-center].

ASPICE (Automotove SPICE) is a framework for the assessment of the maturity of software development processes specific to the automotive industry.

It is employed for the evaluation of the capability to develop reliable and high-quality automotive software. As it is an *Assessment Model*, rather than a *Development Model*, it is more abstract than most of the other technical standards, ensuring its applicability also to adjacent sectors (e.g. all transportation sectors).

The ASPICE framework uses an approach to evaluate and improve development processes, which covers the following domains:

- System, Software, Hardware, Machine Learning Engineering: requirements management, architecture, design, implementation, testing, and integration.
- Management: project management, quality, configuration, and risk management
- Support: quality assurance, verification, documentation

- Acquisition: supplier selection and monitoring
- Reuse: reusable asset management

The ASPICE framework assists in the development of validation strategies through the following mechanisms:

1. Complete traceability, which requires bidirectional links to ensure the validation of every feature.
2. Structured V&V, which is employed to denote a specific set of processes for the purposes of V&V.
3. Multi-level testing is defined as the implementation of unit, integration, system, and acceptance testing, with defined coverage criteria.
4. Risk Management: identification and mitigation of technical risks which could compromise validation
5. Objective metrics: they are defined as providing measurable indicators for the purpose of assessing validation completeness and effectiveness.
6. Systematic reviews: they are required at each stage to identify issues before final validation.

Exida has several SPICE Assessors in its ranks and it's very active in ASPICE-related Working Groups, including one on Hardware and one on Machine Learning.

Ikerlan has expressed the interest of organizing an ASPICE assessment of its Railway Demo, with a narrow scope centred around MLE (Machine Learning Assessment) processes.

The assessment has been thoroughly planned with a pre-assessment that took place in July 2025 and has concluded with the official assessment in early September. Preliminary results shown that the CL1 (Capability Level 1) target has been already achieved, then confirmed in the final official assessment. More details in Sect 10 and Annex I of D6.8 Final Exploitation Report.

5 Acronyms and Abbreviations

Below is a list of acronyms and abbreviations employed in this document:

ACC	Accuracy
AEB	Autonomous Emergency Braking
AI	Artificial Intelligence
ASPICE	Automotive SPICE
AP	Average Precision
AR	Average Recall/Anti-Clockwise Rotate
ATO	Automatic Train Operation
BS	Bottom Shift
CENELEC	Comité européen de normalisation en électronique et en électrotechnique
CL	Capability Level
CMMI	Capability Maturity Model Integration
COCO	Common Objects in COntext
COTS	Commercial Off-The-Shelf

CPU	Central Processing Unit
CR	Clockwise Rotate
DDT	Dynamic Driving Task
DL	Deep Learning
DMR	Dual Module Redundancy
DR	Dropout
DRS	Diverse Redundancy Schemes
E/E	Electrical and/or Electronic system
e.g.	Exempli Gratia (*for example*)
ECSS	European Cooperation for Space Standardization
Ego vehicle	vehicle fitted with functionality that is being analysed for the SOTIF
EQ	Equalization
ESA	European Space Agency
FN	False Negative
FP	False Positive
FPS	Frames Per Seconds
FuSa	Functional Safety
GB	Gaussian Filter
GC	Gamma Correction
GN	Gaussian Noise
GNC	Guidance, Navigation and Control
GPU	Graphics Processing Unit
HARA	Hazard Analysis and Risk Assessment
HF	Horizontal Flipping
HMI	Human-Machine Interface
HW	Hardware
i.e.	Id Est (*that is*)
IEC	International electrotechnical commission
II	Information Items
IoU	Intersection over Union
ISO	International organization for standardization
ISO/PAS	International organization for standardization - Publicly Available Specification
LS	Left Shift
mAP	Mean Average Precision
MB	Median Filter
ML	Machine Learning
MLE	Machine Learning Engineering
NMS	Non-maximum suppression
ODD	Operational Design Domain
OEM	Original Equipment Manufacturer
PAM	Process Assessment Model
PRM	Process Reference Model

ROS2	Robotic Operating System 2
RS	Right Shift
RV	Raising pixel Value
SH	Sharpening
SOTIF	Safety Of the Intended Functionalities
SP	Salt and Pepper Noise
SPICE	Systems Process Improvement and Capability dEtermination
STPA	System Theoretic Process Analysis
SW	Software
TMR	Triple Modular Redundancy
TP	True Positive
TS	Top Shift
TÜV	Technischer Überwachungsverein
V&V	Verification and Validation
VF	Vertical Flipping
w.r.t.	with respect to

6 Bibliography

[1] Li, Chih-Yang, Accessed in Oct-2023. YOLOv4 Tensorflow Keras Implementation. https://github.com/taipingeric/yolo-v4-tf.keras.

[2] Tsung-Yi Lin, Michael Maire, Serge Belongie, Lubomir Bourdev, Ross Girshick, James Hays, Pietro Perona, Deva Ramanan, C. Lawrence Zitnick, and Piotr Dollár, Microsoft COCO: Common Objects in Context. arXiv:1405.0312, 2015.

[3] Mark Everingham, Luc Van Gool, Christopher K. I. Williams, John Winn, and Andrew Zisserman, The pascal visual object classes (VOC) challenge. International Journal of Computer Vision 88, 2, 2010.

[4] Rafael Padilla, Sergio L. Netto, and Eduardo A. B. da Silva, A Survey on Performance Metrics for Object-Detection Algorithms. In 2020 International Conference on Systems, Signals and Image Processing (IWSSIP). 237–242. https://doi.org/10.1109/IWSSIP48289.2020.9145130, 2020.

7 Annexes

This section collects the annexes attached together with the deliverable D2.4.

7.1 Annex A: ESA meeting presentation

This annex collects the power point shared with ESA meeting attendants. In that presentation, the SAFEXPLAIN consortium presented its latest results to representatives of several aerospace and embedded system industries including Airbus DS; BrainChip, the European Space Agency (ESA), Gaisler, and Klepsydra, showcasing major strides in making AI safe and certifiable in critical autonomous embedded systems.