Advances in Reliability, Safety and Security, Part 3 – Kołowrocki, Dąbrowska (eds.)
 ISBN 978-83-68136-15-9 (printed), ISBN 978-83-68136-02-9 (electronic)

Advances in Reliability, Safety and Security

ESREL 2024
Monograph Book Series

Review On Safety Assessment And Validation Methods Of AI Module In Railway

Han Liu, Fei Yan, Yunlai Sun, Yiwei Zheng

School of Automation and Intelligence, Beijing Jiaotong University, Beijing, China

Abstract

With the rapid development of AI, it has significantly impacted various domains, including railway transportation systems. Due to the stringent safety requirements of railway systems, the application of AI technology necessitates safety assessment and verification to mitigate risks. This paper first provides a detailed overview of safety assessment and verification methods for AI modules from three perspectives: testing-based methods, robustness evaluation-based methods, and safety assurance case-based methods. Subsequently, it illustrates the practical application of safety assessment and verification technologies for AI through the example of Intelligent Train Eye system. Finally, the paper summarizes the current challenges in ensuring the safety of AI modules and proposes three aspects that need attention in subsequent technological development.

Keywords: ai, safety, testing, assessment, validation

1. Introduction

1.1. Application of AI in the railway

Since the 1980s, urban rail train control systems have progressed through various technological stages, including the fixed block system, quasi-moving block system, and moving block system. Given the rapid progress of AI technology, the development trend for urban rail transit is characterized by the integration of digitization, networking, and intelligence. Neural networks (NN), cloud computing, and 5G technologies are poised to play pivotal roles in the rail transit field, leveraging their respective strengths to address key challenges in the railway.

In the field of railway transportation, the autonomous obstacle detection system for trains utilizes AI technologies such as image recognition and CNN to intelligently monitor and safeguard the clearances ahead of train movements. This allows trains to autonomously respond to potential obstacles, enhancing operational safety (Yu et al., 2018; He et al., 2021). Intelligent maintenance and diagnostic technologies employ various classifiers to analyze railway signal equipment, efficiently identifying subtle faults that can significantly impact system operations (De Bruin et al., 2016; Sun et al., 2022). In terms of intelligent scheduling, optimization strategies and virtual coupling method are employed to effectively address issues related to reduced operational efficiency caused by train faults (Zhang et al., 2022; Zhou et al., 2023). These approaches maximize operational efficiency to the greatest extent possible. Furthermore, the introduction of an integrated dispatch control solution aims to reduce human error rates in scheduling by leveraging big data and cloud computing technologies. This comprehensive solution seamlessly connects various components, using data-driven intelligent analysis to achieve real-time monitoring and adjustments of train operational states. This approach enhances the accuracy and efficiency of scheduling decisions.

Overall, the application of AI in the field of railway transportation contributes to accident prevention, improved train scheduling, delay prediction and optimization of overall administration and operations. This serves to enhance the operational efficiency, safety, and reliability of the system. The introduction of these

technologies not only innovates the railway sector but also lays a strong foundation for advancing the transportation industry as a whole.

1.2. Problems faced by railway application of AI

The railway transportation system, being a safety-critical system, carries the potential for serious consequences and significant maintenance costs in the event of errors. Consequently, the safety of AI modules is directly linked to the risk and severity of incidents during system operation. Thus, the use of AI carries a series of concerns.

AI systems are typically employed in environments where the complexity is challenging for humans to fully analyze and describe. They can autonomously learn rules without relying on manually generated representations of analysis, details, and intricate environmental conditions. Given that AI relies on extensive training samples to learn complex rules, challenges emerge when deploying AI in open environments, particularly when addressing non-distributed test samples. This challenge encompasses not only the performance of the model but also its robustness and the reliability in handling uncertainty, forming a comprehensive and intricate issue. The question of how AI technology and systems should be employed to ensure safety has not yet been addressed in mature international standards for functional safety.

Additionally, from a safety perspective, research indicates that Deep Neural Networks (DNNs) are susceptible to adversarial sample attacks involving subtle perturbations to input samples, imperceptible to the human. Due to the lack of effective validation techniques for DNNs, model validation based on AI modules is typically confined to performance metrics on standardized test sets and end-to-end simulations in operational design domains. This imposes certain limitations on the assessment of model safety and robustness.

Therefore, ensuring the safety of AI-based systems throughout their entire lifecycle is an urgent concern. Through an extensive review of relevant research, this paper categorizes three types of safety assessment and validation methods for AI modules, namely testing-based methods, robustness evaluation-based methods, and safety assurance case-based methods. The aim is to provide a thorough and comprehensive understanding for ensuring the overall safety of AI-based systems.

2. Safety validation and testing methods of AI module

2.1. Testing-based methods

Before the deployment of AI modules, systematic testing is essential to identify potential defects in the system at an early stage. AI software testing encounters challenges due to the uncertainty, unpredictability, and randomness associated with AI applications. Traditional programs express logic using control flow statements, while NN autonomously learn their logic from extensive data, using edge weights and non-linear activation functions between different neurons to represent it. These differences set AI testing apart from traditional testing. Hence, the challenge in verifying the safety of AI systems lies in the inapplicability of traditional testing methods.

Coverage-guided testing has shown some achievements in software defect detection (Huang et al., 2021; Xie et al., 2019). We approve that programs (software) with high testing coverage execute more source code during testing, resulting in lower chances of undiscovered software defects and increased safety. A new testing metric-Neuron Coverage (NC) has been proposed in deep learning field (Pei et al., 2017), which guides testing by observing and calculating the activation of neurons in the neural network. Given a set of neurons N and an input test set T, NC is defined as the ratio of the number of activated neurons N_{act} to the total number of neurons N_{all} in the neural network.

$$NC = \frac{N_{act}}{N_{all}} \tag{1}$$

A new NC test metric was introduced by Ma L et al. (2018), considering the corner neurons and enriching the dimensions of testing considerations. Tian Y et al. (2018) proposed DeepTest, a tool for automatically generating test cases. This tool systematically explores different parts of DNN logic by generating test inputs that maximize the activation of neurons. Zhang M et al. (2018) proposed a deformation relationship based on GAN networks for automatically synthesizing images under adverse weather conditions. While coverage-guided testing has achieved certain research progress, the connection between coverage metrics and probabilistic assurance remains a topic of ongoing investigation (Huang X, 2021).

Symbolic execution initially proposed by James C. King in 1976 (King, 1976). However, constrained by the solving capability of constraint solvers, symbolic execution technology did not develop until 2008 when Cristian Cadar designed the "execute-generate testing" technique. This approach involves first performing symbolic execution and then generating test cases based on the results of symbolic execution. Traditional symbolic execution involves executing a program on symbols rather than specific inputs and systematically explores program paths up to a given depth range (Cadar et al., 2008). However, DNNs are typically highly nonlinear. So, the methods for solving DNN systems are not yet complete. Additionally, NN consist of thousands of neurons, far exceeding the current capabilities of symbolic reasoning tools, posing serious scalability issues. Gopinath D, Wang K et al. (2018) proposes a lightweight solution. In cases of the network has no constraints, it utilizes the DeepCheck method to convert the network into an imperative program. Then, by inspecting the result of symbolic execution, it checks the coverage scope of the neural network, extracts an interpretation of the behavior. Additionally, Monte Carlo Tree Search (Wicker et al., 2018) can be used to test DNN system. This method and gradient-based search in coverage testing (Pei et al., 2017; Ma et al., 2018; Tian et al., 2017) both concrete execution.

Subsequently, inspired by the Modified Condition/Decision (MC/DC) coverage standard, researchers proposed a hybrid execution technique known as Concolic Testing for the structural features and semantic information of DNNs (Sun et al., 2018, 2019). While Concolic testing is a powerful tool, it still faces challenges in dealing with the complexity of large programs and ensuring the effectiveness of generated test cases.

Testing methods serve as a pathway to verify the safety of AI systems, with the primary focus on how to generate high-quality counterexamples that induce incorrect model outputs. Therefore, scholars consider various scenarios and boundary conditions, striving to thoroughly explore corner cases. These high-quality counterexamples can not only help researchers better understand the weaknesses of the model, but also provide developers with feedback on the network and improve the model. In the process of interpreting this feedback information, the interpretability of the model and testing activities becomes another focal point of research.

2.2. Robustness evaluation-based methods

Significant achievements and effects have been attained by DNNs in image classification. However, the output instability demonstrated by DNNs when confronting adversarial perturbations shows potential negative impacts on critical tasks in railway field, such as obstacle detection. This has raised concerns about its safety.

The definition of robustness is that AI components output the same predicted labels for inputs within a small region (typically defined in a L_p-norm ball) (Huang et al., 2017). This signifies the defensive capability of model against disturbances and is a critical metric for evaluating system safety (Hampel, 1971).

The classic approach is to transform robustness evaluation into a satisfiability problem (Katz et al., 2017, 2018; Singh et al., 2019), which is solved by the SAT/SMT solver. The main idea of this method is to convert the robustness assessment problem into a logical expression. For a given input point x, varying the local perturbation distance δ (constraint) and determining whether there exists a satisfiable solution under this constraint. The limitation of this method is scalability. As the size of Machine Learning (ML) models and input dimensions increases, solvers need to handle more constraints and variables. SAT/SMT solvers typically employ exhaustive search methods to find solutions, leading to increased solution times. Furthermore, this method can only provide binary conclusions, indicating whether there exists an adversarial sample within a certain range of input perturbation, without offering additional information about the robustness of the model. Theoretical foundations for transforming robustness evaluation into a local Lipschitz constant estimation problem are provided in Weng T W (2018).

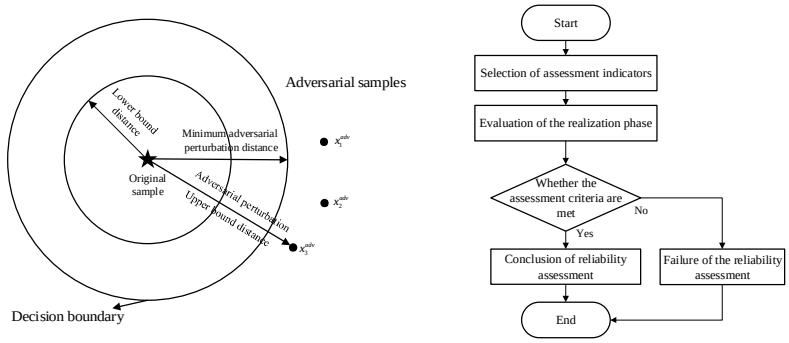

Fig. 1. (a) Adversarial perturbation; (b) Flowchart of white-box testing.

Adversarial attacks represent another commonly used method for robustness evaluation (Bai et al., 2021). Fawzi A et al. (2018); Moosavi-Dezfooli S M et al. (2016); Weng L et al. (2019) use gradient-based approaches to find adversarial samples, considering the computation of minimal perturbation distances. A larger perturbation distance indicates better local robustness of the model, but the search is often computationally expensive. As illustrated in Figure 1 (a), when the perturbation distance is greater than the upper bound distance, it implies that at least one sample with this perturbation has been misclassified by the model. Conversely, when the perturbation distance is less than the lower bound distance, any sample with this perturbation can be correctly classified.

This method can also be extended to the global robustness level. Madry A et al. (2017) introduces the average adversarial risk, while Ruan W et al. (2019); Fawzi A et al. (2018) introduces the average minimal perturbation distance. Essentially, these methods use statistical approaches to assess the local robustness of individual input samples, ensuring that the model can still produce correct outputs when facing the worst-case perturbation for all inputs. However, this requirement is stringent, leading to a lack of generalization ability of the model. Similarly, it does not provide additional information about the model's robustness. Researchers have confirmed that practical NN are difficult to meet such stringent demands (Sinha et al., 2017).

Considering the limitations of the above method, Mangal R et al. (2019) proposed a new concept—probabilistic robustness. Mangal R et al. argue that practical inputs for NN mainly originate from non-adversarial probability distributions, obviating to focus on worst-case. Instead, it suffices to ensure that the NN is robust to at least $(1-\varepsilon)$ probability of the input distribution. Based on the concept of probabilistic robustness, the robustness of the model is assessed by Webb S et al. (2018) through considering the proportion of inputs violating specified system properties, providing an informative concept of how robust the model is. Wang B et al. (2021) evaluated the overall probabilistic robustness of a classifier by averaging the loss function over all possible input perturbation distributions. Zhao X et al. (2021) proposed the concept of probabilistic astuteness, which focused on the accuracy of model output and used operational profiles and robustness verification evidence to quantitatively evaluate model reliability.

Methods based on robustness assessment are a classical approach to evaluating AI safety. Researchers concentrate on defining suitable metrics for robustness assessment and determining acceptable safety levels. Researchers aspire to acquire comprehensive information about the global robustness of models, but often face challenges such as input dimension explosion, long computation times. As AI applications become more widespread, there is a growing desire to obtain information about how safe the model is, enhancing the precision of AI model evaluations.

2.3. Safety assurance case-based method

While testing methods can identify vulnerabilities in AI algorithms, they are unable to traverse all paths within the program. Moreover, an excessive focus on internal program structures and implementation details in testing approaches may lead to the oversight of external factors' impact on the program. Evaluating AI modules through a singular testing method may inadequately reflect the safety of module.

Among the standardization efforts to advance autonomous AI safety analysis and assurance, the Assurance Case (AC) is widely acknowledged by scholars as a pragmatic approach to ensuring AI safety (International Organization for Standardization, 2019; Underwriter Laboratories Inc., 2019). AC is defined as “a structured argument, supported by a body of evidence, that provides a compelling, comprehensible and valid case that a system is safe for a given application in a given environment” (Denney et al., 2018).

Goal Structuring Notation (GSN) (Kelly et al., 2004) and Claims-Argument-Evidence (CAE) (Bishop et al., 2000) are two commonly used representations. GSN places greater emphasis on the hierarchical structure of goals, the dependency relationships of solutions, and their roles in overall system safety. CAE focuses more on logical relationships, emphasizing the logical coherence of arguments. A reasonable choice can be made based on the characteristics of the problem under analysis.

Inspired by Bloomfield R et al. (2009), a safety assurance framework in the development process of ML models is established by delineating the lifecycle of ML models. It meticulously specifies the requirements that ML models should meet at each stage and point out the required proofs of effectiveness, evidence, etc. (Hawkins et al., 2022). Picardi C et al. (2020) provides detailed guidance on generating assurance artifacts supporting safety claims for specific systems at different lifecycles. It also completes the overall structure of the safety assurance framework, offering the necessary direction for executing these activities.

Decomposing high-level system-level specifications into lower-level verifiable specifications understandable by ML models is challenging (Rahimi et al., 2020). A safety assurance case of pedestrian detection at crossroads in autonomous driving exemplifies this difficulty by translating the system-level concept of recognizing pedestrians into a ML-understandable safety requirement, such as "the generated bounding box must not exceed

10% of the minimum bounding box capable of accommodating the entire pedestrian at any size" (Gauerhof et al., 2020). This formulation serves as a reference for subsequent researchers. Additionally, extensive research has emerged in areas such as creating safety arguments (Hawkins et al., 2011) and specifying data management requirements (Ashmore et al., 2021). Hawkins R et al. (2021) present a guidance on the Assurance of Machine Learning in Autonomous Systems (AMLAS), summarizing issues, assurance artifacts, safety requirements, and more encountered in the ML development and deployment process. The ML lifecycle is divided into six stages, with an innovative addition of "feedback and iteration" threads for each stage. When the safety requirements of the latter stage are not met, the previous life cycle stages must be re-examined based on the feedback information, or even that the ML requirements themselves must be reconsidered.

The above assurance frameworks are all static AC. With the constantly update of ML components, fixed static assurance frameworks face certain limitations. Asaadi E et al. (2020) introduces the general framework of Dynamic Assurance Cases (DAC), which captures dynamic assurance metrics with the support of various evidence and safety risk management principles. It conducts safety analysis on the entire life cycle by combining static assurance artifacts and assurance metrics (Denney et al., 2015). Corresponding software for developing assurance caseshas also emerged (Calinescu et al., 2017).

In recent years, there has been widespread attention on AC. The advantage of this method is to ensure that the ML model meets the given safety requirements throughout the entire life cycle by improving the rigorous certification framework. However, the semantic gap between "system requirements and component requirements" or "component requirements and data management requirements.", the challenge of how to generate data that satisfies relevance, accuracy, balance, and completeness requirements and how to understand the erroneous results during model testing and guide them to the insufficiently considered stage ahead are all difficult problems to be solved. Nevertheless, the outlook for development is promising.

3. Application of testing and validation methods

The Intelligent Train Eye (ITE) system shown in Figure 2 is a non-contact obstacle detection system with two operational modes. In the Level-Explore (LE) mode, the system lacks positioning capabilities, featuring only fundamental obstacle detection based on radar point clouds and image deep learning models. In the Level-High Precise (LH) mode, the system incorporates location functionality based on high-precision maps.

Fig. 2. The operational diagram of the ITE system. (Light Rail Vehicle System, 2020)

3.1. Black-box testing

The functional requirements achievable by ITE can be broadly categorized into three aspects. Firstly, in LH mode, obstacle detection relies on high-precision maps and logical maps, enabling train obstacle detection up to 300m, small obstacle detection up to 50m, and pedestrian obstacle detection up to 120m.

The second aspect involves autonomous high-precision positioning, achievable only in LH mode. This functionality should exhibit positioning errors $\leqslant$10cm across various terrains. Lastly, in LH mode, the ITE system is expected to accurately identify signals within the range of 5-70m in front of the train. Simultaneously, the signal's position, ID information, and color details should be detected. This test mainly uses the equivalent partitioning method combined with the boundary value method to design test cases.

The conclusions of the black-box test of the ITE system are as follows: Firstly, the ITE system accurately performs obstacle detection in various scenarios. However, occasional abrupt changes in obstacle detection distance occur when the train is running at high speed, indicating a need for improvement in this aspect. Secondly, the positioning accuracy of the train meets requirements in various scenarios, with occasional

hardware issues that should be addressed. Lastly, concerning signal recognition functionality, the ITE system fulfills the requirements of the system design for each test case.

3.2. White-Box testing

The obstacle detection function in LE mode is implemented using a deep learning model. Therefore, white-box testing methods will be used for testing. This method helps comprehend how the system operates internally, identifying performance bottlenecks, and suggesting optimizations. In contrast, black-box testing lacks this detailed analysis.

The process of white-box testing is shown in Figure 1 (a). Considering the specific functional requirements of the ITE system's obstacle detection, the following evaluation metrics are selected: Leak Detected Rate (LDR), Mistakenly Detected Rate (MDR), Mean Average Precision (mAP), Recall Rate, Precision Rate (Precision), Frames Per Second (FPS), Code Conformity, and Code Vulnerabilities. The research plan for deep model testing involves six steps: (1) Testing with the original dataset; (2) Dataset augmentation and image transformation; (3) Neuron coverage testing; (4) Sensitivity of the model to injected faults; (5) Optimizing the testing dataset; (6) Analyzing result.

In white-box testing, to assess whether LDR and MDR can meet the requirements, our paper utilized a dataset of 35,720 consecutive images fused with LiDAR data. There were 798 frames with trains, resulting in 10 frames of false positives and 7 frames of false negatives. The detection outcomes are presented in Table 1.

Table 1. Testing result of LDR and MDR.

Metric	Result
LDR	0.0087719
MDR	0.00028635

Our paper established a test dataset by collecting video frames from the actual operation of trains. This dataset was used to evaluate the algorithm's mAP, Recall, FPS, and Precision. It consists of images with other trains, pedestrians, or obstacles in front of the train. The dataset comprises a total of 979 images with pedestrians, including 1170 instances; 4303 images with trains, and 634 images with obstacles. The resolution of the dataset is 1280*720. The test results, presented in Table 2, include Recall rate and Precision outcomes at an IoU threshold of 0.5, with results obtained at a confidence threshold of 0.5. Following the testing process, all six metrics for the algorithm's functional implementation meet the required standards, demonstrating the correct functionality of the algorithm.

Table 2. Testing result of mAP, Recall rate, FPS, and Precision.

Metric		Result
mAP		0.9874
Recall	person	0.991845
	train	0.998486
	stop	0.995268
Precision	person	0.99357
	train	0.989975
	stop	0.956625
FPS		26

Finally, a code review is needed. The implementation of the obstacle detection deep learning code involves three main steps: model training, model encapsulation, and model deployment. Initially, for model training, Python code was utilized with PyTorch as the framework. The code adheres to Python Code-style conventions, meeting the specified requirements. The second step involves model encapsulation, where the trained model is converted into an ONNX model. Visual inspection confirmed the model's structure and accuracy match those of the PyTorch model, meeting the requirements. The third step, model deployment, involves converting the ONNX model into a TensorRT model, and ultimately deploying it using C++. The code adheres to the Google C++ code style and demonstrates minimal loss of accuracy. The code meets the specified standards. It has undergone testing using Pprof, valgrind, and QA, achieving a test coverage of over 90%, with no evident vulnerabilities. The code satisfies the requirement for correct implementation.

After the above tests, it can be concluded that the obstacle detection function in LE mode meets the requirements and the test is passed.

3.3. Safety assurance case of ITE system

Applying the AMLAS method to the ITE system and introduce the first three stages in detail.

3.3.1 Stage one: ML safety assurance scoping

(1) *ITE system description*

The ITE system is composed of data acquisition module, perception module and decision-making module.

- Input: real-time video, point cloud data, high-precision map, IMU, MMU and other data.
- Output: The three-dimensional position and attitude of the train, the signal status, the types and distances of all obstacles within a certain range ahead, and the alarm signals.

(2) *Environment description*

The system is installed on the front of the train to ensure the widest possible field of view and scan the track in depth. Trains run on underground tunnels or viaducts, and there are customized code signs, signals, contact lines and other trackside equipment.

(3) *Component description*

In LH mode, the ML component consists of an object detection algorithm, which is used to detect the type and distance of obstacles on the track section in front of the train, and use a bounding box to frame the detected obstacles.

(4) *System security requirements*

The system safety assessment method was used to identify 2 hazards of the ITE system. For each hazard, a number of safety requirements were defined as shown in the Table 3.

Table 3. System Safety Requirement (SSR).

Hazard1- When there are obstacles on the track, the system does not sound an alarm
SSR1-The system should be able to detect train obstacles within 300m
SSR2-The system should be able to detect small obstacles within 50m
SSR3- The system should be able to detect pedestrian obstacles within 120m
SSR4—The point cloud matching between lidar and high-precision map should correctly divide the track segments, and the error should not exceed 1.4m.
SSR5—High-precision maps should cover the entire train line
Hazard2- System false alarm
SSR6- Train positioning accuracy must reach 10cm level
SSR7- False positive rate should be below the specified requirements

The safety requirements assigned to ML components are 1, 2, 3, and 7. SSR4 depends on the point cloud matching algorithm, SSR5 is related to the prior knowledge of system operation, and SSR6 depends on the train positioning component and has nothing to do with the obstacle detection ML component.

3.3.2 Stage two: ML safety requirements assurance

Based on the ML component system safety requirements defined in stage one, this stage obtains a set of component safety requirements suitable for ML component implementation and verification. According to the AMLAS guidelines, ML safety requirements (MLSR) include performance and robustness requirements.

The rationale for MLSR are as follows:

MLSR1 is derived from system requirement SSR1. Similarly, SSR2-MLSR2, SSR3-MLSR3 are all based on project requirements.

MLSR4, MLSR5 are derived from system requirements SSR1, 2, 3. MLSR9 is derived from SSR8. The above component-level requirements are proposed based on the "Artificial Intelligence Deep Learning Algorithm Evaluation Specification (2018)" and combined with the specific functional requirements of ITE system obstacle detection.

MLSR6 is used to ensure the continuity of object detection. Lee J et al. (2022) mentioned that when detecting very fast objects, the application requires a high speed of 10 FPS or more, but for applications that detect slower objects, detecting objects at 2~3 FPS is sufficient. The video frame rate of the ITE system is 30fps. Considering the model deployment limitations, it is believed that the system can continuously detect objects in 5 frames.

MLSR7 is used to ensure the sensitivity of target detection. Due to camera installation limitations, pedestrians may not be fully exposed to the camera's field of view, but they do exist in reality and should be detected (Hamnert et al., 2022).

MLSR8-When encountering an obstacle that the module cannot recognize, the module should promptly alarm and guide manual intervention to avoid dangers caused by unknown obstacles or ML module detection errors.

MLSR10 is used to ensure the correctness of components in the operating domain.

Table 4. ML Safety Requirement (MLSR).

Performance	MLSR6- In the image sequence input from video, any detected object should not be missed by more than 1 in 5 frames.
MLSR1- The component should be able to detect the train ahead within 300m, including minimum and maximum sizes.	MLSR7- The ML component should log when a pedestrian's bounding box enters 50% or more of the detection area.
MLSR2- The component should be able to detect small obstacles ahead within 50m, including minimum and maximum sizes.	MLSR8—When the component recognizes an uncertain obstacle, an alarm should be issued in advance and the cause of the alarm should be communicated, so that humans can intervene in a timely manner.
MLSR3- The component should be able to detect pedestrian obstacles within 120m, including minimum and maximum sizes.	***Robustness***
MLSR4-The LDR is less than or equal to 1%.	MLSR9- The MDR is less than or equal to 1%.
MLSR5-The precision of obstacle recognition is greater than 99%, and the mAP is greater than 90%.	MLSR10- All input samples should meet ML performance requirements within the class scope determined in the operational domain.

3.3.3 Stage three: data management

Table 5. Data Safety Requirement (DSR).

Relevance	***Completeness***
DSR1- Sample collection of key features should reflect the general situation in the country where the system will operate.	DSR5- The data set needs to include all combinations of feature dimensions in the operating domain.
DSR2- Images of areas outside the defined expected operating range should not be included in the dataset.	DSR6- The data set must include situations without obstacles.
DSR3- The format of each data sample should represent an image captured using sensors deployed on the train	***Accuracy***
DSR4- The data should include the main features to be detected.	DSR7- All 2D boxes generated must be large enough to contain the entire obstacle.
Balance	DSR8- All generated 2D boxes do not exceed 10% of the smallest size box that can contain the entire obstacle.
DSR10- The distribution between classes in each dataset should be reasonable.	DSR9- All obstacles present in the data sample must be correctly labeled

ML data requirements specify the relevance, completeness, accuracy, and balance of the data. Data requirements are shown in the Table 5. The rationale for MLSR are as follows:

DSR1-Since different countries have different rail line layouts and environments, it is necessary to collect data samples of the general characteristics of the place where the system operates.

DSR2 - Dataset collection should be restricted to the operational domain.

DSR3- The sensor will provide image, depth and other information to the ML component in a specific format. Therefore, only information in this format should be used during model development.

DSR4-Guarantee the validity of data collection.

DSR5 - Ensure that the dataset includes combinations of features.

DSR6-The ML components need to avoid false positives, so the dataset must include normal operating scenarios.

DSR7-The 2D bounding box needs to be large enough to ensure that no obstacle is missed.

DSR8-The 2D bounding box should not be too large, and it is necessary to ensure the accuracy of obstacle position recognition. (Gauerhof et al., 2020)

DSR9- Ensure the validity of training data.

DSR10-Avoid component performance deviation on different data.

We consider components that meet all of the above safety requirements to be acceptably safe when deployed in a system. When safety requirements are not satisfied, a bottom-up inspection of each step can be carried out

through the examination of development logs, validation logs, testing logs, etc. This approach imparts a degree of traceability to the testing and assessment results. In general, the safety assurance framework ensures that the safety assessment process for components is no longer isolated. Meanwhile, a more comprehensive assurance of overall safety is achieved by combining various testing methods and evidence.

4. Conclusions

This paper introduces the safety assessment and verification techniques of AI module from three perspectives: test-based methods, robustness evaluation-based methods, and safety assurance case-based methods. The current research is relatively mature, but there is a lack of clear explanation of what standards the test or robustness indicators should reach when the system is in an acceptable safety state.

Furthermore, the following issues also need to be considered by researchers:

- In addition to paying attention to the final test results when testing the system, we should also focus on whether the test results exhibit a progressive logic based on variables such as distance, light, and obstruction. A safe system should avoid mutations, hopping behaviours.
- Under the safety assurance framework, testing-improvement-evaluation can be combined to form a closed loop. Testing finds problems to guide system improvement. Evaluation process are used to judge whether the proposed improvement plan is effective, and then continue testing to discover more problems.
- During the training of the ML model, attention should be paid to whether the training data matches the actual operating domain, whether the relevance, accuracy, comprehensiveness, and balance of the data meet the requirements, and corresponding proofs should be given. If the training data of the ML model does not meet the standards, it will be difficult to guarantee the safety of the system no matter how many safety tests and verification methods are used.

Acknowledgements

Research supported by National Natural Science Foundation of China Regional Innovation Joint Fund Key Project (U22A2053), Central Universities Basic Research Business Fund Project (2022JBZY024) and Guangxi Key R&D Project (Gui scientificAB 22035008).

References

Asaadi E, Denney E, Menzies J, et al. 2020. Dynamic assurance cases: a pathway to trusted autonomy. Computer 53(12), 35-46.

Ashmore R, Calinescu R, Paterson C. 2021. Assuring the machine learning lifecycle: Desiderata, methods, and challenges. ACM Computing Surveys (CSUR), 54(5), 1-39.

Bai T, Luo J, Zhao J, et al. 2021. Recent advances in adversarial training for adversarial robustness. arXiv preprint arXiv:2102.01356.

Bishop P, Bloomfield R. 2000. A methodology for safety case development. In Safety and Reliability 20, 1, 34-42.

Bloomfield R, Bishop P. 2009. Safety and assurance cases: Past, present and possible future–an Adelard perspective. In Making Systems Safer: Proceedings of the Eighteenth Safety-Critical Systems Symposium, 51-67.

Cadar C, Ganesh V, Pawlowski P M, et al. 2008. EXE: Automatically generating inputs of death. ACM Transactions on Information and System Security (TISSEC), 12(2), 1-38.

Calinescu R, Weyns D, Gerasimou S, et al. 2017. Engineering trustworthy self-adaptive software with dynamic assurance cases. IEEE Transactions on Software Engineering, 44(11), 1039-1069.

De Bruin, T., Verbert, K., Babuška, R. 2016. Railway track circuit fault diagnosis using recurrent neural networks. IEEE transactions on neural networks and learning systems 28(3), 523-533.

Denney E, Pai G, Habli I. 2015. Dynamic safety cases for through-life safety assurance. In 2015 IEEE/ACM 37th IEEE International Conference on Software Engineering, Vol. 2, 587-590.

Denney E, Pai G. 2018. Tool support for assurance case development. Automated Software Engineering, 25(3), 435-499.

Fawzi A, Fawzi O, Frossard P. 2018. Analysis of classifiers' robustness to adversarial perturbations. Machine learning, 107(3), 481-508.

Gauerhof L, Hawkins R, Picardi C, et al. 2020. Assuring the safety of machine learning for pedestrian detection at crossings. In Computer Safety, Reliability, and Security: 39th International Conference, 197-212.

Gopinath D, Katz G, Păsăreanu C S, et al. 2018. Deepsafe: A data-driven approach for assessing robustness of neural networks. In Automated Technology for Verification and Analysis: 16th International Symposium, 3-19.

Gopinath D, Wang K, Zhang M, et al. 2018. Symbolic execution for deep neural networks. arXiv preprint arXiv:1807.10439.

Hamnert J, Hägglund D. 2022. How Safe Is Machine Vision?: An Evaluation of the AMLAS Process in a Machine Vision Environment.

Hampel, F. R. 1971. A general qualitative definition of robustness. The annals of mathematical statistics, 42(6), 1887-1896.

Hawkins R, Kelly T, Knight J, et al. 2011. A new approach to creating clear safety arguments. In Advances in Systems Safety: Proceedings of the Nineteenth Safety-Critical Systems Symposium, 3-23.

Hawkins R, Paterson C, Picardi C, et al. 2021. Guidance on the assurance of machine learning in autonomous systems (AMLAS. arXiv preprint arXiv:2102.01564.

Hawkins R, Picardi C, Donnell L, et al. 2022. Creating a safety assurance case for an ML satellite-based wildfire detection and alert system. arXiv preprint arXiv:2211.04530.
He, D., Zou, Z., Chen, Y., et al. 2021. Rail transit obstacle detection based on improved CNN. IEEE Transactions on Instrumentation and Measurement, 70, 1-14.
Huang W, Sun Y, Zhao X, et al. 2021. Coverage-guided testing for recurrent neural networks. IEEE Transactions on Reliability, 71(3), 1191-1206.
Huang X, Kwiatkowska M, Wang S, et al. 2017. Safety verification of deep neural networks. In Computer Aided Verification: 29th International Conference, Part I 30, 3-29.
Huang, X. 2021. Safety and reliability of deep learning: (brief overview). In Proceedings of the 1st International Workshop on Verification of Autonomous & Robotic Systems, 1-2.
International Organization for Standardization, 2019), "Road vehicles — Safety of the intended functionality," Standard ISO/PAS 21448:2019.
Katz G, Barrett C, Dill D L, et al. 2017. Reluplex: An efficient SMT solver for verifying deep neural networks. In Computer Aided Verification: 29th International Conference, Part I 30, 97-117.
Kelly T, Weaver R. 2004. The goal structuring notation–a safety argument notation. In Proceedings of the dependable systems and networks 2004 workshop on assurance cases, Vol. 6.
King, J. C. 1976. Symbolic execution and program testing. Communications of the ACM, 19(7), 385-394.
Lee J, Hwang K. 2022. YOLO with adaptive frame control for real-time object detection applications. Multimedia Tools and Applications, 81(25), 36375-36396.
Light Rail Vehicle System[EB/OL]. 2020-04-23)[2024-02-28]. https://railvision.devly.me/light-rail-vehicle-system/.
Ma L, Juefei-Xu F, Zhang F, et al. 2018. Deepgauge: Multi-granularity testing criteria for deep learning systems. In Proceedings of the 33rd ACM/IEEE international conference on automated software engineering, 120-131.
Madry A, Makelov A, Schmidt L, et al. 2017. Towards deep learning models resistant to adversarial attacks. arXiv preprint arXiv:1706.06083.
Mangal R, Nori A V, Orso A. 2019, May. Robustness of neural networks: A probabilistic and practical approach. In 2019 IEEE/ACM 41st International Conference on Software Engineering: New Ideas and Emerging Results (ICSE-NIER), 93-96.
Moosavi-Dezfooli S M, Fawzi A, Frossard P. 2016. Deepfool: a simple and accurate method to fool deep neural networks. In Proceedings of the IEEE conference on computer vision and pattern recognition, 2574-2582.
Pei K, Cao Y, Yang J, et al. 2017. Deepxplore: Automated whitebox testing of deep learning systems. In proceedings of the 26th Symposium on Operating Systems Principles, 1-18.
Picardi C, Paterson C, Hawkins R D, et al. 2020. Assurance argument patterns and processes for machine learning in safety-related systems. In Proceedings of the Workshop on Artificial Intelligence Safety, 23-30.
Rahimi M, Guo J L C, Kokaly S, et al. 2019. Toward requirements specification for machine-learned components. In 2019 IEEE 27th International Requirements Engineering Conference Workshops, 241-244.
Ruan W, Wu M, Sun Y, et al. 2019. Global robustness evaluation of deep neural networks with provable guarantees for the hamming distance. IJCAI-19.
Singh G, Gehr T, Püschel M, et al. 2019. An abstract domain for certifying neural networks. Proceedings of the ACM on Programming Languages, 3(POPL), 1-30.
Sinha A, Namkoong H, Duchi J C. 2017. Certifiable Distributional Robustness with Principled Adversarial Training. CoRR, abs/1710.10571. arXiv preprint arXiv:1710.10571.
Sun Y, Cao Y, Li P. 2022. Contactless fault diagnosis for railway point machines based on multi-scale fractional wavelet packet energy entropy and synchronous optimization strategy. IEEE Transactions on Vehicular Technology, 71(6), 5906-5914.
Sun Y, Huang X, Kroening D, et al. 2019. Structural test coverage criteria for deep neural networks. ACM Transactions on Embedded Computing Systems (TECS), 18(5s), 1-23.
Sun Y, Wu M, Ruan W, et al. 2018. Concolic testing for deep neural networks. In Proceedings of the 33rd ACM/IEEE International Conference on Automated Software Engineering, 109-119.
Tian Y, Pei K, Jana S, et al. 2018. Deeptest: Automated testing of deep-neural-network-driven autonomous cars. In Proceedings of the 40th international conference on software engineering, 303-314.
Underwriter Laboratories Inc., 2020. Standard for Safety for the Evaluation of Autonomous Products UL 4600.
Wang B, Webb S, Rainforth T. 2021. Statistically robust neural network classification. In Uncertainty in Artificial Intelligence, 1735-1745.
Webb S, Rainforth T, Teh Y W, et al. 2018. A statistical approach to assessing neural network robustness. arXiv preprint arXiv:1811.07209.
Weng L, Chen P Y, Nguyen L, et al. 2019. PROVEN: Verifying robustness of neural networks with a probabilistic approach. In International Conference on Machine Learning, 6727-6736.
Weng T W, Zhang H, Chen P Y, et al. 2018. Evaluating the robustness of neural networks: An extreme value theory approach. arXiv preprint arXiv:1801.10578.
Wicker M, Huang X, Kwiatkowska M. 2018. Feature-guided black-box safety testing of deep neural networks. In Tools and Algorithms for the Construction and Analysis of Systems: 24th International Conference, Part I 24, 408-426.
Xie X, Ma L, Juefei-Xu F, et al. 2019. Deephunter: a coverage-guided fuzz testing framework for deep neural networks. In Proceedings of the 28th ACM SIGSOFT international symposium on software testing and analysis, 146-157.
Yu, M., Yang, P., Wei, S. 2018. Railway obstacle detection algorithm using neural network. In AIP Conference Proceedings 1967(1). AIP Publishing.
Zhang M, Zhang Y, Zhang L, et al. 2018. Deeproad: Gan-based metamorphic testing and input validation framework for autonomous driving systems. In Proceedings of the 33rd ACM/IEEE International Conference on Automated Software Engineering, 132-142.
Zhang Q, Lusby R M, Shang P, et al. 2022. A heuristic approach to integrate train timetabling, platforming, and railway network maintenance scheduling decisions. Transportation Research Part B: Methodological, 158, 210-238.
Zhao X, Huang W, Bharti V, et al. 2021. Reliability assessment and safety arguments for machine learning components in assuring learning-enabled autonomous systems. arXiv e-prints, arXiv-2112.
Zhou W, Huang Y, Deng L, et al. 2023. Collaborative optimization of energy-efficient train schedule and train circulation plan for urban rail. Energy, 263, 125599.