Federal Office
for Information Security

AIMobilityAuditPrep

Final Results – Documentation

Document History

Version	*Date*	*Editor*	*Description*
0.1	30.05.2022	Fabian Woitschek	Initial creation
0.2	24.06.2022	Fabian Langer	Add AP3 documentation
0.3	27.06.2022	Fabian Langer	Include additional data requirements in AP3 (Req. 28, 29, 30, 31 & 43)
0.4	28.06.2022	Fabian Woitschek	Use justification in AP3 documentation Unify caption formatting Add AP2 and AP4 documentation
0.5	22.07.2022	Fabian Woitschek	Add AP5 documentation
1.0	12.08.2022	Fabian Woitschek	Include feedback from BSI
1.1	07.09.2022	Thora Markert	Include feedback from BSI
1.2	14.09.2022	Thora Markert	Include feedback from BSI
1.3	09.11.2022	Fabian Woitschek	Add author list
1.4	17.11.2022	Fabian Woitschek	Add disclaimers

Federal Office for Information Security
P.O. Box 20 03 63
53133 Bonn
Internet: https://www.bsi.bund.de

Authors

The report was prepared by the companies ZF Friedrichshafen AG and TÜV Informationstechnik GmbH. The authors of the report are the following members from the companies:

- ZF Friedrichshafen AG
  - Fabian Woitschek
  - Devi Padmavathi Alagarswamy
  - Dr. Georg Schneider
- TÜV Informationstechnik GmbH
  - Thora Markert
  - Fabian Langer
  - Vasilios Danos

To contact the authors, please use firstname.surname@zf.com or initial_letter_firstname.surname@tuvit.de.

Disclaimer

Nowadays, AI-based systems are becoming increasingly popular in the automotive industry (e. g. driving assistance systems, autonomous driving functions etc.). However, appropriate standards, methodologies and requirements to cover the AI-specific risks are still missing. The aim of this project is to develop generic requirements, corresponding test methods and tools to assess such risks of AI-based systems. Due to the complexity and the wide range of applications, the scope of this work is limited to certain aspects in the automotive context. In selected use-cases the previously defined requirements are evaluated and tested as a proof-of-concept. Hence, the results shall be considered as preliminary and as a guidance for best practices. Furthermore, future development and additional use-cases are needed to perform adjustments and continue the specification of these requirements.

Preliminary Remark

The document is based on the technically oriented and work package (in German “Arbeitspaket” or short “AP”) based documentation of the project. Therefore, you will find references to the work packages AP 1-7 throughout the document. These serve internal purposes and can be ignored by the reader.

Table of Contents

1 Introduction

AI-based systems are increasingly used as part of AD/ADAS systems. Especially, DNNs achieve an impressive performance on most task and are the most promising solution to achieve higher level of autonomous driving. At the same time, different manufacturers already use DNN-based solutions as part of L2 ADASs that are operating on public roads. However, current DNNs introduce new and specific vulnerabilities into the systems which can impact the performance of AD/ADAS systems negatively. This requires a detailed analysis of existing vulnerabilities and potential mitigation strategies. To still enable the usage of such DNN-based solutions for high-risk applications (like L4/L5 autonomous driving) clear guidelines and regulations are required. This assures that systems with a high degree of autonomy are of a high quality and include mitigation strategies to known vulnerabilities. However, currently no regulations or standards exist that are tailored towards the use of AI-based systems for AD/ADAS and include AI specific vulnerabilities.

To increase the required knowledge in the areas of risk mitigation and auditing the goal of this project “Vorbereitung der Erprobung und Weiterentwicklung von Anforderungen an KI-Systeme anhand praktischer Use-Cases im Bereich Mobilität (AIMobilityAuditPrep)” is to explore how auditing guidelines can ensure the IT-Security of AI-based AD/ADAS systems for high-risk application areas. This project (in combination with the follow-up project “Erprobung und Weiterentwicklung von Anforderungen an KI-Systeme anhand praktischer Use-Cases im Bereich Mobilität (AIMobilityAudit)”) lays the foundation of a technical guideline which should be contributed to national and international standardization committees.

This document is the final report of work package six “Dokumentation, Publikation & Präsentation Ergebnisse, Handlungsempfehlung für Folgeprojekt " of the current project. Hence, it contains the results of the previous work packages two to five which are already available in the individual reports (1), (2), (3) and (4). Therefore, we cover the SOTA literature review, the creation of general auditing guidelines, the suitability analysis of use cases and the creation of a toolchain and toolbox for training and auditing DNNs. Here, we combine the individual reports using a common terminology and provide a joint summary and outlook.

2 State-of-the-Art Report (AP2)

This chapter is the final report of work package two “Erstellung State-of-the-Art-Dokument” of the project “Vorbereitung der Erprobung und Weiterentwicklung von Anforderungen an KI-Systeme anhand praktischer Use-Cases im Bereich Mobilität (AIMobilityAuditPrep)”. Hence, it contains the results of the research on the State-of-the-Art of the topics that are described in the service description. These consist of the following:

- The entire AI-Lifecycle, from the planning phase, through the data collection and data quality assurance phase, the training phase, the evaluation phase and the operating phase
- At least the following relevant aspects: IT security, safety, robustness, explainability and documentation
- Relevant use cases from the mobility sector
- The role of simulation and synthetic data
- The integration of AI systems in an overall system, which typically consists of different software and hardware components and is embedded in an environmental context
- Strategies and tools for the mitigation
- National and international activities for standardization/auditing
- Current scientific developments from conferences and journals

2.1 AI Lifecycle

The entire artificial intelligence (AI) lifecycle is considered only by a limited amount of research work. Instead, in most cases only individual steps in the entire lifecycle are considered. Still, in the following we present a selection of the most interesting and relevant publications to cover the AI lifecycle from the perspectives of different stakeholders.

In (5) the authors present a detailed survey of all technologies surrounding an AI system and highlight many technologies that are involved in an AI system. They cover the complete lifecycle from data collection to human-machine interfaces and present relevant tasks of each step in the lifecycle. For these steps, they introduce the basic algorithmic ideas and present current advances from the research. Additionally, hardware technologies are discussed that are used for data collection, performant computing and deployment. Different application domains are considered, and practical examples are presented.

The authors in (6) describe the AI lifecycle from the viewpoint of a maturity framework for enterprises. The considered AI lifecycle is shown in Figure 1 and starts at the very beginning with the setting of goals and the definition of business use cases. Then, the general steps of data collection, feature preparation, model training, model evaluation and deployment are considered. For each step, the authors describe the concrete tasks that need to be performed and provide best practices for companies from their experience. The developed framework to measure the overall maturity of the AI lifecycle management is derived from the software capability maturity model introduced in (7).

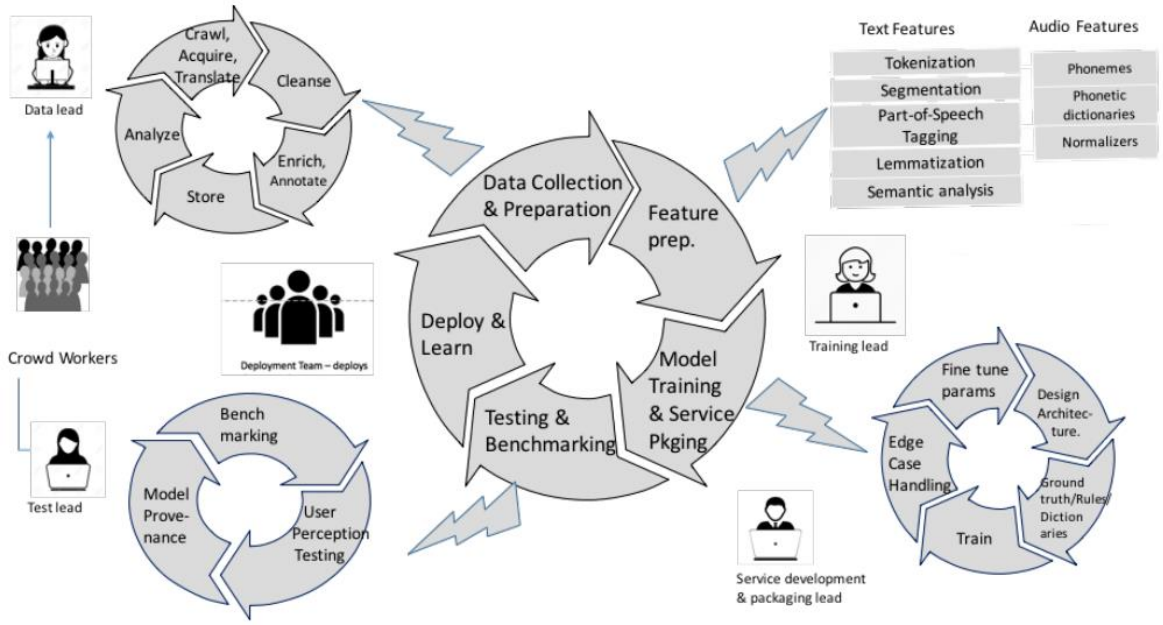


Figure 1 Overview of the AI lifecycle from (6)

Furthermore, work exists that looks at the AI lifecycle from the viewpoint of trustworthy AI. In (8) the authors perform an extensive survey on different aspects of trustworthiness at each step in the entire AI lifecycle. They cover important aspects, like robustness, generalization, explainability and transparency that are also considered in this project. Additionally, they also cover the aspects of fairness and alignment of AI systems with human values so that the entire topic of trustworthiness is considered. In Figure 2 an overview of a subset of the considered aspects and the categorization in the AI lifecycle is shown. All mentioned properties are described in detail and the existing challenges at each step in the AI lifecycle are discussed. The authors also provide concrete actions for practitioners at each step that are taken from current advances in the research. In general, this publication seems to have a similar focus than our project and hence covers the same aspects. A very similar viewpoint is also covered by the authors in (9). The complete AI lifecycle is considered, as well as the integration of AI systems in larger IT systems. Vulnerabilities at each point in the lifecycle are discussed and promising recently published mitigations strategies are described.

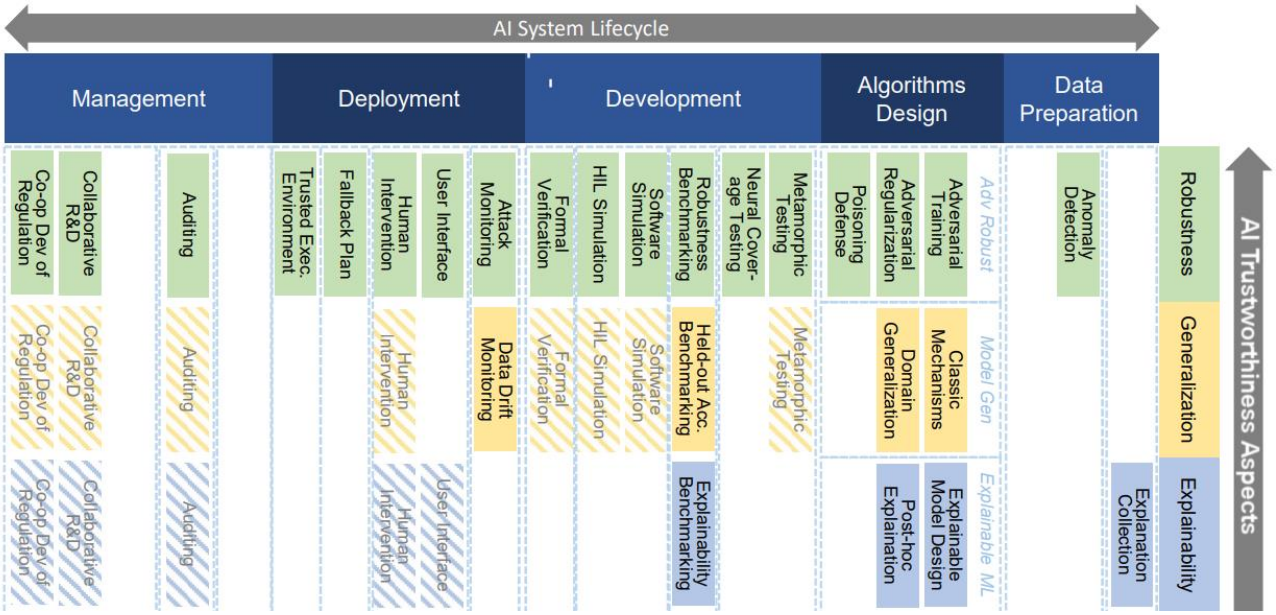


Figure 2 Partial overview of some aspects of AI trustworthiness depending on the steps of the AI lifecycle from (8)

Finally, the last viewpoint is based on the cloud lifecycle management of AI systems. In (10) the authors focus on different tools that can support the management of AI systems and track the complete lifecycle. They base

their work on the principles of software lifecycle management and show how these can be adapted to the AI lifecycle. This enables the reliability, traceability and reproducibility of the AI development process. Concrete software tools that enable the described AI lifecycle tracking are presented later in Chapter 2.3.4.2 as part of the chapter on the documentation of AI systems.

Additionally, the publications (11) and (12) look at the AI lifecycle from the viewpoint of documentation. For internal documentation, different report structures at each point in the AI lifecycle are proposed that can also be handed to third parties for external auditing. We discuss these publications in greater detail in Chapter 2.3.4.1.

2.2 Challenges of AI Systems

The use of AI-based systems leads to unique challenges that do not exist to the same extent for traditional algorithmic IT systems. In comparison some properties of AI-based systems complicate the auditing process to ensure the safe usage in high-risk applications like autonomous driving (AD). Especially, the more complex lifecycle discussed in Chapter 2.1, the often very large input and system parameter space, the high data dependency and the black-box property of AI systems pose new challenges. In this chapter an overview of all arising challenges is given before we present potential mitigation strategies for each challenge in Chapter 2.3. Additionally, it is important to note that the strict assignment of challenges to the following subchapters is not uniquely defined. There exist overlaps between the individual challenges discussed in the remainder of this chapter and depending on the viewpoint some aspects could also be assigned to other subchapters.

2.2.1 IT Security

In this chapter, an overview of IT security threats on AI-based systems is given. An adversary's goal is to compromise the confidentiality, integrity, availability and privacy of the system (13). Most IT security attacks pose a threat on the integrity and availability of a model. Here, evasion attacks and data poisoning attacks are attack methods that emerged for data-driven systems, which do not exist to the same extent for traditional IT systems. Both attack methods are also strongly related to the robustness of an AI-based system which is discussed in Chapter 2.2.2. In contrast, threats on data privacy and confidentiality are excluded from the scope of this project and therefore will not be addressed in detail. Namely, model extraction, membership inference and model inversion attacks, pose a threat on the confidentiality and the privacy of the system. However, model stealing attacks can also be used to enhance evasion attacks and due to the transferability property of adversarial examples these methods can also pose a threat on the integrity and availability of a system (13).

2.2.1.1 Model Extraction Attacks

Model extraction attacks attempt to copy the functionality of a black-box victim model. The general concept of such attacks can be described in two steps. First, a victim model is queried with data chosen by the adversary and then a surrogate model is trained on the data and the queried predictions with the objective to be as similar as possible to the victim model. These attacks can extract a white-box model similar to the victim model, which enables the adversary to craft white-box adversarial attacks without any environmental or domain constraints. Due to the before mentioned transferability property of adversarial attacks, the crafted adversarial examples can be used to impose a threat on the integrity and availability of the victim model. The attack success of the transferred adversarial examples depends on the similarity between the extracted model and the victim model as discussed in (14).

Basic model extraction attacks utilize publicly available datasets and models that similar to typical architectures used in the victim's domain to create the extracted model. Depending on the dataset, domain and the model architecture this can result in a high difference between the victim model and the extracted model. Recently, works have emerged to enhance these model extraction methods through information extraction attacks, to increase the similarity between the two models.

Since the attack success of adversarial attacks on the victim model depends on the similarity between the surrogate and victim model, the aim of an attacker is to gain more knowledge of the model architecture and

training of the victim model. Therefore, recent research demonstrates that combining information extraction attacks of the model architecture or the training data enhances model extraction attacks.

The authors of (15) successfully infer the architecture of a black-box model by snooping information at the PCIe bus and memory bus. They use a three-step method described in Figure 3 to obtain the victim model's architecture. During the first "Run-time-Layer Sequence Identification"-step, they train a long short-term memory (LSTM) connectionist temporal classification network to predict each layer type of the victim model based on the kernel features during run-time. In the next step, they reconstruct the connection between the extracted layers through the read-after-write access patterns for each layer. Finally, they use the read and write volumes to estimate the parametrization of each layer. The authors are able to construct a surrogate model with a high similarity to the victim model and additionally achieve an 50,4% increase of the attack success rate compared to model extraction methods based on publicly available model architectures.

In the publication (16) the authors demonstrate that data extraction attacks can enhance model extraction as well. Concretely, they utilize a model inversion attack to improve the training phase of their surrogate model. They initiate a substitute model by querying a victim model based on data from a public dataset. To improve the substitute model, they select data samples from the public dataset where the victim model has a high confidence score and inverse them using an autoencoder like the one designed in (17). Finally, they retrain the surrogate model on the inversed data samples and the predictions from the victim model. The inversed samples have resemblance with the training data of the victim model, which enables the surrogate model to learn features similar to the victim model. With this approach, they are able to achieve a higher agreement with the victim model than other SOTA model extraction attacks. Additionally, they can evade the SOTA defense method from (18), which detects suspicious model queries linked to model extraction attacks.

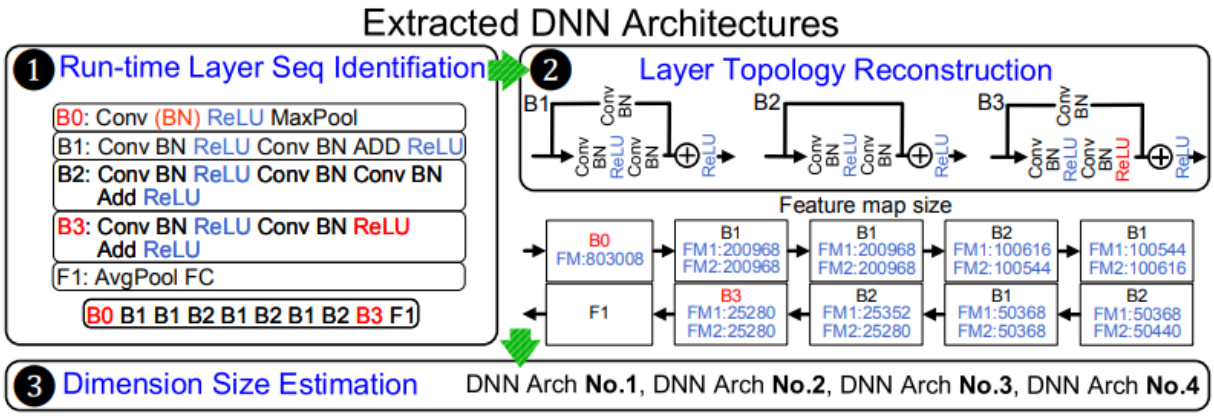


Figure 3 Illustration of the steps necessary to extract model architectures from (15)

2.2.1.2 Evasion Attacks

In this chapter an overview over current state-of-the-art (SOTA) evasion attacks within the autonomous driving domain is given. Evasion attacks, also known as adversarial attacks, are carefully perturbed input samples (adversarial examples) that change the prediction of AI-based systems according to the adversary's will. This imposes a threat on the integrity and availability of the system. Therefore, evasion attacks pose a significant threat for safety-critical systems such as autonomous driving systems, as a wrong prediction could result in fatal accidents. Adversarial attacks are a highly researched attack type, especially with a focus on DNNs within the image domain, which results in a vast variety of attacks with high attack success rates. Another crucial threat imposed by adversarial attacks is their transferability. As explained in Chapter 2.2.1.1, adversarial examples calculated for one model could be transferred to another with a high attack success.

2.2.1.2.1 Categorization

Figure 4 gives an overview over the different types of evasion attacks. Within the autonomous driving domain, there are two attack types that should be considered: digital adversarial attacks and physical adversarial attacks. Digital adversarial attacks are attacks that aim to fool the AI-based model within the autonomous driving system by digitally perturbing an image that is given directly as input to the model. These attacks can be further categorized into three types: gradient-based attacks, optimization-based attacks and generative-based attacks.

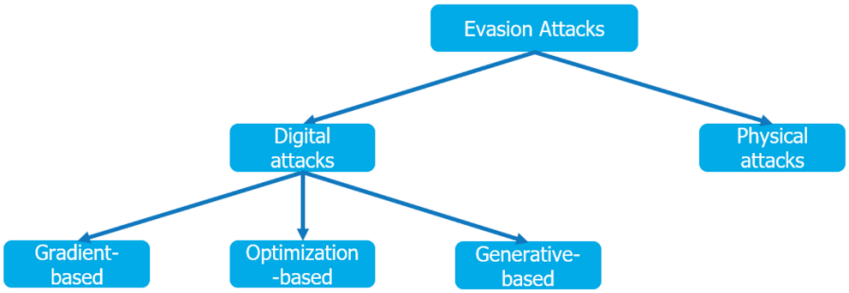


Figure 4 Categorization of evasion attacks

Since digital attacks are not realistic within the autonomous driving domain, a lot of research emerged applying adversarial attacks to a physical and real-world setting, for example in (19). Physical adversarial attacks aim to project adversarial perturbations into a physical scene that is captured by the cameras or other sensors of the AD system, rather than perturbing single inputs that are fed to the AI-based model directly (20). Additionally, an evasion attack is either designed to have black-box or white-box access to the model, meaning the adversary has either limited knowledge or full knowledge about the model. Even though white-box attacks pose a significant threat to an AD system, for these systems black-box attacks are more feasible as discussed in (20). Furthermore, there are evasion attacks that can target a specific class for the misclassification. In this case, the attack optimizes the worst-case perturbation towards being misclassified as a target class. On the other hand, if the attack aims to achieve a general misclassification of the input towards any class different from the true class of the input, the attack is described as untargeted.

2.2.1.2.2 Gradient-based Adversarial Attacks

Gradient-based adversarial attacks generate perturbations based on the gradients of the model. Hence, these attacks require white-box access to the model. Two prominent examples of such attacks are the fast gradient sign method (FGSM) introduced in (21) and the projected gradient descent (PGD) attack proposed in (22).

FGSM calculates the optimal perturbation for a misclassification based on the direction (i.e., the sign) of the adversarial loss gradient. Figure 5 shows an example of a FGSM attack results on a 3D object detection system within the AD domain from (23). Since FGSM is a well-studied and well-performing attack, many attacks have been derived based on its concept. These include the IT-FGSM (24), the basic iterative method (BIM) (23) and the PGD attack.

Figure 5 FGSM attack results on a 3D object detection system from (23). The images on the left contain the clean stereo image with the result of the 3D object detection underneath. The images on the right show the FGSM perturbed stereo image and the corresponding 3D object detection result underneath.

The PGD attack randomly initializes the perturbation, updates it based on the direction of the adversarial loss and projects it back into the perturbation budget of the adversary. Figure 6 gives an example of a PGD attack on a 3D object detection system within the AD domain from (23).

Figure 6 PGD attack results on a 3D object detection system from (23). The images on the left contain the clean stereo image with the result of the 3D object detection underneath. The images on the right show the PGD perturbed stereo image and the corresponding 3D object detection result underneath.

2.2.1.2.3 Optimization-based Adversarial Attacks

Optimization-based adversarial attacks calculate adversarial examples formulated as an optimization problem. They aim to minimize the distance between the adversarial example and the original image, while achieving a misclassification of the adversarial example.

In (25) the authors introduce the DeepFool attack. It aims to solve the optimization problem by starting with the assumption that a DNN is linear and finding an optimal solution for this simplified assumption. Because

DNNs are not linear in practical applications, the authors adapt their solution to fit this non-linearity. Further, DeepFool is a non-targeted attack, i.e., it aims to find an adversarial example that produces a misclassification as any other class. Figure 7 shows examples for adversarial attacks created with the DeepFool attack.

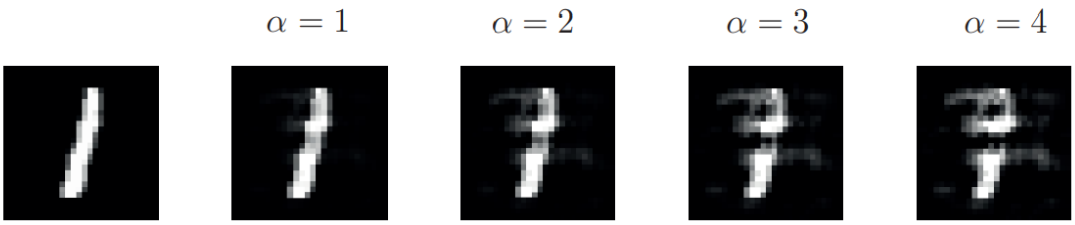


Figure 7 DeepFool attack results on MNIST images with different attack configurations from (25)

A prominent white-box optimization-based attack is the Carlini-Wagner (C&W) attack. This attack is introduced in (26) and finds the smallest perturbation via minimization and box constraints. The attack is designed to break SOTA defensive distillation techniques, which will be further discussed in Chapter 2.3.1.2.3. Additionally, the authors are able to show that this attack performs well in a transferability setting and therefore suggest that it can be utilized in black-box settings. Figure 8 shows L_2 adversarial examples generated for MNIST with different target labels for misclassifications.

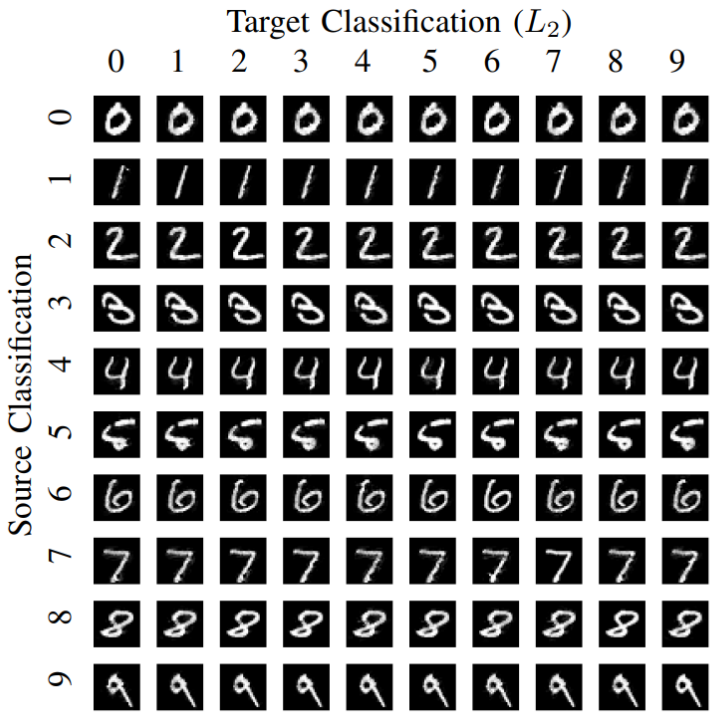


Figure 8 Adversarial examples crafted with the C&W attack on MNIST from (26)

2.2.1.2.4 Generative-based Adversarial Attacks

Generative-based attacks generate adversarial examples using generative adversarial networks (GANs) proposed in (27). The authors of (28) propose a framework called AdvGAN that utilizes a GAN with a discriminator to create visually imperceptible adversarial examples from an original image within a white-box and black-box setting. Figure 9 shows adversarial examples created by AdvGAN for both settings on the CIFAR-10 dataset introduced in (29). This method is further extended in (30) by generating adversarial examples based on latent features rather than an input image.

Figure 9 Adversarial examples created by AdvGAN on MNIST from (28)

In (31) GANs are used to produce image-dependent and universal perturbations. The image-dependent perturbations, displayed in Figure 10, are generated to resemble the natural image as much as possible and fool a specific target model. Whereas the universal perturbations, displayed in Figure 11 are generated to fool multiple target models at once. Both methods were compared to other adversarial attack methods in (24). In a white-box setting they are able to achieve high attack success rates on three different models trained on the (32) dataset containing real-world road images. However, with black-box access to the models and the dataset the attack did not perform well with attack success rates of only $0{,}1\% - 8{,}4\%$ on the same model.

Figure 10 Image-dependent adversarial examples from (31)

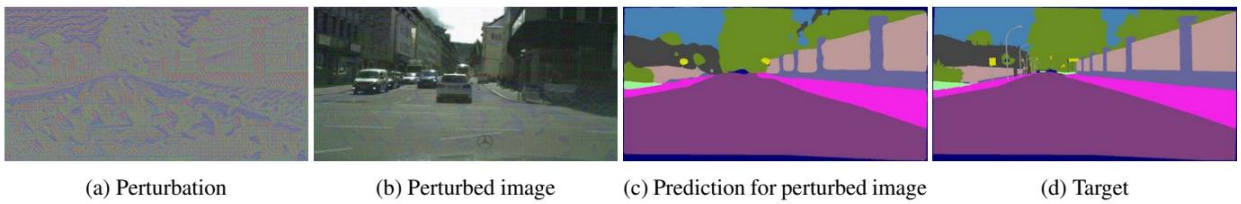


Figure 11 Universal adversarial examples from (31)

2.2.1.2.5 Physical Adversarial Attacks

As explained in Chapter 2.2.1.2.1, physical attacks aim to create physical-world-resilient adversarial examples to mislead AD systems. The following chapter gives examples for SOTA physical attacks that are tested in real-world settings.

In (33) and (34) the transferability of generative-based attacks to a real-world setting on black-box models for traffic sign recognition (TSR) is investigated. Both works use different generative-based attacks to digitally

generate stickers and patches that cause a misclassification and apply these stickers on physical traffic signs to test their transferability. During an experiment using self-taken images of real-world traffic signs with patches on them the authors of (33) are able to reduce the victim model's accuracy from 86,7% to 17,2%. The authors of (34) carry out their experiment with a camera attached to the dashboard of a moving vehicle and achieve an attack success rate of $60\% - 80\%$ at speeds of 6 km/h and 30 km/h on traffic-signs with the generated patches on them.

Finally, in (35) the authors propose the generative-based framework PhysGAN that is able to generate realistic attacks against an end-to-end driving model. Their method creates adversarial examples from existing billboards that are almost imperceptible to the human eye. With these adversarial examples, they are able to produce a steering angle deviation by 19,17°.

Additionally, the authors in (36) investigate approaches to perform physical attacks on TSR systems that are not based on generative adversarial attacks when only black-box access is available. They evaluate the success rate of transferring a physical perturbation generated for an extracted white-box model and executing an optimization-based attack but using a gradient approximation method to estimate the required gradients of the unknown black-box system. Both targeted attack methods are successful by achieving a success rate of $\geq 90\%$ under the best configuration but it shows that the attack is significantly more expensive when only black-box access is available.

2.2.1.2.6 Attacks on Mobility Use Cases

After presenting a general overview of evasion attacks, we now present a short summary of evasion attacks on use cases that are specifically relevant for an AD system. For each perception use case presented later in Chapter 2.4.1.1 we discuss the most interesting publications and provide an example of a digital and a physical adversarial attack.

2.2.1.2.6.1 Image Data

As described previously most research work focuses on evasion attacks on DNNs that use image data as input. Here, the first perception use case is represented by object detection. The authors in (37) consider this use case in combination with semantic segmentation and generate digital perturbations that fool different DNNs for both use cases at the same time. Physical attacks are explored in (38) where the authors generate patches that can be stuck on traffic signs and evade the detection by current SOTA object detectors. They also observe a transferability between different detectors which enable black-box attack with $\approx 40\%$ success rate when tested in reality.

The second perception use case from Chapter 2.4.1.1 is segmentation. In (39) the authors explore digital attacks to fool DNNs for semantic segmentation by removing certain objects from the prediction or changing the entire predicted segmentation. Moving to physical attacks, the authors in (40) focus on local attacks on DNNs for semantic segmentation. They introduce a local attack that fools current segmentation models but only evaluate the success rate on synthetic images and not on images captured in reality. The resulting perturbation and segmentation prediction is shown in Figure 12 for different DNN models. As can be seen, using the local patch to perform tests in reality is difficult because the patch is additive to the unperturbed images.

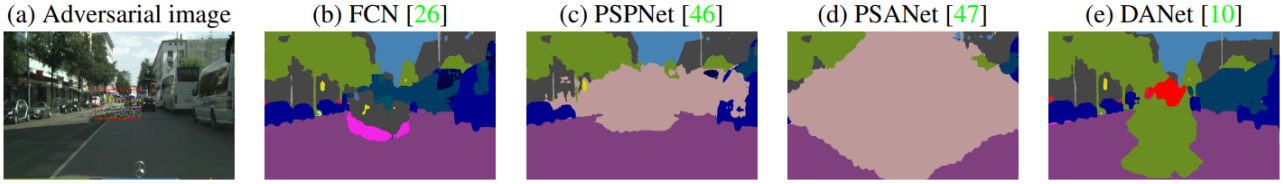


Figure 12 Exemplary visualization of a patch-based evasion attack on semantic segmentation from (40)

Optical flow represents the third perception use case. Here, the authors in (41) present different digital attack methods to fool DNNs for optical flow prediction either globally or locally. In (42) the authors then consider patch-based perturbations which can be applied in the physical-world. These are shown in Figure 13 for an

example. However, they only perform synthetic experiments and do not report success rates when the patch is printed and applied in reality.

Figure 13 Exemplary visualization of a patch-based evasion attack on optical flow prediction from (42)

Lastly, different evasion attacks are also proposed for the depth prediction use case. The authors in (43) generate digital perturbations that impact the prediction globally or locally. They are able to remove objects from the prediction or change the overall distance that is predicted. Considering patch-based attacks the authors in (44) test different patch-based perturbations against SOTA DNNs for depth estimation. We show some examples in Figure 14. Again, the authors evaluate the attack success rate only on synthetic images and do not experiment with printed patches in the real-world.

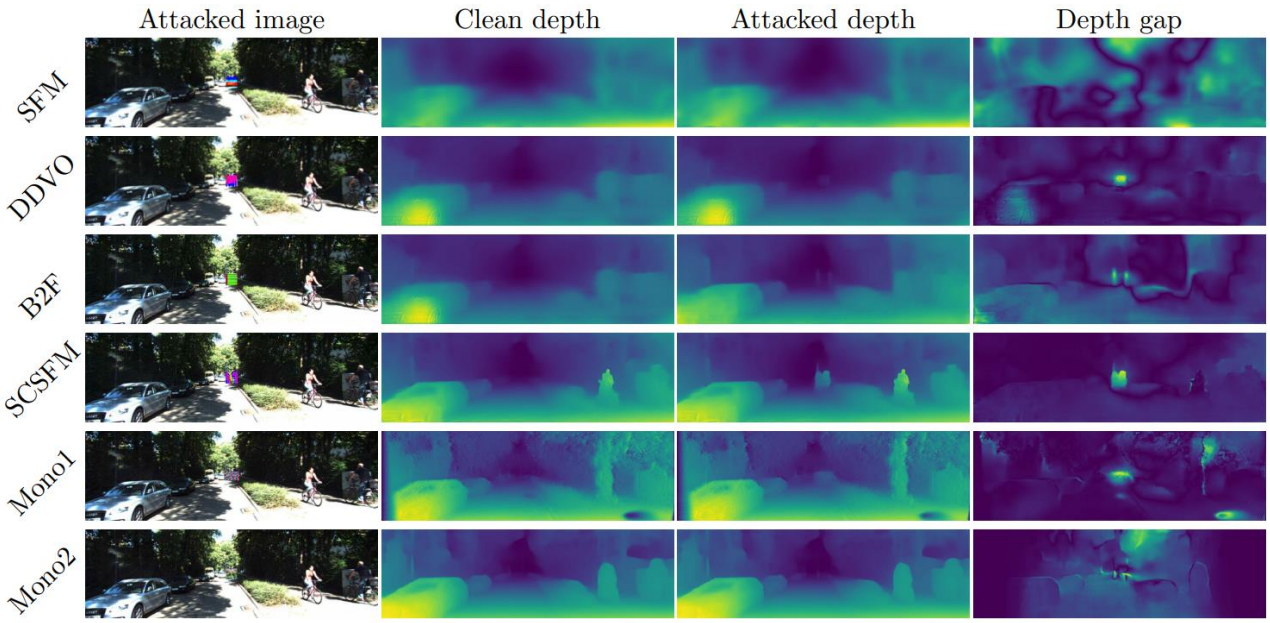


Figure 14 Exemplary visualization of a patch-based evasion attack on depth prediction from (44)

Concluding, it is important to note that most evasion attacks are designed for the white-box use case. In most experiments the threat of black-box attacks is only evaluated by testing the transferability of the generated perturbations. Only a limited number of works integrate different methods for attacking black-box systems in their evaluations. Also, in most cases physical attacks based on patches are only evaluated when the patch is added synthetically to an image. Performing the evaluation in the reality is most challenging and typically also leads to reduced success rate of various attack methods.

2.2.1.2.6.2 Point Cloud Data

For evasion attacks on perception use cases of an AD system that are based on point cloud data only a very limited number of publications exist. One reason is that the required LiDAR sensors are more expensive and experiments cannot be done as easily as for camera sensors. Also, only the use case of object detection is covered because here point cloud data is most useful. In (45) the authors consider this task and propose an attack method that generates point clouds that are marginally perturbed starting from original, unperturbed points clouds. A human does not notice a significant difference meaning the presented perturbations are similar to imperceptible digital perturbations on images. Alternatively, the authors in (46) consider the task of physical adversarial attacks. They generate 3D objects which are not detected by a SOTA object detection model. An exemplary visualization of the generated perturbation is shown in Figure 15. The authors also test the success rate in reality by 3D-printing the adversarial example and performing a drive-by test using a real LiDAR sensor. They find that their attack is still successful and fools the model under different environmental conditions.

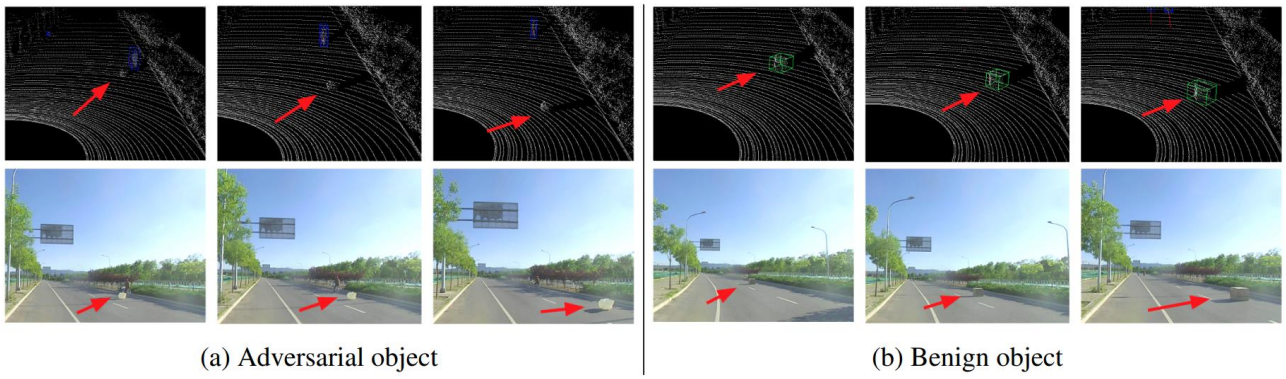


Figure 15 Exemplary visualization of an evasion attack on LiDAR-based object detection from (46)

In addition to research work that considers evasion attacks on camera-based and LiDAR-based perception systems separately, recent methods consider the task of fooling both perception systems by the same perturbation. Here, the authors in (47) propose an attack method to generate an adversarial object that is placed on the roof of vehicles. For camera-based perception the object looks like a patch and for LiDAR-based perception the object is a 3D mesh like the one described previously. In Figure 16 some examples of the resulting object are shown for different scenes in both image and LiDAR space. The authors test the attack success rate on simulated data and find that the attack is successful at fooling both perception systems at the same time. However, the resulting perturbation looks highly suspicious and is quite large in comparison to the vehicle it is applied on.

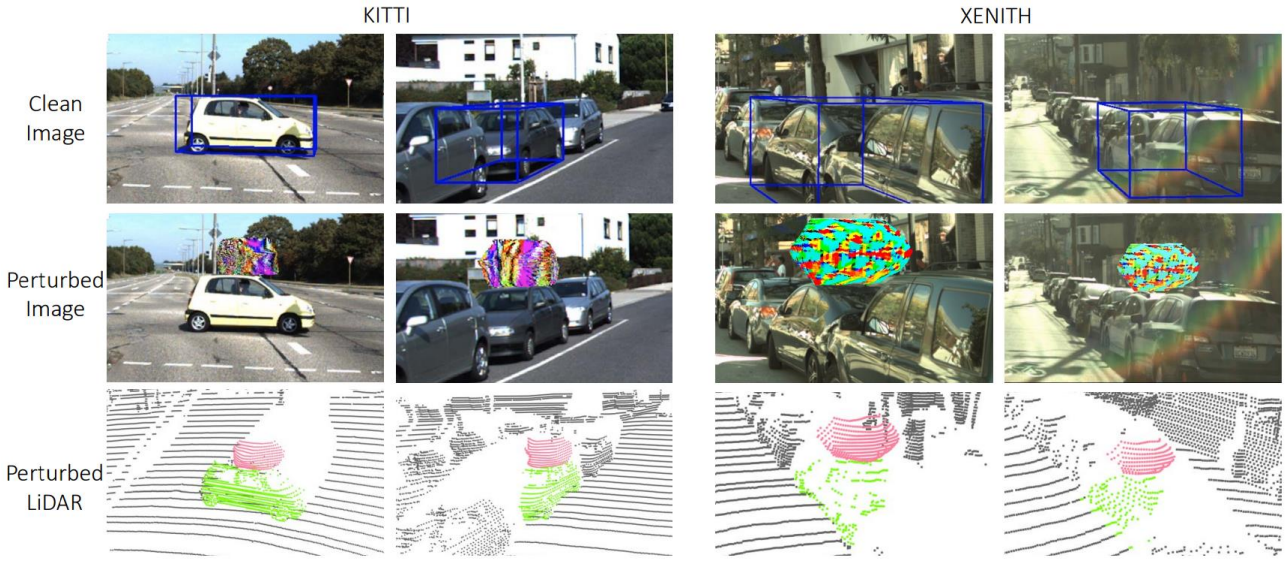


Figure 16 Exemplary visualization of an evasion attack on LiDAR and camera-based object detection from (47)

Finally, there are also methods that attack LiDAR-based perception by sensor spoofing. In (48) the authors transmit laser signals to a LiDAR sensor which creates a fake vehicle in the current scene. However, this attack does not exploit any weaknesses of an AI-based system but instead tricks the physical capabilities of a sensor.

2.2.1.3 Data Poisoning

Data poisoning describes the injection of poisoned data samples in the training dataset. An adversary actively tries to manipulate a part of the dataset by adding new data samples, changing existing data samples or changing the associated labels. This impacts the behavior of all data-driven systems that are learned using the poisoned dataset. The behavior is degraded depending on the specific goals of the adversary. In the following we first present a categorization of different attacks and then discuss concrete attack methods for different goals of an adversary.

A survey of the threat posed by data poisoning is given in (49). The authors cover different attack goals and present methods from current research works. Additionally, they also cover defense strategies and point out open problems in this research field. Open problems are also investigated in (50). The authors perform a thorough evaluation of recently proposed attack methods and find that many methods perform worse when evaluated under different settings. They observe a high sensitivity to variations in the evaluation setup and conclude that many methods do not generalize to realistic attack scenarios. Hence, it is unclear how big the threat posed by data poisoning is and whether stronger attacks can generalize better to realistic settings. To improve the thoroughness of future evaluations of attack methods they develop a standardized benchmark and show pitfalls in the evaluation process.

2.2.1.3.1 Categorization

In Figure 17 we present a taxonomy of data poisoning attacks that can be used to categorize different attack methods. Most important is the first category that determines the type of the poisoning attack and thus the goals of an adversary. This category is used to structure the remainder of this chapter, where we present different methods for each attack type. Another important category is the required model access. Here, the main distinction is whether a method needs access to a concrete model or can generate poisons for a variety of models. The next important category is the required data access of a method. Some methods need access to the concrete training dataset while others only need access to the rough test distribution the trained model should operate in. Lastly, the last two categories focus on the perceptibility of the generated poisons. Here, it is important to first distinguish the perceptibility. This can be analyzed either in visual image space or in some latent feature space that only a system sees. Both determine how well a poisoning can be detected on revision of the poisoned dataset. Similarly, the last category distinguishes whether an attack needs to corrupt the associated label of data samples. Some methods keep the original label and as a result the poisoning is more difficult to detect when the dataset is analyzed.

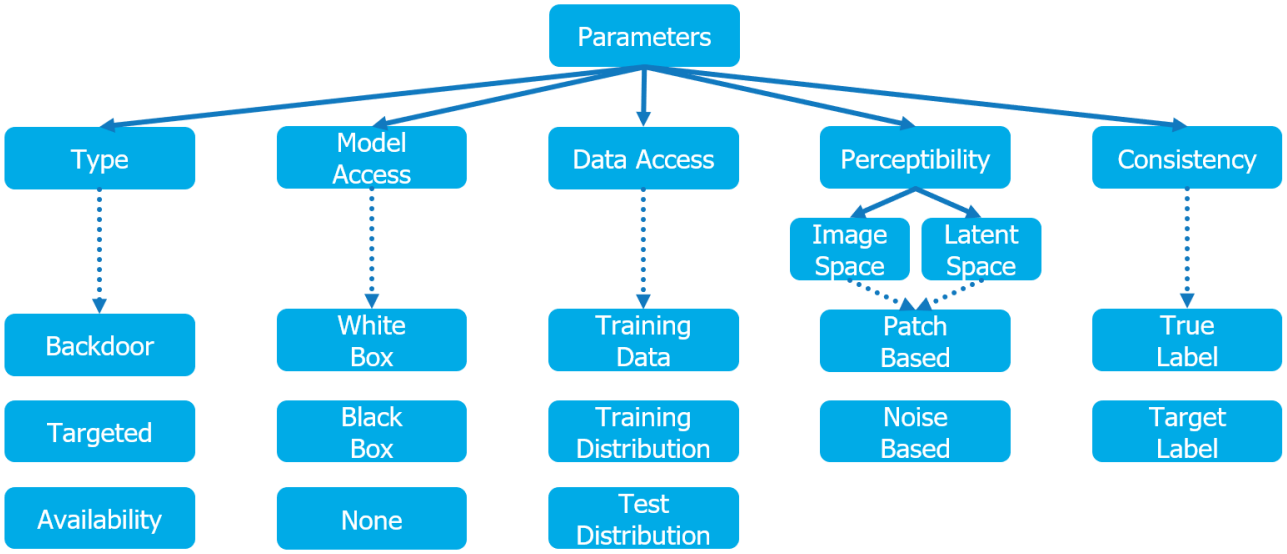


Figure 17 Categorization of attacks for data poisoning

2.2.1.3.2 Backdoor Attacks

The first attack type are backdoor poisoning attacks. Here, an adversary tries to install a backdoor trigger in the training dataset. When this backdoor trigger exists on an image the model should classify this image as a specific class and not as the correct class. However, when the backdoor trigger is not part of an image the model should show a normal, inconspicuous behavior. An overview of different backdoor poisoning attacks and the used backdoor triggers is shown in Figure 18. For example, the patched-based backdoor introduced in (51) inserts a small rectangular patch in an image. When this patch exists, the model should not classify the data sample as "speed limit 30" but instead as a different target class which the adversary specifies. To achieve this, the authors add images that contain the patch into the training dataset and assign the adversarial target label to these images. Hence, they introduce images into the training dataset that are wrongly classified for human observers and would be noticeable when inspecting the dataset. Also, in recent years different methods are proposed that change the style of the backdoor trigger which is also shown in Figure 18. Now, it is also possible to insert a backdoor trigger that is imperceptible for humans but can still be used to introduce malicious behavior into the model.

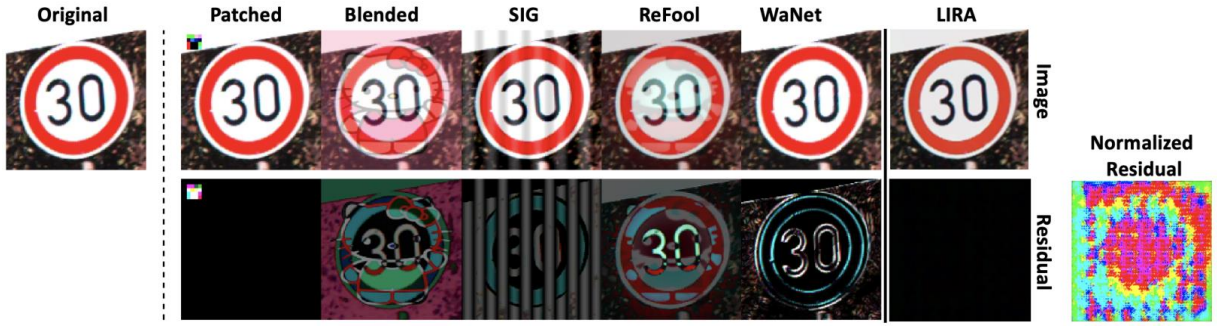


Figure 18 Overview of different attack methods for backdoor poisoning from (52)

However, the described backdoor poisoning attacks can easily be detected when the labels of a dataset are analyzed. Hence, the authors in (53) and (54) extend existing attacks to the clean label settings. Here, the poisoned data samples are no longer assigned a wrong label but are inserted into the training dataset with the correct label. This prevents the easy detection of poisoned data samples by humans and increases the threat posed of such attacks. In Figure 19 an overview of an exemplary attack method for clean label backdoor poisoning is shown. The authors expand the patch-based attack introduced earlier and propose a new method to generate the poisoned data samples. Instead of inserting images that contain the patch and are labeled as the target class, the authors insert poisoned examples of the target class into the dataset. These examples are generated by adding specific noise with a small magnitude to original data samples such that a human does

not notice a difference between the images. Nevertheless, the poisoned images lead to a change in the learned decision boundaries of the model, which leads to a successful attack once the original backdoor is used during inference.

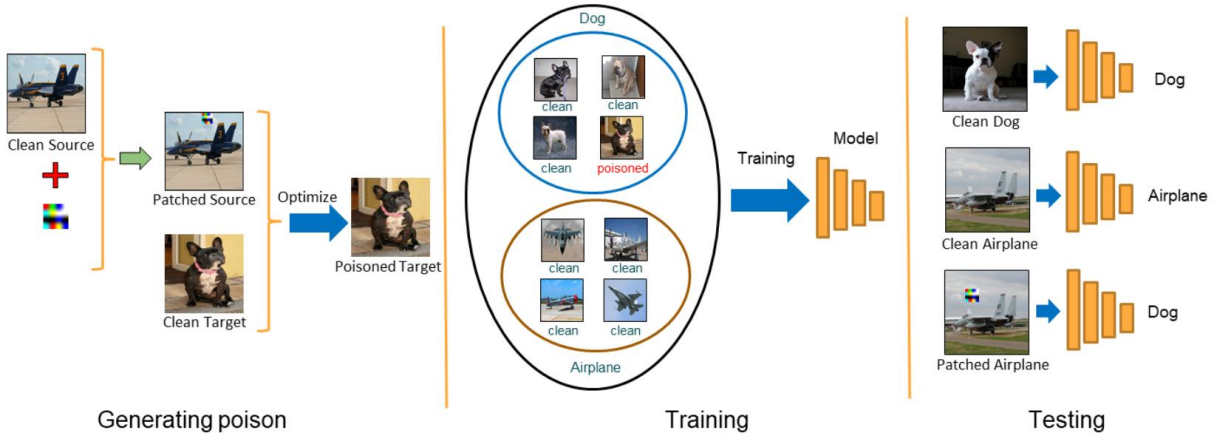


Figure 19 Exemplary visualization of label consistent backdoor poisoning from (54)

Along with the development of improved attack methods defense methods are proposed, which are covered in detail in Chapter 2.3.1.3. Some of these defenses use the fact that most backdoor attacks can be well separated in a latent feature space of a trained model. To close this defense option the authors in (55) introduce further improvements of backdoor attacks. They optimize the perceptibility with respect to some latent feature space that the model learned. This hides the backdoor from detection based on the difference of the representations in the latent feature space.

Recently, backdoor attacks are also proposed for models trained in a semi-supervised way using different forms of self-training, e.g. using FixMatch from (56). The main advantage of such methods is that training on largely unlabeled datasets is cheaper and faster to perform. However, the authors in (57) show that it is possible to poison the unsupervised part of the dataset and hide backdoor triggers there. Hence, it is questionable whether training on largely unlabeled datasets is desirable if the data quality cannot be guaranteed.

Finally, it is important to note that the evaluation of backdoor attacks is done only on synthetic data. To the best of our knowledge no research work exists that test whether the backdoor is still successful when the backdoor triggers are applied in the physical world and recaptured by a camera. For example, it would be interesting to print the patch-based backdoor triggers and stick them to a real traffic sign. Then, one should evaluate the attack success rate when images of the attacked traffic sign are captured from different angles and under different lighting conditions.

2.2.1.3.3 Targeted Attacks

The second attack type for data poisoning attacks are targeted poisoning attacks. Here, the adversary no longer aims to introduce a general backdoor into the behavior of a model, but instead tries to change the classification of a predefined, unmodified target image. This different goal of the adversary is shown in Figure 20. The goal is that the specific image of an otter is misclassified as the target class Labrador. For this attack type only clean label poisons are relevant since otherwise the target image would just be assigned the wrong label. For this no advanced methods would be needed since the target images with the wrong label can just be added to the training dataset.

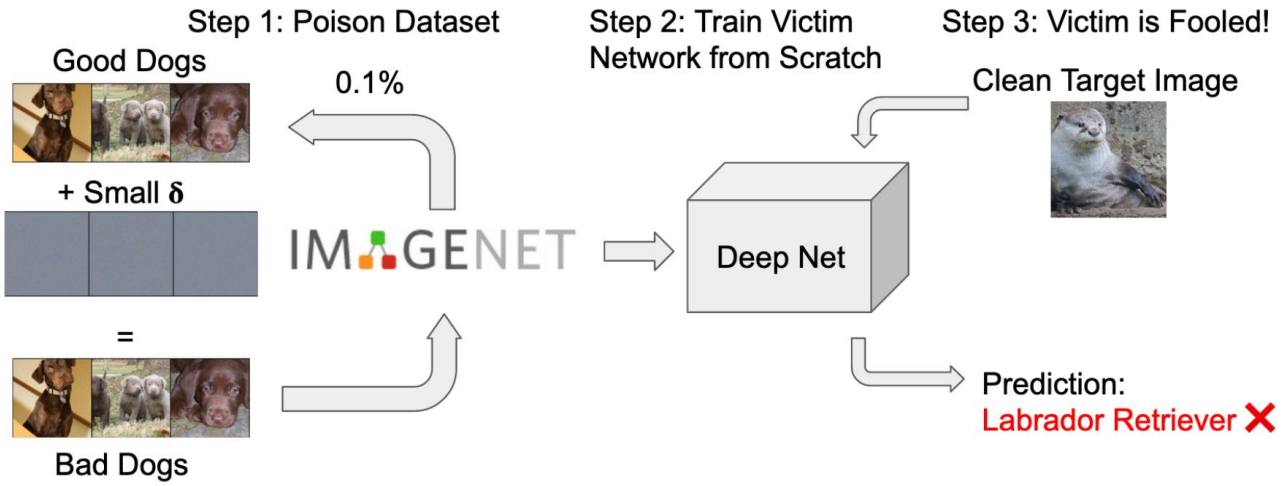


Figure 20 Exemplary visualization of targeted poisoning from (58)

In recent years multiple methods are proposed to perform targeted poisoning. One of the first is (59) where the authors introduce an optimization-based method that needs knowledge of the model architecture that is trained on the poisoned dataset. They poison data samples in the training dataset such that the latent feature representation collides with those from the target image. An alternative approach is introduced in (60). Here, the authors introduce a new optimization that uses an ensemble of models to generate the poisoned data samples. They show that the resulting poisons transfer well to other models with unknown architectures and training setups also when these are trained from scratch. Very recently, the authors in (58) introduce a new targeted poisoning attack that is more efficient than previous attacks. For the first time they can attack models that are trained from scratch on the entire ImageNet dataset introduced in (61). Their main idea is to explicitly match the gradient direction of malicious examples such that they produce a malicious gradient signal during training.

Similar to work on backdoor poisoning recently an attack is proposed that introduces targeted poisoning for semi-supervised learning. In (62) the authors present a strategy to poison the unsupervised part of a dataset by inserting multiple images that form a path from a correctly labeled image to the target image. Then, during the semi-supervised training these images are iteratively assigned the label of the starting image until the target image is reached.

Concluding, it shows that targeted poisoning attacks are not that relevant for this project. Like physical adversarial attacks for evasion, data poisoning attacks are only relevant if they can be performed on an AD system during deployment. To do so in the case of targeted poisoning attacks the adversary needs to insert the target image into the AD system during deployment. It is not enough to place a backdoor in the surrounding scene, but instead the original image captured by the camera needs to be exchanged with the target image. To be able to do this the adversary would need access to the interface between the camera sensor and the perception systems and perform the exchange during run-time. If an adversary can do so, it can perform far easier and more efficient attacks. Therefore, only backdoor poisoning attacks form a principled threat on AD systems if the backdoor trigger remains effective after application in reality.

2.2.1.3.4 Availability Attacks

The last attack type are availability poisoning attacks. The goal of these attacks is slightly different to both attack types discussed previously. Instead of introducing a backdoor or targeted poison that the adversary can exploit during inference, the goal is to change all data samples such that these cannot be used for training a data-driven model. Hence, these attacks are important to prevent the unauthorized use of personal data that is scraped from some publicly available database on the internet. They can be used to release secure datasets where the data samples cannot be used for training a data-driven system. Therefore, these attacks are not relevant for this project since they do not pose a threat on the security or safety of a model and are only mentioned for completeness.

The most promising methods for this attack type add some kind of noise to the original data samples. In (63) the authors add error-minimizing noise on some data samples which tricks the training process into thinking there is no information to learn from these data samples and thus ignoring them during training. A further improvement is presented in (64) where the authors generate the noise by adversarial attacks discussed in Chapter 2.2.1.2. They find that training a model only on adversarial perturbed images generated for a different model basically prevents any training progress.

2.2.2 Robustness

This chapter presents different threats on the robustness of a data-driven system. Specifically, we cover the general threat of encountering natural perturbations during inference where a system sees data samples from a different distribution than the training dataset. Mitigation strategies to all presented robustness threats are later covered in Chapter 2.3.

2.2.2.1 Natural Perturbations

In general, OOD data describes the presence of data samples that deviate from the exact training distribution used during training of a data-driven system. This effect occurs naturally when systems are deployed in the real-world outside of a completely supervised environment. For example, in the case of AD/ADAS it is impossible to capture all possible environmental scenes or driving scenarios in the training dataset. During deployment unseen situations will occur and the data quality of the sensors is impacted by different factors like lighting, rain, heat, etc. These might lead to a shift in the data distribution and present a challenge on the generalization of a system. Thereby, data distribution shifts exist at different levels which range from minor shifts to full OOD data where new concepts exist during inference. At this strongest level it can be impossible that the system keeps a correct prediction. For example, it might occur that a new traffic sign or parking space marking is observed during deployment of an AD system. If the system is not trained on this specific sign or marking the resulting distribution shift is so large that a correct prediction is impossible for the AD system. Additionally, the problem is exacerbated by the fact that DNNs often show high confidence on OOD data samples. This often prevents the detection that a data sample is OOD, which could be used to trigger safety measures. A more detailed discussion on the confidence estimation is done in Chapter 2.3.2.2.

Since distribution shifts occur naturally in the real-world, research recently started into looking at the performance of deep neural networks (DNNs) on OOD data. Here, different datasets are introduced that mimic important and often used datasets but contain corruptions or perturbations. One of the first is introduced in (65). Here, the authors present ImageNet-C which extends the original ImageNet dataset from (61) by applying 15 different corruptions with five different strength levels on the original images. In (66) the authors further propose an extension to the ImageNet-C dataset by using different corruptions that are significantly different to the original corruptions. They argue to use this new dataset as a test dataset to measure the generalization of systems that are trained on the original ImageNet-C dataset. Lastly, the authors in (67) introduce ImageNet-R which contains various renditions like paintings or embroidery of the image classes from the original ImageNet. This represents the strongest domain shift since only the semantic concepts of a class are similar. In addition to the presented datasets to measure the generalization of standard classifiers for image data, recently similar datasets for OOD data are proposed for perception tasks that are relevant for the mobility use cases described in Chapter 2.4.1.1. We discuss these specific datasets in Chapter 2.6.1.

The problem of generalization of data-driven systems is typically attributed to the underlying problem of shortcut learning. In (68) the authors describe shortcut learning as "[learning] decision rules that perform well on standard benchmarks but fail to transfer to more challenging testing conditions, such as real-world scenarios". Hence, the learned rules have a far worse performance on OOD data or under evasion attacks described in the Chapter 2.2.1.2. The systems do not learn robust feature representations but instead use every piece of information available in the training dataset even if it is extremely brittle. This effect also shows when comparing the robustness of human visual perception and current convolutional neural networks (CNNs) in (69). First, humans are more robust than current CNNs when considering different corruption types.

However, when CNNs are trained on a specific corruption type they consistently surpass human performance on this specific corruption type. Nevertheless, on most other corruption types the CNNs perform worse and do not have an increased robustness. Hence, training on a single corruption type does not necessarily imply an increased robustness on other corruption types. The CNNs do not learn generally robust feature representations and decision rules but still exploit the shortcuts for a specific corruption type.

2.2.2.2 Adversarial Perturbations

In addition to natural perturbations another threat on the robustness of a data-driven system is posed by adversarial perturbations. These specific perturbations are the result of an evasion attack where an adversary actively tries to harm the quality of a system. Such attacks are previously covered in Chapter 2.2.1.2. We again mention these attacks here to highlight the tight interaction when considering the robustness and IT-Security of AI systems. Adversarial attacks impact both categories and depending on the viewpoint can be assigned to one or the other.

2.2.3 Explainability

The existing black-box character of AI-based systems combined with the complexity and number of parameters of DNNs complicates the possibility to explain the behavior of a system. In contrast to traditional IT systems that are based solely on algorithms, the behavior of the system is not directly determined by human developers but is instead derived indirectly from the training dataset. Hence, it is largely unclear how a system comes to its predictions and which features of a data sample are most important for a concrete prediction. Therefore, the need arises for methods that can explain the decision or general behavior of a system that is learned from data.

The topic of explainability also gained lots of interest from researchers in recent years. Hence, quite a number of surveys emerged that summarize recent methods and discuss various types of explainability. For example, in (70), (71) or (72) the authors conduct a survey on different aspects of explainability. All try to systematically assess different methods and provide a comparison of the advantages and disadvantages for each method. In (71) the authors only consider the task of explaining CNNs, while the other two surveys also discuss the explainability of more traditional AI systems like decision trees or support-vector machines.

Additionally, it is important to point out that there is currently no agreement over the form of explanations needed for AD systems to be deployed in the public. Important questions must be discussed and answered, which include the following:

- For whom should an explanation be provided?
    - Developers, auditors, customers, etc.
- Which part of an AD system should be explained?
    - Perception, behavior prediction, path planning, end-to-end behavior, etc.
- When does an explanation have to be provided?
    - During audits, in case of accidents, on demand, etc.
- Which details should an explanation provide?
    - Highlight important features, exact traceability, etc.

However, such questions are not unique to AD systems, but arise in general as soon as AI-based systems are used in safety-critical contexts. The authors in (73) make the same observation that the need for explainability of black-box AI systems increases but only little work is done around discussing what is exactly needed. Hence, they conduct a close-door workshop with different stakeholders from academics, industry, policymakers and legal scholars. They discuss important unanswered questions about the design and deployment of explainability methods and provide the main takeaways. It is key to involve stakeholders in the development process and pay attention to the context in which an explanation is used.

2.2.4 Documentation

The uniform documentation of AI systems only emerged recently as a relevant research topic. Up to now, companies that deal with AI systems have their own workflow and solutions for the documentation of AI requirements, the development process, deployment statistics, etc. Only limited work and efforts exist that focus on the development of standardized documentation. This hinders the overall transparency of the development process and of the resulting AI system.

Existing documentation practices for traditional IT systems can only be partially applied to AI-based systems since the lifecycle of AI-based systems presented in Chapter 2.1 is more complex. Unique aspects like the data collection or data-driven training process lack traditional documentation practices. In addition, there are no common techniques that can be used by developers of AI-based systems to show that they addressed the different challenges presented so far in this chapter. This is required to demonstrate the accordance which serves to increase the transparency and trust in AI-based systems when deployed in reality. Additionally, this is required such that an auditing process is able to properly assess the quality of a system for the different aspects in the lifecycle.

2.2.5 Safety

Safety considerations within safety critical applications (e.g. automotive, industrial) are crucial since any failure or malfunction can cause major impact to the physical integrity of the users and any person within the application environment. Despite other important aspects like security, usability etc., safety is the highest priority in any stage of the lifecycle of an application. In case data-driven algorithms and especially machine learning (ML) algorithms determine the behavior, safety considerations also play an important role in the ML-specific lifecycle. Thus, to minimize potential threats or hazards which can lead to risk or harm, ML-based systems must comply with safety requirements mitigating the issues of the data-driven algorithms.

Despite the big advantages of ML which enable the current progress on autonomous driving or advanced driver assistance systems (ADAS), ML-based algorithms imply several limitations and risks affecting the following topics that are summarized in Figure 21:

- Design specification
    - Lack of formal specification of the design & functionality
    - Gap between design objectives and actual behavior
- Implementation transparency
    - No traditional rule-based design
    - Variables are unknown
    - Lack of interpretability
- Formal verification
    - Test coverage and residual risk almost unknown
    - Common formal verification methods not fully applicable
- Performance and robustness
    - Test set performance often unlike real-world performance
    - Uncertain robustness against perturbations and unknown input data
- Monitoring
    - Limited failure prediction
    - Current prediction probabilities often unlike reliable uncertainty estimation

- Optimal ground truth behavior often unknown

Figure 21 Main limitations of ML algorithms with regard to safety

Independently of a specific safety standard or framework, the above-mentioned limitations impede safety considerations. In contrast to classical software applications, examination of the safety properties of ML must take the data-driven decision-making into account. Each safety consideration and its limitations have a strong overlap to other chapters in this document e.g. IT Security, Robustness, Mitigation Strategies, Standardization Activities AI & AD, etc.

2.2.6 Certification and Verification

Robustness certification (respectively verification of neural networks) is the procedure of examining the functionality of a model, in particular when facing specially manipulated input data such as adversarial examples discussed in Chapter 2.2.1.2. A DNN is robust against these manipulations if its predictions are still considered as correct. The goal of verification/certification is to state that for a given input (space), the model's output will lie within a specified output space. The procedure may give a guarantee under certain constraints that the output is in the specified output space or a probability that this is the case. In other words, a lower bound for the robustness of the DNN is determined.

The need for robustness certification results from an increasing number in attack approaches trying to deceive DNNs and modify their predictions. Especially in security and safety critical applications, e.g. autonomous driving, this could cause severe harm. Therefore, it is crucial to be able to measure the vulnerability/robustness of DNNs against adversarial attacks to ensure a certain security level or reveal a backlog in the matter of defense mechanisms.

Ideally, certification is embedded in the development phase of the AI-based system and is part of the testing and benchmarking as shown in Figure 2. In this way, a potential lack of robustness is identified early on and relevant countermeasures can be applied. Furthermore, the certification/verification can be part of the assessment or auditing processes after development. For instance, this might be required in the case of systems that are continuously retrained during deployment.

2.2.7 Standardization

The last challenge for using AI-based systems in safety-critical applications is based on the fact that currently no standardization specific for AI exists. There are no uniformly acknowledged principles and practices that the development, testing or deployment of AI-based system must fulfill. We discuss this further in Chapter 2.3.5 with the focus on the safety of AI-based systems.

Nevertheless, there are approaches to develop a standardization of AI-based systems. Best known here is the EU AI Act introduced in (74) that tries to lay down uniform regulations on AI-based systems. They present a horizontal regulatory approach with necessary requirements to address different risks and challenges when AI is used without focusing on the needs for specific application areas. Additionally, there are also vertical regulatory approaches which aim to develop standards for concrete application areas. Here, standardization approaches for AI in mobility applications are most relevant for this project. Therefore, in Chapter 2.7 we discuss existing and developing standardization approaches when AI-based systems are used in the area of AD or ADAS.

2.3 Mitigation Strategies

In this chapter specific mitigation strategies for the challenges discussed in Chapter 2.2 are presented. The remainder of this chapter is structured in the same way as Chapter 2.2 so it is possible to directly map a mitigation strategy to the associated challenge that is posed on AI-based systems.

2.3.1 IT-Security

In the following, mitigation strategies for the challenges on IT-Security from Chapter 2.2.1 are discussed. As mentioned previously these are also strongly connected to robustness challenges from Chapter 2.2.2 where the specific mitigation strategies are presented in Chapter 2.3.2.

2.3.1.1 Model Extraction Attacks

Within this chapter, dedicated privacy preserving methods are presented and discussed. To keep the focus on safety and security related threats, only those methods are introduced that prevent adversaries to craft successful adversarial examples with a lower attack effort.

2.3.1.1.1 Differential Privacy

Training data often includes sensitive information or represents intellectual property of the manufacturer. Any leakage or exposure of the training data might result in a data privacy violation or commercial issues. However, after the training of a data-driven model this model contains the information in an abstract manner which allows an extraction in certain circumstances. To avoid any successful information extraction, differential privacy provides a defense method based on adding random noise during the training phase (75). Similar approaches add noise to weights, data and gradients and use specific noise distributions.

As shown in Figure 22, this defense method increases the robustness against information leakage but at the same time decreases the performance of a model. Therefore, achieving differential privacy is always a tradeoff between noise strength and model performance (75).

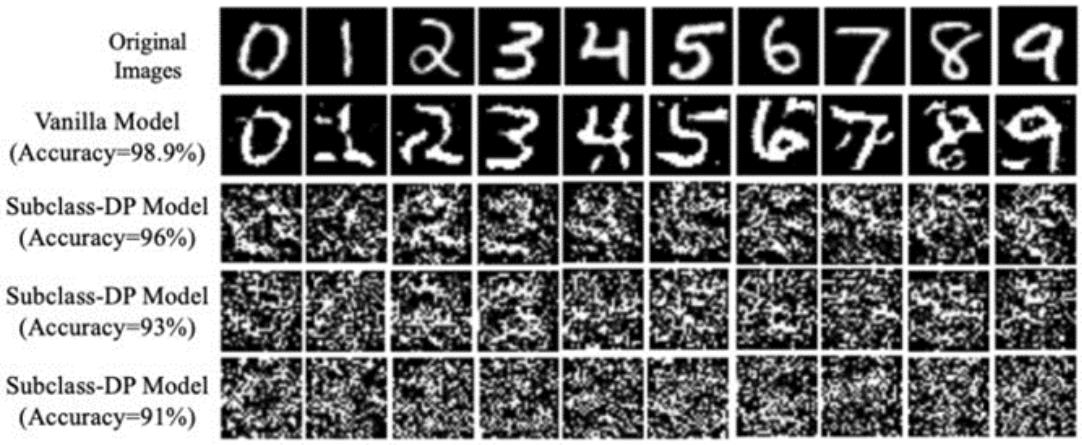


Figure 22 MNIST data set with different thresholds of added noise from (75)

2.3.1.1.2 Homomorphic Encryption

Another method to mask and hide the actual data is homomorphic encryption (76). Due to the homomorphic properties, the computations can be executed on encrypted data without touching the sensitive, non-encrypted content. After finalizing the computations, the output can be decrypted and provided to the user. Limitations of this method are the restriction to suitable models due to polynomial constraints and the applicability only during inference. The polynomial constraint entails that this method is only applicable to polynomial functions, which harshly restricts the supported AI architecture. Additionally, homomorphic

encryption creates a computational overhead and increases the run-time of the system. However, the field of homomorphic encryption in ML requires more research efforts and is under further development (76).

2.3.1.1.3 Trusted Execution Environment

Trusted execution environments (TEEs) (77) provide a trustworthy and controlled environment where sensitive data can be computed. The data within a TEE can be processed unencrypted whereas data outside of the TEE can be encrypted. An example of a TEE is given in Figure 23. Nevertheless, TEEs require a relatively high effort in secure coding and do not provide full guarantees regarding information leakage. Side-channel analysis or similar attacks are still applicable. However, it has to be noted that Side-channel attacks require an extensive effort and can require physical access to the system (78).

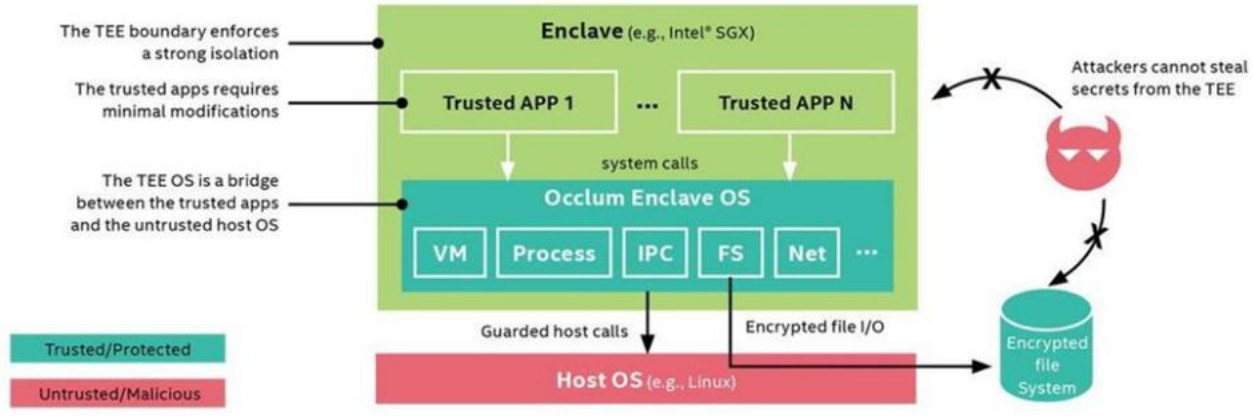


Figure 23 Basic architecture of a TEE from (79)

2.3.1.2 Evasion Attacks

In this chapter SOTA mitigation strategies against evasion attacks are presented. An overview of evasion attacks is given in Chapter 2.2.1.2. The general problem of evaluating mitigation strategies is described in (80) where the authors note that adaptive attacks can often break previously proposed mitigations. Hence, they list recommendations for an evaluation checklist to improve the general quality of the evaluation of mitigation strategies. Similarly, the authors in (81) propose a new ensemble of evasion attacks and use this to evaluate current defense strategies. They discover several broken defenses which again shows the difficulty of correctly evaluating a mitigation strategy. All of the following methods create a tradeoff between the efficiency of the defense and the computational and development effort.

2.3.1.2.1 Categorization

Figure 24 displays a categorization of mitigation strategies against evasion attacks. Mitigation strategies can either be proactive by improving the robustness of a model before deployment, or reactive by mitigating adversarial attacks on a deployed model. Adversarial training encompasses methods that aim to improve the robustness of a model by retraining the model on adversarial examples. Defensive distillation describes methods that aim to distill the information of the original model by distilling the original model predictions through another model. By aggregating different models with each other, model ensemble methods exploit the complexity of transferring adversarial examples over different models to improve the robustness. Additionally, there exist a lot of methods to defend a model by either detecting adversarial input samples or transforming the input data of a model in such a way, that the clean input data is reconstructed as best as possible from the adversarial examples.

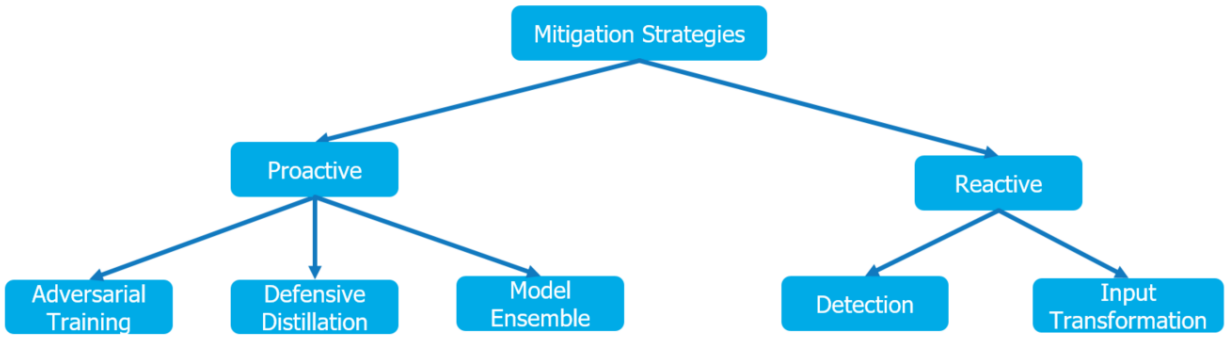


Figure 24 Categorization of mitigation strategies against evasion attacks

2.3.1.2.2 Adversarial Training

Adversarial training is probably the most studied defense method against adversarial attacks. The basic concept is to train a model on adversarial examples to make it more robust against such examples. This enables the model to connect adversarial perturbations to their correct label and therefore prevents misclassifications.

In (22) the authors first introduce the PGD attack, which is further explained in Chapter 2.2.1.2.2, and then perform worst-case adversarial training using this attack. They train the model solely on adversarial examples generated with PGD and are able to achieve high robustness against several attacks with white and black-box access. However, it has been shown that this defense method leaves the model still vulnerable to certain types of attacks (82).

The authors of (83) propose to improve the robustness of a model by training it with adversarial examples transferred from other models. During the training process, they alternate between adversarial examples created for the model to defend and the transferred adversarial examples from other models. This improves overfitting issues of adversarial training, since the adversarial examples are more diverse. Additionally, the training on transferred adversarial examples improves the black-box robustness of the models compared to regular adversarial training.

Adversarial training also is a known defense against data poisoning attacks, which is described further in Chapter 2.3.1.3. However, adversarial training requires very large datasets and must be done carefully to prevent the model from overfitting, since overfitting in turn makes the model more vulnerable against attacks. Additionally, it requires changes to the training process of the model.

2.3.1.2.3 Defensive Distillation

Defensive Distillation, as introduced in (84), is a method that distills the information of a model. The authors train a second model with the probabilities of the logit layer from the original model as soft labels. A schematic overview of the training with distillation is shown in Figure 25. This creates smoother classifiers and decreases the classifiers sensitivity to input perturbations. However, it has been proven that defensive distillation can be successfully evaded by attacks such as the C&W attack, for example in (20).

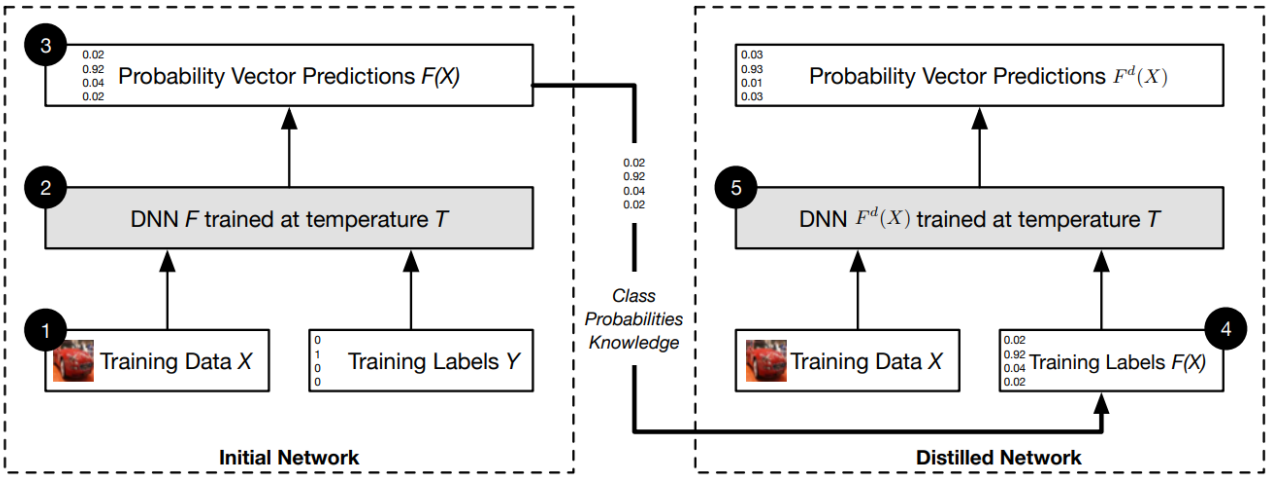


Figure 25 Training procedure under defensive distillation from (84)

2.3.1.2.4 Model Ensemble

In (85) the authors introduce random self-ensembles, where they inject noise layers before each convolutional layer during the training and inference phase of a model as shown in Figure 26. They show that this approach is equivalent to combining an infinite number of random models in an ensemble. Under the C&W attack this mitigation strategy is able to maintain an accuracy of 86,1%.

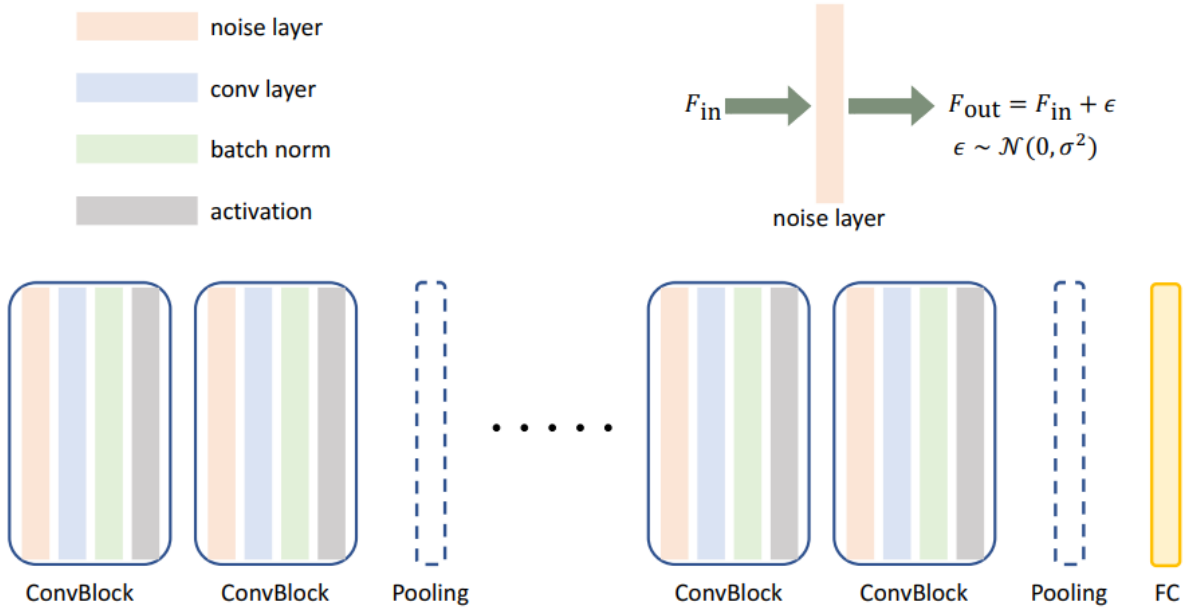


Figure 26 Illustration of a Random Self-Ensemble model with noise layers before each convolutional layer from (85)

The authors of (86) propose the use of an adaptive diversity promoter (ADP) training procedure to introduce more diversity into their model ensemble. They use an ADP regularizer that encourages the low confidence-predictions of each model to be orthogonal to each other. An illustration of the ADP regularizer during training is shown in Figure 27. By introducing more diversity between the models within the ensemble, the transferability of adversarial attacks on each of the models is more challenging. This in turn results in a higher robustness against adversarial attacks. Finally, while model ensemble mitigate SOTA adversarial attacks, they require higher computational effort and require the training process of the model to be adjusted extensively.

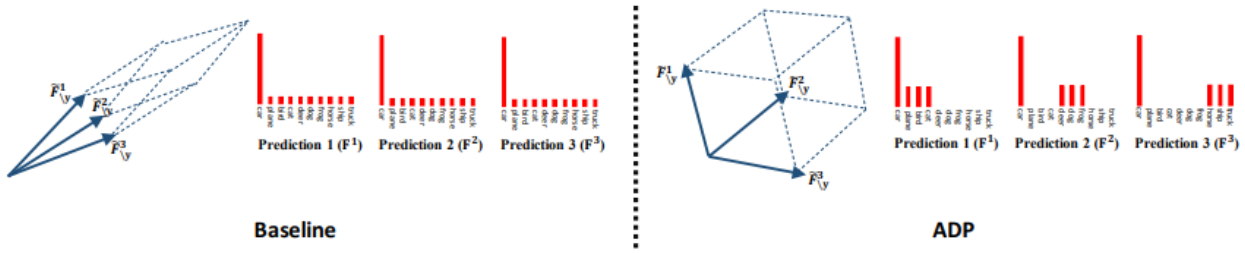


Figure 27 Side-by-side comparison between baseline model ensemble predictions and model ensemble predictions under an ADP regularizer from (86)

2.3.1.2.5 Detection of Adversarial Attacks

Instead of improving the robustness of a model, there are mitigation strategies that rather focus on the detection of adversarial examples. Hence, these methods aim to prevent the system from incorrect predictions by filtering out adversarial examples. Similar to proactive mitigation strategies it also shows for detection-based reactive strategies that stronger adversaries are able to break defense methods that are considered as secure at the time of release. For example, in (87) the authors build on the PGD attack and break four recently proposed methods for the detection of adversarial examples.

A method that is able to detect adversarial examples is called feature squeezing. As introduced in (88) the input data of a model can be squeezed to make adversarial noise more perceptible by the model. They suggest two methods for squeezing the original input samples in size and using a smoothing filter to change pixel values of the input image. The adversarial examples are detected by comparing the resulting predictions on both data samples.

In (89) the authors propose a detection method for both out-of-distribution samples and adversarial attacks. They assume that the pre-trained features of a model follow a class-conditional Gaussian distribution. Based on this assumption they use Gaussian discriminant analysis to measure the probability density of a test sample. They are able to achieve high detection rates against out-of-distribution samples by an $45{,}3\%$ increase compared to other detection methods. Additionally, on adversarial samples created by the C&W attack they are able to detect $95{,}8$ % correctly.

2.3.1.2.6 Input Transformation

Input transformation methods aim to reproduce the clean data by removing any adversarial perturbations before a data sample is presented to the model. In (90) a framework for regenerating the original image of an adversarial example is proposed. The authors utilize a GAN-based framework to project input samples onto a range of samples generated by the generator. Benign samples are expected to be closer to the generated samples than adversarial examples. By projecting the input sample onto the generated range, the adversarial perturbation is reduced. The projected sample is then presented to the model. An illustration of this is presented in Figure 28. This method has been found effective against adversarial attacks such as C&W and FGSM.

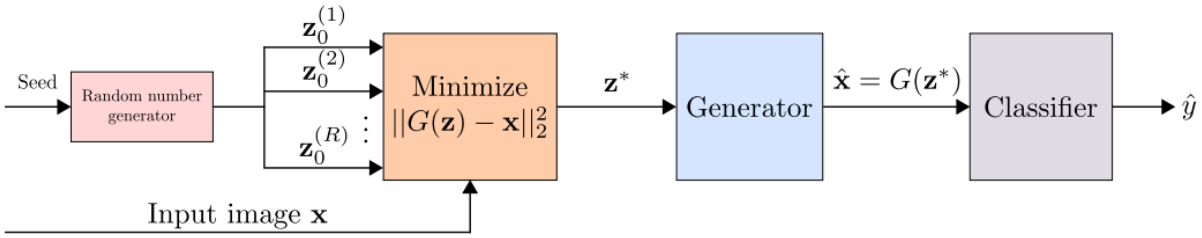


Figure 28 Illustration of defense-GAN from (90)

The authors of (91) propose high-level representation guided denoiser which is a method to remove adversarial examples through a denoiser model. This denoiser is trained to generate negative noise maps that

are based on the distance of the high-level representation of the example. These noise maps are added to the adversarial example to remove adversarial noise (i.e. reduce the distance of the high-level representation of the example). Furthermore, the authors show that the denoiser is trained more efficiently to provide robustness than adversarial training on the ImageNet dataset. Additionally, the denoiser can be transferred to other classifiers.

As stated above input transformation methods can be enhance the robustness of a model against adversarial perturbation, however as they adjust the input data before it is given to the model, they increase the computational effort needed for each classification.

2.3.1.3 Data Poisoning

We present current mitigation strategies for data poisoning attacks from Chapter 2.2.1.3. It is important to understand that the best mitigation is to ensure that all datasets are processed only inside the company and measures are taken such that it is not possible to exchange any data samples by an adversary. This also includes that the capturing of the dataset is controlled and it is not possible that images which contain malicious intent are inserted. However, this can often not be guaranteed or it would be too expensive or time consuming. Hence, in the following we present methods that can be applied if the correctness of a dataset cannot be guaranteed.

In general, this research field shows effects similar to the research on the mitigation of evasion attacks presented in Chapter 2.3.1.2. Multiple mitigation strategies are proposed that defend against the data poisoning attacks that existed at that time. However, many strategies are broken a short time later by stronger attack methods that exploit certain mitigation ideas. This results in an arms race between attackers and defenders. Hence, in (92) the authors combine a list of common pitfalls that should be avoided when evaluating new defenses. Most importantly, this includes evaluating against adaptive poisoning attacks which is the current SOTA when evaluating defenses against evasion attacks.

2.3.1.3.1 Categorization

Before presenting concrete mitigation strategies we first introduce a common taxonomy to categorize different methods. This is like our approach for data poisoning attacks in Figure 29. Most important are again the different defense types that exist. These will be used to structure the remainder of this chapter. Then, the time when a defense method is applied is relevant. Some methods are applied before training while others take a trained model and thus are applied after training is done. Similarly, the number of iterations that a mitigation strategy uses is important. Some are efficient to apply while others require extensive changes of the training process. Another important category is the type of data poisoning attacks that are prevented by a defense method. In the best case a defense method prevents data poisoning in general, but some methods only prevent specific attacks. Lastly, it is important to distinguish the data that a defense method requires. Here, some methods need access to at least some clean and correctly labeled data samples, while others also work if only the poisoned dataset is available.

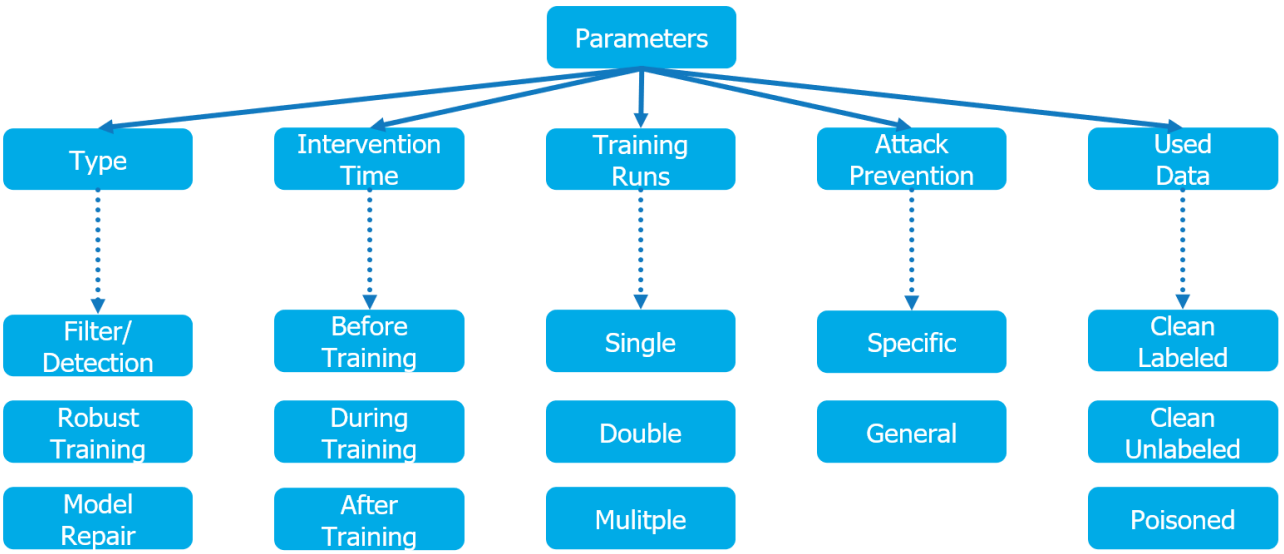


Figure 29 Categorization of mitigation strategies for data poisoning

2.3.1.3.2 Filter/Detection

The first category of mitigation strategies consists of methods that try to detect poisoned data samples and remove these from the dataset. This is first proposed in (93) and (94). In both methods a model is first trained on the complete dataset that might contain poisoned data samples. After the training is done all data samples from the training dataset are classified with the trained model and different parameters of the model are analyzed. Here, the authors analyze the learned representations or activations of the penultimate layer of the trained model. They find that those representations differ significantly on clean and poisoned data samples. Hence, they filter out all data samples that can clearly be distinguished from the majority of data samples based on their feature representations.

However, in (95) the authors are able to break both described methods. An adaptive attacker can optimize the indistinguishability of the feature representations of poisoned and clean data samples. The resulting poisoned data samples cannot be distinguished and the defense fails. In general, it shows that defense methods that are based on outlier detection are typically ineffective against an adaptive adversary. If the criterion to detect anomalous behavior can be integrated into the attack process an attacker can exploit this and break a defense. A similar observation is made in (96) where the authors consider defenses that perform the anomaly detection on the dataset and not on learned feature representation. They introduce a general approach to craft attacks that can evade anomaly detectors and are able to break various defenses that try to sanitize a dataset from poisoned data samples.

2.3.1.3.3 Robust Training

Methods for robust training represent the second category of defense strategies. In Figure 30 the effect of different defense methods is visualized in case of the targeted poisoning attack from (58). Here, the tested filter defenses are discussed previously in Chapter 2.3.1.3.2 and adversarial training represents the standard method from (22) discussed previously in Chapter 2.3.1.2.2. It is important to note that this method is also very successful at preventing targeted data poisoning and not only against evasion attacks. However, it has the disadvantage that the standard performance is significantly reduced. All remaining methods in Figure 30 are based on robust training and are presented in the following.

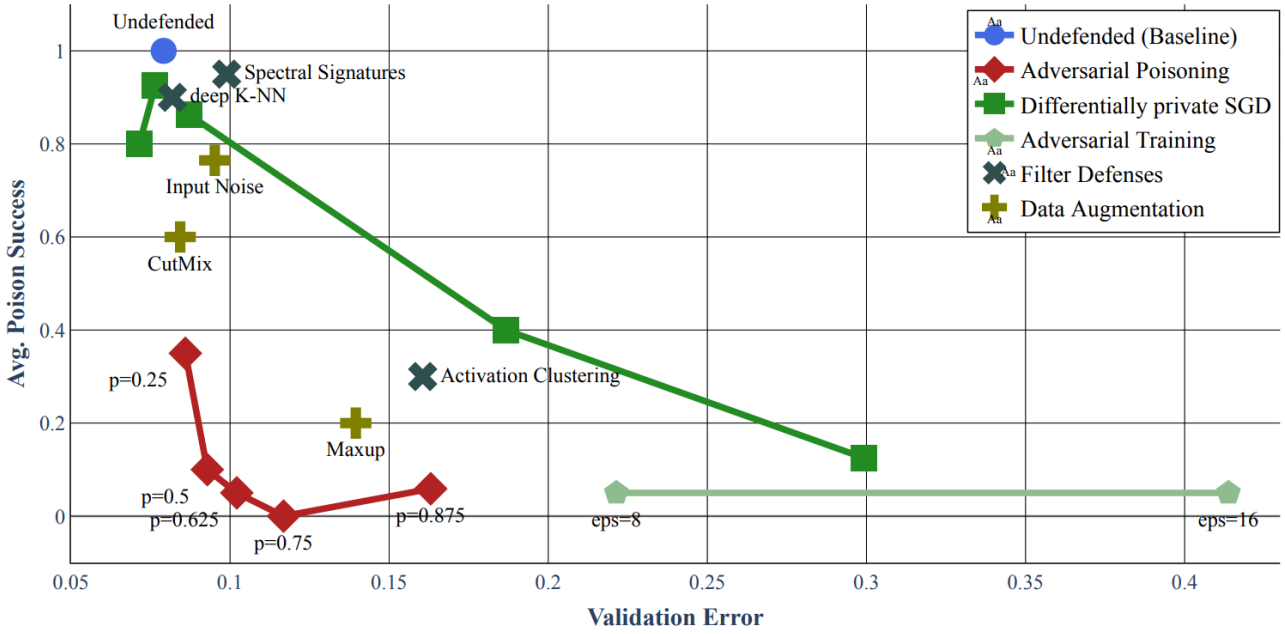


Figure 30 Overview of different defense methods against the targeted poisoning attack from (97)

The first method group uses standard data augmentations that are used to increase the performance of models on standard data samples. For the first time, in (98) the authors propose to use strong data augmentations as a simple but effective method to reduce the effect of backdoor and target poisoning attacks. They find that especially using the MaxUp augmentation technique from (99) reduces the success of data poisoning attacks. Thereby, MaxUp generates a set of augmented data samples for each original data sample in a dataset and only minimizes the worst-case loss over the augmented data samples.

Another group of methods is based on models that are trained with differentially private stochastic gradient descent (DP-SGD) introduced in (100). This method is originally introduced to ensure that learning is done privately and no individual examples overly influence the training. To do so the training gradients are clipped and Gaussian noise is added. In (101) these methods are extensively explored to mitigate data poisoning attacks. The authors find that using DP-SGD increases the robustness of models even when no meaningful privacy guarantees can be given.

The last method is recently proposed in (97). Here, the authors adapt standard adversarial training by generating the adversarial samples using a targeted data poisoning attack instead of an evasion attack. Their method achieves a similar robustness against data poisoning as standard adversarial training while suffering a smaller drop in performance on standard data.

Concluding, the authors in (102) apply the concept of randomized smoothing from (103) to defend against backdoor data poisoning attacks and provide a certifiable robustness bound. They provide the first benchmark for certified robustness against backdoor attacks but only consider binary classification problems. The concept of randomized smoothing is later discussed in Chapter 2.3.6.2.

2.3.1.3.4 Model Repair

Lastly, some mitigation strategies are based on repairing a model after this is trained on a dataset that might contain poisoned data samples. In (104) the authors try to remove neurons that are inactive on clean data samples. Hence, they test a model on data samples from a clean dataset and prune all neurons that show a low activation on these samples. In addition, they further fine-tune the pruned model. First, this recovers lost performance and secondly can change neurons that contain backdoor behavior which are not pruned earlier.

A similar, idea is also used by the authors in (105). Here, they first try to identify existing backdoors in a model and then try to reconstruct the trigger. Based on these results they propose multiple defense strategies. On the one hand, they try to prune neurons that are only active when the identified trigger exists in an image.

On the other hand, they try to unlearn the backdoor trigger by retraining the model on images that contain the reversed trigger.

However, for both described defense methods it is possible that adaptive attacks bypass the defense, because they exploit the distinguishability of clean and poisoned data samples based on the activity of neurons. An adaptive attacker can mimic the activity of neurons on clean samples on the poisoned data. For example, in (55) the authors show that their backdoor is not detected by NeuralCleanse.

2.3.2 Robustness

This chapter presents mitigation strategies that are specific to the challenges on robustness discussed in Chapter 2.2.2. Additionally, we present methods for the confidence or uncertainty estimation of DNNs which can be seen as a general mitigation strategy not focused on a specific threat.

2.3.2.1 Natural Perturbations

In this chapter we present mitigation strategies that can be used to increase the performance of DNNs on OOD data samples. Here, the most important category uses augmentation methods to improve the generalization of DNNs during training. As discussed in Chapter 2.2.2.1 augmenting the training dataset using only a single corruption type is not helpful for an increased performance against different corruptions. Hence, most recent methods utilize strong data augmentation and the combination of different augmentation operations. In (106) the authors combine the concept of MixUp from (107) with the use of standard augmentations for computer vision. An example of their resulting method is shown in Figure 31. It consists of applying different augmentation operations in parallel to an image. At the end, all different paths are combined via a weighted summation and the resulting image is a mixture of the effect of all augmentations. This method achieves a significantly increased robustness also on corruptions that are not used as part of the augmentation operations. In (67) the authors further improve this method by intruding a new powerful augmentation operation. Concretely, they pass a clean image through an image-to-image network and apply various random perturbations at different stages in the reconstruction process. Hence, the reconstructed image is not a perfect copy but will be visually diverse. They show that their method preserves the main semantics of an image and at the same time generates unique distortions. Alternatively, the authors in (108) introduce adversarial noise training. They first train a generative DNN that learns to find worst-case noise for a standard image classifier depending on the current input. Then, the generative DNN is used as noise generator and is incorporated in the training process. Here, they follow the method of standard adversarial training from (22) but use the noise generator to find a perturbation instead of the PGD attack. The fundamental method of adversarial training is previously discussed in Chapter 2.3.1.2.2.

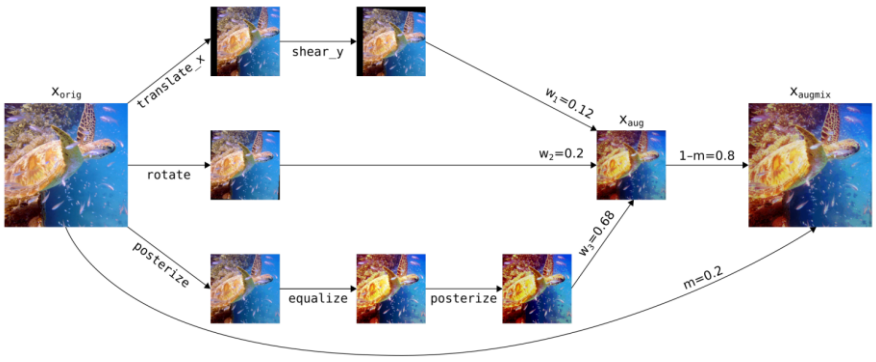


Figure 31 Overview of the augmentation strategy from (106)

A second category of methods for an improved performance on OOD data uses improvements to the architecture of DNNs. Here, one recent approach is (109) where the authors note that current CNN architectures are not shift invariant since commonly used methods for down sampling like MaxPooling ignore the Nyquist-Shannon sampling theorem. To fix this behavior they propose to include an anti-aliasing filter in MaxPooling layers. This leads to an improved performance on standard data and to a better

generalization on corrupted data samples. Another method is proposed in (110). Here, the authors integrate a technique for unsupervised online domain adaptation into the DNN architecture. They do this by estimating the statistics for batch normalization (introduced in (111)) online using recently observed data samples. Then, these values are used instead of the statistics that are estimated over the training dataset. This significantly improves the robustness of a DNN and can be combined with methods for data augmentation described previously for the best performance.

The final category for mitigation strategies for OOD generalization is based on online detection of data samples that are OOD. Hence, these methods do not increase the robustness of the predictions of a DNN but instead filter samples that deviate from the training distribution. If such samples are found these can then be passed to a specialized DNN or traditional algorithms that can handle a shift in the data distribution better. An overview of different methods for OOD detection is presented in (112). The authors define a standardized evaluation procedure to reliably assess performance improvements and perform an extensive evaluation of recently proposed methods based on this evaluation procedure. In general, one of the first methods is proposed in (113). The authors observe that the maximum softmax probability that is output by a DNN for classification tends to be higher on standard data samples than on OOD data samples. Hence, they define a threshold on the softmax probability and assume that all data samples with a lower value come from a OOD distribution. In (114) a more advanced method is introduced. The authors propose a decomposition of the standard softmax layer and introduce a preprocessing step that is applied on all data samples during inference. Both improvements help to significantly improve the detection performance. In addition to specific methods for OOD detection it is possible to detect OOD samples based on the general confidence of a DNN. In Chapter 2.3.2.2 we present different methods to reliably estimate the confidence of a DNN which can be used to detect OOD data samples when the confidence is below a predefined threshold.

2.3.2.2 Confidence Estimation

As a last category of mitigation strategies against robustness or IT-Security challenges, it is possible to use methods for confidence estimation of DNNs. For each prediction these methods also provide a confidence or uncertainty value that should be correlated with the actual reliability of a prediction. Then, this information can be used to detect whether the current prediction of a DNN can be trusted. In the best case the confidence of a DNN is high on in-domain data samples and is lower on OOD data samples or adversarial perturbed data samples. In these cases, it is then possible to contact an alternative backup system if the confidence of a DNN is below a certain threshold. This concept is especially important to incorporate safety systems into an entire AD system. For example, it is possible to use DNN-based systems most of the time for performance reasons and contact an alternative verifiable safety system if the prediction of the DNN is not reliable in a certain situation.

One important category for confidence estimation are methods that are based on using an ensemble of models instead of only a single model. This is first proposed in (115) where the authors use dropout layers from (116) not only during training. Instead, dropout stays active during inference and an ensemble of models is generated indirectly by predicting each data sample multiple times. Then the final prediction of the model is the average of all individual predictions, which allows to estimate the uncertainty. The authors in (117) take this approach even further and use a real ensemble. Hence, they train multiple models for the same task and average the prediction of the individual models during inference. This has the disadvantage that multiple models must fit in the available memory of the compute device at the same time and thus the authors in (118) propose a more efficient method. Instead of using multiple models they expand a single model. For each layer the normal weight matrix is shared among all members of the ensemble, but additional rank-one weight matrices are introduced for each member. This allows to compute the prediction of the entire ensemble using a single forward pass through the network by repeating the input data sample over a batch.

Another category of methods for confidence estimation consists of methods that introduce different probabilistic layers in a DNN that output a more meaningful probability distribution than using the standard softmax activation. Here, the authors in (119) exchange the softmax activation in the last layer with a rectified linear unit (ReLU) activation and learn an evidence score for each class. This score is then used to place a

Dirichlet distribution over the class probabilities, which allows to better estimate the uncertainty of each classification. Alternatively, it is also possible to use Bayesian neural networks (BNNs), where the weights are no longer deterministic but instead model a probability distribution, for example in (120). However, BNNs typically underfit the dataset and lead to at least a doubling of the model parameters which prevents their use on high dimensional data or in resource-constrained environments. Hence, the authors in (121) propose an efficient approximation of a BNN. They recycle the approach from (118) and model the member specific rank-one weight matrices using a Gaussian distribution instead of deterministic weights. Therefore, they combine the idea of BNNs with techniques for efficient ensembles.

A third class of methods for confidence estimation also changes the architecture of a DNN, but without introducing probabilistic concepts. For example, the authors in (122) introduce a hybrid model combining a DNN and the k-nearest neighbors algorithm. For each data sample during inference, they estimate the distance to the closest data samples seen during training and generate a confidence score based on this distance. A simpler method is proposed in (123). Here, the authors introduce temperature scaling which adds a scaling coefficient to the final logit layer of a DNN before passing the results through the final softmax activation. This scaling coefficient is optimized on a hold-out dataset and improves the general calibration of the probabilities.

Lastly, there are also methods that introduce a single metric which represent the confidence score of a DNN. In (124) the authors add an additional output to a DNN which learns the confidence value for each prediction. Hence, they train this single value to represent whether the classification of the DNN is correct or likely incorrect. In contrast, the authors in (125) introduce a score that can be calculated for a trained DNN. They first use gradient information to generate the pixel-wise feature importance. Then, this is used to generate perturbed data samples and the confidence is generated by averaging over all perturbed samples. Since gradient information is required to efficiently compute the attribution this method can only be used for white-box systems. This is further improved upon by the authors in (126) that extend the approach to estimate the confidence of black-box systems and enable a more efficient use in mobility applications.

2.3.3 Explainability

In this chapter we discuss the explainability of AI systems and present an overview of the most relevant methods from current research. We focus mainly on methods that are applicable to explain and interpret DNNs, since these are primarily used in AD systems, as we describe in Chapter 2.4.1.1. First, we introduce a taxonomy to categorize different methods for explainability that is derived from recent surveys on explainability. Then, methods for local explainability are discussed which try to explain the output of an AI system on a single input sample. Afterwards, we discuss methods for global explainability which try to explain the general behavior of an AI system instead of justifying the prediction of a single data sample.

2.3.3.1 Categorization

Most surveys on explainability discussed in Chapter 2.2.3 derive their own taxonomy for different explainability methods. We try to combine these taxonomies and present an overview in Figure 32 that allows to categorize each explainability method which is relevant in the context of AI for mobility systems. This allows to compare different methods and easily determine which methods are feasible and handle the complexity of the different mobility use cases which are described in Chapter 2.4.

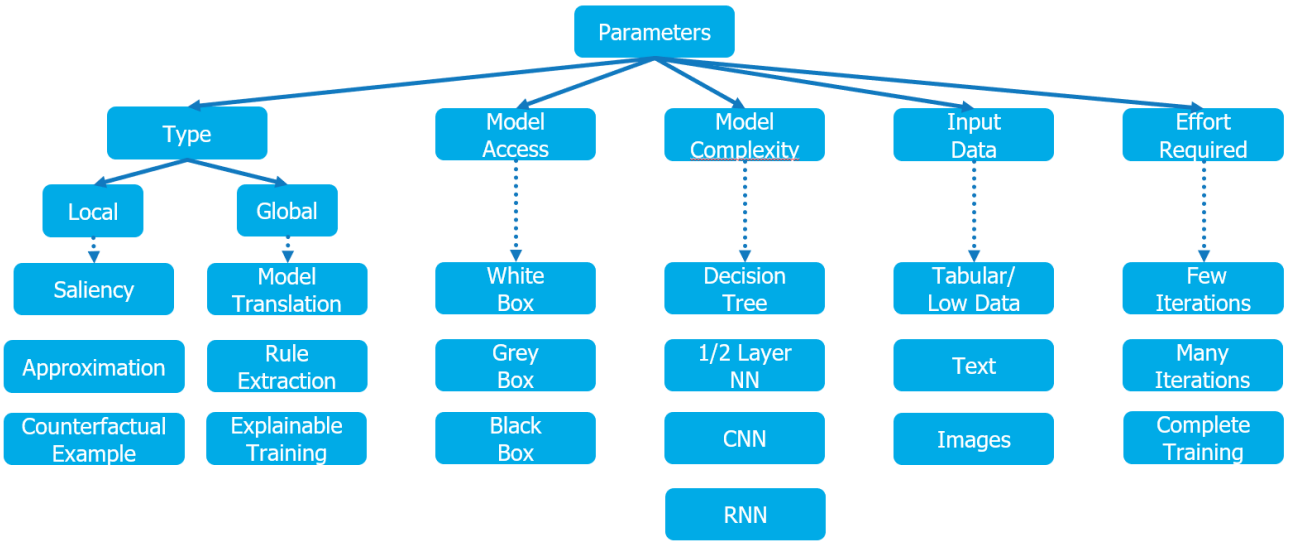


Figure 32 Categorization of methods for explainability

The first category is most important and groups explainability methods by the type of explanation they provide. Here, local and global methods can be distinguished and we organize Chapter 2.3.3.2 and Chapter 2.3.3.3 using this categorization. Next, the required model access of explainability methods needs to be considered. Some methods are model-agnostic and can be used to explain black-box systems, while others require information of the system or gradients and can only be used to explain white-box systems. Another important category to distinguish explainability methods is the feasible model complexity. On the one hand it is only possible to apply some methods on tree-like systems. On the other hand, there are also methods that can be used to explain complex systems based on large DNNs. Similarly, also the feasible input data differs a lot between different methods. A lot of methods can only be applied reasonably on tabular-like data that has a low dimensionality. Explaining image-based systems is harder due to the high-dimensionality and nonexistence of simple feature categories. Lastly, the required effort to execute an explainability method varies a lot. For some methods, a single forward pass of a data sample is enough, while others require extensive approximation using many samples.

In addition to the already presented general surveys on explainability there are also some publications that focus on a specific type of explainability. Here, the authors in (127) focus on methods that provide causal explanations. They present a summary of the literature from this viewpoint and consider both traditional and DNN-based systems. Focusing on the application of explainability methods to AD, the authors in (128) and (129) first provide a general survey on different methods for explainability. Next, they focus on the application on the different tasks that are relevant for AD and present a summary of publications that applied some form of explanations in the context of AD.

2.3.3.2 Local Methods

Methods for local explainability try to explain the decision of a system on a single data sample. Hence, they do not cover the entire application domain and need to be applied repetitively. This kind of explanation is useful to understand a concrete prediction and provide justification of a system for a concrete data sample. Also, methods for local explainability are easier to scale to complex DNNs than trying to explain the behavior of a DNN globally. This is the main advantage and the reason most research work focuses on local explainability of DNNs.

2.3.3.2.1 Saliency

The most prominent methods for local explanations of DNNs are saliency-based methods. Typically, these generate a heatmap on the input data which is the visualization of features that are relevant for the prediction. Some examples of different methods for saliency-based explanation are shown in Figure 33. In general, the main advantage of saliency-based methods is that these can be applied to any DNN and on high dimensional

data. Therefore, the focus of research lies on such methods and improvements are continuously introduced in recent years.

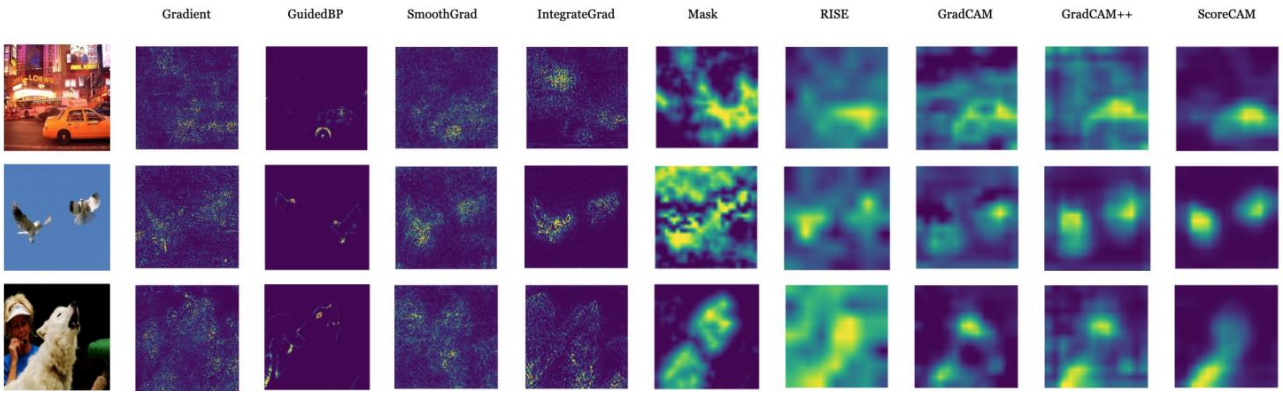


Figure 33 Exemplary visualization of different saliency methods from (130)

Current methods for saliency-based explanations typically require white-box access to the system. One kind of methods is based on using gradient information to aggregate the contribution of internal components of a DNN to a prediction. Here, some current methods are (131) and (132)[1]. These can be easily applied to any white-box DNN and only require a few iterations of passing data samples through a DNN. Alternatively, another kind of promising methods is based on visualizing the internal activation patterns of a DNN and projecting this to the input. Here, some recent methods are (130) and (133). Additionally, a third kind is based on backpropagating the prediction output of a DNN through each layer back to the individual features. For this kind, (134) and (135) are recently proposed methods. The three method kinds presented are mostly used and most research work is focused in these areas.

Nevertheless, there exist also other approaches that use different concepts to generate saliency maps. An interesting one is proposed in (136) where the authors use adversarial attacks that generate coherent visual perturbations which is very similar to the concept of physical adversarial attacks discussed in Chapter 2.2.1.2.5. Also, interestingly the authors in (137) train a separate DNN that upscales the feature maps learned in the original DNN. They train their model to learn a saliency map that can be used in parallel to the original DNN. However, the title of both publications is slightly misleading because they still require gradients to perform the adversarial attack or train the upscaling model. Hence, these methods require white-box access and cannot be used to explain true black-boxes.

Due to the success of saliency-based methods there are also publications that investigate whether the provided explanations can be trusted. In (138) the authors analyze different saliency methods and discuss that only relying on the visual inspection of saliency maps can be misleading. They show that some methods, based on the guided backpropagation of the DNN output to the input data introduced in (134), are largely invariant to the parameters of a DNN and basically just act as an edge detector in images. To increase the practical trust in saliency-based methods they then propose two simple tests that assess the quality of saliency-based explanations.

2.3.3.2.2 Approximation

Methods for approximation typically try to locally approximate a larger system by inherently explainable models. Hence, most methods describe the local behavior of a system by a linearly weighted combination of the input features. This allows to explain the impact of each feature in a very understandable fashion. However, the linear combination also represents the main disadvantage because these do not scale well to input data with high dimensionality like images or videos. Hence, the applicability to DNNs and specially to explain image data is very limited and typically only allows very rough explanations based on super-pixels from (139).

[1] The open-source project is available at: https://github.com/PAIR-code/saliency

The first method that used a local linear approximation is LIME proposed in (140). The authors train a linear model by generating new samples locally around the data sample that should be explained and observing the output of the original system. The approximated model is locally correct and is used to explain the decision of the original system for this specific data samples. An alternative is proposed in (141) where the authors generate so called "anchors", which are those features that are mainly responsible for the prediction of a system locally. Any changes to the rest of the features does not have an influence on the prediction of the system. Again, this method does not scale well to high dimensional input data and requires an intensive search procedure to find the right "anchors". A further alternative is DeepSHAP proposed in (142). The authors use a linear approximation of a system and calculate the importance of input features based on Shapley values. Therefore, this method suffers from the same disadvantages discussed previously.

2.3.3.2.3 Counterfactual Example

Another type of methods where research interest picked up in recent years are explanations based on counterfactual examples. These methods try to answer the question "How should a specific data sample be changed such that the prediction of a system changes to a specific value or class?". Thereby, it is also important that the change of the data sample is easy to explain for humans. For example, this allows to explain a classifier by statements of the form "If the pupils of the animal in the image were made larger the probability of a cat would decrease by 20%". Hence, these approaches provide local explanations for a specific image and show exactly the impact of a group of features. An example is shown in Figure 34 where the explanation of the saliency-based method Grad-CAM is compared with a counterfactual example from the method GANalyze for a Lion vs. Cheetah classifier. Typically, methods to generate counterfactual examples have similar requirements to saliency-based methods discussed in Chapter 2.3.3.2.1. They require white-box access but can be applied to large DNNs with high-dimensional input data.

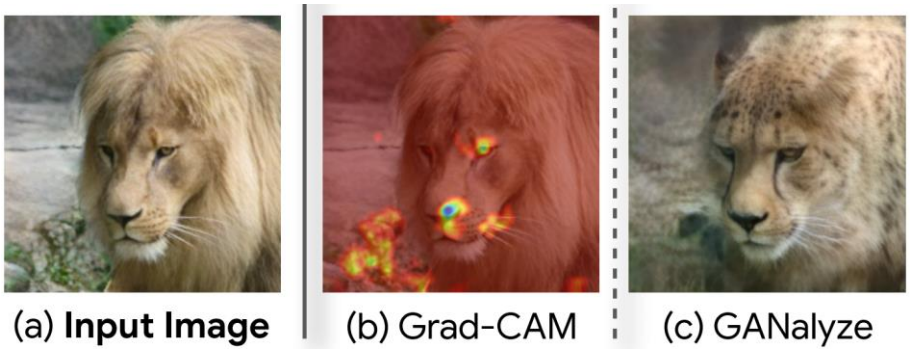


Figure 34 Visualization of the explanations provided by counterfactual examples in comparison to saliency-based explanations from (143)

One of the first publications to use counterfactual examples to explain vision-based systems is (144). Here, the authors explain why an image was classified as a class c rather than class c'. To do so, they replace the most relevant region of the input image for the classification c with a part from an image of class c'. The resulting image would then be classified as class c' and the generated explanation provides information about the mainly responsible feature region. Alternative methods to generate counterfactual explanations are based on generative DNNs instead of exchanging specific image parts. Here, (145) and (143) both use GANs to synthetically generate the parts of the image that must be changed instead of extracting these from an existing image of class c'. The basic principle behind GANs is discussed in more detail in Chapter 2.6.3.

Also, there is an approach that takes the opposite direction of counterfactual examples. In (146) the authors explain the prediction of a system by showing the training data samples that are most responsible for this prediction. Instead of generating a counterfactual example they provide multiple reinforcing examples from the train dataset.

2.3.3.3 Global Methods

In contrast to methods for local explanations, methods for global explanations try to explain the complete behavior of a system. Hence, these do not only explain the prediction on a single data sample but provide a model that is able to explain each prediction. Therefore, methods for global explainability achieve perfect coverage. However, it is significantly more difficult to explain a complex system like a DNN globally instead of providing a local explanation. Most of the methods that are presented in the following are limited in their applicability or only achieve poor performance in comparison to the original system.

2.3.3.3.1 Model Translation

The idea behind methods for model translation is to transfer the behavior of a complex system to a simpler model that is more explainable. Hence, the explainable model should mimic the behavior of a complex DNN as best as possible. Here, one approach is to train a surrogate model that is inherently more interpretable. In (147) the authors train a gradient boosted decision tree for any given black-box model. In principle, they perform a model extraction attack and then provide explanations for the extracted model. These explanations are again based on SHAP values that we already discussed in Chapter 2.3.3.2.2. Hence, this method has the same disadvantages and cannot be scaled well to high dimensional data samples like images.

Alternatively, there are also approaches that integrate explainable models in a DNN and use a hybrid system. In (148) the authors insert multiple autoencoder structures in a CNN to extract a limited number of high-level latent features. The extracted features are then used to construct a Bayesian network that shows the exact effect of each extracted feature. Hence, the feature extraction is still performed by a CNN, but the final prediction is made in an understandable way by a graphical model. This represents a tradeoff between increasing the explainability of a system but consequently reducing the prediction performance. A similar method is also used in (149), where the authors learn a graphical model that represents the learned features of a CNN. They use this to visualize the knowledge hierarchy learned in a CNN, which can be used to explain the learned representation of a CNN.

2.3.3.3.2 Rule Extraction

Methods based on rule extraction try to extract the exact decision rules from a complex system and build an exact replica of the system. If such methods are applied for DNNs typically decision trees are used that model the behavior of each neuron depending on the current input values. Then, all trees are combined and intermediate rules are extracted that simplify the resulting tree. As a result, these methods provide perfect explainability because a decision tree is used instead of a DNN and for these each prediction is perfectly traceable. However, these methods are extremely limited and can only be applied on shallow networks. Also, they do not scale well with the input dimensionality. Hence, explainability based on rule extraction is more an academic research field than something that can be used well in praxis.

This conclusion is also supported by the authors in (150), which review different methods for rule extraction of DNNs. They conclude that only a limited number of publications consider this problem and satisfactory solutions do not exist. This also shows in the evaluation of the currently most promising method DeepRED in (151). The authors experiment with a binary classification task for two classes from the MNIST dataset introduced in (152). Even for this purely academic task they first must prune the DNN and the extracted decision tree only achieves a fidelity of $\approx 90\%$.

2.3.3.3.3 Explainable Training

Instead of trying to explain a system after training there are also approaches that incorporate explainability directly in the system during training. Most notably in (153) the authors consider the task of providing explanations for an end-to-end AD system. See Chapter 2.4.2 for a general introduction to these systems. Here, the authors consider the task of predicting the steering angle of a vehicle from monocular images. They use a DNN and incorporate visual explanations based on saliency maps into the architecture. In Figure 35 the resulting architecture is shown and an exemplary explanation is visualized. The architecture consists of a

standard CNN to perform the feature extraction from the raw input images and a following LSTM to predict the steering angle based on the current input and remembered information of the last seen input images. Additionally, the authors insert an attention mechanism that is trained to visualize the regions in the learned feature representation that are most relevant for the current prediction. As a result, this attention map can be upscaled to the size of the original image which results in a saliency map. This allows a human to observe the regions in the input image that had the most influence on the current prediction.

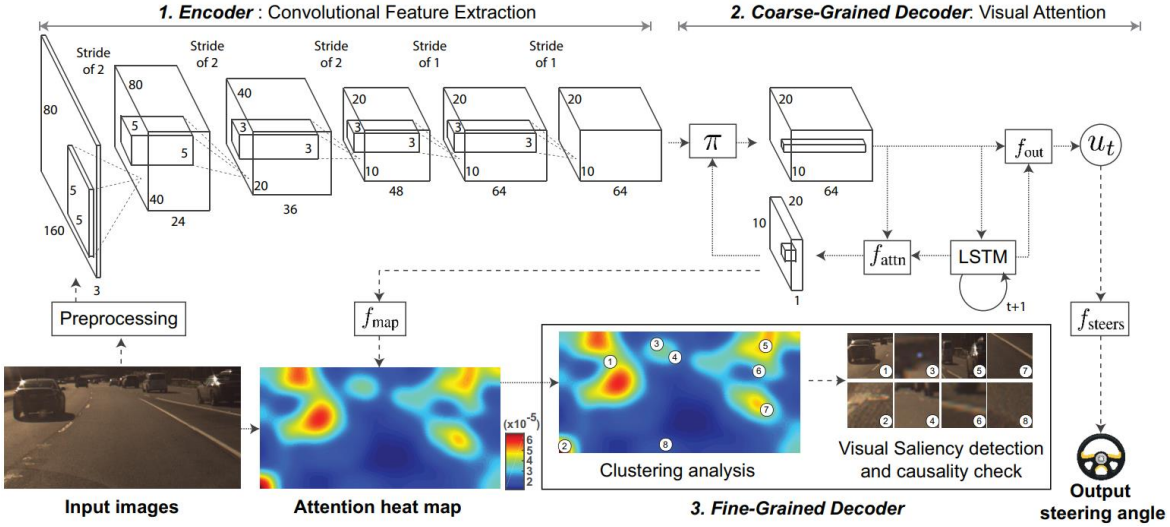


Figure 35 Overview of an explainable AD system from (153)

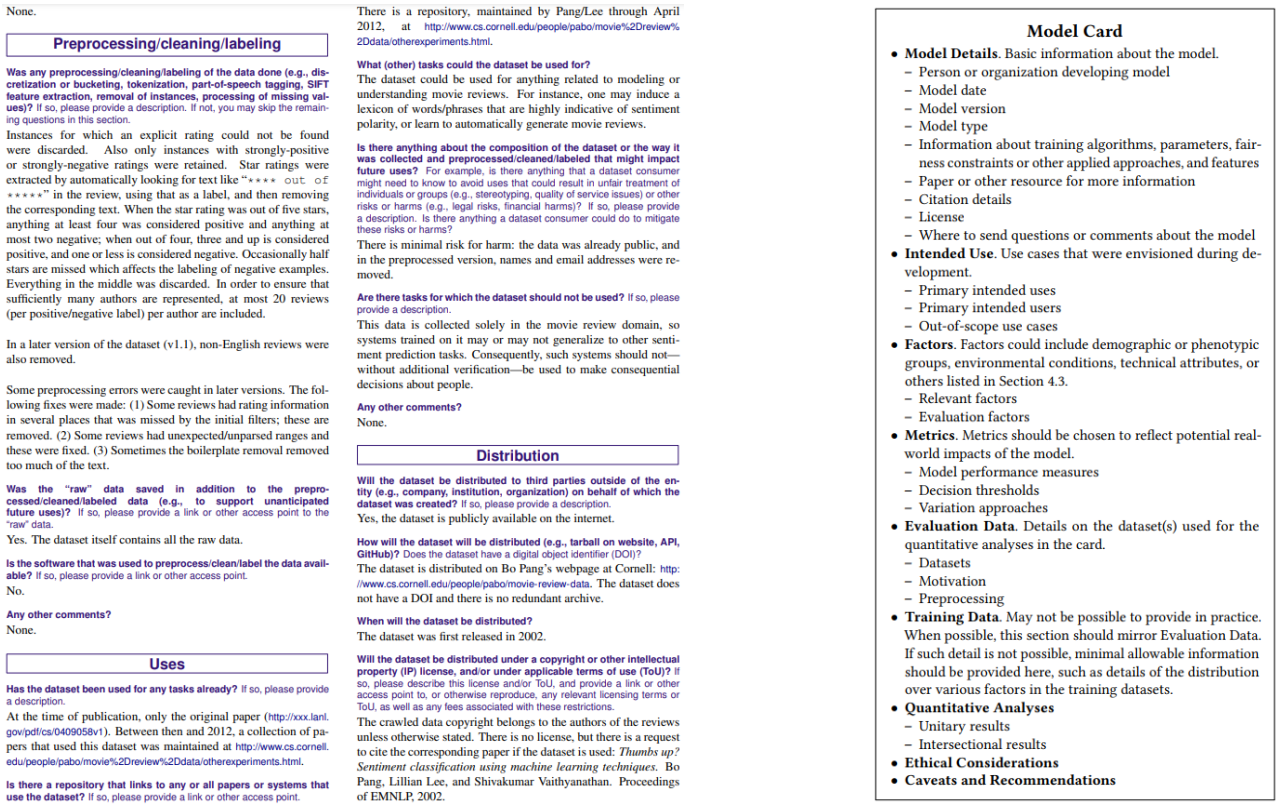
In another recent approach the authors in (154) learn a weighted linear combination of individual DNNs. Thereby, each DNN only has a single input feature which allows to explain the overall prediction of the combination of DNNs. Since each prediction results from a weighted combination and each DNN has a single input feature the impact of each input feature on the final prediction can easily be explained. However, this approach only works well for tabular-like data and currently does not scale to high dimensional data.

2.3.4 Documentation

This chapter summarizes potential strategies for the challenge of a uniform documentation of AI-based systems discussed in Chapter 2.2.4. Additionally, we present software tools that can help to document the development process, the evaluation of AI systems and track the complete AI lifecycle.

2.3.4.1 Proposals for Unification

Proposals that gained the most attention in recent years come from leading industry players and are based on their practical experience. On the one hand, several works discuss the need for standardized documentation of the datasets used for the training of data-driven systems. In (155) the authors propose to create a mandatory datasheet that accompanies every release of a dataset. They provide a question catalogue that covers all aspects that are relevant during the data acquisition and processing, ranging from the fundamental motivation over the collection process to the distribution and maintenance. Some examples of the question catalogue are shown in Figure 36 on the left side. The datasheets are also created for datasets that are only available internally and can be handed to external parties on request. This increases transparency, accountability and reproducibility. Alternatively, in (156) the authors also consider the problem of dataset quality, but from a specification and verification perspective. First, they provide an overview of existing standards for datasets in certain safety-critical domains. These standards can only partially be applied to AI systems due to the increased input dimensionality, inability to formally verify strict completeness and data-dependent behavior. Following, the authors provide a list of recommendations for the management of datasets and propose the usage of three additional artifacts for the documentation of datasets, consisting of the definition standard, the requirement specification and the verification plan.



None.

Preprocessing/cleaning/labeling

Was any preprocessing/cleaning/labeling of the data done (e.g., discretization or bucketing, tokenization, part-of-speech tagging, SIFT feature extraction, removal of instances, processing of missing values)? If so, please provide a description. If not, you may skip the remaining questions in this section.

Instances for which an explicit rating could not be found were discarded. Also only instances with strongly-positive or strongly-negative ratings were retained. Star ratings were extracted by automatically looking for text like "★★★★ out of ★★★★★" in the review, using that as a label, and then removing the corresponding text. When the star rating was out of five stars, anything at least four was considered positive and anything at most two negative; when out of four, three and up is considered positive, and one or less is considered negative. Occasionally half stars are missed which affects the labeling of negative examples. Everything in the middle was discarded. In order to ensure that sufficiently many authors are represented, at most 20 reviews (per positive/negative label) per author are included.

In a later version of the dataset (v1.1), non-English reviews were also removed.

Some preprocessing errors were caught in later versions. The following fixes were made: (1) Some reviews had rating information in several places that was missed by the initial filters; these are removed. (2) Some reviews had unexpected/unparsed ranges and these were fixed. (3) Sometimes the boilerplate removal removed too much of the text.

Was the "raw" data saved in addition to the preprocessed/cleaned/labeled data (e.g., to support unanticipated future uses)? If so, please provide a link or other access point to the "raw" data.

Yes. The dataset itself contains all the raw data.

Is the software that was used to preprocess/clean/label the data available? If so, please provide a link or other access point.

No.

Any other comments?

None.

Uses

Has the dataset been used for any tasks already? If so, please provide a description.

At the time of publication, only the original paper (http://xxx.lanl.gov/pdf/cs/0409058v1). Between then and 2012, a collection of papers that used this dataset was maintained at http://www.cs.cornell.edu/people/pabo/movie%2Dreview%2Ddata/otherexperiments.html.

Is there a repository that links to any or all papers or systems that use the dataset? If so, please provide a link or other access point.

There is a repository, maintained by Pang/Lee through April 2012, at http://www.cs.cornell.edu/people/pabo/movie%2Dreview%2Ddata/otherexperiments.html.

What (other) tasks could the dataset be used for?

The dataset could be used for anything related to modeling or understanding movie reviews. For instance, one may induce a lexicon of words/phrases that are highly indicative of sentiment polarity, or learn to automatically generate movie reviews.

Is there anything about the composition of the dataset or the way it was collected and preprocessed/cleaned/labeled that might impact future uses? For example, is there anything that a dataset consumer might need to know to avoid uses that could result in unfair treatment of individuals or groups (e.g., stereotyping, quality of service issues) or other risks or harms (e.g., legal risks, financial harms)? If so, please provide a description. Is there anything a dataset consumer could do to mitigate these risks or harms?

There is minimal risk for harm: the data was already public, and in the preprocessed version, names and email addresses were removed.

Are there tasks for which the dataset should not be used? If so, please provide a description.

This data is collected solely in the movie review domain, so systems trained on it may or may not generalize to other sentiment prediction tasks. Consequently, such systems should not—without additional verification—be used to make consequential decisions about people.

Any other comments?

None.

Distribution

Will the dataset be distributed to third parties outside of the entity (e.g., company, institution, organization) on behalf of which the dataset was created? If so, please provide a description.

Yes, the dataset is publicly available on the internet.

How will the dataset will be distributed (e.g., tarball on website, API, GitHub)? Does the dataset have a digital object identifier (DOI)?

The dataset is distributed on Bo Pang's webpage at Cornell: http://www.cs.cornell.edu/people/pabo/movie-review-data. The dataset does not have a DOI and there is no redundant archive.

When will the dataset be distributed?

The dataset was first released in 2002.

Will the dataset be distributed under a copyright or other intellectual property (IP) license, and/or under applicable terms of use (ToU)? If so, please describe this license and/or ToU, and provide a link or other access point to, or otherwise reproduce, any relevant licensing terms or ToU, as well as any fees associated with these restrictions.

The crawled data copyright belongs to the authors of the reviews unless otherwise stated. There is no license, but there is a request to cite the corresponding paper if the dataset is used: *Thumbs up? Sentiment classification using machine learning techniques.* Bo Pang, Lillian Lee, and Shivakumar Vaithyanathan. Proceedings of EMNLP, 2002.

Model Card

- **Model Details**. Basic information about the model.
  - Person or organization developing model
  - Model date
  - Model version
  - Model type
  - Information about training algorithms, parameters, fairness constraints or other applied approaches, and features
  - Paper or other resource for more information
  - Citation details
  - License
  - Where to send questions or comments about the model
- **Intended Use**. Use cases that were envisioned during development.
  - Primary intended uses
  - Primary intended users
  - Out-of-scope use cases
- **Factors**. Factors could include demographic or phenotypic groups, environmental conditions, technical attributes, or others listed in Section 4.3.
  - Relevant factors
  - Evaluation factors
- **Metrics**. Metrics should be chosen to reflect potential real-world impacts of the model.
  - Model performance measures
  - Decision thresholds
  - Variation approaches
- **Evaluation Data**. Details on the dataset(s) used for the quantitative analyses in the card.
  - Datasets
  - Motivation
  - Preprocessing
- **Training Data**. May not be possible to provide in practice. When possible, this section should mirror Evaluation Data. If such detail is not possible, minimal allowable information should be provided here, such as details of the distribution over various factors in the training datasets.
- **Quantitative Analyses**
  - Unitary results
  - Intersectional results
- **Ethical Considerations**
- **Caveats and Recommendations**

Figure 36 Exemplary visualization of datasheets from (155) on the left and model cards from (157) on the right

In addition to research on the documentation of datasets there is also some work to standardize the documentation of a trained AI system. In (157) the authors propose to provide documentation for each release of an AI system, which should describe the performance characteristics. Their model cards contain the basic information of the model architecture, the training parameters and a detailed evaluation in a variety of conditions in the intended domain. For each of the categories they provide guidance in determining a useful question catalogue from which model cards can be derived. An example is shown in Figure 36 on the right side. This clarifies the intended use case and shortcomings of each model leading to improved transparency.

Combining the different areas where documentation of AI systems is required, the authors in (158) and (159) propose a documentation for AI systems that is inspired by supplier's declarations of conformity that are used in many industries for traditional (hardware) products. These include not only the final performance of a system but also capture various aspects about the development, the testing, the robustness or the safety and security. Again, the authors provide a question catalogue that can be used to create FactSheets for any AI system and provide detailed examples from their experience. The proposed documentation forms the basis that can be used to get an overview of a system. For the specific threats on IT-Security from Chapter 2.2.1 and robustness from Chapter 2.2.2, it would be possible to extend the original FactSheets to also contain more detailed information on the implemented mitigation strategies of the described threats and the evaluation against exemplary attacks.

Similar to the previous concepts, the authors in (12) introduce a framework for the auditing of AI systems that should be done internally at companies developing AI systems. At each stage of the lifecycle described in Chapter 2.1 a set of documents is required before moving further in the lifecycle. Combined, these documents form an overall audit report to assess decisions made during the development process, the evaluated test scenarios and the quality of the AI system. An overview of the introduced auditing and documentation process is shown in Figure 37. The authors propose to use the introduced framework to close the accountability gap during large-scale deployment. They provide detailed descriptions of each step in the internal development and parallel audit of an AI system and mention the required documentation at each step. The framework for the documentation of AI systems is derived by combining lessons learned from auditing practices in other industry areas.

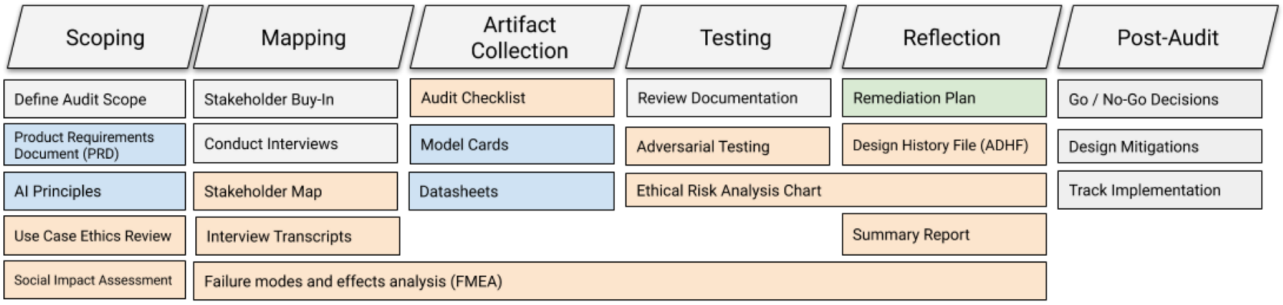


Figure 37 Overview of the proposed internal auditing process and associated documentation from (12)

In (11) the authors present the ABOUT ML[2] project which is an initiative to increase the transparency of AI systems across different stakeholders. They focus on the need for uniform documentation of the AI lifecycle and standardized practices. The goal of the project is to consolidate past efforts for standardized documentation and develop guidelines and templates to support documentation of an AI system. This should lower the barrier to integrate thorough documentation in any AI development process for any industry. To achieve this, during the project different pilot use cases are explored to test the applicability of the proposed documentation guidelines in practice. Additionally, the project wants to develop means to tackle the inherent transparency limitations resulting from intellectual property protection or information security.

2.3.4.2 Software Tools

To support the documentation of AI systems different tools exist that allow to track an AI system over the complete lifecycle. In recent years multiple commercially available platforms were developed that provide features ranging from data visualization over experiment tracking (including data and model versioning) to code packaging for deployment. Heavily used and funded platforms include (160), (161) and (162). However, more platforms exist where the focus is slightly different or less functionality is available. The platforms mentioned also provide more advanced features like feature importance analysis, visual explainability or automated parameter search. Also, the upscaling of AI operations in a company is a focus-point where the platforms provide simple user interfaces, easy-to-use apps and the option to perform no-code experiments. One limiting factor in using more advanced features is that in most cases only tabular and low dimensional data is heavily supported. Explaining or evaluating vision or point cloud AI systems is more difficult due to high input dimensionality and large system architectures as discussed in the associated chapters in this report.

In contrast to the previous commercial platforms for AI system tracking there are also open-source alternatives that are largely developed by an independent community. Here, prominent examples include (163) and (164). These provide the same basic features but do not yet include more advanced features. Nevertheless, the basic need of tracking an AI system is fulfilled by having a model registry and training parameter tracking. The usability is more focused on expert users and not on an as easy as possible integration and development of AI systems. Still, the mentioned open-source platforms are a viable alternative that is also used in the industry and consistently new features and improvements are introduced.

The mentioned platforms currently mainly support the internal documentation of the lifecycle of an AI system and the tracking of experiments including training and evaluation. Companies might use this information to produce the reports mentioned in Chapter 2.3.4.1 and expand it with additional information. These platforms provide the basis to monitor the AI lifecycle internally and aggregate information to provide to third parties.

2.3.5 Safety

Any safety requirement must consider the limitations mentioned in Chapter 2.2.5 in order to manage and minimize the risks. In the automotive sector, there are several major norms, standards and frameworks with respect to safety driven design and development (functional safety) of automotive products. One of the most

[2] The project homepage is available at: https://partnershiponai.org/workstream/about-ml/

common standards for functional safety is the ISO 26262 (165) including functional safety requirements for electronic systems embedded in road vehicles. The major approach of the ISO 26262 is shown in Figure 38 and is a risk driven approach that is comprised of requirement specification, architecture, implementation, integration, verification and validation.

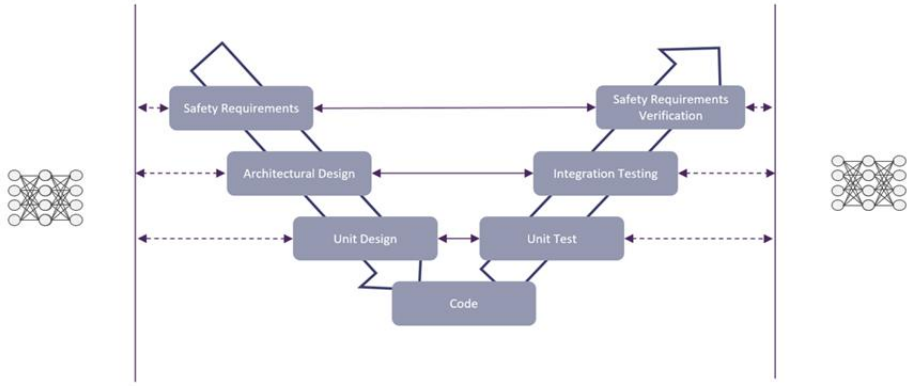


Figure 38 V-Model approach of the ISO 26262 requires mapping to the ML specific properties

Furthermore, the standard provides different automotive safety integrity levels (ASILs) where each level represents a set of requirements regarding the acceptable residual risk. The higher the ASIL, the higher the safety requirements and the lower the residual risk of the application. In general, high ASILs require a high transparency, explainability, test coverage and (semi-) formal verification of the systems design and functionality. Unfortunately, all these requirements collide with the limitations of ML-based systems listed in Chapter 2.2.5 and lead to different strategies to achieve the demands. Figure 39 summarizes some strategies to map and cover ML- specific aspects with safety requirements.

Safety Strategy	Practical AI Safety Opportunities
Inherently Safe Design	Design Specification ((Amodei et al. 2016; Leike et al. 2017; Seshia et al. 2018))
	Implementation Transparency ((Bojarski et al. 2018; Zeiler and Fergus 2014; Adebayo et al. 2018))
	Formal Verification ((Seshia, Sadigh, and Sastry 2016; Dvijotham et al. 2018; Wang et al. 2018))
Safe Fail	Uncertainty Estimation ((Lakshminarayanan, Pritzel, and Blundell 2017; Gal and Ghahramani 2016))
	In-distribution error detection ((Geifman and El-Yaniv 2017; 2019; Guo et al. 2017))
	Out-of-distribution error detection ((Mohseni et al. 2020; Golan and El-Yaniv 2018; Vyas et al. 2018))
Safety Margin	Domain Generalization ((Zhang et al. 2019; Ganin and Lempitsky 2014; Lee, Eum, and Kwon 2019))
	Perturbation and Corruptions Robustness ((Geirhos et al. 2019; Huang et al. 2018; Hendrycks, Lee, and Mazeika 2019))

Figure 39 Suitable safety strategies for ML-based systems from (166)

In the following, some aspects of the listed strategies will be discussed in extracts. The most promising approach is the use of inherently safe architectures realized by the possibility for formal verification of the ML-based model. Formal verification for ML models is strongly connected to the robustness considerations described in Chapter 2.3.2 and is covered more extensively in Chapter 2.3.6. The scope of formal methods for verification is to find flaws or to prove that the system is flawless. Assigned to ML, formal verification shall reveal robustness issues and/or enable robustness guarantees. Besides applying formal verification, the ISO 26262 demands (semi-) formal specifications of the design and functionality. The aim of this requirement is to model the system's reasoning and decision making within the application environment. Accordingly, the literature provides some strategies to fulfill the demands of (semi-) formal specification. Here, Figure 40 shows some relevant approaches.

Challenges	Principles
Environment (incl. Human) Modeling	Active Data-Driven, Introspective, Probabilistic Modeling
Formal Specification	Start at System Level, Derive Component Specifications; Hybrid Boolean-Quantitative Specification; Specification Mining
Modeling Learning Systems	Abstractions, Explanations, Semantic Feature Spaces
Scalable Design & Verification of Data and Models	Compositional Reasoning, Controlled Randomization, Quantitative Semantic Analysis
Correct-by-Construction Methods	Formal Inductive Synthesis, Safe Learning by Design, Run-Time Assurance

Figure 40 Challenges and possible mapping for (semi-) formal specification of ML based systems from (167)

Due to the limited applicability of formal methods (in particular for complex ML-based models) high-risk applications represented by high ASILs are difficult to achieve or are even unfeasible. The majority of the contradictions is based on the lack of formal verification of the model's specification and behavior. Thus, residual risk estimation remains uncertain and the risk driven approach of the ISO 26262 cannot be justified point to point. Nonetheless, a successful mapping and satisfying the safety requirements is constrained to the selected safety level and the risk impact of the applications. The lower the safety level, the higher the successful safety approval. Residual limitations in higher safety levels can be mitigated by redundant (traditional) systems or safe fail architectures. As an example, automated driving systems and autonomous vehicles (AVs) use additional sensor data (e.g. LiDAR, RADAR) to improve the decision process of the ML-based algorithms and lower the risk in case of a misclassification based on the image data from a camera.

2.3.6 Certification and Verification

Here, we present the mitigation strategies for the challenges on certification presented in Chapter 2.2.6. When applying certification, several things must be considered and an appropriate algorithm has to be chosen accordingly. This involves the power of an attacker (e.g. l_1, l_2 or l_∞ influence), the size and the architecture of the DNN to certify. Broadly, there are two different categories of approaches for robustness certification: complete and incomplete verification. These have distinct strengths, but also restrictions. There are also attempts to combine the best of the two worlds to overcome their shortcomings. In the following, the individual categories are discussed.

2.3.6.1 Complete Verification

Complete verification is the most exact, but also the most demanding approach. A successful complete verification guarantees (under certain conditions) that for a given input sample, there is no adversarial example, which results in a prediction other than the ground truth. Hence, the resulting boundaries for robustness are exactly defined. However, the problem of complete verification of a DNN is NP-complete as discussed in (168). The worst-case time complexity of this type of algorithms is exponential, although they exhibit practical run times for smaller networks. Most approaches work with l_∞ adversaries as well as simple feed-forward ReLU networks. A principled introduction to the use of ReLU as an activation function is given in (169).

One approach for complete verification is using logical solvers for satisfiability modulo theories (SMT) proposed in (170) or mixed integer linear programming (MILP) discussed in (171). When considering feed-forward neural networks with ReLU activation functions, the components of the model architecture can be transformed in mathematical functions, equations and inequalities. SMT and MILP solvers can determine if these are satisfiable for a given input sample respectively if the network is robust. Though solver-based approaches can successfully verify smaller neural networks under specific constraints (e.g. feed-forward structure, ReLU activations), they do not offer an adequate scalability. The size of a model as well as its architecture, which might not be convertible, are often the reason why this kind of approach is not feasible for DNNs used in complex practical applications.

A more scalable approach to complete verification is branch-and-bound algorithms (172). Here, the linear characteristic of ReLUs and affine mappings is exploited. This property reflects in the output as well, i.e., for a

given input sample the output is locally linear. A bounding step determines approximate lower and upper boundaries for possible output deviations by incomplete verification. The corresponding lower bound can result in a successful verification, while an upper bound may result in a failed one. If both boundaries do not yet lead to a verification decision, a neuron branching is applied. The neuron is transformed into two linear constraints and for each the bounding step is applied again. If evaluation of both branches results in the same decision, the verification of this branch is finished else the branching continues recursively with the following neurons. If all neurons were branched and there are still branches not leading to a verification decision, then the output deviation can be determined by linear programming. Though branch-and-bounding approaches are more scalable than solver-based ones, the maximum size of neural networks is still limited. Furthermore, the architecture must be suitable for the branching step, e.g. using ReLU activations.

Although leading to a robustness verification with exact boundaries, complete verification is only suitable for small DNNs and datasets and is often restricted to model architectures using ReLUs. Additionally, complete verification methods are limited to a model size of up 10^5 neurons and about 6 layers (173).

2.3.6.2 Incomplete Verification

In contrast to complete verification, incomplete verification only offers an approximate verification. The advantage of this kind of approach is that the relaxations enhance the scalability of the used methods, thus allowing the verification of larger networks.

Algorithms based on linear relaxations (174), again make use of the linear property of ReLU functions. A stable ReLU can be relaxed to a linear equation. For an unstable ReLU, a lower and an upper output boundary is determined which allows for bounding of the neuron by linear constraints. The bounded space is called ReLU polytope. With the combination of linear equations and the ReLU polytopes, an approximate overall output space for the whole network can be determined, which in turn can be used for an (approximate) robustness verification. When dealing with different activation functions than ReLU, the authors in (175), propose a verification for arbitrary neural network architectures. Linear relaxation methods scale to model sizes of 10^5 neurons and approximately 10 layers (173).

Probabilistic approaches are based on the concept of randomized smoothing. A smoothed model is created by adding random noise to the input data. The noise shall eliminate the effects of adversarial perturbations (176). Then, the new model is verifiable, but the verification only holds for this smoothed version. The applied noise distribution must be chosen carefully, since the smoothing process can have a serious impact on the accuracy of the model and in addition influences the tightness of the verification boundaries. Besides the effect on prediction accuracy, the noise addition and even stronger the transition back into the correct prediction can bring a computational overhead as shown in (177). In return, this approach is independent from the model architecture. Most randomized smoothing based algorithms are conceived for l_2 adversaries, but there are also more flexible approaches regarding l_p norms, e.g. (178), (179), (180) and smoothing distribution from (181).

Incomplete verification is more flexible and scalable than complete approaches, but large datasets and DNNs are still a challenge. Furthermore, the resulting robustness bounds are not exact, only hold for the smoothed model or do not have a reasonable tightness in comparison to the actual robustness. Additionally, for incomplete verification methods there exists a tradeoff between the scalability of the method and the tightness of the calculated robustness bounds (173).

2.3.6.3 Hybrid Approaches

Next to pure algorithms from the classes of complete and incomplete verification, there are additional hybrid methods. These try to combine the approximation techniques of incomplete verification, such as linear relaxation, with procedures from complete verification. The result shall inherit the advantages of both categories, scalability as well as precision or tightness of verification boundaries. Though enhanced in comparison to complete verification, scalability still remains a constraint. For example, (182) combines MILP solvers with the approach of linear relaxation for incomplete verification. In particular successful for l_2

verification, is the method proposed in (183). Here, the input space is divided into convex polyhedral regions, in which the output has a linear behavior. The regions can then be analyzed by geometric projections leading to more precise verification results than many incomplete approaches.

2.4 Mobility Use Cases

The different use cases of AI in mobility applications are best discussed on the concrete example of a system for AD. In Figure 41 a general overview of the components of such a system is shown. The fundamental goal of AD is to capture the surrounding scene of a vehicle using different sensors and deriving concrete actuator signals to maneuver the vehicle in this scene. Fundamentally, two different approaches can be distinguished to perform the described task. On the one hand, end-to-end approaches exist which consist of a single complex AI that takes the raw sensor data as input and predicts the control signals for the actuators. Current approaches use DNNs for the complex AI and such systems are discussed in Chapter 2.4.2. On the other hand, a modular approach exists where the complete task is split in different components. These components can be represented by an AI system or also traditional algorithms. This approach is used for the current generation of AD systems that are tested in specific regions on public roads. In Chapter 2.4.1 we discuss the tasks and techniques for each of the modular components in greater detail.

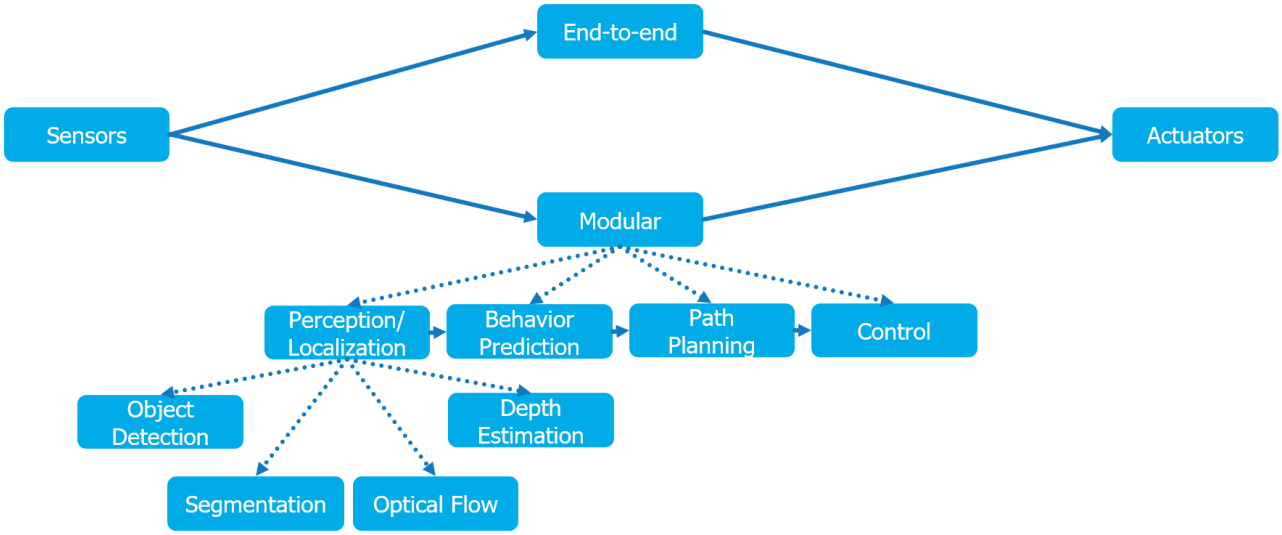


Figure 41 Overview of the components of a system for AD

Generally, different surveys exist that cover relevant use cases of AI for AD. In (184) and (185) the authors start by discussing different sensor types and an overview of the general architecture of an AD system. Then, they cover all tasks of the modular approach and present a comparison of techniques based on AI with traditional algorithms in detail. In contrast, the authors in (186) only focus on deep learning (DL) techniques. They first present a general introduction to DL and then show DL-based solutions for each task in the modular approach. Additionally, hardware requirements to perform real-time capable inference are discussed and general hardware components are presented.

In addition to using AI in an AD system there is also the option to use AI as part of an ADAS. However, most use cases are very similar because an ADAS is specialized on understanding only a part of the current scene. All use cases that are solved by an ADAS are also incorporated in an AD system. Therefore, we focus on presenting the modular components of an AD system and discuss how AI can be used to solve each individual task. Nevertheless, also for ADAS some works exist that summarize the use cases. In (187) the authors provide a short description of the main ADAS features that are deployed on public roads. Mainly, the authors discuss different camera locations and data processing steps. A more detailed summary of all existing ADAS features where AI could be used is given in (188). Again, a focus point is the discussion of different sensors and how the fusion of the data from different sensor types can be performed.

2.4.1 Modular Components

In the following we provide an overview of the tasks of the modular components of an entire AD system. We discuss the goals and main techniques of each task and provide additional references for a detailed discussion of the tasks.

2.4.1.1 Perception / Localization

The goal of perception modules is to understand the surrounding scene of a vehicle. The inputs are sensory readings and different perception modules try to extract information about higher-level concepts from the data. These higher-level concepts are then fed to modules further down the AD system. Some of the relevant higher-level concepts are shown in Figure 42 for the case of visual perception from camera sensors. In general, these concepts capture all information that is relevant for the understanding of the current scene, like how humans use their sensory system to get a representation of a scene.

In (189) a very detailed survey is done for all perception tasks based on image data. The authors compare traditional algorithms to extract the higher-level concepts with current approaches based on DL. For all tasks the SOTA regarding accurate perception is achieved by DNNs. Traditional approaches do not match the perception qualities of vision-based models that are trained on large datasets. Also, for AD/ADAS systems that are tested in reality the perception is done exclusively by using different DNNS, as shown in (190), (191) or (192). Therefore, in the following we discuss the individual perception tasks with DNNs in mind.

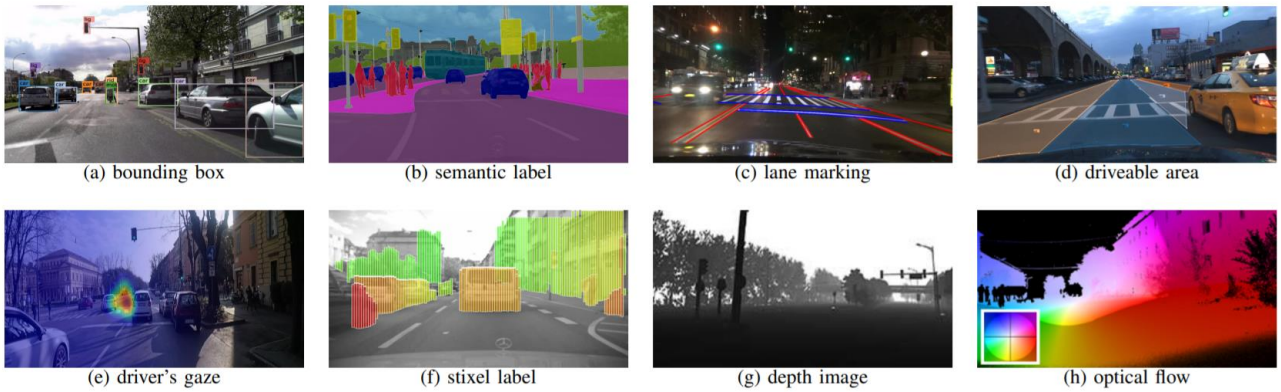


Figure 42 Overview of different visual perception tasks required for AD from (193)

2.4.1.1.1 Object Detection

The idea of object detection is to locate object instances from predefined categories in the current scene. Typically, all detected objects are assigned a bounding box that is provided to following components in the AD system. Traditionally, bounding boxes are predicted in the 2D image space (see Figure 42a), but some approaches also perform 3D bounding box prediction. If only image sensors are used the prediction of the depth of a bounding box is only an extrapolation, since no real depth information exists. In contrast, when RADAR or LiDAR sensors are used the input data is not an image but a point cloud (see Figure 43). Here, it is possible to have more accurate predictions of the depth component of the 3D bounding box.

The principal problem to detect objects is fundamental to computer vision and not a unique problem for mobility systems. Hence, in (194) the authors present a detailed survey on techniques for object detection in image data for different applications. In (195) the authors consider only the case of AD and discuss the evolution of DNN architectures used for object detection. They present recent approaches and perform a uniform evaluation on datasets specific for AD, but mainly focus on image data. Considering object detection on point cloud data the authors in (196) present a summary of all relevant approaches. They show that this is a more complex task due to high sparsity and irregularities of the point cloud data. Therefore, interest on this task only started recently, but the number of publications increased largely in the last years.

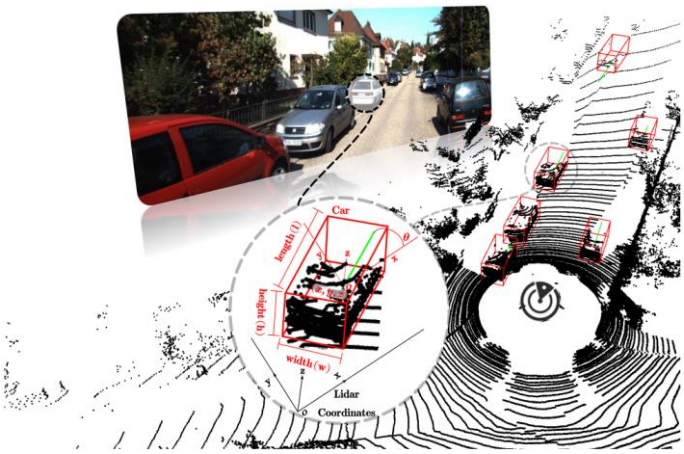


Figure 43 Visualization of object detection using 3D LiDAR data from (196)

2.4.1.1.2 Segmentation

Segmentation involves partitioning an image into multiple higher-level concepts. Thereby, segmentation is in principle the individual classification of each pixel, where a concrete class is assigned to each pixel. The individual pixels then form larger groups that can be interpreted as higher-level concepts. For example, in Figure 42b the semantic segmentation of a driving scene is shown. Here, specific classes are used for all higher-level concepts, like pedestrians, vehicles, signs, road surface, etc. Special cases of segmentation are shown in Figure 42c and Figure 42d, where only specific aspects of the overall scene are relevant for the segmentation.

Similar to object detection the segmentation of images is not a task that is specific to AD. In (197) the authors compare different DNN architectures to perform segmentation and introduce the basic idea behind each. They also cover different applications and extensively discuss remaining challenges. In contrast, the authors in (198) focus on the application of AD and summarize segmentation architectures that are applied on datasets specific for AD. They also shortly cover object detection, but then discuss segmentation in great details. The comparison of different DNN architectures is done for different segmentations tasks and for each task the relevant datasets are described.

2.4.1.1.3 Optical Flow

Theoretically, optical flow is the motion of brightness patterns between two successive images. The motion of patterns results from the movement of an object in a physical scene which is captured in successive images. Hence, the resulting optical flow can be interpreted as the projection of the true motion of objects in a scene. Ideally, it shows the displacement which maps all pixels of the former image to the new location in the second image. An example of the resulting optical flow in an image is shown in Figure 42h. The usage of optical flow estimation can give further insights into the layout of the current scene that the AV navigates and the decomposition into the movement of individual objects. Such movement information can then be used by following components as an alternative to track objects based on mapping bounding boxes from associated frames.

In (199) the authors introduce the problem of object flow for AD. They discuss the underlying task and summarize relevant datasets. However, they only use traditional algorithms to estimate the optical flow and do not use any AI-based approaches. Such AI-based approaches were only introduced more recently and are covered in (200). There the authors provide a systematic review of recently proposed techniques for optical flow estimation and focus on techniques that use DNNs. Comparable to other perception tasks DNNs start to outperform traditional algorithms also for optical flow estimation.

2.4.1.1.4 Depth Estimation

The goal of depth estimation is to estimate the distance of the camera to each point of a scene captured in an image (see Figure 42g). In principle, estimating the depth only from a monocular image is an ill-posed

problem. Nevertheless, this is an interesting approach since it reduces the need for RADAR or LiDAR sensors. It would be beneficial for the costs and simplicity of AD systems if an accurate depth estimation were possible from image data only and a major industry player is using only image data as discussed in (191). Alternatively, depth estimation from images is also relevant because the estimated depth might be used as a backup when RADAR or LiDAR sensors only provide point clouds with poor quality due to environmental conditions.

Monocular depth estimation is considered in (201). The authors introduce the problem and summarize relevant datasets and quality metrics. Also, they present traditional approaches before considering recent approaches using DNNs. These provide depth estimations that clearly outperform traditional approaches and do not suffer from sparsity issues, since the depth can be estimated effortless for every point in an image. Further, the authors in (202) consider depth estimation when stereo image data is available. They structure their survey in a similar way and start by presenting relevant datasets. Alike, they conclude that DNN-based approaches significantly outperform traditional approaches based on matching hand-crafted features across multiple images.

2.4.1.1.5 Localization

Localization aims to determine the current position of the AV as it navigates through the scene. Thereby, the position consists of the location and orientation of the vehicle on the current road or lane. The most straightforward option is to use the global positioning system, but since the average accuracy of the position can deviate up to multiple meters the usage is limited for AD. Here, the localization must be exact in terms of decimeter level to stay in the correct lane. Available alternative solutions are discussed in (203) or (204) and we show an overview in Figure 44. Most approaches are using traditional algorithms to perform the concrete localization, e.g. for map matching, hence the localization is not a typical AI use case. However, AI approaches can be used to derive landmarks or key points from the current sensory inputs, which is again an underlying perception task. Here, the most relevant case is the detection of lane markings (see Figure 42c) which can be used by further algorithms to exactly locate the AV in the lane.

Figure 44 Overview of different approaches to perform the localization of an autonomous vehicle from (203)

2.4.1.2 Behavior Prediction

The principled goal of behavior prediction is to predict the possible future behavior of all participants in the current scene. Hence, it is based on the results of the perception modules since future behavior can only be predicted for known objects. In Figure 45 an exemplary behavior prediction is shown for the case of pedestrian path prediction from a birds-eye view perspective. Here, the perception provides a surface annotation and relevant information of the detected person. Based on this information the most likely future path of the pedestrian is predicted. Instead of only predicting the most likely future behavior it is also possible to predict multiple future behaviors. These can be ranked by probability of occurrence and provide a deeper understanding of possible future scenarios that can be exploited by the following components in the AD system.

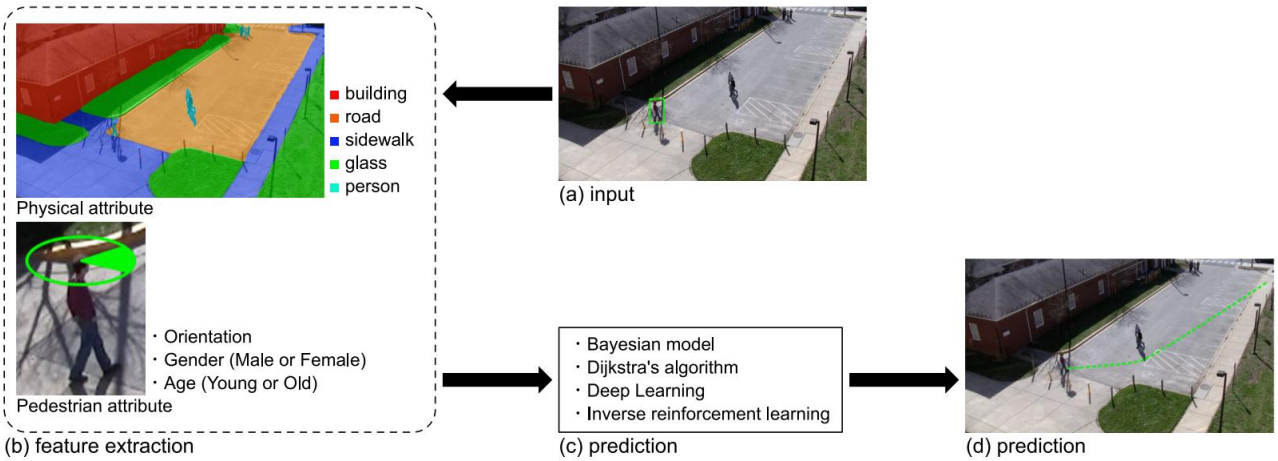


Figure 45 Overview of behavior prediction and required data representations from (205)

A first survey on path prediction of pedestrians and vehicles is done in (205). The authors shortly cover different methods to perform the feature extraction from images and then cover different algorithms for the behavior prediction. This includes traditional approaches, as well as more recent approaches based on DNNs. Concluding they provide an extensive overview of available datasets. In (206) the authors focus on the behavior prediction of pedestrians. They present a variety of approaches and categorize them according to the introduced taxonomy. Traditional algorithms are first extensively summarized before the authors present more sophisticated approaches that use DNNs. Concluding, the authors in (207) focus on behavior prediction using only DNNs. They present an overview of different prediction tasks and highlight the details of the used architecture, training methods and datasets. Also, all relevant datasets and evaluation metrics are listed. Looking at available literature, data-driven AI systems have recently surpassed traditional algorithms for behavior prediction if enough high-quality data exists. Such systems are mostly based on DNNs and are deployed in systems operating on public roads as we show for some examples in Chapter 2.5.2.

2.4.1.3 Path Planning

In the path planning component, the goal is to plan a concrete path for the vehicle to navigate in the current scene. To achieve this goal multiple intermediate tasks must be performed. These different tasks are shown in Figure 46. First, the global planning of the route is performed based on the user destination and the available road network. Next, the local planning is used to navigate between the global waypoints that are planned. Here, information of the previous perception and prediction components on the scene and moving objects are used to decide the motion to perform. The possible motions are higher-level concepts like stay in the lane, start parking maneuver, take evasive actions, etc. Based on the selected motion the detailed local planning is done which outputs the path that the vehicle should use to navigate through the current scene. Additionally, in Figure 46 the last step of an AD system is also shown, which is the translation of the planned local path in actual commands for the actuators. We cover this component explicitly in Chapter 2.4.1.4.

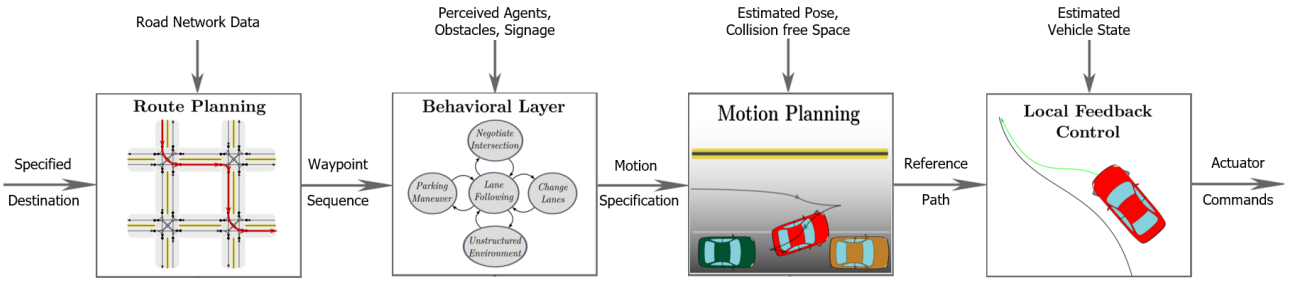


Figure 46 Overview of the steps required for path planning from (208)

The authors in (208) cover traditional techniques for path planning and present the interaction with control techniques. They strictly focus on the traditional planning steps shown in Figure 46 and do not discuss recent approaches based on more sophisticated AI methods like DNNs. In (209) the authors focus on the path

planning task and discuss traditional approaches as well as recent data-driven approaches. These do not necessarily follow the traditional planning steps, but sometimes incorporate the behavior layer and the motion planning from Figure 46 into a single learned component. They conclude that data-driven approaches are on the rise and mostly outperform traditional approaches. However, for the more sophisticated approaches the questions of safety, verification and explainability are largely unanswered which limits their use in practice. Finally, the authors in (210) and (211) survey recent approaches on data-driven planning using DNNs. They focus on reinforcement learning and imitation learning-based approaches, which are heavily researched in recent years. They discuss different options to integrate such approaches into the AD system and review different setups for the reinforcement learning environment. Additionally, the role of simulators and synthetic data is discussed for training and validation. Summarizing, they point out the great potential but also conclude that some challenges still must be solved before mature solutions are available.

2.4.1.4 Control

The control component is the last component of an AD system and selects actuator commands to execute the planned local path. It incorporates direct feedback from the vehicle and uses a kinematic model that describes the general capabilities of the vehicle. The goal is to minimize the tracking error between the planned path and the executed motion. In Figure 46 the main idea of the control component is shown on the right side as the last step.

In (208) the authors cover traditional control algorithms and discuss formal derivations for different control tasks. They provide a comparison which evaluates the used vehicle model, the time complexity and the path stability guarantees. Instead of traditional algorithms, the authors in (212) focus on using DNNs in the control component. Again, this trend emerged in recent years due to the ability of DNNs to model complex driving environments and their principled ability to generalize learned rules to new scenes. The authors present different approaches to use DNNs for controlling a vehicle and extensively discuss remaining challenges. Again, the black-box character of DNNs complicates the verification because it is impossible to test these systems on all scenarios that can occur in the real-world. Further research is required before learned control systems are ready for deployment on public roads.

2.4.2 End-to-End System

In contrast to the modular approach, in end-to-end AD system all individual components that are described earlier are exchanged with a single large DNN. This takes raw sensor data as input and directly outputs the actuator commands to navigate the vehicle. An overview of different approaches to learn an end-to-end system is given in (189). The authors also discuss available datasets and metrics to evaluate the quality of the navigation.

In this younger research field, the first relevant publication is (213). Here, the authors use a full end-to-end system based on a DNN that outputs steering commands using raw pixel data from a single camera. No explicit training incentive is used to force the network to learn the representation of higher-level concepts, like vehicle detection or lane marking detection. Instead, all representations are learned automatically during training. The main advantage of this approach is that it enables the simultaneous optimization of all components. Instead of optimizing human-like modular tasks the DNN has the freedom to find the most useful representations. Eventually, this will lead to better performing systems because the direct optimization of the goal is possible. However, this approach has the direct disadvantage that explainability is lost and the black-box character of the whole AD system is even amplified. Also, no explicit control of the safety of the system exists. The DNN must be trusted to learn robust representations that also perform well on unknown input data, because no safety modules exist. These are critical disadvantages and the reason why currently full end-to-end systems are not used for commercial AD systems, as we describe in Chapter 2.5.

In (214) the authors also use an end-to-end approach to train a DNN to navigate a vehicle. However, the DNN in this system outputs the next waypoint where the vehicle should navigate instead of actuator commands. Comparing with Figure 41, this means that the control module is excluded from the DNN and only the

modules up to the path planning are learned from the raw sensor input data. The final actuator commands are still generated by traditional control algorithms. Hence, this is some intermediate approach between a modular and a full end-to-end system. It seems promising since it combines the potential of an increased performance by optimizing most modules together with the possibility to include safety mechanisms in the control algorithm. Here, it is also possible to include an additional safety backup system that the control module can use as fall back if required.

An additional problem of most end-to-end approaches is that they need to collect data samples online during training. Learning from expert human drivers is not sufficient to train a system that performs well under all conditions. Hence, simulating worst-case conditions, like accidents or bad weather, is needed to train a system that is as robust as possible. However, this suffers from the currently poor correlation of online driving performance and offline testing, as shown in (215). Also, the domain gap between simulation and reality must be reduced to safely transfer end-to-end learned systems to commercial AD systems. We present potential solutions to closing this domain gap in Chapter 2.6.3.

2.5 Entire Mobility Systems

In this chapter we describe how the use cases from Chapter 2.4 can be integrated into an entire AD system that can operate on public roads. We discuss the interaction with hardware components and the integration of individual AI systems into the overall AD software. It is important to note that the concrete system architecture of major industry players for AD is not known, since this is the intellectual property of the respective company. Nevertheless, current research publications are available and these serve well to introduce the main ideas for the integration of individual components into an entire system for AD.

2.5.1 System Overview

Early research work focuses on autonomous driving on special routes and not the ability to drive unknown routes. Also, initially no DNNs are used and all modular components are represented by traditional algorithms. Nevertheless, it is still interesting to investigate these systems, since the main architecture is very similar for current systems. For example, this can be seen by comparing Figure 47 and Figure 49 which we discuss in the following.

One of the first publications that present an entire system for AD is (216). Here, the authors provide an overview of the vehicle that Stanford University used for competing at the 2007 defense advanced research projects agency urban challenge. They discuss the used hardware and present the traditional algorithms used for all components of the modular approach presented in Figure 41.

In (217) the authors present their approach to design a vehicle that travelled the Bertha Benz memorial route autonomously. They provide a detailed overview of the autonomous vehicle and the algorithms for perception, localization, planning, etc. In Figure 47 an overview of the used sensor setup and the general system architecture is shown. As can be seen on the right side of the figure, the used architecture is very similar to the overview of the modular approach in Figure 41. In addition to the previously presented components the system contains a reactive layer. This layer includes safety logic in case a sudden emergency maneuver becomes necessary. In this case the signal flow does not have to go through the entire modular pipeline but is generated in this special layer. Again, the authors use no AI components, meaning the system relies only on traditional algorithms. However, it is easy to integrate AI systems into the architecture shown on the right side in Figure 47. For example, each component of the perception category could be replaced by an AI system, e.g. using DNNs as described in Chapter 2.4.1.1. This is a plug-in replacement since the AI system takes the same input and generates the same output. All interfaces stay the same, but the AI-based system takes over the role of traditional perception algorithms.

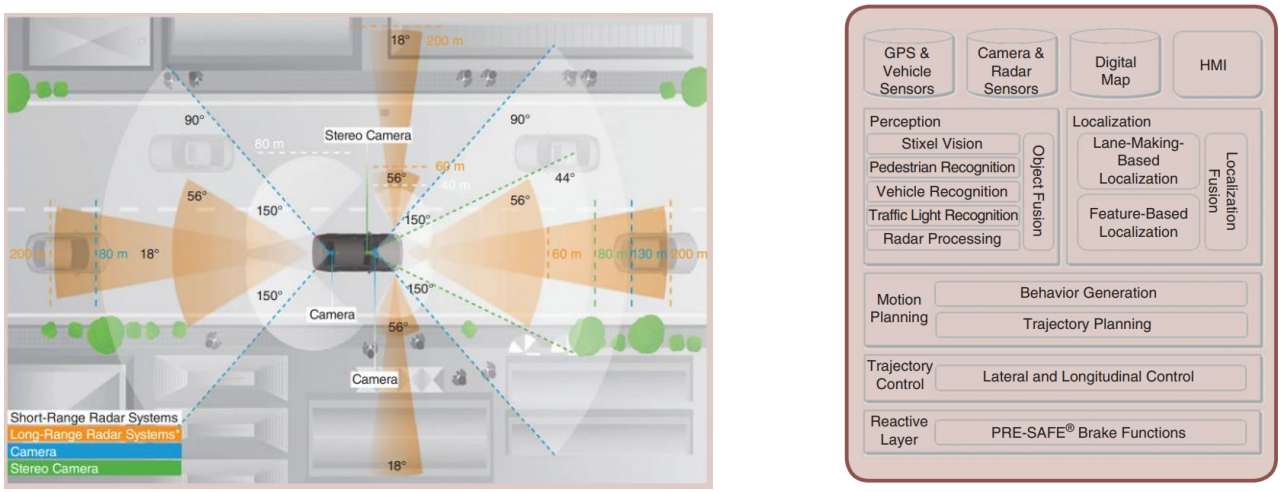


Figure 47 Sensor setup on the left and architecture on the right of the AD system from (217)

Like the last publication the authors in (218) present an overview of a system they developed for driving autonomously in a specific scenario. Here, they focus on driving on German highways and present the sensor setup, architecture overview and perception algorithms. Summarizing they discuss remaining challenges in their approach that serve as the basis for future research.

In (185) the authors first present a general survey on the progress of autonomous driving in recent years. Then they introduce their intelligent autonomous robotics automobile (IARA) research vehicle. They present a detailed overview of the software architecture with the data flow and interaction of different components. This architecture overview is shown in Figure 48. Again, they do not explicitly mention the use of AI components, but it can well be seen how different components could be exchanged by AI components without any change in the overall architecture. Additionally, the authors cover autonomous driving in the industry and present all companies that focus on developing systems for highly automated driving. They compare the sensors used and the maturity of the respective systems. As previously mentioned no concrete system architectures are provided, since these are the secret of each company.

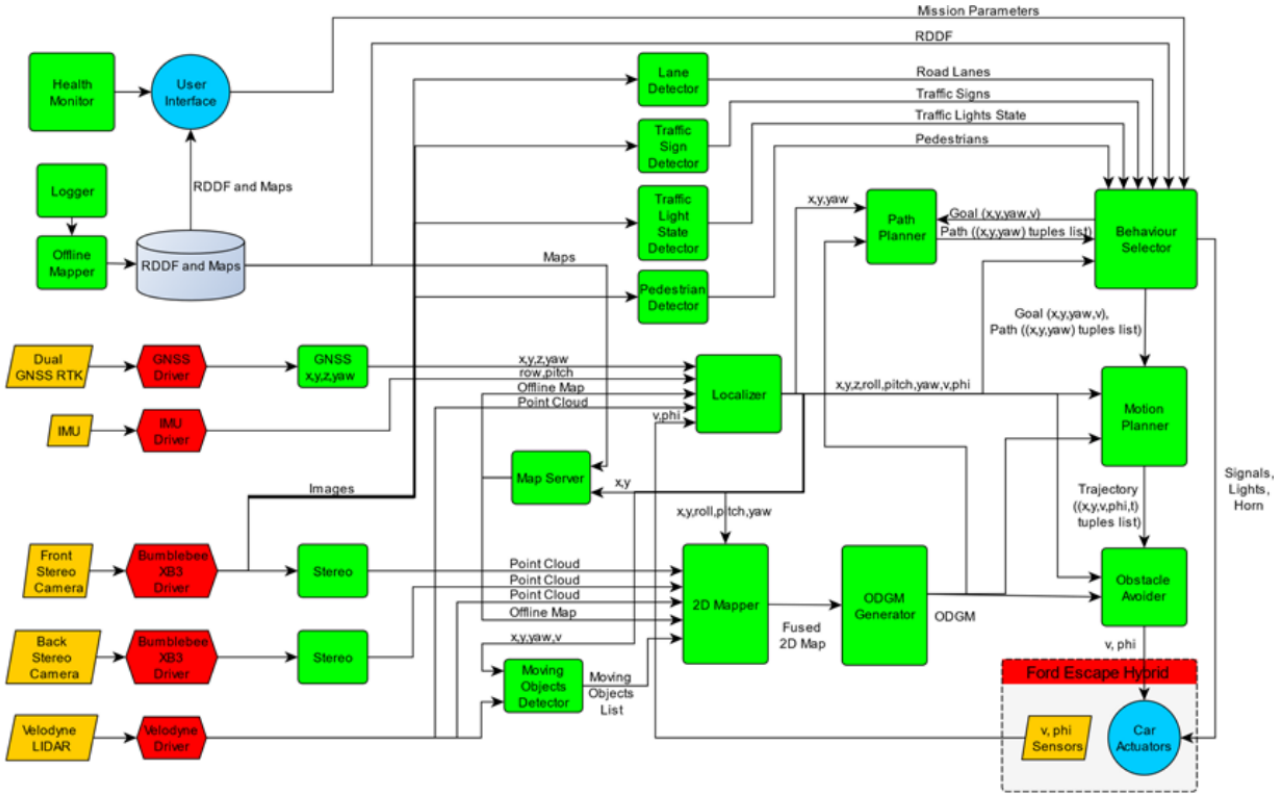


Figure 48 Detailed overview of the data flow in the IARA from (185)

Finally, the authors in (219) present a survey on hardware components used for AD. They compare different sensors and list main advantages and disadvantages. Also, they describe the perception use cases where each sensor is relevant and compare the performance in contrast to the human visual system. Then, they focus on presenting available hardware components that are used for real-time capable AD and provide a comparison between them. They cover the integration of software and conclude by a detailed discussion of remaining challenges.

2.5.2 AI Integration

After presenting the architecture of systems for AD, the interaction with hardware components and the basic idea how to integrate AI components into the entire architecture, we now discuss available information on the concrete integration of AI components. As already discussed in Chapter 2.4.2 in (214) and (213) the authors focus on using end-to-end approaches. They also present the software architecture and output of the DNN. Especially, in (214) the interaction of different DNN components learned for different tasks is discussed. Here, it shows again that AI components can be used as an alternative for most modular components. These can be exchanged without a significant change in the overall system architecture.

Regarding the integration of AI components into commercially available systems only limited publications exist. In (191)[3]the authors provide some information on autonomous driving research done at Tesla. They cover the creation of a large-scale dataset and then present a very rough overview of the architecture of the DNN used for autonomous driving. Here, it is again interesting to observe how different modular components are merged in a single DNN that can have multiple heads for each required component output. These results are then consumed by traditional algorithms to generate the control commands for the actuators.

More concretely, another overview of an architecture of a commercial system for AD is given in (220). Here, Baidu provides the open source[4] architecture Apollo that is also used as the basis for their commercially available AD software Apollo Go which is tested in different Chinese metropolitan areas. The Apollo project provides software utilities to develop, test and deploy systems for autonomous driving. It ships with pre-trained DNN components for perception and prediction and has different algorithmic options for the modular components. In Figure 49 we show an overview of the AD system, the hardware components and the required connections between them. As can be seen on the left side the general AD system is very similar to the system shown in Figure 47. It contains the same modular components, but now DNN-based systems exist to perform some respective tasks. Overall, we can conclude that this architecture is representative of most AD systems developed by different companies or research teams.

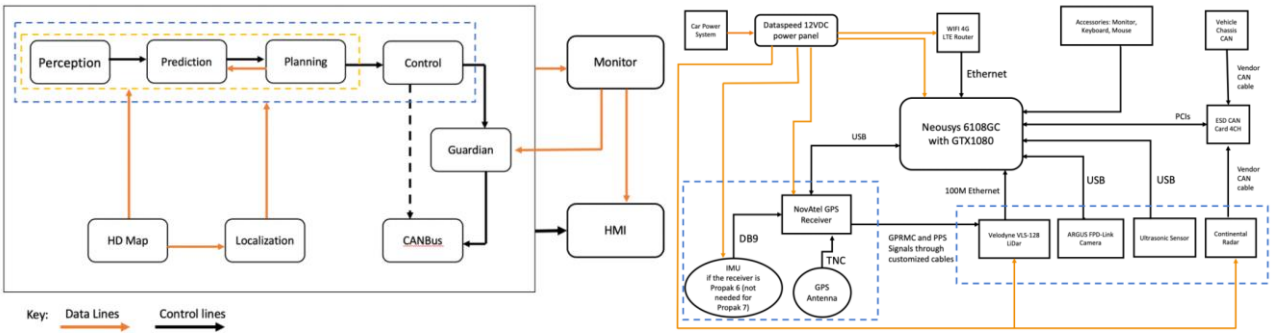


Figure 49 Overview of the AD system on the left and hardware components on the right of the AD system Apollo from (220)

2.6 Mobility Datasets & Simulation

After presenting the different use cases of AI systems and the integration into a complete AD system, we now discuss available datasets for data-driven AI systems. Then, we present options for generating synthetic data

[3] The video presentation is available at: https://www.youtube.com/watch?v=g6bOwQdCJrc
[4] The open-source project is available at: https://github.com/ApolloAuto/apollo

based on special purpose simulators. This is very important to cover corner cases and rare scenarios that cannot all be captured in reality in a timely matter. However, the image quality of current simulators is not representative of camera sensors used in the real-world. Therefore, we discuss recent approaches to improve the image quality from simulators and close the domain gap between simulation and reality.

2.6.1 Datasets

At first, a principled overview of available datasets is part of most surveys focused on AD. Hence, in (184), (186) and (189) the authors provide a summary of available datasets and a comparison between them. Additionally, there are also publications that explicitly focus on comparing the quality and diversity of datasets. Here, (193) is the most relevant publication. Additionally, the authors try to characterize the drivability of a scenario and discuss what driving scenarios are currently not covered by existing datasets.

Also, consistently new datasets are proposed which include the main difference to previous works. Such a publication is (221) and their comparison of automotive datasets is shown in Figure 50. It compares the most used datasets using different relevant categories. Here, the most important aspects to distinguish datasets are the used sensor types, the quality of the annotations, the scope and the diversity of the data. It is also important to point out that some datasets contain real data, while others contain only synthetic data from simulation. In general, we can conclude that enough datasets exist for developing performant systems for perception and prediction on typical scenarios. However, rare edge cases are typically not covered in public datasets because it is impractical to cover all possible scenarios that can occur. Here, simulation of interesting and relevant scenarios is a promising direction that we discuss later in Chapter 2.6.2.

Dataset	# cities	# hours	# sequences	# annotated images	Stereo	Depth	LiDAR	Radar	SPAD-LiDAR	IMU/GPS	All 360°
KITTI [12]	1	1.5	22	15k	✓	✓	✓			✓	
Cityscape [9]	27	2.5	0	5k	✓					✓	
Mapillary Vistas [35]	30	-	-	25k							
ApolloScape [17, 55]	4	-	-	140k	✓		✓			✓	
SYNTHIA [44]	1	2.2	4	200k		✓					✓
H3D [38]	4	0.8	160	27k			✓			✓	
SemanticKITTI [1]	1	1.2	22	43k			✓				
DrivingStereo [52]	-	5	42	180k	✓	✓	✓			✓	
Argoverse [8]	2	0.6	113	22k	✓		✓			✓	✓
EuroCity [4]	**31**	0.4	-	47k							
CADC [41]	1	0.6	75	7k			✓			✓	
Audi [13]	3	0.3	3	12k	✓	✓	✓			✓	✓
nuScenes [7]	2	5.5	1k	40k			✓	✓			✓
A*3D [40]	1	**55**	-	39k	✓		✓				
Waymo Open [53]	3	6.4	**1150**	230k			✓				
Ours (AIODrive)	8	2.8	100	**100k**	✓	✓	✓	✓	✓	✓	✓

Figure 50 Overview of often used datasets for AD/ADAS from (221)

In addition to general datasets for AD that contain annotations for multiple perception tasks, there are also datasets that focus on a single use case. For example, in (222) the authors present a dataset only for German traffic signs. A similar dataset is presented in (223) for Chinese traffic signs or in (224) for American traffic signs.

Special datasets do not only exist for traffic signs but also for all other basic perception tasks. An additional example are datasets that focus on pedestrians. Here, (225) contains cropped images of humans from personal photos and (226) contains pedestrians in different traffic scenes in an urban environment.

Additionally to the discussed datasets for different perception tasks, recently some research works propose datasets with corrupted image quality that mimic corruptions which occur in reality like noise, rain, snow, reflections, etc. First, such ideas are introduced for standard image classification described in Chapter 2.2.2.1. Then, the authors in (227) expand the idea and add corruptions to a widely used dataset for object detection. An overview of the used corruptions and resulting images is shown in Figure 51. Similarly, the authors in (228) add corruptions to datasets that are widely used for semantic segmentation.

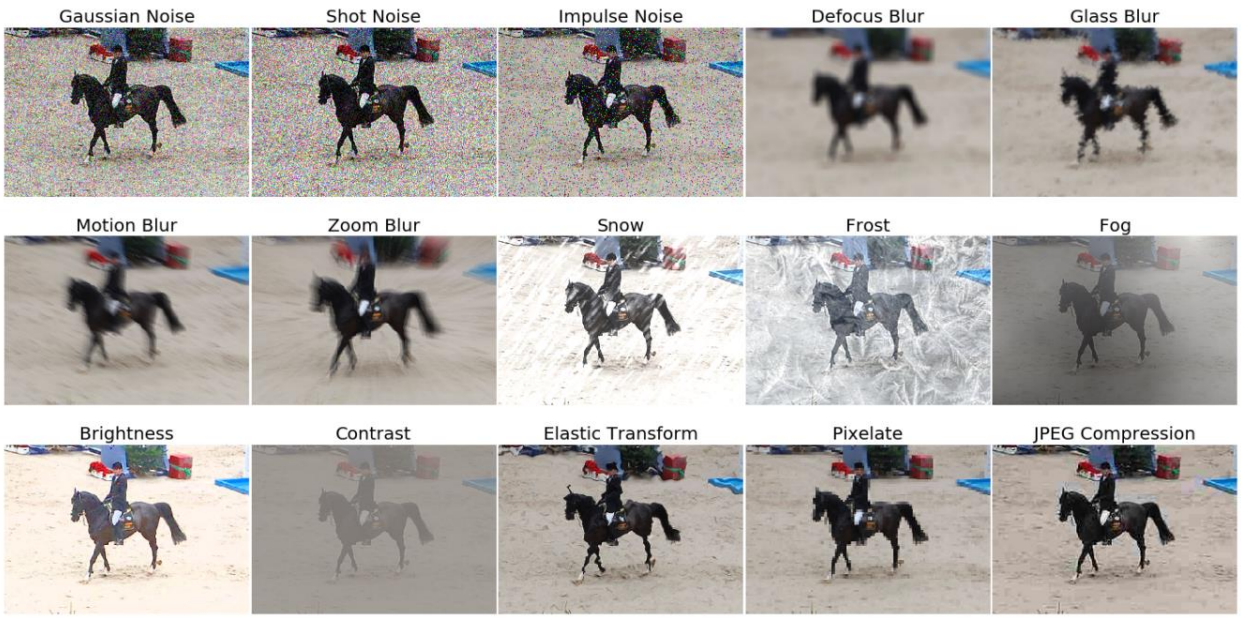


Figure 51 Overview of corrupted images used for object detection from (227)

2.6.2 Simulators

Using only data captured from reality is both time-consuming and expensive. Simulators provide an easy alternative that can generate labelled data quickly for selected interesting situations. Hence, simulation plays a vital role in the development and especially the verification of AD systems.

On the one hand, open-source simulators exist that are free to use and where the contribution is possible for everyone. Here, by far the most used option is CARLA introduced in (229). It is built as special purpose simulator to generate synthetic data for various traffic scenarios and is also used in the industry due to its high flexibility. It ships with already included assets for scene layout, buildings or vehicles and can easily be extended to include custom objects or road layouts. Also, the usage of different sensor types is possible and different environmental conditions can be simulated out-of-the box. CARLA can generate labels for different modalities and includes all relevant semantic classes for scene participants. In Figure 52 an overview of three different sensing modalities is shown, where the left shows a normal image from a RGB camera, the middle shows the ground-truth depth and the right shows the ground-truth semantic segmentation. As can be seen the normal RGB image does not look photorealistic and there exists a clear gap to images captured in reality. This is a typical phenomenon which represents the main disadvantage of using simulators but does not only exist for CARLA. Currently, for all simulators it is possible to observe a difference to images from real data and we discuss different techniques to reduce this gap in Chapter 2.6.3.

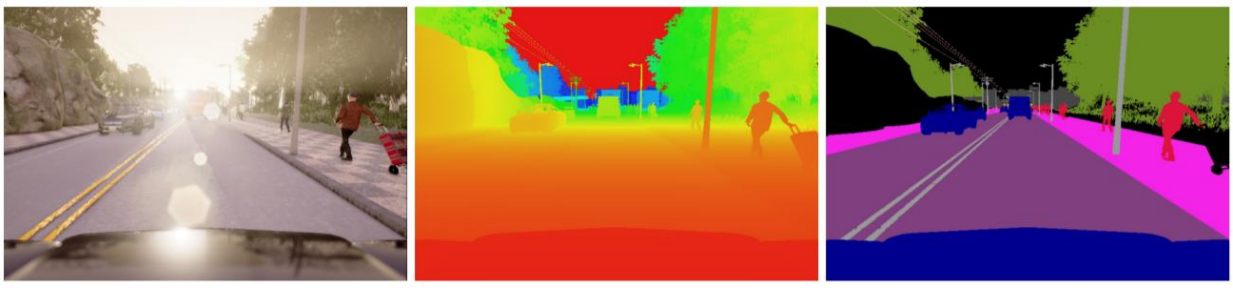


Figure 52 Example of three sensing modalities available in CARLA from (229)

An alternative open-source option to CARLA is AirSim introduced in (230). However, this simulator is not tailored towards the requirements of automotive simulation but can be used to simulate a range of vehicles also including flying vehicles. Hence, in the automotive industry CARLA is typically used when an open-source simulator is needed.

In addition to open-source simulators there also exist different proprietary simulators that are commercially available. These options include CarMaker from (231) and DriveSim from (232). Both have similar features as described for CARLA and additionally can be included in Hardware-in-the-Loop testing stations. Also, they produce more realistically looking images but the gap to reality can still be observed. Nevertheless, for the testing and validation of automated driving functions CarMaker represents the industry standard due to the maturity of the product and the available functionality that also allows simulation of powertrain and motion control use cases.

2.6.3 Image Quality Enhancements

As described earlier, current simulators do not produce images that look photorealistic. There exists a gap between simulated data and data that is captured during drives in reality. Hence, using only synthetic data is often considered as doomed to fail as soon as systems are tested offline in reality. Therefore, the authors in (233) use recent progress in image-to-image translation to successfully close the domain gap. They are able to train an end-to-end AD system using only simulated data and operate a vehicle in reality[5]. Their key idea is to transfer images captured while driving to look like the simulated images the system is trained on before feeding the images to the AD system.

The exact opposite approach is motivated by the authors in (234). Here, they also use image-to-image translation but try enhancing the photorealism of a simulator[6], instead of worsening the quality of real images. They postprocess simulated videos and transfer the image style and quality to look like the images in well-known datasets for AD. This improves the realism of the simulated data significantly and makes it hard for humans to distinguish between simulated data and data captured in reality. Some examples of their successful image-to-image translation are shown in Figure 53.

Figure 53 Examples of successful image-to-image translation from (234)

Both described methods use image-to-image translation to achieve the significant improvements and reduce the domain gap. Here, GANs form the basis of the best performing methods. These special CNNs are first introduced in (27) as a new approach for generative models. They can learn a latent representation of the

[5] The video presentation is available at: https://www.youtube.com/watch?v=D7ZglEPu4lM
[6] The video presentation is available at: https://www.youtube.com/watch?v=P1IcaBn3ej0

features of an image dataset and then generate entirely new images that look like they originate from the original distribution. Following, the authors in (235) adapt the basic idea of GANs but use synthetic images as input data of the generator instead of random sampling from the latent representation. In recent years this approach is further refined, for example by (236) or (237). Now, such approaches can be used to reduce the domain gap between simulated and real images. This might enable the deployment of AD systems trained with simulated data in reality without a significant decrease in quality.

Very recently another approach emerged for image-to-image translation that seems to match the quality of GAN-based approaches or even further improve upon them. In (238) and (239) the authors use diffusion models instead of GANs for image-to-image translation. Diffusion models are initially introduced in (240) and use an iterative denoising process to convert samples from a Gaussian distribution to samples from the training distribution. Applying this concept to image-to-image translation the denoising is no longer conditioned on samples from a Gaussian distribution but instead on the images that should be transferred to the reference style. This enables the application to enhance images from a simulator. However, to the best of our knowledge up to now no approaches exist that explore this concrete translation, mainly because the presented approaches were only proposed very recently. Nevertheless, it is still interesting to observe the progress in this area and see if further improvements of the quality of the enhanced images can be achieved.

2.7 Standardization Activities AI & AD

Finally, the following chapter gives an overview on existing and developing standardization approaches. Since most of these standards were created for general software systems, some of their contents and requirements do not map well to AI and AD systems. Details on this gap are given in Chapter 2.2.5 and Chapter 2.3.5. Since this gap within the existing standardization is well known, currently there are ongoing standardization efforts for AI and AD specific systems to fill this gap.

2.7.1 Existing Standardization

Table 1 gives an overview of existing standards related to either AD or AI systems.

Table 1 Existing safety and security standards in the domain of road vehicles

Standard	Topic	AI specific	AD specific
ISO 26262:2018 Road vehicle - Functional Safety	Requirements ensuring the functional safety of road vehicles.	No	Partially
ISO/PAS 21448:2019 Road vehicles - Safety of the intended functionality	Guidance on design, verification and validation measures to ensure safety of the intended functionality in absence of failure.	No	Partially
ISO/IEC TR 24028 AI Trustworthiness	Survey on approaches regarding the trustworthiness of AI systems, such as explainability, risk/threats and their mitigation strategies.	Yes	No
ISO/IEC TR 24029-1 Assessment of the robustness of neural networks	Background of existing methods for robustness assessment of neural network.	Yes	No
IEC 61508:2010 Functional safety of electrical/electronic/programmable electronic safety-related system	General safety requirements towards electrical/electronical/programmable systems.	Partially	No

The ISO 26262: "Road vehicle – Functional Safety" (165) is an automotive standard that provides requirements to ensure the functional safety of vehicles. The standard defines processes and requirements to design and operate automotive products. Further, it defines the ASILs that classify different hazards. The ASILs are calculated based on the severity, exposure and control of the hazard. Severity refers to the number of injuries the hazard could cause, the exposure describes the expected times of occurrence of the hazard and the control refers to the possibility to prevent the hazard (241). There are four ASILs from ASIL A to ASIL D (low to high risk) and a quality management level for non-safety-related functionalities. Based on the ASIL of a hazard, safety requirements can be derived.

The ISO/PAS 21448: "Road vehicles - Safety of the intended functionality (SOTIF)" (242) aims to cover foreseeable misuse by the driver. It only covers the SAE L1 and L2, which are only driver-assisted systems. Hence, it does not integrate SAE L3 and higher, which are highly automated driving systems (243). Additionally, as discussed in (244) AI-based systems are not sufficiently covered by it. However, the ISO/PAS 21448 is going to be replaced by the ISO/FDIS 21448 (245), which is currently under development.

2.7.2 Standardization in Progress

Concluding this report, we list AI and AD specific standardization activities that are in progress in Table 2.

Table 2 Overview of ongoing AI and AD standardization activities

Standard	Topic	Status
ISO/CD TR 4804	Safety and cybersecurity for automated driving systems - Design, verification and validation	In review
IEEE P2846	Guidance on design, verification and validation measures to ensure safety of the intended functionality in absence of failure	Draft status
ISO/IEC AWI TR 24030	Representative use cases of AI applications in a variety of domains	Under development
VDE AR 2842-61	Application rule for the entire life cycle of trustworthy autonomous (AI) systems	Partially published
ISO/FDIS 21448	Guidance on design, verification and validation measures to ensure safety of the intended functionality in absence of failure. (Will replace ISO/PAS 21448:2019)	Under development
DIN Standardization Roadmap AI (2nd edition)	Framework for early development of AI Standardization in Germany	Under development
ISO/AWI PAS 8800	Safety-related properties and risk factors of AI components within road vehicles.	Under development
ISO/IEC AWI TR 5469	Functional Safety & AI Systems	Under development
ISO/IEC DTS 4213.2	Assessment of machine learning classification performance	Under development
ISO/IEC TR 24029-2 Assessment of the robustness of neural networks	Methodology for the use of formal methods for robustness assessment of neural networks.	Under development
ISO/IEC FDIS 22898 AI Definitions	Artificial intelligence concepts and terminology	Under development

3 Generic Requirements (AP3)

The aim of this chapter is to derive and formulate generic requirements towards systems for autonomous driving (AD) and advanced driver assistance systems (ADAS) to ensure their safety. Due to the robustness threats towards AD and artificial intelligence (AI) systems discussed in the AP2 report (2), there are risks and vulnerabilities for the safety and security of such systems that have to be addressed during the entire development process.

As Figure 54 illustrates, the safety and security of an AD or AI system is influenced by its performance, robustness, interpretability, monitoring and documentation. Therefore, the generic requirements are formulated to address these aspects along the entire lifecycle of the system. Additionally, the system's compliance to existing standardization and norms has to be considered.

The requirements are derived in a two-step process. First, existing safety and security standards are introduced. Their concepts and requirements, especially from ISO 26262 (165; 246; 247), are used to derive a first set of requirements. In a second step, supplementary requirements are formulated to reduce the residual risk and to address gaps arising from the involvement of AI components. Additionally, the requirements address the entire lifecycle of a system from the design and conceptualization phase of the system to the integration and testing of software units.

Figure 54 Overview of composition of requirements for AI systems

Overall, the requirements are formulated to fit all general AD/ADAS systems (i.e. perception system, traffic sign recognition, etc.). Since these different systems differ in data, complexity and risk exposure, thresholds and test cases are defined generically and have to be refined for each individual use case. As an example, some

systems might have sensor-induced boundary values, whereas other systems have an infinite input space and therefore have to be tested against corner cases. The specific definition of the corner cases and boundary values also may differ between different specific applications. Therefore, the requirements in this document are to be understood as generic technical requirements and have to be refined for each application. For the refinement of such requirements a concise rationale shall be given to enable and enhance auditability and communication between stakeholders.

First, in Chapter 3.1, existing security and safety standards in automotive are introduced and the elicitation process for the generic requirements is defined. Afterwards, in Chapter 3.2 the requirements towards the entire system are introduced and in Chapter 3.3 requirements for the specific AI subsystems of the entire system are defined. Chapter 3.4 discusses the applicability of the generic requirements to different use cases. Finally, Chapter 3.5 categorizes the testability of each requirement.

3.1 Requirements Elicitation

In this chapter, the process used to derive the general requirements is introduced. As explained in the previous chapter the scope of this document is to define safety and security requirements specific to AI and AD/ADAS relevant systems. The general requirements are defined along the entire lifecycle of the system and for the aspects of performance, robustness, interpretability, monitoring and documentation. Moreover, they are defined to address characteristic features of AI systems, e.g., the difficulty of explainability or the vulnerability against adversarial inputs. Several sources, i.e., norms, standards, technical reports and research studies are consulted. Especially the ISO 26262 exhibits an extensive coverage of requirements and processes for functional safety in automotive regarding electronical systems. Therefore, it is used as groundwork for deriving a set of general requirements specific for AI systems. To further support the elicitation of safety requirements, the ANSI/UL 4600 (248) was used as guidance to address remaining gaps regarding the safety of the system. Since these norms focus on the safety of the system, the ISO/SAE 21434 (249) and the UNECE R 155 (250) were used to derive security-related requirements. Lastly, additional requirements are defined to fill remaining safety and security concerns.

Overall, the aim of the resulting list of generic requirements is to form a guidance for the development of safe and secure AI components in AD and ADAS systems. For each use case there might not be the need to implement all of the defined general requirements, but a subset or combination of them. If this is the case, a rationale should be provided to ensure that the safety and security of the system is assured with the chosen combination of requirements.

The first section of this chapter gives an introduction to security standard ISO/SAE 21434 (249) and the UNECE R 155 (250) regulation. The next chapter provides a short overview on the safety standards ANSI/UL 4600 (248) and ISO 26262. The third chapter shows how the requirements in this document are derived with ISO 26262 as foundation. In the last section, a justification is given for formulation of additional requirements and how these correspond with the first set of requirements regarding risk classification.

3.1.1 Security Standards

There are several standards regarding automotive cyber security. The most important one being ISO/SAE 21434 proposing measures and processes to protect the vehicle against cyber attacks during the entire life cycle. This standard provides the base for compliance with the UNECE R 155 regulation, a mandatory part of European vehicle type approval.

3.1.1.1 ISO/SAE 21434

The ISO/SAE 21434 (249) is a norm that focusses on the cybersecurity of systems within an automotive context. It gives guidance and normative information on measures, such as the definition of cybersecurity goals, requirements and activities, to improve and control cybersecurity throughout the entire product lifecycle of a system.

According to the ISO/SAE 21434 a cybersecurity goal is formulated during the concept phase as high-level requirements related to possible threats. Then more detailed cybersecurity requirements are derived from these goals and the cybersecurity requirements are then refined into technical cybersecurity activities and controls to be implemented.

These cybersecurity activities can be classified in relation to the risk of possible threats through Cybersecurity Assurance Levels (CALs). These levels are formulated as non-technical requirements defining the level of rigor of the cybersecurity activities to be implemented in order to ensure an appropriate risk reduction for the defined threats. During the concept phase of the product lifecycle, the CALs are determined and assigned to the cybersecurity goals and inherited by their corresponding cybersecurity requirements. Additionally, the CALs can be used during the product development phase to further control the rigor of cybersecurity measures by, for example, CAL-dependent levels of independence or test parameters for specific activities.

In the scope of this project, the ISO/SAE 21434 was used to derive and refine requirements relevant to the cybersecurity of the AD system.

3.1.1.2 UNECE R 155

The United Nations Economic Commission for Europe (UNECE) Regulation 155 for Cyber Security (250) addresses the topic of cyber security attacks on vehicles. It will become a mandatory part of the type approval in all UNECE member countries: applicable to new vehicle types from July 2022 and from July 2024 to all vehicles. The regulation demands the implementation of a Cyber Security Management System (CSMS), "a systematic risk based approach defining organisational processes, responsibilities and governance to treat risk associated with cyber threats to vehicles and protect them from cyber attacks" (250).

For type approval, the technical service or approval authority has to verify by document review if the requirements of the standard are implemented by the CSMS. Supplier-related risks shall be identified and managed. Additionally, the standard requires the implementation of suitable measures for mitigating cyber security threats and processes to verify that the applied techniques are effective. The processes and measures should cover the entire life cycle, from development, production to post-production.

UN R 155 provides lists for threats and mitigations which have to be taken into account. The lists are not complete and additional relevant threats and mitigations, e.g., specific for the vehicle type, should be considered as well.

A risk assessment based on design and development documents has to be conducted including results of testing and design choices for risk mitigation or enhancing risk assessment. It has to be ensured that risk assessment is always up-to-date and includes the manufacturer's expertise and findings as well as external sources, e.g., vulnerability databases.

Incident detection and response mechanisms shall be supported by monitoring and logging for forensic analysis. The occurrence of new cyber security threats and their coverage by existing measures has to be checked continuously. New cyber security attacks shall be reported to the approval authority.

The implemented measures for protecting the vehicle against cyber security threats are verified by sample testing. The testing process shall focus on high risks, but still cover all kinds of risks identified during risk assessment.

The regulation demands a structured approach to cyber security of vehicles with defined processes and measures without giving specific advice for their implementation.

3.1.2 Safety Standards

The most extensive safety standard in automotive is ISO 26262. Its implementation provides functional safety to road vehicles by work products and defined requirements. ANSI/UL 4600 proposes an argumentation framework to ensure an automotive system is safe. Thereby, ANSI/UL 4600 can cooperate with other standards, e.g., ISO 26262, to help build the argumentation structure.

3.1.2.1 ANSI/UL 4600

The American National Standards Institute (ANSI) / Underwriters Laboratories (UL) 4600 standard for safety for evaluation of autonomous products (248) targets the safe operation of fully autonomous vehicles without driver. Its goal is to establish an argumentation structure to prove that an autonomous vehicle is sufficiently safe. The safety standard is goal based. It helps in defining safety goals and gives practical advice on how to achieve them, but does not state mandatory instructions. UL 4600 complements and can cooperate with other standards, e.g., by using work products from ISO 26262 to fulfill a safety goal.

The scope of the standard incorporates hazard identification, safety cases, risk mitigation and the overall context of the use case including, for example, the ODD. Since security is also affecting safety, the standard demands a security plan, but does not go into detail how it should be implemented. The approach is on system level and concerns the whole life cycle of AD systems, e.g., planning, development and deployment.

The main concept of UL 4600 is the definition of a safety case, an argumentation structure ensuring that the autonomous vehicle performs sufficiently safe and is ready for deployment. A safety case contains safety-ensuring goals or claims towards the autonomous vehicle (functions). Arguments supporting the claims have to be constructed. Evidence should prove that the arguments are valid and thus the respective claim is fulfilled. One is free in choosing the form of notation, as long as the safety case is described adequately and entirely. The standard does not dictate how a claim should be fulfilled, but gives practical advice in this regard. This freedom in implementation makes the standard quite flexible, which can be especially beneficial when dealing with an AI system whose behavior might be not entirely comprehensible.

The standard's clauses give suggestions for topics that should be addressed in the safety case, but do not specify how these should be targeted. Each clause has prompt elements refining the corresponding topic. The prompt elements come with a use case-independent application note. Mandatory prompt elements form a group of safety topics that shall be implemented by all means, e.g., hazard identification is strictly necessary. Required prompt elements can only be omitted when they are inherently not viable. Highly recommended elements can be omitted when a coherent justification is given, while recommended prompt elements are optional. Additionally, a conformance paragraph describes how the addressed clause can be assessed to verify if the requirements of the standard are met, thus ensuring safe operation.

For risk categorization, the UL 4600 risk evaluation demands that a criticality level, describing the risk, shall be assigned to each hazard. Furthermore, a prompt element of the standard requires the use of integrity levels, e.g., from ISO 26262, or similar techniques.

The safety case and its claims have to be constantly monitored and evaluated during deployment. This especially holds for assumptions and unknowns. For this purpose, safety metrics shall help monitoring safe functionality. Correcting measures have to be implemented, if necessary. These feedback loops help to reduce the risks of unknowns and potential gaps of the safety cases, thus continuously increasing safety even during deployment.

3.1.2.2 ISO 26262

ISO 26262 „Road vehicles – Functional safety" is an ISO standard for the safety of electronical systems in vehicles. Its goal is the correct and safe functionality of a system in its intended environment. The standard has 12 parts covering topics such as management activities and organizational aspects, hazard analysis and risk assessment, as well as requirements for development, production and monitoring.

The main part deals with system, hardware and software development and provides procedures, work products and requirements towards vehicle safety. For the latter, different methods for implementation can be utilized depending on the identified integrity level of the system.

The Automotive Safety Integrity Level (ASIL) concept is one of the key elements of ISO 26262. It is a four-level classification from ASIL A to ASIL D to describe the risk of a system within a road vehicle. The levels are defined based on the effect a hazard for the considered system can have. Namely, it is calculated based on the

severity of injuries from possible accidents. Table 3 gives an overview on how these three factors impact the ASIL classification of the system. ASIL D is the highest classification level linked to the highest exposure, severity and uncontrollability of the system. Quality management (QM) methods can already mitigate certain basic risks. For a system whose risks are already covered by QM, it is not necessary to apply methods suggested by the ASIL recommendations.

Table 3 Overview on the derivation of the ASII classifications taken from (165)

Severity	*Exposure*	*Controllability*		
		Simple	*Normal*	*Difficult, uncontrollable*
Light and moderate injuries	Very low	QM	QM	QM
	Low	QM	QM	QM
	Medium	QM	QM	A
	High	QM	A	B
Severe and life threating injuries, survival probable	Very low	QM	QM	QM
	Low	QM	QM	A
	Medium	QM	A	B
	High	A	B	C
Life threatening and fatal injuries	Very low	QM	QM	A
	Low	QM	A	B
	Medium	A	B	C
	High	B	C	D

The ASIL recommendations are formulated along the entire product development process. For each of the different aspects of the development process, several methods for ensuring the functional safety of the system are defined. For each aspect of the development process a table is defined mapping the suggested method to an ASIL recommendation for each ASIL. The recommendation is described as not recommended (o), recommended (+) and highly recommended (++). Despite calling them *recommendations*, ISO 26262 states that highly recommended methods shall be implemented. However, for each aspect it is sufficient to implement an appropriate combination of entries, if a rationale is given that this combination is suitable to address the safety concerns (165).

3.1.3 ASIL-derived Requirements

To formulate requirements for ADAS and AD systems, the requirements of ISO 26262-6 (247) for product development at the software level provide a groundwork. Specific ASIL recommendations are chosen that are either impacted by AI components or have to be further refined to fit AI components. It is important to note that the ASIL recommendations not explicitly mentioned in this document are not affected by the AI components and can therefore be implemented and addressed by techniques suited for regular software systems. The following sections give an overview of those chosen ASIL recommendations from which the final general requirements will be derived.

3.1.3.1 System Level

The ISO 26262 starts by defining ASIL recommendations at the system level. These recommendations address functional safety concerns along the product development at the stage of integrating all existing components, including potential AI subsystems, into the vehicle.

First, Table 4 gives an overview of ASIL recommendations for integration testing at the system level. The recommended methods should be used to derive test cases for the entire system, after all components are integrated.

Table 4 Methods for deriving test cases for integration testing (DI) taken from (247)

Method		ASIL A	ASIL B	ASIL C	ASIL D
DI1	Analysis of requirements	++	++	++	++
DI2	Analysis of external and internal interfaces	+	++	++	++
DI3	Generation and analysis of equivalence classes for hardware-software integration	+	+	++	++
DI4	Analysis of boundary values	+	+	++	++
DI5	Error guessing based knowledge or experience	+	+	++	++
DI6	Analysis of functional dependencies	+	+	++	++
DI7	Analysis of common limit conditions, sequences and sources of dependent failures	+	+	++	++
DI8	Analysis of environmental conditions and operational use cases	+	++	++	++
DI9	Analysis of field experience	+	++	++	++

Secondly, test methods to ensure the correct and consistent implementation of interfaces are defined in Table 5.

Table 5 Methods for consistent and correct implementation of external and internal interfaces (CI) at the hardware-software level taken from (247)

Method		ASIL A	ASIL B	ASIL C	ASIL D
CI1	Test of external interfaces	+	++	++	++
CI2	Test of internal interfaces	+	++	++	++
CI3	Interface consistency check	+	++	++	++

Since the robustness of AI systems has to be estimated differently to regular software systems, the ASIL recommendations in Table 6 will be affected by the AI component.

Table 6 Level of robustness at the system (RS) level taken from (247)

Method		ASIL A	ASIL B	ASIL C	ASIL D
RS1	Resource usage test	o	+	++	++
RS2	Stress test	o	+	++	++
RS3	Test for interference resistance and robustness under certain environmental conditions	++	++	++	++

The last recommendations to be included in this section are towards the testing of performance and the safety at the vehicle level defined in Table 7. In general, the testing procedure of the system has to be adjusted to fit AD/ADAS and AI system.

Table 7 Methods for correct functional performance, accuracy and timing of safety mechanisms at the vehicle level (FP) taken from (247)

Method		ASIL A	ASIL B	ASIL C	ASIL D
FP1	Performance test	+	+	++	++
FP2	Long-term test	+	+	++	++
FP3	User test under real-life conditions	+	+	++	++
FP4	Fault injection test	o	+	++	++
FP5	Error guessing test	o	+	++	++
FP6	Test derived from field experience	o	+	++	++

3.1.3.2 Software Integration

In addition to the recommendations catered towards the derivation of test cases for system level integration testing, the ISO 26262 defines recommendations for the integration testing at the software level. These recommendations describe the integration steps that should be taken until the software component is fully integrated into the system.

The methods in Table 8 focus on the notations for the documentation of the software architectural design. These methods can be converted to serve the purpose of describing the AI model and lifecycle of the architectural design when deriving the general requirements.

Table 8 Notations for the software architectural design (NA) taken from (247)

Method		*ASIL A*	*ASIL B*	*ASIL C*	*ASIL D*
NA1	Natural language	++	++	++	++
NA2	Informal notations	++	++	+	+
NA3	Semi-formal notations	+	+	++	++
NA4	Formal notations	+	+	+	+

Table 9 introduces methods regarding the verification of the software integration into the system. Later, these can ensure that the integration of the AI subsystems into the entire system is tested and verified thoroughly. Here, it is important to note that the language of the table is taken from the ISO 26262, which is formulated towards conventional software product. Therefore, the term model that is referenced in method IV5 relates to the modelling of the problem and software system and is not unanimous with the AI model within AI subsystems.

Table 9 ASIL recommendations to verify the software integration (IV) taken from (247)

Method		*ASIL A*	*ASIL B*	*ASIL C*	*ASIL D*
IV1	Requirements-based test	++	++	++	++
IV2	Interface test	++	++	++	++
IV3	Fault injection test	+	+	++	++
IV4	Resource usage evaluation	++	++	++	++
IV5	Back-to-back comparison test between model and code, if applicable	+	+	++	++
IV6	Verification of the control flow and data flow	+	+	++	++
IV7	Static code analysis	++	++	++	++
IV8	Static analyses based on abstract interpretation	+	+	+	+

Additionally, to ensure that the software integration into the system is verified properly, the ISO defines recommendations on the testing environment of the system (see Table 10). They give a high recommendation for Hardware-in-the-loop testing and testing in electronic control unit network environments for all ASILs. Vehicle tests are highly recommended starting from ASIL C to ASIL D.

Table 10 ASIL recommendations for software testing (ST) taken from (247)

Method		*ASIL A*	*ASIL B*	*ASIL C*	*ASIL D*
ST1	Hardware-in-the-loop	++	++	++	++
ST2	Electronic control unit network environments	++	++	++	++
ST3	Vehicles	+	+	++	++

The types of tests that are suggested in Table 11 for the software integration stage are requirements-based tests highly recommended for all ASILs and fault injection testing is highly recommended for ASIL D.

Table 11 ASIL recommendations for embedded software testing (ET) taken from (247)

Method		*ASIL A*	*ASIL B*	*ASIL C*	*ASIL D*
ET1	Requirements-based test	++	++	++	++
ET2	Fault injection test	+	+	+	++

To derive the actual test cases to be used for the above-suggested testing methods, the ISO 26262 defines a number of different methods in Table 12.

Table 12 ASIL recommendations for deriving test cases for embedded software testing (DE) taken from (247)

Method		*ASIL A*	*ASIL B*	*ASIL C*	*ASIL D*
DE1	Analysis of requirements	++	++	++	++
DE2	Generation and analysis of equivalence classes	+	++	++	++
DE3	Analysis of boundary values	+	+	++	++
DE4	Error guessing based on knowledge or experience	+	+	++	++
DE5	Analysis of functional dependencies	+	+	++	++
DE6	Analysis of operational use cases	+	++	++	++

3.1.3.3 Software Unit

Since the ISO 26262 defines techniques along the entire product development process, it also addresses the design, development and testing of the software unit. In the case of an AI/AD system, the software unit is composed of the AI subsystems. Requirements formulated towards the following ASIL recommendations are defined in Chapter 3.3.

Analogous to the architectural design documentation of the entire system, Table 13 holds the notations for the documentation of the software unit design. Additionally, guidelines for modelling and coding of the software unit (see Table 14) are recommended. These methods on the one hand address techniques for clean and robust coding of the software unit (e.g. MC4) and on the other hand further address the documentation and design of the software unit (e.g. MC6).

Table 13 Notations for the software unit design (NU) taken from (247)

Method		*ASIL A*	*ASIL B*	*ASIL C*	*ASIL D*
NU1	Natural language	++	++	++	++
NU2	Informal notations	++	++	+	+
NU3	Semi-formal notations	+	+	++	++
NU4	Formal notations	+	+	+	+

Table 14 ASIL recommendations for modelling and coding guidelines (MC) taken from *(247)*

Method		*ASIL A*	*ASIL B*	*ASIL C*	*ASIL D*
MC1	Enforcement of low complexity	++	++	++	++
MC2	Use of language subset	++	++	++	++
MC3	Enforcement of strong typing	++	++	++	++
MC4	Use of defensive implementation techniques	+	+	++	++
MC5	Use well-trusted design principles	+	+	++	++
MC6	Use of unambiguous graphical representation	+	++	++	++
MC7	Use of style guides	+	++	++	++
MC8	Use of naming conventions	++	++	++	++
MC9	Concurrency aspects	+	+	+	+

Finally, Table 15 completes the software unit recommendations, by defining techniques towards the derivation of test cases for the software unit. These methods cater to derive actual test cases to support the testing and verification methods defined in Table 16.

Table 15 ASIL recommendations for deriving test cases for software unit testing (DU) taken from (247)

Method		*ASIL A*	*ASIL B*	*ASIL C*	*ASIL D*
DU1	Analysis of requirements	++	++	++	++
DU2	Generation and analysis of equivalence classes	+	++	++	++
DU3	Analysis of boundary values	+	++	++	++
DU4	Error guessing based on knowledge or experience	+	+	+	+

Again, it is important to note that just like in IV5 in Table 9, UV14 references the modelling of the system and not an AI model.

Table 16 ASIL recommendations for software unit verification (UV) taken from (247)

Method		*ASIL A*	*ASIL B*	*ASIL C*	*ASIL D*
UV1	Walk-through	++	+	o	o
UV2	Pair-programming	+	+	+	+
UV3	Inspection	+	++	++	++
UV4	Semi-formal verification	+	+	++	++
UV5	Formal verification	o	o	+	+
UV6	Control flow analysis	+	+	++	++
UV7	Data flow analysis	+	+	++	++
UV8	Static code analysis	++	++	++	++
UV9	Static analyses based on abstract interpretation	+	+	+	+
UV10	Requirements-based test	++	++	++	++
UV11	Interface test	++	++	++	++
UV12	Fault injection test	+	+	+	++
UV13	Resource usage evaluation	+	+	+	++
UV14	Back-to-back comparison test between model and code, if applicable	+	+	++	++

3.1.3.4 Monitoring

Finally, the ISO 26262 touches upon techniques for error detection and error handling. However, the techniques come without specific ASIL recommendations. Since error detection and handling methods play a vital role for monitoring of AD and AI subsystems, the methods are included in this document.

To provide consistency to the other method suggestions by the ISO 26262, risk levels were defined based on the risk classification defined in Section 3.1.4. Table 17 gives an overview on the methods for error detection and Table 18 gives an overview of suggested error handling methods.

Table 17 Error detection methods (ED) from (247) with additional risk levels

Method		*Risk*			
		Low	*Medium*	*High*	*Very high*
ED1	Range checks of input and output data	+	++	++	++
ED2	Plausibility check	+	+	+	++
ED3	Detection of data errors	+	++	++	++
ED4	Monitoring of program execution by an external element	o	+	++	++
ED5	Temporal monitoring of program execution	o	+	++	++
ED6	Diverse redundancy in the design	o	+	++	++
ED7	Access violation control mechanisms	o	+	++	++

Table 18 Error handling methods (EH) from (247) with additional risk levels

Method		*Risk*			
		Low	*Medium*	*High*	*Very high*
EH1	Deactivation in order to achieve or maintain a safe state	+	+	++	++
EH2	Static recovery mechanism	+	+	+	+
EH3	Graceful degradation	+	+	++	++
EH4	Homogenous redundancy in the design	o	o	+	++
EH5	Diverse redundancy in the design	o	o	+	++
EH6	Correcting codes or data	+	+	+	++
EH7	Access permission management	o	+	++	++

3.1.4 Additional Requirements

As explained above, the ASIL recommendations are formulated for regular software systems. However, AD systems and AI subsystems face different risks and vulnerabilities than regular software systems. Therefore, there is a need to address these risks and vulnerabilities with supplementary requirements. Since the ASIL classification is well defined (see Table 3), the additional requirements are categorized along a four-level risk and damage classification (low, medium, high and very high risk) that can be directly mapped to ASIL A to ASIL D.

The resulting list of ASIL-derived and additional requirements is formulated to address a large number of possible risks for AD/ADAS systems. Therefore, it has to be determined for each specific use case, whether a requirement is relevant or even feasible. However, if a requirement is not implemented a rationale should be given to reason that the requirement is not feasible to be implemented, is not relevant to the use case or that it is already addressed through other requirements.

3.2 Entire System

In this chapter, general requirements for entire AD systems consisting of multiple (hardware) components and AI subsystems embedded in an environmental context are discussed. The requirements address the performance, robustness, interpretability, monitoring and documentation of the system. Requirements specific for AI subsystems are described later in Chapter 3.3.

3.2.1 General

Here, some general requirements towards the entire AD system are defined. These requirements are not assigned to one of the categories performance, robustness, interpretability, monitoring and documentation and are composed in this general category.

3.2.1.1 ASIL-derived Requirements

3.2.1.1.1 Environment Compliance

Requirement 1: *The environmental context shall correspond to the operational design domain (ODD).*

Requirement 1 is derived from the ASIL recommendations DI8 and DI9 addressing the methods to derive test cases for the integration testing on the system level defined in Table 4. It is important that the system is only used in the context it is developed for. The environment the systems is deployed in must be a part of the ODD that is specified during the development of the system. In case the environmental context changes both suddenly and significantly and the system falls outside of the ODD, processes must exist that lead the system back into a safe state when the system risk is high or very high.

3.2.1.1.2 Component Interaction

Requirement 2: *The communication, interfaces, signals, etc. between different components shall be coordinated.*

The next requirement describes that individual components are connected correctly and that the signal flow works as intended. It combines the interface testing methods CI1 – CI3 defined in Table 5, as well as IV2 from Table 9. The interplay between different components is correctly initialized and the output from prior components is in the correct form to be used by the following ones. Also, it is required that error or warning messages generated by one component are not ignored by other connected components but are reacted upon.

Table 19 General ASIL-derived requirements with ASIL recommendations from the ISO 26262 (165; 246; 247) for the entire system

Method		*ASIL recommendation*			
		ASIL A	*ASIL B*	*ASIL C*	*ASIL D*
1	The environmental context shall correspond to the operational design domain (ODD).	+	++	++	++
2	The communication, interfaces, signals, etc. between different components shall be coordinated.	+	++	++	++

3.2.1.2 Additional Requirements

3.2.1.2.1 Sensor Setup Compliance

Requirement 3: *The sensor setup shall be similar to the development/training setup.*

Like Requirement 1, it is important that the sensor setup that captures the relevant environmental context is comparable to the setup used during development or training. This ensures that the respective output can be trusted and that the environmental context is captured correctly. Likewise, it is also required that all sensors (hardware components) operate correctly and lie in their own ODD. If a calibration is required, it is performed correctly and verified. Again, processes must be in place that bring the entire system into a safe state when a significant divergence in the sensor setup occurs. As explained, this requirement ensures the safety of the system, therefore it should be implemented starting from medium risk application (see Table 20).

In addition to requirements that are generally applicable to general purpose AI systems that consist of multiple components, there are also requirements that are needed specifically for systems in the mobility area. Most importantly, for systems that are used for automated/autonomous driving it is required that the vehicle model/capabilities are similar to the development/training setup. This is an extension to the general Requirement 3 for the case that the system also has actuators in addition to sensors. However, not every AI system is connected to actuators and thus the general requirement is formulated only for sensors.

3.2.1.2.2 General Validity

Requirement 4: *The requirements for AI subsystems shall apply to the entire system (if applicable).*

The last overall requirement is that individual requirements for AI subsystems are also applied to the entire system if possible. For example, this includes the requirements described later in this document in Chapter 3.3. If it is applicable, these individual aspects shall be fulfilled for each component of the entire system. This in term enables that all individual aspects can also be fulfilled for the entire system by combining the information of individual components and enhancing it to be representative of the behavior of the entire system.

Table 20 General additional requirements with risk levels

Method		*Risk*			
		Low	*Medium*	*High*	*Very high*
3	The sensor setup shall be similar to the development/training setup.	+	++	++	++
4	The requirements for AI subsystems shall apply to the entire system (if applicable).	++	++	++	++

3.2.2 Performance

In the following, requirements specific for the aspect of performance are listed. Thereby, the focus on performance is in terms of the quality of a system and not in terms of computational resources and efficiency. In addition, in this requirement category on performance it is possible to derive specialized requirements for systems in the mobility sector. For example, in the case of AD/ADAS systems concrete KPIs can be specified to meaningfully represent Requirement 6 and Requirement 7 in this domain. One example for a high-level KPI would be to measure the average number of kilometers driven between any accidents. Then, a concrete requirement could assign a certain lower limit on the number of kilometers, which need to be fulfilled. Alternatively, the number of critical accidents in a certain time/distance frame serves as another exemplary very high-level KPI. A more detailed discussion on the applicability of the requirement is presented later in Chapter 3.4.

3.2.2.1 ASIL-derived Requirements

3.2.2.1.1 Performance Guarantee

Requirement 5: *The adequate performance shall be guaranteed for a certain timeframe after initial deployment.*

This requirement focusses on the behavior of the system when in operation and it implements ASIL recommendation FP2 from Table 7. Here, it is needed that the behavior of the system is guaranteed for a certain amount of time after the initial deployment of a system. This is similar to a warranty for traditional hardware products. The system must behave correctly when operated in the ODD for at least the specified timeframe. A voluntary extension of the mandatory timeframe is possible to provide behavior guarantees for example over the entire lifetime of the system.

Table 21 ASIL-derived performance requirements with ASIL recommendations from the ISO 26262 (165; 246; 247) for the entire system

Method		*ASIL recommendation*			
		ASIL A	*ASIL B*	*ASIL C*	*ASIL D*
5	The adequate performance shall be guaranteed for a certain timeframe after initial deployment.	+	+	++	++

3.2.2.2 Additional Requirements

3.2.2.2.1 Performance KPIs

Requirement 6: *The performance on key performance indicators (KPIs) shall be as high as possible.*

The next performance requirement for every general-purpose AI system is that it achieves the highest possible KPIs for the concrete task on test data. Concretely, the KPI depends on the task the AI system is supposed to solve. Thus, this requirement must be specified for each specific application when certain KPIs can be defined. For example, for stock value prediction the absolute divergence at a given point in time might be a suitable KPI, whereas for trajectory prediction of traffic participants the displacement error at a given point in time is suitable.

Table 22 Requirement regarding the performance on KPIs with risk levels

Method		*Risk*			
		Low	*Medium*	*High*	*Very high*
6	The performance on key performance indicators (KPIs) shall be as high as possible.	+	++	++	++

3.2.3 Robustness

After addressing the functionality in terms of performance, it is equally important to address that the system's performance is correct under different circumstances. Hence, it is necessary to define requirements ensuring its robustness.

3.2.3.1 ASIL-derived Requirements

3.2.3.1.1 Performance on Worst-Case Error

Requirement 7: *The performance shall be compliant to the allowed worst-case error.*

In addition to performing well in terms of the respective KPIs for a given task, it is desirable that the system is compliant to the allowed worst-case error or deviation. Hence, this requirement reflects whether the performance of a system is in the allowed boundary conditions on the entire test data set. Whereas Requirement 6 measures the average performance, this requirement relates to the allowed risk in safety critical applications. This requirement addresses the testing methods for the system level RS2 and RS3 from Table 6, as well as FP4, FP5 and FP6 from Table 7, IV3 from Table 9 and finally ET2 from Table 11. Further, this requirement addresses the methods for deriving test cases for the embedded software testing DE3 and DE4 from Table 12.

3.2.3.1.2 Performance Reproducibility

Requirement 8: *The performance shall be reproducible in the real environment for operation.*

Requirement 6 and Requirement 7 relate to the main performance characteristics on test data, which is representative of the environment the system operates in after deployment. However, it is also relevant that the behavior of the system can be reproduced in the real environment it operates in. This is reflected in this requirement, which states that Requirement 6 and Requirement 7 need to be fulfilled also during operation and not only on previously captured/determined/generated test data. Thereby, the usage in the real operating environment inside the ODD is enabled. Overall, this requirement combines ASIL recommendations FP3 from Table 7, ST3 from Table 10 and IV4 from Table 9.

Table 23 ASIL-derived robustness requirements with ASIL recommendations from the ISO 26262 (165; 246; 247) for the entire system

Method		*ASIL recommendation*			
		ASIL A	*ASIL B*	*ASIL C*	*ASIL D*
7	The performance shall be compliant to the allowed worst-case error.	++	++	++	++
8	The performance shall be reproducible in the real environment for operation.	+	++	++	++

3.2.4 Monitoring

Here, requirements are proposed that indicate which aspects of the system should be supervised to enable downstream evaluation or active regulation, e.g. by updating the AI system.

3.2.4.1 ASIL-derived Requirements

3.2.4.1.1 Operation Monitoring

Requirement 9: *The feedback of the system shall be tracked while in operation.*

Requirement 9 focusses on the tracking of the system behavior when in operation, by refining ED4 and ED5 from Table 17. During operation, it is important to track when any deviation from the ODD occurs, because this could show that the ODD is chosen not ideally or the user misuses the system. Similarly, it is important to track any feedback or intervention from the user. This could again show that the system does not operate in the ODD or does not show an ideal behavior. Additionally, any (critical) mistakes of the system should be tracked to analyze potential shortcomings in the design.

3.2.4.1.2 Error Correction

Requirement 10: *The performance shall be corrected when critical errors occur after deployment.*

Lastly, the final ASIL-derived monitoring requirement from EH6 in Table 18 is that the performance of the system is continuously tracked and that (critical) errors are corrected after deployment for high-risk use cases. As long as the system is used in the intended ODD any repeatedly reoccurring errors need to be fixed. This is again similar to the common recall practice that is often used in many different industries. In the case of AI systems where the behavior is largely determined by software, it is also possible that the recall/update is conducted via an over-the-air software update.

Table 24 ASIL-derived monitoring requirements with ASIL recommendations from the ISO 26262 (165; 246; 247) for the entire system

Method		*ASIL recommendation*			
		ASIL A	*ASIL B*	*ASIL C*	*ASIL D*
9	The feedback of the system shall be tracked while in operation.	o	+	++	++
10	The performance shall be corrected when critical errors occur after deployment.	+	+	+	++

3.2.4.2 Additional Requirements

3.2.4.2.1 Reproducibility

Requirement 11: *The system state shall be tracked in a reproducible way while in operation.*

In the case of errors or accidents, it is necessary to obtain as much information as possible of the system. For this purpose, it is important that the system state is tracked in such a way that the entire system's behavior can be reproduced. The information to be tracked for this has to be defined for each system and component individually.

Table 25 Requirement for the reproducibility of the system with risk levels

Method		*Risk*			
		Low	*Medium*	*High*	*Very high*
11	The system state shall be tracked in a reproducible way while in operation.	+	+	++	++

3.2.5 Documentation & Lifecycle

In this chapter, requirements for documentation purposes are introduced. They are related to the AI lifecycle and are relevant for the overall system.

3.2.5.1 ASIL-derived Requirements

3.2.5.1.1 Architectural Documentation

Requirement 12: *The architectural design shall be described explicitly.*

This requirement includes the recommendations from Table 8. A documentation of the architecture of the entire system is crucial for evaluation and assessment. The architectural description shall comprise the overall structure and the interfaces between the specific SW units. A clear and unambiguous description in natural language is always required. For low security applications, an additional informal description is sufficient. For (very) high security applications, at least a semi-formal notation is necessary. This requirement addresses the documentation of the overall system, therefore it may be partially covered by the documentation for regular non-AD/AI components.

3.2.5.1.2 Developer Eligibility

Requirement 13: *The quality & trustworthiness for developers shall be assessed.*

Also, requirements can be imposed on the developers of a system that contains AI components. Depending on the safety criticality and risk of the intended usage different variants of this requirement are possible. On the one hand, it is important that a proper qualification of the developers is ensured. This includes required training assignments and raising the risk awareness for potential risks that need to be considered during development depending on the overall risk level of the system. On the other hand, this can also include the need for a security clearance or screening process when working on extremely high-risk use cases where the influence of external parties/countries/etc. must be ruled out. This requirement could be defined as an additional requirement, however it touches on the recommendations ED7 and EH7 from Table 17 and Table 18 by controlling the access to the development facility.

Table 26 ASIL-derived documentation requirements with ASIL recommendations from the ISO 26262 (165; 246; 247) for the entire system

Method		*ASIL recommendation*			
		ASIL A	*ASIL B*	*ASIL C*	*ASIL D*
12	The architectural design shall be described explicitly.	++	++	++	++
13	The quality & trustworthiness for developers shall be assessed.	o	+	++	++

3.2.5.2 Additional Requirements

3.2.5.2.1 Development Documentation

Requirement 14: *The development process shall be tracked.*

It is required that important aspects of the development of the system are tracked. Most importantly, this includes the training and the architectural changes made during development. Additionally, any changes in the ODD that occurred during development need to be tracked and justified. Finally, this also includes to track the principles of the internal evaluation and the evolution of the tests performed. This is strongly connected to requirements on individual aspects like performance or robustness.

Table 27 Development documentation requirement with risk levels

Method		*Risk*			
		Low	*Medium*	*High*	*Very high*
14	The development process shall be tracked.	+	++	++	++

3.2.6 Summary of Requirements

Table 28 summarizes all requirements[7] presented in this chapter with their corresponding risk level. For requirements that were derived from several ASIL recommendations the strictest ASIL classification was chosen. For the additional requirements, the previous sections presented definitions alongside descriptive paragraphs elucidating their ASIL recommendations.

Table 28 Summary of generic requirements for the entire system[7]

Requirement			*Risk level*			
ID	*Description*	*Type*	*ASIL A/ Low*	*ASIL B/ Medium*	*ASIL C/ High*	*ASIL D/ Very high*
1	The environmental context shall correspond to the operational design domain (ODD).	ASIL	+	++	++	++
2	The communication, interfaces, signals, etc. between different components shall be coordinated.	ASIL	+	++	++	++
3	The sensor setup shall be similar to the development/training setup.	Additional	+	++	++	++
4	The requirements for AI subsystems shall apply to the entire system (if applicable).	Additional	++	++	++	++
5	The adequate performance shall be guaranteed for a certain timeframe after initial deployment.	ASIL	+	+	++	++
6	The performance on key performance indicators (KPIs) shall be as high as possible.	Additional	+	++	++	++
7	The performance shall be compliant to the allowed worst-case error.	ASIL	++	++	++	++

[7] The aim of our efforts is to develop firm requirements that mobility systems containing AI modules have to follow. In the work presented here the requirements are a first draft and the basis for further iterations. In their current state some of them are more to consider as a best practice and an ideal state in terms of safety and security. Nevertheless, the boundaries, values and wording of the requirements will be adjusted within further projects and actual automotive applications.

Requirement			*Risk level*			
8	The performance shall be reproducible in the real environment for operation.	ASIL	+	++	++	++
9	The feedback of the system shall be tracked while in operation.	ASIL	o	+	++	++
10	The performance shall be corrected when critical errors occur after deployment.	ASIL	+	+	+	++
11	The system state shall be tracked in a reproducible way while in operation.	Additional	+	+	++	++
12	The architectural design shall be described explicitly.	ASIL	++	++	++	++
13	The quality & trustworthiness for developers shall be assessed.	ASIL	o	+	++	++
14	The development process shall be tracked.	Additional	+	++	++	++

3.3 AI Subsystem

In this chapter, the generic requirements towards the AI subsystems within AD/ADAS systems are defined. The requirements are defined for each of the aspects performance, robustness, interpretability, monitoring and documentation. As discussed in Chapter 3.1, for each aspect, ASIL-derived requirements and additional requirements are formulated. At the end of this chapter, Section 3.3.6 provides an overview of all requirements and their risk level recommendations.

3.3.1 Performance

Since AI subsystems are part of the entire system, the requirements for the entire system are hierarchical. Therefore, Requirement 5 and Requirement 6 from Chapter 3.2.2 still hold for the AI subsystems.

3.3.2 Robustness

As discussed in (1), AI systems are susceptible to various robustness threats, which poses a high risk for the entire AD system. To address these concerns, robustness requirements for these AI systems are introduced in this section.

3.3.2.1 ASIL-derived Requirements

Based on the ASIL recommendations for modelling and coding (MC) guidelines, deriving test cases for software unit testing (DU) and software unit verification (UV), the following robustness requirements can be derived.

3.3.2.1.1 Robustness Improvement

Requirement 15: *The AI model shall be implemented using mitigation strategies against robustness threats.*

From the ASIL recommendations for the modelling and coding guidelines listed in Table 14, Recommendation MC4 needs to be further specified in Requirement 15. To enhance the robustness of the AI model against attacks and robustness threats, suitable mitigation strategies as described in (1) shall be used.

3.3.2.1.2 Software Verification and Testing

The ISO 26262 recommends several methods aiding the software unit verification for road vehicle systems, which are listed in Table 16 in combination with their ASIL recommendation. It is important to note that the table follows the ISO 26262 definition, which focusses on software development for non-AI systems.

Therefore, the term *model* referenced in UV14 describes the software modelling of a regular software system and is not equivalent to an AI model. From the listed methods UV4, UV5, UV10 and UV12 have to be defined further for AI models.

Requirement 16: *The AI model shall be verified with formal robustness verification techniques.*

UV5 recommends the formal verification of the system for ASIL C and ASIL D, which can be further defined into Requirement 16. As stated in (1), formal robustness verification of AI models is not a trivial problem and not feasible for all models. Therefore, in cases where formal robustness verification is infeasible, Requirement 17 shall be used as a substitution.

Requirement 17: *The robustness of the AI model shall be verified with empirical robustness estimation techniques.*

Requirement 17 addresses the recommendation for semi-formal verification UV4, as robustness estimation can be used to statistically verify the robustness of the model. Since empirical robustness is based on fault-injection testing with data that is perturbed by worst-case perturbations generated by, for example, adversarial attacks or translations, this requirement additionally ensures UV12.

Requirement 18: *The AI model shall be tested against out-of-distribution data.*

By completing the testing procedure through out-of-distribution data testing defined in Requirement 18, UV10 can be ensured.

3.3.2.1.3 Deriving Test Cases for Software Unit Testing

To derive test cases for software units, ASIL defines methods to guide the derivation of test cases, defined in Table 15. DU3 and DU4 address the robustness of the AI model and are further refined in Requirement 19, Requirement 20 and Requirement 21.

Requirement 19: *Test cases at the boundary values of the input of the AI model shall be derived.*

To derive adequate test cases that determine the robustness of the model at the input boundary values, the input boundary values shall be analyzed. As mentioned in the beginning of Chapter 3, the input boundary values are system-dependent. Some use-cases will come with clear input boundaries, e.g., defined by documentation or domain experts, while others have infinite input space and the requirement is not applicable. The latter shall be covered by a rationale.

Requirement 20: *Test cases based on corner cases of the AI model shall be derived.*

If boundary values can not be defined, Requirement 20 shall be implemented to ensure that the model behaves robust in corner cases. Corner cases depict extreme values or parameter combinations which might lead to unpredictable system behavior. As for input boundary values, corner cases could be already defined in the system's documentation or have to be determined by system analysis of a domain expert. Notably, DU3 has to be implemented for any system with a risk and damage potential above ASIL A.

Requirement 21: *Test cases shall be derived through error guessing based on knowledge and experience of the system.*

Concluding Requirement 21 states that error guessing based on knowledge or experience of domain experts shall be conducted to derive representative test cases for the AI model of the AD system.

Table 29 ASIL-derived robustness requirements with ASIL recommendations from the ISO 26262 (165; 246; 247) for the AI subsystem

Method		*ASIL recommendation*			
		ASIL A	*ASIL B*	*ASIL C*	*ASIL D*
15	The AI model shall be implemented using mitigation strategies against robustness threats.	+	+	++	++

16	The AI model shall be verified with formal robustness verification techniques.	o	o	+	+
17	The robustness of the AI model shall be verified with empirical robustness estimation techniques.	+	+	++	++
18	The AI model shall be tested against out-of-distribution data.	++	++	++	++
19	Test cases at the boundary values of the input of the AI model shall be derived.	+	++	++	++
20	Test cases based on corner cases of the AI model shall be derived.	+	++	++	++
21	Test cases shall be derived through error guessing based on knowledge and experience of the system.	+	+	+	+

3.3.2.2 Additional Requirements

3.3.2.2.1 Fault injection test

Requirement 22: *The AI model shall be tested against possible robustness threats.*

The ISO 26262 recommends fault injection testing in Table 16 for ASIL A to ASIL C and highly recommends it only at ASIL D. Since AI models, different to conventional software systems, base their decisions solely on the input data, it is important to test them against damaged or manipulated data. Additionally, as discussed in (1), AI subsystems are highly susceptible to robustness threats. Therefore, it is highly important to test the AI model against these possible threats, such as IT security threats or data poisoning attacks. For the choice of possible robustness threats and for the test coverage of these threats a rationale shall be given depending on the use case.

Due to the high susceptibility to these threats, there is a higher need for fault injection testing for AI models than suggested in the ISO 26262. To encompass this, Table 30 defines the risk levels for Requirement 22, where fault injection testing shall be done starting from a medium risk and damage potential.

3.3.2.2.2 Data Validation

To ensure robustness for AI subsystems, additional requirements towards the datasets used for training and testing of the AI subsystems are needed. The datasets determine the quality and decisions of the entire subsystem. Therefore, additional requirements regarding data validation (DV) are defined.

Requirement 23: *The source of the datasets shall be traceable.*

To mitigate vulnerabilities like data poisoning or backdoor attacks, the origin of the used datasets has to be clear. It is highly recommended to rely on self-generated datasets of valid, unmanipulated data for high security applications. Well-known open-source datasets can provide an alternative for lower security levels. These should be tested, e.g., for unwanted manipulation (see Requirements 25, 26 and 27).

Requirement 24: *The source of the dataset shall be verified.*

Requirement 25: *The training, test and evaluation datasets shall have adequate coverage of the operational input domain.*

Requirement 25 warrants that the model is trained on representative data of the operational input domain of the system. As stated above the structure and size of the datasets are vital for the AI model's functionality. Therefore, it is important that the datasets have an adequate coverage of the input domain and reflect its features properly incorporating potential environmental constraints of the overall system. A rationale about the coverage of the dataset of the input domain in support of the use case shall be provided.

Requirement 26: *The datasets shall be verified against the safety requirements.*

Additionally, Requirement 26 ensures that the datasets comply with the safety requirements defined for the application. Since specific safety requirements are defined for each use case in detail, they state further information on corner cases, distributions and environmental factors of each individual use case. If the dataset does not comply to or does not contain the needed information, such as for example necessary lighting conditions or corner case scenarios, the datasets have to be adjusted.

Requirement 27: *The uncertainty of the datasets shall be analyzed and quantified.*

Finally, Requirement 27 states that the uncertainty within a dataset shall be analyzed and modelled. This gives more insight into the quality of the datasets and the testing can be adjusted accordingly.

Requirement 28: *The datasets used for training, testing and evaluation shall not contain any errors.*

In order to ensure proper training and testing of the model, the datasets shall not contain any errors. Therefore, the developers shall ensure that the datasets are properly analyzed and any identified errors shall be corrected. Additionally, to further support this requirement, it has to be ensured that the labels used for training are correct and reflect the input domain.

Requirement 29: *The training, test and evaluation datasets shall have sufficient size.*

Since DNN require are large amount of data to efficiently learn generalized features, it is important that the datasets used for training and testing the model are of sufficient size. "Sufficient size" depends on the classification problem of the model, the domain, the data type and environmental constraints. Furthermore, the split-ratio of the datasets shall fit the intended purpose of each dataset. Therefore, a rationale on the choice and size of the dataset shall be provided.

Requirement 30: *The training, test and evaluation datasets shall be independent from each other.*

To ensure a proper training and testing process of the model, the model shall be trained, tested and evaluated on mutually independent datasets. This ensures that accuracy achieved during the training of the model can be projected on newly introduced data. Additionally, this enables the developers to detect issues during the training such as over- or underfitting.

Requirement 31: *The training, test and evaluation datasets shall be prepared in an adequate way.*

The necessary steps shall be taken to prepare the data according to the use case. This can for example entail a data cleaning step removing artifacts or errors from the dataset or a dimensionality reduction step.

Table 30 Additional robustness requirements with risk levels for the AI subsystem

Method		*Risk*			
		Low	*Medium*	*High*	*Very high*
22	The AI model shall be tested against possible robustness threats.	+	++	++	++
23	The source of the datasets shall be traceable.	+	+	++	++
24	The source of the dataset shall be verified.	o	+	++	++
25	The training, test and evaluation datasets shall have adequate coverage of the operational input domain.	+	+	++	++
26	The datasets shall be verified against the safety requirements.	o	+	++	++
27	The uncertainty of the datasets shall be analyzed and quantified.	o	+	++	++
28	The datasets used for training, testing and evaluation shall not contain any errors.	++	++	++	++
29	The training, test and evaluation datasets shall have sufficient size.	+	++	++	++
30	The training, test and evaluation datasets shall be independent from each other.	++	++	++	++
31	The training, test and evaluation datasets shall be prepared in an adequate way.	+	++	++	++

Table 30 gives an overview over the above defined dataset validation requirements. For low-risk applications, there is no need to validate the datasets used for training and testing further, as there is low risk and damage potential stemming from the model. Medium risk applications are recommended to implement these requirements; however, they are not required as the risk and damage is not as severe. Especially, high and very high-risk application shall implement the requirements or a combination of them.

3.3.3 Interpretability

Another challenge of AI subsystems within safety-critical systems is their black-box nature. Since their decisions give no insight into their decision-making process, it is hard to retrace whether a model is trained properly and focusses on relevant data points within the input to make its decisions. As an example, irrelevant features within the training data such as watermarks could influence the model's feature learning and result in the misclassification of real-world data. In this section, requirements are formulated to enhance the insight of the model's decisions by interpretability methods. Possible interpretability methods can be found in (1).

3.3.3.1 ASIL-derived Requirements

3.3.3.1.1 Deriving Test Cases for Explanations

Requirement 32: *The requirements shall be analyzed to derive test cases for interpretable model decisions.*

Based on the ASIL DU1 in Table 15 for deriving test cases for software unit testing, Requirement 32 is introduced. It states that the requirements towards the software unit shall be analyzed to derive test cases that should be explained. This ensures that some test cases are derived to support the requirements.

3.3.3.1.2 Comparing Requirements and Model Decisions

Another challenge of AI systems is to check whether the model decisions meet the requirements formulated for the software unit. The ISO 26262 defines ASIL recommendations towards the verification of the software unit to ensure this. The corresponding recommendations are presented in Table 16.

Requirement 33: *The model's decisions shall be explained to aid the comparison between the modelling of the system and the trained model.*

Requirement 34: *The model's decisions shall be explained to check if the requirements of the system are met.*

UV10 and UV14 are specified in Requirement 33 and Requirement 34 to support the interpretability of the model. As explained in Section 3.1.3.3, the term model in UV14 references the software modelling of the software unit and is different from the AI model.

Table 31 ASIL-derived interpretability requirements with ASIL recommendations from the ISO 26262 (165; 246; 247) for the AI subsystem

Method		ASIL recommendation			
		ASIL A	ASIL B	ASIL C	ASIL D
32	The requirements shall be analyzed to derive test cases for interpretable model decisions.	++	++	++	++
33	The model's decisions shall be explained to aid the comparison between the modelling of the system and the trained model.	++	++	++	++
34	The model's decisions shall be explained to check if the requirements of the system are met.	+	+	++	++

3.3.3.2 Additional Requirements

To gain further information on the model's decision-making process some additional requirements have to be defined.

3.3.3.2.1 Decision Explanations

Requirement 35: *The model's decision at the boundary values shall be explained.*

Requirement 36: *The model's decision on corner cases shall be explained.*

Requirement 35 and Requirement 36 ensure that during the testing phase of the AI system, possible errors or inconsistencies in the model's decision-making process shall be identified by explaining the model's decisions on boundary values and corner cases. Further, those explanations will give insight into the reasoning of the model's behavior on safety-critical scenarios. These two requirements are highly recommended to be implemented starting from a high risk level, since it is plausible that an AD system will encounter corner cases and boundary values.

Requirement 37: *The model's decision on failed tests shall be explained.*

Additionally, during the testing process, the model's decisions on failed test scenarios shall be explained to aid the adjustment of the model or training data. To ensure this, Requirement 37 is recommended for systems with low and medium risk and highly recommended for high to very high-risk systems.

3.3.3.2.2 Interpretable Model Architecture

Requirement 38: The least complex model architecture needed to solve the task shall be chosen.

Since the testability and interpretability of AI models decreases with the increase in complexity level, the least complex architecture possible shall be chosen. For example, for neural networks the complexity of a model architecture relates to its layer types (e.g. convolutional layer, recurrent layer, etc.), the number of parameters

and the connection between layers. As for all requirements, if it is not possible to choose the model architecture with the least complexity, a rationale shall be given.

Requirement 39: *A model architecture shall be chosen to maximize the interpretability of decisions.*

To maximize the interpretability of the model's decisions, an interpretable architecture shall be chosen. Since interpretable model architectures are not feasible for every use case, this requirement shall be chosen for suitable use cases and systems. However, as for all other requirements, if an interpretable architecture is not feasible for the system at hand, it should be justified.

Table 32 Additional interpretability requirements with risk levels for the AI subsystem

Method		*Risk*			
		Low	*Medium*	*High*	*Very high*
35	The model's decision at the boundary values shall be explained.	+	+	++	++
36	The model's decision on corner cases shall be explained.	+	+	++	++
37	The model's decision on failed tests shall be explained.	+	+	++	++
38	The least complex model architecture needed to solve the task shall be chosen.	+	+	++	++
39	A model architecture shall be chosen to maximize the interpretability of decisions.	o	+	+	++

3.3.4 Documentation & Lifecycle

In this chapter, requirements for documentation purposes related to the AI lifecycle are introduced that are relevant for the AI subsystem.

3.3.4.1 ASIL-derived Requirements

3.3.4.1.1 Software Unit Documentation

Requirement 40: *The SW unit design shall be described explicitly.*

Requirement 40 combines the recommendations from Table 13 with recommendation MC6 from Table 14. A documentation of the design of the SW units is crucial for evaluation and assessment. Especially for the AI SW units, the AI model's architecture shall be included. A clear and unambiguous description in natural language is always required. For low security applications, an additional informal description is sufficient. For (very) high security applications, at least a semi-formal notation is necessary.

Table 33 ASIL-derived documentation and lifecycle requirement with ASIL recommendations from ISO 26262 (165; 246; 247) for the AI subsystem

Method		*ASIL recommendation*			
		ASIL A	*ASIL B*	*ASIL C*	*ASIL D*
40	The SW unit design shall be described explicitly.	++	++	++	++

3.3.4.2 Additional Requirements

3.3.4.2.1 Traceability

Requirement 41: *The dataset & model shall be versioned.*

The evolution of the used datasets and AI model shall be tracked. It is important to have a version history of the dataset and the changes/additions/cleanings/etc. performed on it. Additionally, the establishing data collection process and the captured environmental contexts need to be described. This allows to see whether the dataset is representative of the intended ODD and does reflect the real environmental context. Similarly, for a data-driven model that is trained on a dataset it is important to track the used hyperparameters and which version of the datasets is used. As a result, a version history can be generated that shows the evolution of the data and the model and the associated differences for each version.

Requirement 42: *Standardized methods to record characteristics of datasets, AI models and key processes shall exist and be followed.*

For traceability, the main characteristics of datasets, models and processes have to be documented. The utilized procedures shall be consistent throughout the whole project to establish comparability among recorded data. If there exist any general or industry-specific standards, these are preferable.

Requirement 43: *The labelling process of the dataset shall be documented and tracked.*

This requirement ensures that possible biases or incorrect label assumptions can be identified and tracked.

Table 34 Traceability requirements with risk levels

Method		*Risk*			
		Low	*Medium*	*High*	*Very high*
41	The dataset & model shall be versioned.	o	+	++	++
42	Standardized methods to record characteristics of datasets, AI models and key processes shall exist and be followed.	o	+	++	++
43	The labelling process of the dataset shall be documented and tracked.	o	+	++	++

3.3.5 Monitoring

The following section introduces the requirements towards the monitoring of the AI subsystems within an AD system. The requirements can be categorized into run-time monitoring requirements that aim to monitor either the input or output of the model; and fail-safe requirements that ensure that the AI subsystems fail safely to mitigate an entire system failure.

3.3.5.1 ASIL-derived Requirements

As discussed in Chapter 3.1.3.4, the ISO 26262 defines error detection and handling methods without ASIL recommendations. Therefore, Table 17 and Table 18 define risk levels for these methods according to the risk and damage potential defined in Chapter 3.1.4.

3.3.5.1.1 Error Detection

Requirement 44: *The input shall be monitored and checked before it is given into the AI model.*

Requirement 45: *The plausibility of the AI model's output shall be checked.*

Requirement 44 and Requirement 45 refine ED1, ED2 and ED3 to ensure that the input and output of the AI model are monitored. On the input side, this will prevent the model from evaluating damaged or manipulated data and providing incorrect output. Suitable detection techniques can be found in (1). The output of the model shall be checked for plausibility errors to detect implausible behavior of the AI subsystems.

Requirement 46: *The AI model shall be monitored during the program execution.*

ED4 and ED5 are refined in Requirement 46 where a monitoring component of the AI subsystem is required. This helps to identify failures within the AI subsystem before the entire system is possibly affected. This

requirement enables the logging of events concerning the AI subsystem and provides the groundwork for Requirement 47.

Requirement 47: *Errors of the model shall be logged.*

Including and enhancing ED3, this requirement provides the possibility for evaluation of the AI subsystem in case of failures or accidents.

3.3.5.1.2 Error Handling

Requirement 48: *Damaged or manipulated inputs shall be corrected when it is safely possible.*

As recommended in EH6, Requirement 48 states that damaged data or code passed to the AI subsystem shall be corrected. As stated above, detection methods for manipulated data were introduced in the AP2 report.

Requirement 49: *Fail-safe methods shall be implemented to mitigate entire system failures.*

Requirement 49 integrates recommendations EH1, EH2 and EH3, by stating the need for fail-safe methods.

Requirement 50: *Parallel redundant AI models shall be implemented.*

Additionally, EH4 and EH5 can be addressed on the AI subsystem level by implementing a parallel and redundant AI model. Depending on the use case, the use of different sensors for the redundant models is advised.

Table 35 ASIL-derived monitoring requirements with ASIL recommendations from the ISO 26262 (165; 246; 247) for the AI subsystem

Method		*Risk level*			
		Low	*Medium*	*High*	*Very high*
44	The input shall be monitored and checked before it is given into the AI model.	+	+	++	++
45	The plausibility of the AI model's output shall be checked.	+	+	++	++
46	The AI model shall be monitored during the program execution.	o	+	++	++
47	Errors of the model shall be logged.	+	++	++	++
48	Damaged or manipulated inputs shall be corrected when it is safely possible.	o	o	+	++
49	Fail-safe methods shall be implemented to mitigate entire system failures.	+	+	++	++
50	Parallel redundant AI models shall be implemented.	o	o	+	++

3.3.6 Summary of Requirements

In Table 36 a summary of the generic requirements[8] defined in this chapter is presented. As explained in Section 3.2.6, the risk level is either based on the strictest ASIL recommendation of the involved ISO requirements or on a rationale that was provided for each additional requirement in this chapter.

[8] The aim of our efforts is to develop firm requirements that mobility systems containing AI modules have to follow. In the work presented here the requirements are a first draft and the basis for further iterations. In their current state some of them are more to consider as a best practice and an ideal state in terms of safety and security. Nevertheless, the boundaries, values and wording of the requirements will be adjusted within further projects and actual automotive applications.

Table 36 Summary of generic requirements for the AI subsystems[9]

Requirement			Risk level			
ID	*Description*	*Type*	*ASIL A/ Low*	*ASIL B/ Medium*	*ASIL C/ High*	*ASIL D/ Very high*
15	The AI model shall be implemented using mitigation strategies against robustness threats.	ASIL	+	+	++	++
16	The AI model shall be verified with formal robustness verification techniques.	ASIL	o	o	+	+
17	The robustness of the AI model shall be verified with empirical robustness estimation techniques.	ASIL	+	+	++	++
18	The AI model shall be tested against out-of-distribution data.	ASIL	++	++	++	++
19	Test cases at the boundary values of the input of the AI model shall be derived.	ASIL	+	++	++	++
20	Test cases based on corner cases of the AI model shall be derived.	ASIL	+	++	++	++
21	Test cases shall be derived through error guessing based on knowledge and experience of the system.	ASIL	+	+	+	+
22	The AI model shall be tested against possible robustness threats.	Additional	+	++	++	++
23	The source of the datasets shall be traceable.	Additional	+	+	++	++
24	The source of the dataset shall be verified.	Additional	o	+	++	++
25	The training, test and evaluation datasets shall have adequate coverage of the operational input domain.	Additional	+	+	++	++
26	The datasets shall be verified against the safety requirements.	Additional	o	+	++	++
27	The uncertainty of the datasets shall be analyzed and quantified.	Additional	o	+	++	++
28	The datasets used for training, testing and evaluation shall not contain any errors.	Additional	++	++	++	++
29	The training, test and evaluation datasets shall have sufficient size.	Additional	+	++	++	++
30	The training, test and evaluation datasets shall be independent from each other.	Additional	++	++	++	++
31	The training, test and evaluation datasets shall be prepared in an adequate way.	Additional	+	++	++	++
32	The requirements shall be analyzed to derive test cases for interpretable model decisions.	ASIL	++	++	++	++
33	The model's decisions shall be explained to aid the comparison between the modelling of the system and the trained model.	ASIL	++	++	++	++
34	The model's decisions shall be explained to check if the requirements of the system are met.	ASIL	+	+	++	++
35	The model's decision at the boundary values shall be explained.	Additional	+	+	++	++
36	The model's decision on corner cases shall be explained.	Additional	+	+	++	++

Requirement			*Risk level*			
37	The model's decision on failed tests shall be explained.	Additional	+	+	++	++
38	The least complex model architecture needed to solve the task shall be chosen.	Additional	+	+	++	++
39	A model architecture shall be chosen to maximize the interpretability of decisions.	Additional	o	+	+	++
40	The SW unit design shall be described explicitly.	ASIL	++	++	++	++
41	The dataset & model shall be versioned.	Additional	o	+	++	++
42	Standardized methods to record characteristics of datasets, AI models and key processes shall exist and be followed	Additional	o	+	++	++
43	The labelling process of the dataset shall be documented and tracked.	Additional	o	+	++	++
44	The input shall be monitored and checked before it is given into the AI model.	ASIL	+	+	++	++
45	The plausibility of the AI model's output shall be checked.	ASIL	+	+	++	++
46	The AI model shall be monitored during the program execution.	ASIL	o	+	++	++
47	Errors of the model shall be logged.	ASIL	+	++	++	++
48	Damaged or manipulated inputs shall be corrected when it is safely possible.	ASIL	o	o	+	++
49	Fail-safe methods shall be implemented to mitigate entire system failures.	ASIL	+	+	++	++
50	Parallel redundant AI models shall be implemented.	ASIL	o	o	+	++

3.4 Applicability of Requirements

After presenting all generic requirements for the entire system and AI subsystems, this chapter focusses on the applicability of the generic requirements to concrete use cases. In Table 37 we discuss the applicability of each requirement when considering use cases from the mobility domain. A summary of the most important use cases is presented in the previous report from AP2 in (1). Concretely, we consider whether there exists a need for concretization and whether a requirement can be applied for mobility use cases in a straight-forward way without major adaptions. In doing so, we use the following categories and provide details on the reason of the concrete categorization for each requirement:

- Applicability: This category determines how well a requirement is suited for different mobility use cases. It represents whether a requirement is in principle applicable without focusing on the concreteness of a requirement.
  - Simple: The requirement is well-suited for mobility use cases and can easily be applied.
  - Complex: The requirement is partly suited for mobility use cases and can be applied with slight modifications.
  - Unrealistic: The requirement is not suited for mobility use cases and can only be applied with major modifications or not at all.

- Concretization Effort: This category determines how concretely a requirement is formulated or whether there exists a need to provide further information about a requirement when used in practice for different mobility use cases.
  - None: The requirement is very concrete and can be used in practice without any concretization for mobility use cases.
  - Minor: One part of the requirement needs to be concretized before it can be used in practice for mobility use cases. In addition, the concretization of this part is rather straightforward.
  - Major: Multiple parts of the requirement need to be concretized before it can be used in practice for mobility use cases. Also included are requirements where only one part needs to be concretized but the concretization is not straightforward.

Table 37 Applicability of requirements for mobility use cases

Requirement		*Applicability*	*Concretization Effort*
ID	*Description*		
1	The environmental context shall correspond to the operational design domain (ODD).	Simple	Minor • Suitable measurement for environmental context
2	The communication, interfaces, signals, etc. between different components shall be coordinated.	Simple	None
3	The sensor setup shall be similar to the development/training setup.	Simple	Minor • Suitable definition of similarity
4	The requirements for AI subsystems shall apply to the entire system (if applicable).	Simple	Minor • Suitable applicable requirements
5	The adequate performance shall be guaranteed for a certain timeframe after initial deployment.	Simple	Major • Suitable definition of adequate • Suitable timeframe
6	The performance on key performance indicators (KPIs) shall be as high as possible	Simple	Minor • Suitable KPIs
7	The performance shall be compliant to the allowed worst-case error.	Complex • How to check high-dimensional (internal) data/output values	Major • Suitable definition of worst-case error
8	The performance shall be reproducible in the real environment for operation.	Complex • How to define realistic dummies & simulations	Major • Suitable definition of environment • Suitable coverage of complete environment
9	The feedback of the system shall be tracked while in operation.	Complex • How to log high-dimensional (internal) data/output values • How to limit storage demand/costs	Minor • Suitable tracking methods

Requirement		*Applicability*	*Concretization Effort*
ID	*Description*		
10	The performance shall be corrected when critical errors occur after deployment.	Complex • How to correct specific errors in complex systems	Major • Suitable definition of critical errors
11	The system state shall be tracked in a reproducible way while in operation.	Complex • How to log the state of complex systems • How to limit storage demand/costs	Major • Suitable tracking methods
12	The architectural design shall be described explicitly.	Simple	None
13	The quality & trustworthiness for developers shall be assessed.	Simple	Minor • Suitable assessment methods
14	The development process shall be tracked.	Simple	Minor • Suitable tracking methods
15	The AI model shall be implemented using mitigation strategies against robustness threats.	Simple	Major • Suitable definition of robustness threats • Suitable mitigation strategies • Suitable extension to new threats • Suitable coverage of robustness threats
16	The AI model shall be verified with formal robustness verification techniques.	Unrealistic • How to verify complex systems	Major • Suitable verification techniques
17	The robustness of the AI model shall be verified with empirical robustness estimation techniques.	Simple	Major • Suitable estimation techniques • Suitable coverage of complete robustness
18	The AI model shall be tested against out-of-distribution data.	Simple	Major • Suitable definition of OOD data • Suitable coverage of complete OOD data
19	Test cases at the boundary values of the input of the AI model shall be derived.	Simple	Minor • Suitable coverage of complete boundary values
20	Test cases based on corner cases of the AI model shall be derived.	Simple	Major • Suitable definition of corner cases • Suitable coverage of complete corner cases
21	Test cases shall be derived through error guessing based on knowledge and experience of the system.	Simple	Major • Suitable coverage of complete test cases
22	The AI model shall be tested against possible robustness threats.	Simple	Major • Suitable definition of robustness threats • Suitable coverage of robustness threats

Requirement		*Applicability*	*Concretization Effort*
ID	*Description*		
23	The source of the datasets shall be traceable.	Simple	Minor • Suitable traceability
24	The source of the dataset shall be verified.	Simple	Minor • Suitable verification
25	The training, test and evaluation datasets shall have adequate coverage of the operational input domain.	Simple	Major • Suitable definition of adequate • Suitable coverage of complete input domain
26	The datasets shall be verified against the safety requirements.	Simple	Major • Suitable safety requirements
27	The uncertainty of the datasets shall be analyzed and quantified.	Simple	Minor • Suitable measurement of dataset uncertainty
28	The datasets used for training, testing and evaluation shall not contain any errors.	Complex • How to verify large & high-dimensional datasets	Major • Suitable definition of errors
29	The training, test and evaluation datasets shall have sufficient size.	Simple	Minor • Suitable dataset size
30	The training, test and evaluation datasets shall be independent from each other.	Simple	Minor • Suitable dataset comparison
31	The training, test and evaluation datasets shall be prepared in an adequate way.	Simple	Major • Suitable definition of adequate
32	The requirements shall be analyzed to derive test cases for interpretable model decisions.	Simple	Minor • Suitable coverage of complete test cases
33	The model's decisions shall be explained to aid the comparison between the modelling of the system and the trained model.	Complex • How to explain high-dimensional output values	Minor • Suitable explanations
34	The model's decisions shall be explained to check if the requirements of the system are met.	Complex • How to explain high-dimensional output values	Minor • Suitable explanations
35	The model's decision at the boundary values shall be explained.	Complex • How to explain high-dimensional output values	Major • Suitable explanations • Suitable coverage of complete boundary values
36	The model's decision on corner cases shall be explained.	Complex • How to explain high-dimensional output values	Major • Suitable explanations • Suitable definition of corner cases • Suitable coverage of complete corner cases
37	The model's decision on failed tests shall be explained.	Complex • How to explain high-dimensional output values	Minor • Suitable explanations

Requirement		*Applicability*	*Concretization Effort*
ID	*Description*		
38	The least complex model architecture needed to solve the task shall be chosen.	Simple	Minor • Suitable complexity measure
39	A model architecture shall be chosen to maximize the interpretability of decisions.	Complex • How to interpret complex systems	Major • Suitable interpretability
40	The SW unit design shall be described explicitly.	Simple	None
41	The dataset & model shall be versioned.	Simple/Complex • How to version complex systems	Minor • Suitable versioning methods
42	Standardized methods to record characteristics of datasets, AI models and key processes shall exist and be followed	Simple	Minor • Suitable recording methods
43	The labelling process of the dataset shall be documented and tracked.	Simple	Minor • Suitable tracking methods
44	The input shall be monitored and checked before it is given into the AI model.	Simple	Major • Suitable input checking
45	The plausibility of the AI model's output shall be checked.	Simple	Minor • Suitable plausibility checking
46	The AI model shall be monitored during the program execution.	Complex • How to limit storage demand/costs	Minor • Suitable monitoring metrics
47	Errors of the model shall be logged.	Complex • How to log high-dimensional data • How to limit storage demand/costs	Minor • Suitable timeframe of logging • Suitable extent of logging
48	Damaged or manipulated inputs shall be corrected when it is safely possible.	Complex • How to correct high-dimensional data	Major • Suitable input correction
49	Fail-safe methods shall be implemented to mitigate entire system failures.	Complex • How to limit computational resources / storage demand / costs	Major • Suitable fail-safe methods
50	Parallel redundant AI models shall be implemented.	Complex • How to operate multiple complex systems • How to limit computational resources / storage demand / costs	Major • Suitable extent of redundancy • Suitable redundant models

3.5 Testability of Requirements

In addition to the applicability of the generic requirements defined in Chapter 3.4, the testability of these requirements has to be addressed. Table 38 shows a mapping between each requirement and a short indicator on the test procedure and its overall testability.

The testability of a requirement is categorized as:

- High: The test effort for this requirement is low and can possibly be automated.
- Medium: The tests for this requirement require domain knowledge or interpretations.
- Low/Infeasible: This requirement is not testable without infeasible or very high effort or there exist no methods to test this requirement.

However, the testability might depend on the specific use case, since some test efforts might change for different use cases, systems and environments. To give further insight into the testing procedure for these requirements, the tests can either be:

- Evidence-based: The test for this requirement requires evidence in form of documentation or audits.
- Metric-based: The test for this requirement requires the computation of statistics o metrics.

Table 38 Overview on the testability of each generic requirement

Requirement		*Testability*	*Test*	*Comments*
ID	*Description*			
1	The environmental context shall correspond to the operational design domain (ODD).	Medium	Evidence-based • Documentation on the environmental domain	
2	The communication, interfaces, signals, etc. between different components shall be coordinated.	Medium	Evidence-based • Interface implementation • Interface documentation	
3	The sensor setup shall be similar to the development/training setup.	(High) Depends on the use case	Evidence-based • Documentation on the sensor setup	• Depends on the sensor setup • For similar sensors high testability
4	The requirements for AI subsystems shall apply to the entire system (if applicable).	Depends on the individual requirement	Depends on the individual requirement	
5	The adequate performance shall be guaranteed for a certain timeframe after initial deployment.	(Medium) Depends on the use case	Metric-based • Performance metric over certain timeframe	• Depends on the timeframe and additional system components
6	The performance on key performance indicators (KPIs) shall be as high as possible	High	Metric-based • Calculation of KPIs	
7	The performance shall be compliant to the allowed worst-case error.	(High) Depends on the use case	Metric-based • Performance metric calculated for worst-case error	• Depends on the verification of the worst-case error
8	The performance shall be reproducible in the real environment for operation.	Medium	Metric-based • Assessing environments	

Requirement		Testability	Test	Comments
ID	Description			
			• Comparison of metrics within different environments	
9	The feedback of the system shall be tracked while in operation.	High	Evidence-based • System documentation/Code review	
10	The performance shall be corrected when critical errors occur after deployment.	Medium	Evidence-based/Metric-based • System documentation • Performance metrics on critical errors	
11	The system state shall be tracked in a reproducible way while in operation.	Low	Evidence-based • System documentation • What is reproducible • When to track • What does the system state entail	
12	The architectural design shall be described explicitly.	High	Evidence-based • Documentation of the architectural design	
13	The quality & trustworthiness for developers shall be assessed.	High	Evidence-based • Site audit and documentation	
14	The development process shall be tracked.	High	Evidence-based • Documentation of the development process	
15	The AI model shall be implemented using mitigation strategies against robustness threats.	Medium	Evidence-based • Code review/Documentation • Assessing suitable strategies	
16	The AI model shall be verified with formal robustness verification techniques.	Low	Metric-based • Verification metrics	• Infeasible for complex models (>10^5 neurons and 6 layers); For more information, see Chapter 2
17	The robustness of the AI model shall be verified with empirical robustness estimation techniques.	Medium	Metric-based • Robustness metrics • Assessing suitable metrics • Assessing thresholds	
18	The AI model shall be tested against out-of-distribution data.	High	Metric-based • Performance metric on out-of-distribution data	
19	Test cases at the boundary values of the input of the AI model shall be derived.	High	Evidence-based • Test documentation	

Requirement		*Testability*	*Test*	*Comments*
ID	*Description*			
20	Test cases based on corner cases of the AI model shall be derived.	(Medium) Depends on the use case	Evidence-based • Test documentation	• Assessing the quality of tests • Defining corner cases • Coverage of corner cases • Depends on system and environment
21	Test cases shall be derived through error guessing based on knowledge and experience of the system.	Medium	Evidence-based • Test documentation • Assessing the quality of tests	
22	The AI model shall be tested against possible robustness threats.	(Medium) Depends on the use case	Evidence-based • Test documentation	• Assessing the quality of tests • Defining possible threats • Depends on system and environment
23	The source of the datasets shall be traceable.	Medium	Evidence-based • Dataset documentation	
24	The source of the dataset shall be verified.	Medium	Evidence-based • Dataset documentation	
25	The training, test and evaluation datasets shall have adequate coverage of *the operational input domain.*	Medium	Evidence-based • Dataset documentation	
26	The datasets shall be verified against the safety requirements.	Medium	Evidence-based • Dataset documentation	
27	The uncertainty of the datasets shall be analyzed and quantified.	Medium	Metric-based • Uncertainty metric • How much uncertainty is adequate	
28	The datasets used for training, testing and evaluation shall not contain any errors.	Medium	Evidence-based • Dataset documentation	
29	The training, test and evaluation datasets shall have sufficient size.	Medium	Evidence-based • Dataset documentation	
30	The training, test and evaluation datasets shall be independent from each other.	High	Evidence-based • Dataset documentation	
31	The training, test and evaluation datasets shall be prepared in an adequate way.	Medium	Evidence-based • Dataset documentation • Software modelling documentation	
32	The requirements shall be analyzed to derive test cases	Medium	Evidence-based • Test documentation	

Requirement		Testability	Test	Comments
ID	Description			
	for interpretable model decisions.			
33	The model's decisions shall be explained to aid the comparison between the modelling of the system and the trained model..	Medium	Metric-based/Evidence-based • Explanation metric • Software modelling documentation	
34	The model's decisions shall be explained to check if the requirements of the system are met.	Medium	Metric-based/Evidence-based • Explanation metric • Requirement documentation	
35	The model's decision at the boundary values shall be explained.	Medium	Metric-based/Evidence-based • Explanation metric • Input space (boundary value) documentation	
36	The model's decision on corner cases shall be explained.	(Medium) Depends on the use case	Metric-based/Evidence-based • Explanation metric • Corner case documentation	• Assessing the quality of tests • Defining corner cases • Coverage of corner cases • Depends on system and environment
37	The model's decision on failed tests shall be explained.	Medium	Metric-based • Explanation metric	
38	The least complex model architecture needed to solve the task shall be chosen.	Medium	Evidence-based • Model documentation	• Definition of "least complex" architecture • Assessment of rationale
39	A model architecture shall be chosen to maximize the interpretability of decisions.	Medium	Evidence-based • Model documentation	
40	The SW unit design shall be described explicitly.	High	Evidence-based • Documentation of the SW unit design	• Depending on the degree of testing white box access of model is needed
41	The dataset & model shall be versioned.	High	Evidence-based • Versioning documentation	
42	Standardized methods to record characteristics of datasets, AI models and key processes shall exist and be followed	High	Evidence-based • Entire documentation	
43	The labelling process of the dataset shall be documented and tracked.	High	Evidence-based • Dataset documentation	

Requirement		*Testability*	*Test*	*Comments*
ID	*Description*			
44	The input shall be monitored and checked before it is given into the AI model.	Medium	Evidence-based/Metric-based • Software documentation • Input checking metric	
45	The plausibility of the AI model's output shall be checked.	Medium	Evidence-based • Code review/Documentation • Suitable plausibility checking	
46	The AI model shall be monitored during the program execution.	Medium	Evidence-based • Suitable monitoring metrics	
47	Errors of the model shall be logged.	High	Evidence-based • System documentation • Logging • Code review	
48	Damaged or manipulated inputs shall be corrected when it is safely possible.	Medium	Evidence-based/Metric-based • Detection metric • System documentation • Code review	
49	Fail-safe methods shall be implemented to mitigate entire system failures.	Medium	Evidence-based • Documentation of fail-safe methods	
50	Parallel redundant AI models shall be implemented.	Medium	Evidence-based/Metric-based • System Documentation • Similarity metric of models	

4 Use Case Comparison for Audit Criteria Development (AP4)

This chapter is the final report of work package four "Vergleich relevanter Use-Cases im Hinblick auf Eignung für Prüfkriterienentwicklung" of the project "Vorbereitung der Erprobung und Weiterentwicklung von Anforderungen an KI-Systeme anhand praktischer Use-Cases im Bereich Mobilität (AIMobilityAuditPrep)". Hence, it contains the results of the listing and categorization of potential use cases in the mobility area and specifically use cases that are part of ADAS or AD systems. These use cases are categorized and the most interesting and relevant ones are analyzed regarding their suitability for the development of audit criteria and the practical testing of the generic requirements given in Chapter 3. Based on this the recommendation of use cases for the work packages five and seven is performed.

As a general disclaimer, it is important to point out that the sheer number of potential use cases in the area of AD/ADAS and the complexity of individual use cases makes an unambiguous categorization and comparison close to impossible. Depending on the viewpoint different aspects can be assessed as more important leading to a varying categorization. This also strongly depends on the details that one considers when categorizing the use cases. In this document we try to give a general overview of relevant use cases for AD/ADAS and derive a common categorization as best as possible. This does not claim to be the ideal and all-inclusive solution and there are other options to perform the categorization. Nevertheless, a common categorization is needed to be able to compare various use cases more easily. Only this enables the proper selection of the use cases for the practical tests in work packages five and seven.

4.1 Category Overview

In this chapter we present the categories that are used to group the use cases in Chapter 4.2. We discuss different categories that are used to differentiate the use cases and start by initially listing the categories that are motivated in the description of services. Then, we present additional categories that cover further important aspects of different mobility use cases. For each category we discuss the possible parameters and give practical examples for the meaning of each parameter level. Each concrete parameter that is later used to categorize the use cases is highlighted in **bold face**.

4.1.1 Required Categories from Description of Services

4.1.1.1 Safety Relevance

The first category that is mentioned is the impact and relevance of the use case for the safety of the entire mobility system and external traffic participants. Here, possible parameter values are:

- **High**: The use case impacts the control of a vehicle and is executed at moderate/high speed levels where a failure can cause significant damages.
- **Medium**: The use case impacts the control of a vehicle and is executed at moderate/high speed levels where a failure can cause limited damages. Also included are use cases where a failure can cause significant damages, but the use case only impacts the control of a vehicle as part of an assistance system and not independently.
- **Low**: The use case impacts the control of a vehicle and is only executed at low speeds where a failure can cause limited damages. Also included are use cases that are executed at moderate/high speed levels where a failure can cause limited damages, but the use case only impacts the control of a vehicle as part of an assistance system and not independently.
- **None**: The use case does not impact the driving performance and the control of a vehicle in any way and is a pure comfort system.

4.1.1.2 Input Data

This category covers the respective input data for each use case. Here, possible parameter values are:

- Sensor-based: Data that is generated by different sensors, including both exteroceptive and interoceptive sensors. During the use case discussion in Chapter 4.2 we assume an exteroceptive sensor if not described otherwise. Relevant sensors are:
  - Camera
  - LiDAR
  - RADAR
  - Acoustic
  - Ultrasonic
  - GPS
- **HD Map:** High-definition map that contains a more detailed representation of map and road elements.
- **Internal**: Data/Information that is generated internally in the entire system by previous components or use cases and not by sensors.
- **Fusion**: Data that is combined from different sensors or components.
- **None**: The use case needs no data as input.

4.1.1.3 Modular Components

In this category the modular components of an AD system which are involved in each use case are covered. The modular components are described in detail in the previous Chapter 2. In this category, possible parameter values are:

- **Perception**
- **Localization**
- **Prediction**
- **Planning**
- **Control**
- **None**: No traditional modular component is used.

4.1.1.4 AI Usage

The next category differentiates the AI and ML techniques that are used in each use case. Generally, in the remainder of this document we only cover use cases where an AI involvement is in principle possible and logical. In this category, possible parameter values are:

- Current: This parameter lists the technique that is currently used by the majority of state-of-the-art (SOTA) systems for each use case.
  - **Algos**: Traditional algorithm/methods are used that are not based on ML.
  - Tree
  - SVM
  - DNN
- ML-Future: In case currently traditional algorithms are used this parameter describes the expected arrival time of ML-based techniques.

- **Near**: ML-based techniques are expected to be the SOTA in the next few (<3) years.
- **Far**: ML-based techniques are expected to be the SOTA in reasonable (3-10 years) time.
- **Unlikely**: ML-based techniques are not expected to be the SOTA in reasonable (<10 years) time.

4.1.1.5 Auditability

This category covers the test effort required to derive the residual risk of an AI system implementing the use case. Here, possible parameter values are:

- **Complex**: The damage potential of hazards is medium or high and despite a high test effort, the residual risk of the system is complex to derive.
- **Medium**: The residual risk can be measured with a reasonable test effort.
- **Simple**: The damage potential of hazards is low or the residual risk can be covered with low test effort.

4.1.2 Additional Categories from AP2 & AP3

4.1.2.1 Complexity

The first additional category differentiates the complexity of each use case. Here, different factors are relevant to determine the overall complexity. On the one hand, one factor is the amount of different modular components that is already covered in the requirement in Chapter 4.1.1.3. Additionally, other factors are considered like the availability of relevant literature, whether the use case is completely new or already used for some years and whether the use case is well understood or still heavily researched without commonly used techniques. In this category, possible parameter values are:

- **High**: The use case is new, under research and contains multiple modular components.
- **Medium**: The use case is either new but only contains few modular components or the use case is well understood but contains multiple modular components.
- **Low**: The use case is well understood and contains only a single modular component.

4.1.2.2 Widespread Distribution

This category covers the distribution of the usage of the use case. Here, possible parameter values are:

- **High**: The use case is present in most new high-priced vehicles.
- **Medium**: The use case is present in some new high-priced vehicles.
- **Low**: The use case is not present in most new high-priced vehicles.

4.1.2.3 Attack Applicability

In this category the applicability of potential attacks on the AI component is covered. Principled attack methods are described in detail in Chapter 2. Again, multiple factors are important to consider when assessing the general applicability of an attack in reality. The most important factors are the scalability of an attack, the availability of literature or demonstrations of an attack, the required access interface of an adversary and whether concrete implementations are publicly available. In this category, possible parameter values are:

- **Unrealistic**: The attack cannot be executed without internal access of the adversary.
- **Complex**: The attack can be executed without internal access of the adversary but cannot be scaled to impact multiple vehicles or there exists no public knowledge of the attack.
- **Medium**: The attack can be executed without internal access of the adversary, can be partly scaled to impact multiple vehicles and there exists at least some public knowledge of the attack.

- **Simple**: The attack can be executed without internal access of the adversary, can be well scaled to impact multiple vehicles and there exists extensive public knowledge of the attack.

4.1.2.4 Perception Components

This category focusses specifically on the involved perception components for each use case. Most of the perception components are described in detail in Chapter 2. In this category, possible parameter values are:

- **Detection**
- **Segmentation**
- **Depth**
- **Flow**
- **Clustering:** Clustering is used to group 3D point cloud data.
- **Occupancy Grid Map:** Generate grid map of environment indicating the location of obstacles.
- **Classification**
- **Regression**
- **None**: No perception component is involved.

4.2 Use Case Overview

In this chapter we present an initial overview of use cases in the area of mobility that currently use AI systems or where AI systems are potentially used in the future. A list of potential use cases is developed and the previously introduced categories are used to compare the use cases. We focus on listing rather generic use cases to be able to cover all important aspects of different use cases that arise in the context of AD and ADAS. Later in Chapter 4.3, a more detailed discussion is performed for the most interesting and relevant use cases. First, we discuss generic use cases that are relevant for both AD and ADAS before we focus on use cases that are specific for each field. In both cases there exist different publications that cover use cases from different viewpoints which we discuss in Chapter 2. Most importantly ((184), (185), (186)) cover use cases for AD and ((187), (188)) cover use cases for ADAS. We try to combine these different viewpoints and first extract generic use cases that are relevant for both domains.

4.2.1 Generic Use Cases

Table 39 gives an overview of the considered generic use cases and the grouping using the categories from Chapter 4.1. It serves as an overview to compare the different use cases on a quick glance. In the remainder of this chapter the concrete categorization is further discussed in detail for each use case. Additionally, we argue whether the use case should be selected for the more fine-grained analysis in Chapter 4.3 that forms the basis for the final selection of use cases in Chapter 4.4.

Table 39 Overview of general use cases

ID	1	2	3	4	5	6	7
Use Case	Emergency Braking	Collision Avoidance	Lane Keeping	Lane Changing	Adaptive Cruise Control	Rain/Grip Level	Virtual Sensor Replacements
Safety Relevance	High	High	High	High	High	Medium	Medium
Input Data	Camera, LiDAR, RADAR, Fusion	Camera, LiDAR, RADAR, Fusion	Camera, Fusion	Camera, LiDAR, RADAR, Fusion	Camera, LiDAR, RADAR, Fusion	Camera, Internal	Internal, Fusion
Modular Components	Perception, Control	Perception, Prediction, Control	Perception, Localization, Control	Perception, Localization, Planning, Control	Perception, Control	Perception	Perception
AI Usage	Current: Algos ML-Future: Near	Current: Algos, DNN ML-Future: Near	Current: Algos ML-Future: Near	Current: Algos, DNN ML-Future: Near	Current: Algos ML-Future: Near	Current: DNN	Current: DNN
Auditability	Complex	Complex	Complex	Complex	Complex	Complex	Simple - Complex
Complexity	Medium	High	Medium	High	Medium	Medium	Low
Widespread Distribution	High	Low	Medium	Low	High	Low	Low
Attack Applicability	Camera: Medium – Point Cloud: Complex	Camera: Medium – Point Cloud: Complex	Simple	Camera: Simple – Point Cloud: Complex	Complex	Complex	Unrealistic
Perception Components	Detection, Depth, Classification, Clustering	Detection, Depth, Classification, Clustering	Segmentation	Detection, Depth, Segmentation, Clustering	Detection, Depth	Classification, Regression	Classification, Regression

ID	8	9	10
Use Case	Driver/Passenger Interaction	Global Navigation /Path Planning	Automated Parking
Safety Relevance	None	None	Low
Input Data	Camera, Acoustic	HD Map, GPS	Camera, LiDAR, RADAR, Ultrasonic, Fusion
Modular Components	Perception	Localization, Prediction	Perception, Planning, Control
AI Usage	Current: DNN	Current: Algos, DNN	Current: Algos ML-Future: Near
Auditability	Simple	Simple	Complex
Complexity	Low	Low	Medium
Widespread Distribution	High	High	Medium
Attack Applicability	Camera: Unrealistic – Acoustic: Simple	Unrealistic	Camera: Medium – Point Cloud: Complex
Perception Components	Classification	Classification, Regression	Detection, Segmentation, Depth

4.2.1.1 Emergency Braking (1)

The emergency braking use case contains all functionality that reacts to potential obstacles in the driving path of a vehicle by initiating a deceleration motion. It can be executed at different speed levels and is independent of the road type. Thus, it has a high relevance for the safety. Depending on the manufacturer different input data types are possible. In some cases, only camera-based data is used, while others also include point cloud data based on LiDAR or RADAR to have a more robust depth estimation for potential obstacles. Additionally, the fusion of the described sensor data can be used. Since this use cases exploits sensor data, the perception component is involved. In addition, the use case also impacts the control component of a vehicle by automatically controlling the actuators to initiate a deceleration motion. For systems that are used in current ADAS on public roads, typically traditional algorithms are used for regulatory reasons. Nevertheless, it is already possible to use ML-based methods, which also promise a better performance and therefore will most-likely be used in the near future. The auditability is classified as complex since the use case has a high safety relevance and perception-based ML systems are hard to verify considering the almost infinite input space. Emergency braking is already in use for some years (also on vehicles that operate on public roads) and rather well understood. Therefore, the complexity can be categorized as medium while the widespread distribution is rather high. Also, it is challenging to scale the applicability of attacks. For camera-based data one way to attack the emergency braking functionality is to attach physical perturbations on moving vehicles/obstacles. This does not scale to multiple vehicles at all and thus the only alternative is to create fake obstacles, e.g. using a beamer. For point cloud data it is far more difficult to pretend that an obstacle exists and close to no public knowledge exists here. Finally, the perception techniques of detection and classification are used to detect obstacles in image data, whereas depth and clustering techniques are used to detect obstacles in point cloud data. In Chapter 4.4 we do not consider this use case, because the collision avoidance use case discussed in Chapter 4.2.1.2 can be considered as including the emergency braking use case.

4.2.1.2 Collision Avoidance (2)

This use case is a generalization of the emergency braking use case from Chapter 4.2.1.1. In addition to deceleration, collision avoidance functionalities also include acceleration and steering motions. Also, it is possible to avoid accidents resulting from side or rear-end collisions, instead of only focusing on collisions on the driving path of a vehicle. Thus, most parameters in Table 39 are like the categorization for emergency braking. For the modular components this use case additionally impacts the prediction component. In contrast to emergency braking, which performs an immediate and heavy braking maneuver, general collision avoidance has more degrees of freedom and can avoid accidents that are likely to happen in the future. Here, DNNs are used to predict the future behavior and trajectory of other participants which might need to be avoided. Therefore, this use case has a higher complexity and is currently not widely used yet. Again, the auditability is classified as complex for the same reasons as in Chapter4.2.1.1. This use case is further analyzed in Chapter 4.4 because it is most generic and includes different specific use cases to avoid collisions in certain situations or by executing certain maneuvers.

4.2.1.3 Lane Keeping (3)

The next use case consists of lane keeping, which includes all functionalities that keep a vehicle in the current driving lane. Here, mainly steering motions are performed to tackle the given task. In general, there is no need to perform a detection of other traffic participants and thus no point cloud data is required as input. However, next to perception localization plays a role to determine the current position of the vehicle on a map to understand the lane structure if markings are missing and to accurately estimate the position of the vehicle in the lane. Systems with lane keeping functionality are already deployed in the current generation of ADASs, meaning currently traditional algorithms are used. This also means that the complexity is only medium while the widespread distribution is partly given. The auditability is again classified as complex for the same reasons discussed previously. In contrast to the previously discussed use cases, the attack applicability is rather simple for the lane keeping use case. An adversary can prepare the road surface using markers, which potentially impacts all vehicles that drive by. Since the detection of other traffic participants is not required, only the segmentation task is relevant for the perception of the surrounding scene. For the detailed analysis this use case is included because it covers very relevant tasks for driving functionalities which are also currently deployed.

4.2.1.4 Lane Changing (4)

Like the discussion on use case 1 and use case 2, the lane changing use case has a more complex functionality than the previous lane keeping use case. This use case includes all functionalities that lead to a change in the driving lane of the vehicle. For example, this includes overtaking maneuvers on two lane roads, lane selection on intersections or entering highways. To perform such maneuvers steering motions are again most important. Nevertheless, also acceleration or deceleration motions are required to be able to merge in between two vehicles and adapt to the driving speed of the new lane. Hence, this use case again requires detecting other dynamic traffic participants and typically sensors for point cloud data are used. Also, the planning component is heavily impacted because lane changing is a complex maneuver which requires the trajectory planning of the vehicle for multiple steps in the future. Therefore, this use case is very complex and is not used widely yet. Similarly, the auditability is classified as complex for the reasons discussed in Chapter 4.2.1.1. Like use case 3, the applicability of the attack is simple when only camera perception is used because static markings can be placed on the road. Attacking point cloud sensors for traffic participants detection is more complex as discussed in Chapter 4.2.1.1. In Chapter 4.4 this use case is analyzed in detail because it includes very important driving functionalities with a high safety relevance especially when considering higher levels of automated driving.

4.2.1.5 Adaptive Cruise Control (5)

The adaptive cruise control use case includes functionalities that manage the distance to a vehicle driving in front of the ego vehicle. Here, deceleration and acceleration motions are important to control the distance to the front vehicle adaptively based on its driving maneuvers and speed. Hence, for this use case the detection of the front vehicle is important, which can be based on camera or point cloud data. Based on the detection and the estimated distance to the front vehicle, the speed of the ego vehicle is directly controlled without a relevant involvement of prediction or planning components. Nevertheless, the auditability of an AI system is again classified as complex. The functionalities for this use case are included in current ADASs, meaning currently traditional algorithms with a medium complexity are used. Attacks are very difficult to carry out and scale to impact multiple vehicles. It is possible to hide the front vehicle for perception by adding physical perturbation (e.g., stickers or 3d prints) on the vehicle, but an adversary gains little from this. Like use case 3 the adaptive cruise control use case is further analyzed in Chapter 4.3 because in combination this enables to operate a vehicle autonomously in standard road scenarios.

4.2.1.6 Rain/Grip Level (6)

The next use case covers functionalities that measure the amount of rain on a road or the grip level in general. This can include the detection of ice or snow and an assessment of the general road surface. The goal is to provide an indication of the available grip level of the upcoming road which is integrated by the planning component in potential motions or used by the driver of a vehicle. Thus, the safety relevance is only medium because there is no direct control of driving functionalities. For grip level assessment mainly camera data is used for the perception and DNNs are mainly used to evaluate and interpret the data. The auditability is classified as complex despite the medium safety relevance of the use case, because perception-based ML systems are hard to verify considering the almost infinite input space. In principle, the idea behind the use case is rather simple, however research only started recently and the use case is relatively young. Hence, the widespread distribution is pretty low and the overall complexity is medium because grip level prediction is not extensively researched. Also, performing attacks is rather complex since an adversary would need to perturb the road surface continuously for a rather long distance. This is also the main reason why this use case is not further analyzed in Chapter 4.3. Another reason is that this use case is more niche and not necessarily required for automated driving.

4.2.1.7 Virtual Sensor Replacements (7)

This use case includes all functionalities where physical sensors are replaced by intelligent algorithms that provide the same (or even enhanced) information but use already existing sensor data to derive this information. Hence, the number of required sensors in a vehicle is reduced by using intelligent processing software. Like the discussion on the previous use case, there is no direct control of driving functionalities leading to a reduced safety relevance in case sensors are replaced that are mainly used for convenience functionalities. As mentioned, various internal data sources are used or fused with other external sensors to gather the required data which allows generating the same data that a traditional sensor would capture. Here, DNNs are most powerful to learn the functionality of a traditional sensor and replicate it from the available data. Depending on the level of criticality of the replaced sensor, the auditability of the virtual AI-based sensor can reach from simple to complex. The complexity is low compared to the other use cases, but the use of virtual sensors is also not widely spread yet. Also, it is unrealistic to perform any attacks because the adversary does not have a single attack point that only serves to attack sensor replacements. In addition, such sensor replacements are used for non-critical sensors for convenience functions where an attack does have very little or no damage potential. This use case is also not further analyzed for the same reasons as for use case 6. In future, it is likely that virtual sensor replacements are also used to replace sensors that are relevant for driving functionalities. However, such usages are not the focus for the categorization in Table 39. There, we mainly consider virtual replacements of sensors which are not directly involved in the perception components for AD/ADAS, i.e. Camera, RADAR or LiDAR sensors are not replaced. When these sensors are also replaced, the

complexity and safety relevance of this use case is significantly increased and new attack vectors open for adversaries.

4.2.1.8 Driver/Passenger Interaction (8)

In contrast to all previous use cases, this use case focusses on pure comfort functionalities inside a vehicle. Concretely, all functionalities are included that serve to interact with the driver or other passengers of a vehicle. Hence, different human-machine interfaces are relevant where the most important one is having a voice assistant for different comfort commands, like music/climate control or wishes to adjust the navigation route. Similarly, other interfaces are possible but typically AI is most important for voice interfaces. Here, the perception is based on the data from interoceptive acoustic sensors, which is processed by a DNN to achieve the performance of current voice assistants. The auditability of an AI-based system is ranked as simple because the use case has no direct impact on the safety of the driving functionality. Similarly, the overall complexity is low compared to use cases that include driving functionalities, but voice assistants are widely used in current vehicle generations. The attack applicability depends on the sensor that is used for perception. In case of acoustic sensors, a physical attack interface is to play perturbed sounds (songs, commercials, etc.) over radio (or Spotify, YouTube etc.). This attack scales potentially to a large number of vehicles and thus has a simple applicability. In case other sensor data is used the attack is instead unrealistic because an adversary would have to apply the physical perturbations at the inside of the vehicle. Since this use case has no safety relevance and does not directly impact any driving functionalities it is not included in the analysis in Chapter 4.3.

4.2.1.9 Global Navigation/Path Planning (9)

The next use case includes AI related functionalities for global navigation and path planning. Here, the global route/path of a vehicle is planned, which consists of the rough path from the starting location to the target location. The most famous example are map services (Google Maps, etc.), which plan a route for the vehicle. This route is updated online during driving depending on the current occupancy of roads or the probability of traffic jams. Here, AI-based algorithms play a role to predict potential arrival times or likely traffic jams. In general, global path planning is not based on perception but only utilizes the localization and prediction components. It is already used for quite some years and thus is rather well understood. Therefore, global path planning is widely distributed and available for everyone via apps for smartphones. The auditability is ranked as simple similarly to Chapter 4.2.1.8. Also, the complexity is low in comparison to driving functionalities, but an attack is unrealistic because there exists no physical attack interface an adversary can exploit. In Chapter 4.3 this use case is further analyzed due to the extremely widespread distribution and relevance for path planning of automated driving.

4.2.1.10 Automated Parking (10)

Finally, the last generic use case is automated parking, which covers functionalities for automatically finding a parking spot in a larger parking lot or performing automatic parking at the roadside. The safety relevance is low because the functionalities are only executed at low speeds, but otherwise use a similar sensor suite and detection concepts as functionalities for collision avoidance from use case 2. In addition, the planning component is involved to plan the concrete maneuver that is required to navigate the vehicle into a tight parking spot. Despite operating with low speed, a malfunctioning could harm pedestrians. Therefore, the auditability is classified as complex since perception-based ML systems are hard to verify considering the almost infinite input space. This use case is already available for some years in high-end vehicles and is continuously further included in current ADASs. Hence, a certain degree of widespread distribution is present and the complexity is in the middle of the range. Finally, the attack applicability is comparable to the discussion in Chapter 4.2.1.1. This use case is not included in the detailed analyses because it is a combination of different individual use cases but only executed at lower speeds for a concrete task.

4.2.2 ADAS specific Use Cases

Similar to the previous Chapter 4.2.1, this chapter provides a general list of use cases that are specifically relevant for ADAS which are not yet covered in the use cases presented so far. Like Table 39 we show an overview in Table 40 and discuss the categorization of each use case in the following.

Table 40 Overview of ADAS specific use cases

ID	*Use Case*	*Safety Relevance*	*Input Data*	*Modular Components*	*AI Usage*	*Auditability*	*Complexity*	*Widespread Distribution*	*Attack Applicability*	*Perception Components*
11	Blind Spot Monitoring	Medium	Camera, RADAR, Fusion	Perception	Current: Algos, DNN ML-Future: Near	Complex	Medium	Medium	Complex	Detection, Classification
12	Traffic Sign Assistant	Low	Camera, HD Map, GPS	Perception, Localization	Current: Algos, DNN ML-Future: Near	Simple (Complex)	Low	High	Simple	Detection, Classification
13	Wrong-way Warning	Medium	Camera	Perception, Localization	Current: Algos ML-Future: Near	Complex	Medium	Medium	Simple	Segmentation
14	Driver Monitoring	Medium	Camera	Perception	Current: DNN	Complex	Low	Low	Unrealistic	Detection, Classification

4.2.2.1 Blind Spot Monitoring (11)

The blind spot monitoring use case includes functionalities that provide an aid to the driver by monitoring blind spots of the field of view of the driver. For example, this can include areas that are not visible in the mirrors or which are blocked by the vehicle chassis. In these blind spots the detection of obstacles and other traffic participants is performed which can be based on camera or point cloud data. Since this use case is a pure assistance system, only the perception component is relevant. Overall, there are already systems operating in public which offer blind spot monitoring leading to a partial distribution. The auditability is classified as complex despite a medium safety relevance of the use case for the same reasons discussed in Chapter 4.2.1.6. Also, the complexity is medium because the task is already researched for some years and is strictly simpler than the more complex use case 2 covering collision avoidance. An adversary trying to attack blind spot monitoring has a very complex task. One reason is that it is difficult to scale practical perturbations to impact multiple vehicles because perturbations need to be applied on individual vehicles. For the analysis

in Chapter 4.3 this use case is not covered because use case 2 can be seen as a generalization that includes similar but more complex functionalities.

4.2.2.2 Traffic Sign Assistant (12)

This use case includes all functionalities that show currently relevant traffic signs to the driver. However, this purely acts as an assistance feature and for example does not adapt the speed of a vehicle to the detected speed limit automatically. Therefore, the safety relevance is rather low because only the perception or localization components are involved. For the perception camera sensors are useful if no map data is available or if there is a short-term temporary change in the traffic signs. Since the safety relevance is categorized as low, the auditability activities can be quite simple. In case of a traffic sign assistant that automatically adapts the vehicle speed to the detected speed limit, the auditability might rise with the safety relevance to a complex level. This use case is very specific and includes clearly defined tasks leading to a rather low complexity. Also, there are already functionalities deployed for some years leading to a widespread distribution. The attack applicability is simple since physical perturbations can be applied to existing signs or phantom/spoofing signs can be placed on the roadside. In Chapter 4.3 this use case is further analyzed, because the concrete functionality is not yet represented in the current selection of use cases and provides a rather simple use case which can become more important in the future for general-purpose autonomous driving.

4.2.2.3 Wrong-Way Warning (13)

Like use case 3 the wrong-way warning use case includes functionalities regarding the driving lane of a vehicle. In contrast, only assistance systems are considered that provide a warning to the driver if the vehicle enters the opposite lane or enters a one-way street in the wrong driving direction. Hence, most parameters are similar to Chapter 4.2.1.3, but the complexity and safety relevance is lower because the control component is not directly involved. We do not analyze this use case further because it can be seen as a specific application of use case 3.

4.2.2.4 Driver Monitoring (14)

The last ADAS specific use case is based on the monitoring of the driver. For example, the goal is to detect the drowsiness or distraction of a driver and provide a warning to the driver. Hence, this use case is again based on camera perception facing the inside of a vehicle and does not involve other components than perception. Since driver monitoring is again a pure assistance feature without influence on the control of the vehicle, currently DNNs are used for the highest performance of driver attention prediction. Nevertheless, we want to point out the importance of this use case, because the earliest as possible detection of a driver inattention leads to the highest available prewarning time. This can reduce the number and criticality of accidents and is also important when the driver must monitor assistance functionalities and be able to intervene rapidly. The auditability is classified as complex similarly to the discussion in Chapter 4.2.2.1. The complexity is comparably simple because the use case does not involve multiple traffic participants and only considers the driver. Also, the spread of the use case is rather low since it only came up in recent years. At the same time, it is unrealistic to apply an attack because an adversary would need to apply perturbations at the inside of the vehicle. In the analysis in Chapter 4.3 this use case is included to also analyze a use case which focusses on the perception of the inside of a vehicle.

4.2.3 AD specific Use Cases

Finally, this chapter covers use case for AD that are not included yet. Again, Table 41 provides an overview of the use cases which are discussed in detail in the remainder of this chapter.

Table 41 Overview of AD specific use cases

ID	15	16	17	18	19	20
Use Case	A Priori Map-based Localization	Road Users Detection	Road Elements Detection	Free Space Detection	Behavior Prediction	Local Path Planning
Safety Relevance	High	High	High	High	High	High
Input Data	LiDAR, Camera, RADAR, Fusion	LiDAR, Camera, Acoustic, RADAR, Fusion	LiDAR, Camera, RADAR, Fusion	LiDAR, Camera, RADAR, Fusion	Internal or sensor based, HD Map	Internal, HD Map
Modular Components	Perception, Localization	Perception	Perception, Localization	Perception	Perception, Prediction	Planning, Control
AI Usage	Current: Algos, DNN ML-Future: Near	Current: Algos, DNN ML-Future: Near	Current: Algos, DNN ML-Future: Near	Current: Algos, DNN ML-Future: Near	Current: Algos, DNN ML-Future: Near	Current: Algos ML-Future: Far
Auditability	Complex	Complex	Complex	Complex	Complex	Complex
Complexity	Medium	Medium	Medium	Medium	High	Medium
Widespread Distribution	Low	Medium	Medium	Medium	Low	Low
Attack Applicability	Complex	Camera, Acoustic: Simple- Point Cloud: Complex	Camera: Simple- Point Cloud: Complex	Camera: Simple - Point Cloud: Complex	Unrealistic	Unrealistic
Perception Components	Detection, Classification, Segmentation	Detection, Classification, Segmentation, Clustering	Detection, Classification, Segmenta tion, Clustering	Detection, Segm- entation, Occupancy Grid Map, Detection	Detection, Classification	None

4.2.3.1 A Priori Map-based Localization (15)

Localization aims to determine the current position of the ego-vehicle as it navigates through the scene. In autonomous driving, this is a very important component especially for lane keeping mentioned in Chapter 4.2.1.3 and for global navigation mentioned in Chapter 4.2.1.9. Hence, the safety relevance of localization is categorized as high.

There are several approaches for localization (184):

- GPS-IMU fusion based localization uses data from GPS and IMU with the dead-reckoning principle to estimate the current position of the vehicle. This method alone does not give the accuracy required for autonomous driving because of the inaccuracies in GPS data and the accumulation of errors in dead reckoning.
- Another approach is SLAM which performs the localization of the vehicle and the map creation of its environment at the same time. This method is commonly used for indoor robot navigation. Since this method does mapping and localizing at the same time, it requires high computational costs and hence is not efficient enough for using it in the outdoor world.
- Another approach is doing localization based on a pre-built map. This is called "A Priori Map Based Localization", which is the most commonly used approach for AD localization. Hence, it is considered in the following. This again contains two sub-approaches, namely landmark search and point cloud matching.
    - Landmark search commonly uses camera sensors to detect landmarks (e.g. poles, signs, road markers, etc.) and matches them with a digital landmarks map to localize the vehicle. Hence, the perception component is involved in this approach.
    - In point cloud matching 3D sensors such as LiDAR are used. Here, the point cloud at a given time is pre-processed and matched with an existing 3D map to get the position and orientation of the ego-vehicle at that time.

For global navigation, GPS-IMU sensors together with traditional algorithms are used, which has high widespread distribution. For accurate localization using camera or LiDAR sensors DNNs are used, which has a low widespread distribution. Nevertheless, the auditability is classified as complex for the reason discussed in Chapter 4.2.1.1. The complexity is considered medium as it is a well-understood use case and contains relatively few modular components. However, the attack applicability is complex as attacking this component requires to modify the environment extensively. Lastly, the perception components detection, classification and segmentation are used for finding landmarks. We select this use case for the further analysis in Chapter 4.3 because it is not yet represented in the previously selected use cases and is very relevant for AD/ADAS.

4.2.3.2 Road Users Detection (16)

Road users are the dynamic traffic participants on the scene such as pedestrians, cars, cyclists etc. They can be detected either from camera sensors or 3D sensors or using a fusion approach. In addition, acoustic sensors can be used to detect emergency vehicles. This use case has a high safety relevance because failure to detect road users can lead to high damages. The perception component is involved as it detects the road users around the vehicle. Here, road user detection based on cameras uses DNNs, while road user detection based on 3D sensors uses traditional algorithms like clustering. However, recently DNNs are also being used for 3D sensor data. Like in Chapter 4.2.1.1 the auditability is classified as complex. Like the previous use case, complexity is considered medium as it is a well-understood use case and contains relatively few modular components. It has medium widespread distribution as it is found in many but not most new vehicles (e.g. vehicles from Mazda, Hyundai). The attack applicability is simple in case of camera or acoustic sensors because the system can be fooled by fake emergency vehicle sounds or pedestrians wearing special T-shirts with adversarial prints. It is much more complex to fool the perception based on 3D sensors. When considering the involved perception components, segmentation is camera specific, clustering is point cloud specific and detection & classification are common to both. In Chapter 4.3 this use case is further analyzed since it is very relevant and forms the basis for different functionalities.

4.2.3.3 Road Elements Detection (17)

Road elements are the static elements on the road such as lane markings, landmarks, traffic signs, traffic lights, etc. They have a high safety relevance and failing to detect them can cause high damages. For example, failure

to detect lane markings can make the vehicle drift outside the driving lane and failure to detect traffic signs can result in incorrect speed or behavior of the vehicle. All the attributes are similar to the previous use case. The only difference is that the localization is included in the modular components, as the detection of road elements influences the localization output. Like the previous use case it is selected for the detailed analysis later.

4.2.3.4 Free Space Detection (18)

This use case detects the free space on the road on which the ego-vehicle can potentially drive. It is a complementary use case of the use cases 16 and 17, in the sense that the previous use cases detect the obstacles on the road whereas this use case detects the drivable area on the road. Hence, all the attributes are the same as for use case 17. Here, the perception component additionally includes occupancy grid maps which can be used for free space detection. Since this use case is very similar to use case 17 it is not analyzed further in Chapter 4.3.

4.2.3.5 Behavior Prediction (19)

This use case identifies the behavior and subsequently the trajectory of the traffic participants (e.g. crossing the street, overtaking of a vehicle). It is required for accident-free local path planning discussed in Chapter 4.2.3.6 and has a high safety relevance for the same reasons as use cases 16 and 17. The inputs can be either internally generated information (e.g. the pose of the detected pedestrians/vehicles in the last few frames) or the raw-sensor data. Using HD map information (e.g. location of zebra crossing) can improve the behavior prediction. Based on the type of inputs used, the involved modular components can be perception, prediction or both. Traditionally model based prediction approaches are used, but in recent times DNN-based trajectory prediction is gaining popularity. Again, the auditability is classified as complex as in Chapter 4.2.1.1. The complexity of this use case is rated higher than the complexity of the other use cases because the human factor is heavily involved and it is even for humans difficult to estimate the trajectory of the traffic participants in all cases. Also, the high complexity results in a low widespread distribution. In general, it is unrealistic to attack the system because this requires creating a fake behavior of other traffic participants without having internal access. When sensors are used as input, the involved perception components are detection and classification. This use case represents a different functionality than the other use cases and is therefore included in the detailed analysis later.

4.2.3.6 Local Path Planning (20)

While the use case 9 deals with the end-to-end path planning, local path planning plans the path of the ego-vehicle only in its perception range. The safety relevance is high, similar to the previous use cases. The inputs are the internal inputs which contain knowledge of the environment and the vehicle's current position. The modular components involved are planning and control. While traditionally tree-based and probabilistic algorithms are used, reinforcement learning approaches are currently on the rise. Like DNNs, reinforcement learning approaches face the problem of difficult (formal) verification and thus, considering the high safety relevance, auditability is classified as complex. This use case has a medium complexity with low widespread distribution. Similar to use case 9, attacks on local path planning are unrealistic. In Chapter 4.3 this use case is not further analyzed because it is very similar to use case 9 which is already included in the detailed analysis.

4.3 Use Case Analysis

After presenting a list of potential use cases and their assignment in important categories in Chapter 4.2, we now perform a more detailed analysis of important use cases. For this we use the selection of the most important use cases from the list of all presented use cases and analyze the suitability of these for the development of audit criteria. Afterwards, we also discuss how some use cases can be combined to cover even more driving scenarios and expand the audit criteria development.

4.3.1 Single Use Cases

In the following, we present a detailed analysis of selected use cases from Chapter 4.2. These are selected so that they cover all important use cases that are included in current systems for AD/ADAS. Use cases which are more niche and not widely used are not considered in the following, because the final aim of this project (or following ones) is to develop a technical guideline that allows the auditing of AI-based systems for AD. Therefore, it is most relevant to cover use cases which are actively used or are nearly ready for deployment from a technical point of view. The final selection for use cases for the development of the technical guideline is discussed in Chapter 4.4.

4.3.1.1 Additional Categories

Each important and selected use case is further analyzed in Chapter 4.3.1.2 using the following additional categories.

4.3.1.1.1 Representativity

First, we analyze the representativity of the general use case. Here, related specific tasks and functionalities are listed that are part of the general use case but are more specialized and focus on a concrete application. This gives a good overview of all techniques that belong to the broader use case.

4.3.1.1.2 Generalizability

Next, we discuss the generalizability for each general use case. Here, important factors are involved that mainly limit how well results can be reused for other use cases. Most importantly this includes which sensors are used, which perception components are involved and whether the use case impacts the planning and control of the vehicle.

4.3.1.1.3 Resources

Additionally, for each general use case we discuss the required resources. Again, this includes multiple factors like the availability of open-source datasets or representative implementations, the complexity and model size of involved AI components and the required computational resources for training or inference. Concretely, the following parameters are used to gain further insights on the required resources for each use case:

- Specific dataset: Are specific public datasets available that concentrate on this use case or include this use case amongst others.
    - **Available**: There is at least one public dataset available that is commonly used in research. In addition, a link to this dataset is provided.
    - **Uncommon**: There is no public dataset that is commonly used in research.
- Open-source implementations: Are open-source implementations available that specifically implement this use case or include this use case amongst others.
    - **Available**: There is at least one public implementation available. In addition, a link to this implementation is provided.
    - **Uncommon**: There is no public implementation with sufficient trust level.
- Data dimensionality: The size of the input data dimensionality relative to all use cases.
    - **High**: The dimensionality is on the higher end because:

      (a) Image data from camera sensors with a high resolution is required.

      (b) Point cloud data from LiDAR or RADAR sensors with a high resolution is required.

- **Medium**: The dimensionality is in the middle of the range because:
  (a) Image data with a low or medium resolution is sufficient.
  (b) Point cloud data with a low or medium resolution is sufficient.
- **Low**: The dimensionality is on the lower end because:
  (a) Internal data is required that is already processed and does not come directly from a camera, RADAR or LiDAR sensor.
  (b) Data from other sensors than camera, RADAR or LiDAR is required.

- Computational resources: The required computational resources needed to perform experiments (training, testing, etc.) relative to all use cases.
  - **High**: The required resources are on the higher end because:
    (a) The data from multiple sensors needs to be fused.
    (b) Recurrent/temporal algorithms or DNNs are used.
    (c) Semantic segmentation is performed.
    (d) The raw data needs extensive preprocessing before usage is possible.
  - **Medium**: The required resources are in the middle of the range because:
    (a) The data from a single sensor needs to be processed.
  - **Low**: The required resources are on the lower end because:
    (a) No sensor data is involved in real-time.
    (b) Fast algorithms or small DNNs are sufficient/typically used.
- Development effort: The required development effort for use case functionality or auditing tools relative to all use cases.
  - **High**: The required development effort is on the higher end because:
    (a) Large variance in relevant data attributes is possible.
    (b) Dynamic elements are relevant for the output.
  - **Medium**: The required development effort is in the middle of the range because:
    (a) Variance of relevant data attributes is limited.
    (b) Only static elements are relevant for the output.
  - **Low**: The required development effort is on the lower end because:
    (a) Relevant data attributes are always very similar.
    (b) Only static elements are relevant for the output.

4.3.1.1.4 Standards/Tools

Depending on the use case different tools are available which consider testing and auditing at various stages in the lifecycle. On the one hand, there exist tools which perform virtual tests using simulated data samples. A first approach can be to utilize built-in evaluation capabilities which come with software libraries used during development, e.g. performance evaluation with Scikit-learn[9]. Furthermore, general-purpose tools for

[9] https://scikit-learn.org/

the validation of ML-based systems can be deployed which do not focus on the specific needs of mobility use cases. Such tools are provided by different companies, for example by QuantPi[10] which focusses on explaining ML-based systems, or by LatticeFlow[11] which focusses on the robustness of ML-based systems. Simulation tools which are more use case specific can range from the specific hardware/sensor simulation, like Ansys AVxcelerate Sensors[12] or RT-LAB[13], to the complete driving simulation or digital twins. For open-source driving simulation tools the authors in (251) provide a comparison of different implementations. Additionally, there are various commercial driving simulation tools, for example ASM Traffic[14], CarMaker[15] or rFpro[16]. Furthermore, tools for virtual validation are typically developed by each company independently to fulfill the exact needs and requirements for the validation of their specific use case. Besides pure simulation tools, there are also tools which consider the physical testing in the real-world environment. Here, simulation is combined with physical devices and hardware, potentially mounted to a test vehicle. These tools enable a more realistic evaluation of AD/ADAS systems, but this comes with increased cost and time effort. In Chapter 4.3.1.2, exemplary tools of this category are assigned to the chosen use cases.

In addition to the tools, we cover existing standards, norms or technical guidelines specific to each use case in the tables below. The tables are organized based on the issuing body of the document. Additionally, it has to be noted that all use cases shall conform to existing safety and security standards, such as the ISO 26262 ((165), (246), (247)), ISO/PAS 21448 (242), ISO/SAE 21434 (249) and UNECE R 155 (250). Therefore, these norms are not mentioned explicitly for each use case.

4.3.1.1.4.1 ISO

Table 42 Overview of relevant documents issued by ISO

ID	*Name*
ISO 17387:2008	ISO 17387:2008: Intelligent transport systems –Lane Change Decision Aid Systems (LCDAS) – Performance requirements and test procedures (last reviewed in 2018) (252)
ISO 19377:2017	ISO 19377:2017: Heavy commercial vehicles and buses — Emergency braking on a defined path — Test method for trajectory measurement (253)
ISO 19237:2017	ISO 19237:2017: Intelligent transport systems — Pedestrian detection and collision mitigation systems (PDCMS) — Performance requirements and test procedures (254)
ISO 22078:2020	ISO 22078:2020: Intelligent transport systems — Bicyclist detection and collision mitigation systems (BDCMS) — Performance requirements and test procedures (255)
ISO 3888-2:2011	ISO 3888-2:2011: Passenger Cars— Test Track for a severe lane-change manoeuvre — Part 2: Obstacle Avoidance (256)
ISO 22735:2021	ISO 22735:2021: Road vehicles — Test method to evaluate the performance of lane-keeping assistance systems (257)
ISO 11270:2014	ISO 11270:2014: Intelligent transport systems — Lane keeping assistance systems (LKAS) — Performance requirements and test procedures (258)
ISO/SAE PAS 22736:2021	ISO/SAE PAS 22736:2021: Taxonomy and definitions for terms related to driving automation systems for on-road motor vehicles (259)

[10] https://quantpi.com/
[11] https://latticeflow.ai/
[12] https://ansys.com/de-de/products/av-simulation/ansys-avxcelerate-sensors
[13] https://opal-rt.com/autonomous-vehicle/
[14] https://dspace.com/de/gmb/home/medien/product_info/prodinf_asm_traffic.cfm
[15] https://ipg-automotive.com/de/produkte-loesungen/software/carmaker/
[16] https://rfpro.com/

ID	Name
ISO 19638:2018	ISO 19638:2018: Intelligent transport systems – Road Boundary Departure Prevention Systems (RBDPS) – Performance requirements and test procedures (260)
ISO 21717:2018	ISO 21717:2018: Intelligent transport systems – Partially Automated In-Lane Driving Systems (PADS) – Performance requirements and test procedures (261)
ISO 21202:2020	ISO 21202:2020: Intelligent transport systems – Partially Automated Lane Change Systems (PALS) – Functional / operational requirements and test procedures (262)
ISO 15622:2018	ISO 15622:2018: Intelligent transport systems – Adaptive Cruise Control Systems – Performance requirements and test procedures (263)
ISO 20035:2019	ISO 20035:2019: Intelligent transport systems – Cooperative Adaptive Cruise Control Systems (CACC) – Performance requirements and test procedures (264)
ISO 15622:2002	ISO 15622:2002: Transport information and control systems – Adaptive Cruise Control Systems – Performance requirements and test procedures (265)
ISO/TR 22086-1:2019	ISO/TR 22086-1:2019: Intelligent transport systems (ITS) – Network based precise positioning infrastructure for land transportation – Part 1: General information and use case definitions (266)
ISO/TS 21176:2020	ISO/TS 21176:2020: Cooperative intelligent transport systems (C-ITS) – Position, velocity and time functionality in the ITS station (267)
ISO/TR 16786:2015	ISO/TR 16786:2015: Intelligent transport systems – The use of simulation models for evaluation of traffic management systems – input parameters and reporting template for simulation of traffic signal control systems (268)
ISO 22741:2022	ISO 22741:2022: Intelligent transport systems – Roadside modules AP-DATEX data interface (269)
ISO/AWI TS 5283	ISO/AWI TS 5283: Road Vehicles – Ergonomic aspects of driver monitoring and system interventions in the context of automated driving (still under development) (270)
ISO 19206-3:2021	ISO 19206-3:2021: Road Vehicles – Test devices for target vehicles, vulnerable road users and other objects, for assessment of active safety functions – Part 3: Requirements for passenger vehicle 3D targets (271)
ISO/TS 18506:2014	ISO/TS 18506:2014: Procedure to construct injury risk curves for the evaluation of road user protection in crash tests (272)

4.3.1.1.4.2 SAE

Table 43 Overview of relevant documents issued by SAE

ID	Name
SAE J2400_200308	SAE J2400_200308: Human Factors in Forward Collision Warning Systems: Operating Characteristics and User Interface Requirements (273)
SAE J3029_201510	SAE J3029_201510: Forward Collision Warning and Mitigation Vehicle Test Procedure - Truck and Bus (274)
SAE J3048_201602	SAE J3048_201602: Driver-Vehicle Interface Considerations for Lane Keeping Assistance Systems (275)
SAE J2808_201701	SAE J2808_201701: Lane Departure Warning Systems: Information for the Human Interface (276)
SAE J3240	SAE J3240: Passenger Vehicle Lane Departure Warning and Lane Keeping Assistance Systems Test Procedure (277)
SAE J2399_202110	SAE: J2399_202110: Adaptive Cruise Control (ACC) Operating Characteristics and User Interface (stabilized Oct 2021) (278)

ID	Name
SAE J2365	SAE J2365: Calculation and Measurement of the Time to Complete In-Vehicle Navigation and Route Guidance Tasks (279)
SAE J2678_201609	SAE J2678_201609: Navigation and Route Guidance Function Accessibility While Driving Rationale (Cancelled Sep 2016) (280)
SAE J3114_201612	SAE J3114_201612: Human Factors Definitions for Automated Driving and Related Research Topics (281)
SAE J2396_201705	SAE J2396_201705: Definitions and Experimental Measures Related to the Specification of Driver Visual Behavior Using Video Based Techniques (282)
SAE J2944_201506	SAE J2944_201506: Operational Definitions of Driving Performance Measures and Statistics (283)
SAE J2945/A	SAE J2945/A: Standard for Lane-Level and Road Furniture Mapping for Infrastructure-based V2X Applications (284)
SAE J2945/9	SAE J2945/9: Vulnerable Road User Safety Message Minimum Performance Requirements (285)
SAE J3134_201905	SAE J3134_201905: Automated Driving System (ADS) Marker Lamp (286)

4.3.1.1.4.3 UNECE

Table 44 Overview of relevant documents issued by UNECE

ID	Name
UNECE R79	UNECE R79: Steering Equipment (287)
UNECE R157	UNECE R157: UN Automated Lane Keeping Systems (ALKS) (288)
UNECE GTR 9	UNECE GTR 9: Pedestrian Safety (289)

4.3.1.1.4.4 BSI

Table 45 Overview of relevant documents issued by BSI

ID	Name
BSI Reliability Assessment of Traffic Sign Classifiers	Reliability Assessment of Traffic Sign Classifiers (290)

4.3.1.2 Analysis

An overview of the resulting analysis is shown in Table 46 for all selected use cases. Here, more detailed information is provided which explains the analysis further using the categories presented in Chapter 4.3.1.1. For the resources category we provide the reasons for each categorization using the argumentation from Chapter 4.3.1.1.3 denoted in the parenthesis.

Table 46 Overview of the use case analysis

ID	Representativity	Generalizability	Resources	Standards/Tools
2	• Steering Torque Control • Automated Emergency Steering • Emergency Braking Head-on • Emergency Braking Junction • Rear Emergency Braking • Traffic Jam Assist	• Camera, LiDAR or fusion-based outside perception • Impact on longitudinal and lateral control • Using detection or segmentation or clustering	• Specific datasets: Uncommon • Open-source implementations: Uncommon • Data dimensionality: High (a, b) • Computational resources: Medium (a) or High (a, c)	Standards: • ISO 17387:2008 • ISO 19377:2017 • ISO 19237:2017 • ISO 22078:2020 • ISO 3888-2:2011 • SAE J2400_200308 • SAE J3029_201510 • UN ECE R79

ID	Representativity	Generalizability	Resources	Standards/Tools
	• Collision Warning / Crash Alert		• Development effort: High (a, b)	Tools: • Sensor/driving simulation • Dewesoft[17] • OxTS[18]
3	• Lane Keeping Assist • Lane Centering Assist • Emergency Lane Keeping • Lane Sway Warning • Lane Departure Warning	• Camera or fusion-based outside perception • Impact on lateral control • Using segmentation	• Specific datasets: Available – e.g. CULane[19] • Open-source implementations: Available – e.g. DNN[20] • Data dimensionality: Medium (a) • Computational resources: High (c) • Development effort: Medium (a, b)	Standards: • ISO 22735:2021 • ISO 11270:2014 • ISO/SAE PAS 22736:2021 • ISO 19638:2018 • ISO 21717:2018 • ISO 19377:2017 • SAE J3048_201602 • SAE J2808_201701 • UN ECE R157 Tools: • Sensor/driving simulation • VBOX[21] • Dewesoft • OxTS
4	• Driver Initiated Lane Change • Automated Lane Change • Merge-In Request • Automated Highway Entering • Automated Highway Leaving • Automated Lane Selection Junction • Automated Full Junction Handling	• Camera, LiDAR or fusion-based outside perception • Impact on lateral control • Using detection or segmentation or clustering	• Specific datasets: Uncommon • Open-source implementations: Uncommon • Data dimensionality: High (a, b) • Computational resources: High (a, b, c) • Development effort: High (a, b)	Standards: • ISO 21202:2020 • ISO 17387:2008 • ISO 19377:2017 • ISO 19237:2017 • ISO 22078:2020 • ISO 3888-2:2011 • SAE J3240 • SAE J2808_201701 • UN ECE R79 Tools: • Sensor/driving simulation • VBOX • Dewesoft • OxTS
5	• Cruise Control • Curve Speed Adaption • Stop & Go Control	• Camera, LiDAR or fusion-based outside perception • Impact on longitudinal control • Using detection or clustering	• Specific datasets: Uncommon • Open-source implementations: Uncommon • Data dimensionality: Medium (a, b)	Standards: • ISO 15622:2018 • ISO 20035:2019 • ISO 15622:2002 • SAE: J2399_202110 Tools:

[17] https://dewesoft.com/de/applikationen/fahrzeug-tests
[18] https://www.oxts.com/de/industry/automotive-testing-and-development/
[19] https://xingangpan.github.io/projects/CULane.html
[20] https://github.com/voldemortX/pytorch-auto-drive
[21] https://vboxautomotive.co.uk/index.php/de/

ID	Representativity	Generalizability	Resources	Standards/Tools
			• Computational resources: Medium (a) or High (a) • Development effort: Medium (a)	• Sensor/driving simulation • VBOX • Dewesoft • OxTS
9	• Global Route Planning • Dynamic Route Adjustment • Traffic Jam Prediction • Arrival Time Prediction • Personalized Route Learning	• No perception • No impact on control • Using prediction	• Specific datasets: Uncommon • Open-source implementations: Uncommon • Data dimensionality: Low (b) • Computational resources: Low (a) • Development effort: Low (a, b)	Standards: • ISO/TR 22086-1:2019 • ISO/TS 21176:2020 • SAE J2365 • SAE: J2678_201609 Tools: • Driving simulation • VBOX • Dewesoft • OxTS
12	• Traffic Sign Recognition • Speed Limit Warning • Automatic Speed Adaption • Right-of-Way Assistant • Current Valid Signs Reminder	• Camera-based outside perception • (No) impact on longitudinal control • No impact on lateral control • Using detection	• Specific datasets: Available – e.g. GTSRB[22] • Open-source implementations: Available – e.g. DNN[23] • Data dimensionality: Medium (a) • Computational resources: Low (b) • Development effort: Low (a, b)	Standards: • ISO/TR 16786:2015 • ISO 22741:2022 • BSI Reliability Assessment of Traffic Sign Classifiers Tools: • Sensor/driving simulation • OxTS
14	• Fatigue/Drowsiness Monitoring • Distraction Alerts • Gaze Detection • Driving Suitability Assessment	• Camera-based inside perception • No impact on control • Using detection	• Specific datasets: Available – e.g. Vicomtech DMD[24] • Open-source implementations: Available – e.g. DNN[25] • Data dimensionality: Medium (a) • Computational resources: Low (b) • Development effort: Medium (a)	Standards: • ISO/AWI TS 5283 • SAE J3114_201612 • SAE J2396_201705 • SAE J2944_201506 Tools: • Dewesoft
15	• Ego Motion Estimation • Features/Object Detection • Map Updates	• Fusion-based outside perception • No impact on control • Using feature matching	• Specific datasets: Available – e.g. KITTI[26] • Open-source implementations: Available – e.g. Algorithms[27]	Standards: • SAE J2945/A Tools: • Sensor/driving simulation • VBOX • Dewesoft

[22] https://benchmark.ini.rub.de/gtsrb_news.html
[23] https://github.com/poojahira/gtsrb-pytorch
[24] https://dmd.vicomtech.org/
[25] https://github.com/AleksaArsic/ADAS-ML-Driver-Monitoring-System
[26] http://www.cvlibs.net/datasets/kitti/
[27] https://github.com/RozDavid/LOL

ID	Representativity	Generalizability	Resources	Standards/Tools
			• Data dimensionality: High (a, b) • Computational resources: Medium (a) or High (a, d) • Development effort: Medium (b)	
16	• Pedestrian Detection • Vehicle Detection • Bicycle Detection • Scooter Detection • Emergency Vehicle Detection	• Camera, LiDAR or fusion-based outside perception • No impact on control • Using detection or segmentation or clustering	• Specific datasets: Available – e.g. nuScenes[28] • Open-source implementations: Available – e.g. DNN[29] • Data dimensionality: High (a, b) • Computational resources: Medium (a) or High (a, c) • Development effort: High (a, b)	Standards: • ISO 19206-3:2021 • ISO/TS 18506:2014 • SAE J2945/9 • SAE J3134_201905 • UN ECE GTR 9 Tools: Sensor/driving simulation VBOX Dewesoft OxTS
17	• Lane Markings Detection • Landmarks Detection • Traffic Lights Detection	• Camera or fusion-based outside perception • No impact on control • Using detection or segmentation	• Specific datasets: Available - e.g. Mapillary Vistas[30] • Open-source implementations: Available – e.g. DNN[31] • Data dimensionality: High (a) • Computational resources: High (c) • Development effort: Medium (a, b)	Standards: • ISO 21202:2020 Tools: • Sensor/driving simulation • VBOX • Dewesoft
19	• Pedestrian Motion Prediction • Vehicle Motion Prediction • Driver Style Recognition • Decision Making	• Prediction based on sensor data or perception output and map • No impact on control	• Specific datasets: Available – e.g. Caltech PIE[32] • Open-source implementations: Available – e.g. DNN[33] • Data dimensionality: Low (a) or High (a) • Computational resources: High (c) • Development effort: High (a, b)	Standards: - Tools: • Sensor/driving simulation • A Scenario-Based Platform for Testing Autonomous Vehicle Behavior Prediction Models in Simulation (291)

[28] https://www.nuscenes.org/
[29] https://github.com/xingyizhou/CenterNet
[30] https://www.mapillary.com/dataset/vistas
[31] https://github.com/open-mmlab/mmdetection
[32] https://data.nvision2.eecs.yorku.ca/PIE_dataset/
[33] https://github.com/aras62/PIEPredict

4.3.2 Combination of Use Cases

After analyzing specific use cases, in the following we discuss whether some use cases can be combined and how this impacts the development of audit criteria. It is important to note that most use cases presented in Chapter 4.3.1.2 already have a high complexity for auditing. Therefore, the development of audit criteria is already complex even when no combination of use cases is considered. Hence, we perform the initial selection of use cases in Chapter 4.4 only from the use cases presented so far and do not use a combination of use cases for the initial development of the toolchain and the first tests of audit criteria.

Using a combination of use cases is more relevant later when the goal is to expand the tests and transfer developed audit criteria to more use cases. Here, it is advantageous when it is relatively easy to add an additional use case to the one considered for developing the audit criteria. If such an addition is naturally possible this provides a straightforward way to expand the tests of the audit criteria to include more use cases and driving functionalities.

Additionally, it is interesting to test the transferability of the developed audit criteria to use cases which differ from the one used for the development regarding the generalization aspects discussed in Table 46. Since the goal of this project (or follow-up projects) is to find audit criteria for a modular technical guideline, ideal audit criteria are transferable to different use cases even when for example different sensors or perception components are used. Hence, the evaluation of the proposed audit criteria for a use case outside of the generalization of the use case used for development is interesting. Therefore, we also discuss for each combination whether this combination enables to test the transferability of the audit criteria to a use case with different aspects for the generalizability. Here, in principle all combinations of use cases which complement each other with respect to the development of audit criteria are interesting to investigate, regardless of whether the functionality of the use cases is connected to each other. By considering all possible combinations, independent of whether they functionally complement each other, this perspective allows to find a use case combination which enables to optimally test the transferability of audit criteria to use cases with an entirely different generalizability. However, in the following we only discuss the combinations of use cases that functionally complement each other, which allows the easiest expansion of the audit criteria tests with the lowest amount of additional work.

Summarizing, in the following we will mainly discuss how use cases can be combined with the goal in mind to expand the auditing process and criteria development while allowing on optimal coverage of the entire parameter space. These combinations then serve as a guideline for following projects to expand the audit criteria tests but are not used for the initial selection of use cases for AP5 and AP7.

4.3.2.1 Combination of AD Use Cases

The first sensible combinations arise when combining specific AD use cases (15, 16, 17 and 19 in Table 46) with the other use cases in Table 46. Concretely, all considered AD use cases focus on perception or prediction and do not include direct control functionalities. In contrast, the more generic use cases also mostly include control functionalities. Hence, a sensible approach could be to start by using one of the presented AD use cases for the initial development of audit criteria. Since there is no direct control impact, the development and tests of the audit criteria are more focused to the IT security of the AI component and can be less extensive. This enables an easier start of the criteria development and allows to approach the complexity of an entire system step by step. Following the development of audit criteria for an AD use case, in the next step this use case can then be integrated into one of the generic use cases. This embeds the use case in a higher level to consider concrete driving functionalities that can be used in reality. Hence, the development of audit criteria can be expanded to a use case that includes control components, which enables the exemplary full auditing of an entire system and not only of the AI component. This approach allows to first optimally cover important aspects of AI components while having a lower complexity to allow more feasible tests. Then, the tests are expanded to the full complexity by considering the impact on the control component in the context of a generic use case. Thus, in Table 47 we show where it is sensible to combine an AD use case with a more

generic use case. Here, we also use two different levels for the estimated complexity of the combination with respect to the implementation effort and entire system functionality.

Table 47 Overview of the combination complexity for AD use cases

AD Use Case		*Combination Complexity 1*		*Combination Complexity 2*	
ID	*Name*	*ID*	*Name*	*ID*	*Name*
15	A Priori Map-based Localization	3 12	Lane Keeping Traffic Sign Assistant	4	Lane Changing
16	Road Users Detection	2 5	Collision Avoidance Adaptive Cruise Control	4	Lane Changing
17	Road Elements Detection	3 12	Lane Keeping Traffic Sign Assistant	4	Lane Changing
19	Behavior Prediction	5	Adaptive Cruise Control	2 4	Collision Avoidance Lane Changing

Use case 15 can be straightforwardly combined with use case 3 since the accurate localization plays a role for keeping the driving lane. Additionally, the localization can be used as an alternative/redundancy to traffic sign detection when an accurate map with all relevant traffic signs is available. This use case can also be combined with use case 4 because the localization in the lane also plays a role during lane changing. However, this is strictly more complex than the combination with the lane keeping use case as discussed in Chapter 4.2.1.4. For all presented combinations testing the transferability of audit requirements to use cases with different aspects is possible. The main reason is that the map-based localization differs from all other use cases because a classical perception component is usually not involved. Using feature matching is special and thus any combination with another use case includes necessarily a test of the transferability of the requirements.

Next, use case 16 can be combined with use case 2 and use case 5 since both require the detection of other road users for avoiding or following. Also, the combination with use case 4 is possible because lane change maneuvers require the location of other road users to determine whether the lane is free to change. Again, this is more complex since use case 4 itself is very complex. It is possible to test the transferability of audit criteria for all presented combinations. This depends on the concrete implementation of the included functionalities. For example, one possibility is to use a camera-based pedestrian detection for the initial development of audit criteria. Then the combination with a point cloud-based vehicle detection is possible, which in combination allows to implement the functionalities of the collision avoidance use case. Another example is to start with a camera or point-cloud based detection of vehicles and use a camera-based segmentation to enable the lane changing use case.

For use case 17 it is possible to combine it with use case 3 and 12 because for both use cases the detection of road elements is important. In case of use case 12, the detection of traffic signs is required, whereas use case 3 requires the detection of road/lane markings. Here, we assume in the following camera-based road elements detection which is mainly used. Like the previous use cases the combination with use case 4 is possible but most challenging. In contrast to the discussion for the previous use case, testing the transferability of criteria is only possible for the combination with the collision avoidance use case. The reason is that both use case 3 and 12 use camera-based perception, which is also mainly used for use case 17. Hence, only the combination with use case 4 allows to test the transferability of audit criteria to different sensor setups and thus different perception components.

Finally, use case 19 can be combined with use case 5, where the prediction of the driving behavior of the leading vehicle might increase the overall performance. For the combinations with increased complexity, using use case 2 is possible in addition to the previously discussed use case 4. For use case 2 more collisions could be avoided when the possible behavior of other participants is predicted in advance, instead of only reacting to the observed scene. Like the combinations discussed for use case 15, it is possible to test the transferability of audit criteria for each combination with this use case. This is because use case 19 uses different input data and thus captures an entirely different parameter space.

4.3.2.2 Combination of Generic Use Cases

In addition to expanding the AD use cases by combining them with the generic use cases, it is also possible to combine the generic use cases among themselves. This further allows to increase the complexity and scale the development and tests of audit criteria to more and more complex driving functionalities. It enables the expansion from a single use case and builds on top of the achieved results to transfer them to more use cases that can be meaningfully combined. In Table 48 sensible combinations are again shown with an estimation of the required complexity.

Table 48 Overview of the combination complexity for non AD use cases

AD Use Case		*Combination Complexity 1*		*Combination Complexity 2*	
ID	*Name*	*ID*	*Name*	*ID*	*Name*
2	Collision Avoidance	5	Adaptive Cruise Control	3	Lane Keeping
3	Lane Keeping	5	Adaptive Cruise Control	2	Collision Avoidance
		12	Traffic Sign Assistant	4	Lane Changing
4	Lane Changing	5	Adaptive Cruise Control	3	Lane Keeping
		12	Traffic Sign Assistant		
5	Adaptive Cruise Control	2	Collision Avoidance	-	
		3	Lane Keeping		
		4	Lane Changing		
		12	Traffic Sign Assistant		
9	Global Navigation/Path Planning	-			
12	Traffic Sign Assistant	3	Lane Keeping	-	
		4	Lane Changing		
		5	Adaptive Cruise Control		
14	Driver Monitoring	-			

First, use case 2 can be combined with use case 5, which adds functionalities to react to potential collisions while following a leading vehicle. In addition, the combination with use case 3 is possible to react to collisions while also automatically staying in a driving lane. This has a higher complexity because during adaptive cruise control the number of collisions that can occur with respect to colliding with the leading vehicle is lower than the number of collisions that can occur during automatic lane keeping. All combinations can also be used to test the transferability of audit criteria because it is possible to use different sensors for each use case. For example, one could use LiDAR-based vehicle detection for collision avoidance and combine it with camera-based detection for use case 5 or camera-based segmentation for use case 3.

Use case 3 can most easily be combined with use case 5, which enables to perform the standard driving behavior of following a vehicle in a lane completely automatically. Here, the transferability testing of audit criteria is also possible because different sensors and perception components can be used. The combination with use case 12 allows to also adapt the vehicle speed to the current road speed limit, instead of simply following a leading vehicle. However, this combination does not allow testing the audit criteria transferability because both use cases are based on camera perception. For the more complex combinations, on the one hand, the combination with use case 2 is possible as discussed previously. Additionally, the combination with use case 4 is possible and adds even more complex driving functionalities. This allows to change lanes automatically, for example to perform an overtake maneuver instead of only following a vehicle. Here, testing the criteria transferability is also possible, because for use case 4 it is possible to use point cloud-based perception in contrast to camera-based segmentation for use case 3.

Next, use case 4 can be combined with use case 5, which allows automatic overtakes and with use case 12, which allows to adapt the driving speed to the speed limit in the new lane. For example, on multi-lane highways there are situations where each lane has a different speed limit. Both combinations also enable the testing of the transferability of audit criteria because different sensors and perception components can be used. For example, it is possible to perform point cloud-based detection to detect free spaces for lane changing

and use a camera-based detection for traffic signs or the leading vehicle. As discussed in the previous use case, the combination with use case 3 is also possible.

For use case 5, the combination with the use cases 2, 3 and 4 is possible as discussed previously. Additionally, the combination with use case 12 is possible, which enables the automated adaption of the current speed limit while following a leading vehicle. Again, testing the transferability of criteria is possible, for example by using point cloud-based detection of the leading vehicle and using camera-based detection of traffic signs.

Use case 9 and use case 14 cannot be sensibly combined with other use cases, since they do not consider driving functionalities. Hence, there is no option to combine these use cases with the other ones, which all consider concrete driving functionalities. Nevertheless, both use cases might be useful for a final test of the transferability of audit criteria because they differ considerably from all other use cases. However, it is more sensible to explore a meaningful combination of use cases first, as this allows to build upon already developed tools and software and still allows to test the criteria transferability in most cases.

Finally, use case 12 can be combined with use cases 3,4 and 5, which is discussed previously at the respective use case.

4.4 Use Case Recommendations

After presenting possible use cases for AI in AD and ADAS systems in Chapter 4.2 and performing a detailed analysis of the most relevant and interesting use cases in Chapter 4.3, we now discuss the recommendations of use cases for the following work packages five and seven. For these work packages it is important to have a clearly defined and very specific use case to be able to perform the desired experiments and tests of the audit criteria. The goal is to perform in-depth technical experiments and concretely analyze potential shortcomings of proposed audit criteria. Therefore, it is required to select a narrow use case which allows to focus on a concrete technique and assess potential audit requirements. For this we use the previous analysis described in this document as the basis to first select a basic use case which determines the main functionality. Then, we narrow this use case and select a concrete application of the use case which enables the described procedure for work package five and seven. The result is a selection of a narrow use case which enables the initial development and testing of audit criteria while representing a clear way to integrate the narrow application into a real mobility use case which captures some realistic driving functionalities.

To perform the selection of the basic use case we revisit the analysis in Table 46 and the basic categorization in the associated Table 39, Table 40 and Table 41. Depending on the performed categorization some use cases are more suitable for the initial development and testing of audit criteria than others. In Table 49 a summary of the suitability of all presented categories is shown. For each category a color is assigned which indicates whether the respective parameters are well suited (green) for the initial development of audit criteria, partly suited (yellow) or not suited (red).

First, a certain amount of safety relevance must be given and a high relevance is ideal. This allows to develop audit criteria for critical tasks where an audit is important and required. Next, ML-based systems should be in use or as close as possible to real world usage. If the usage of ML-based systems is unrealistic, the use case is not suited for the development of audit criteria. In the best-case scenario, the auditability of a use case should be simple, which allows for a feasible initial development and tests. However, as discussed in Chapter 4.2 the auditability is typically rather complex, which means a complex auditability cannot be treated as a strict exclusion criterion. Next, the suitability of the complexity of a use case follows the previous discussions on the auditability, because the complexity of a use case should be rather low for the initial development, which allows to perform more extensive tests and developments. A highly widespread distribution of a use case is preferable because this allows to develop audit criteria that are very relevant for practical systems. Nevertheless, having use cases with a less widespread distribution is not an exclusion criterion because these can still be relevant and the distribution could increase rapidly in the near future. Regarding the attack applicability it is important that an attack interface exists and that attacks can be performed theoretically. In the best case, the applicability is simple, which allows the most feasible tests and deals with the most relevant threats. Lastly, it is more suitable when datasets and implementations are publicly available and mainly low

resources are required for a use case. In contrast, when no dataset is available or the complexity of an implementation is estimated as high, the use case is out-of-scope for this project due to the limited timeframe.

Table 49 Overview of the suitability of the analysis parameters per category for the audit criteria development

Requirement	*Suitable (S)*	*Partially Suitable (PS)*	*Unsuitable (U)*
Safety Relevance	High	Medium, Low	None
AI Usage	ML: Current, Near	ML: Far	ML: Unrealistic
Auditability	Simple	Medium, Complex	-
Complexity	Low	Medium, High	-
Widespread Distribution	High	Medium, Low	-
Attack Applicability	Simple	Medium, Complex	Unrealistic
Resources	All the following: • Available dataset & implementation • Mainly low resources	Everything in between	One of the following: • No dataset available • Estimated complex implementation

After discussing the suitability of different parameters of the categories used for the analysis of the use cases, in Table 50 we summarize the suitability of each use case selected in Chapter 4.3 for the development of audit criteria by applying the introduced suitability to each parameter. Based on this overview it shows that the use cases Collision Avoidance (2), Lane Changing (4), Adaptive Cruise Control (5), Global Navigation/Path Planning (9), Driver Monitoring (14) and Behavior Prediction (19) are not suited for the initial development of audit criteria. All remaining use cases are considered in the following for the recommendation. Thus, in Chapter 4.4.1 we first discuss which use case is most suited for the initial development of audit criteria and is therefore recommended for AP5. We also narrow this use case such that a concrete application is selected which allows to perform in-depth technical experiments and gain detailed insights. Afterwards, in Chapter 4.4.2 we perform a complementary recommendation of an additional use case for AP7. Here, we also provide alternative options, discuss the advantages and disadvantages of each use case and again narrow the use cases to arrive at a concrete application which can be tested extensively.

Table 50 Overview of the suitability of use cases for the audit criteria development

ID	*Safety Relevance*	*AI Usage*	*Auditability*	*Complexity*	*Widespread Distribution*	*Attack Applicability*	*Resources*
2	S	S	PS	PS	PS	PS	U
3	S	S	PS	PS	PS	S	PS
4	S	S	PS	PS	PS	PS	U
5	S	S	PS	PS	S	PS	U
9	U	S	S	S	S	U	U
12	PS	S	S	S	S	S	S
14	PS	S	PS	PS	PS	U	PS
15	S	S	PS	PS	PS	PS	PS
16	S	S	PS	PS	PS	S / PS	PS
17	S	S	PS	PS	PS	S / PS	PS
19	S	S	PS	PS	PS	U	PS

4.4.1 Use Case Recommendation AP5

As can be seen in Table 50, the use case Traffic Sign Assistant (12) is green for most categories and thus is most suitable. Mostly the rather low complexity and simple auditability in combination with the availability of datasets and implementations and in general a lower demand of resources are decisive and well suited for the initial development and tests of audit criteria. The only parameter that is not well suited is the low safety relevance of this use case. However, the safety relevance could also be increased by considering driving functionalities that automatically adapt the vehicle speed to the detected speed limit and do not only serve as

an assistance to the driver. This extension could be useful in following projects to increase the complexity and add a direct impact on the control component of a vehicle. For the initial development of audit criteria in AP5, the currently considered assistant functionality is well suited because it allows for easier experiments and more extensive tests.

Summarizing, the use case recommendation for AP5 is based on the traffic sign assistant use case. Specifically, this means that a single outside forward facing camera sensor is used to generate the required input data. Based on this data, the classification (and previous detection) of traffic signs is performed using a DNN-based system. Our recommendation is to first only consider systems that perform a pure classification of traffic signs. The reason is that typically there exists a common detector for all kinds of road elements as discussed in Chapter 4.2.3.3. Based on the detected objects, the content of the detected bounding boxes is fed to specific systems that specialize in concretely classifying the object in a box based on the basic road elements classes. For the case of traffic sign recognition this means that a preceding road elements detector exists which outputs bounding boxes and an associated basic class, e.g. road sign, traffic light, etc. Then a specific classifier exists for each basic class which outputs the concrete class, e.g. stop sign, green bus traffic light, etc. Thus, a classifier that focusses on traffic signs is used to determine the concrete sign type based on the given cut out part of the entire input image. The output of this system is therefore the detected traffic sign in the given image. This specific classifier is selected for the development and tests in AP5 because it has a limited complexity, which allows to perform extensive and in-depth technical experiments. Later, it is possible to combine this specific classifier with a common detector, for example as part of AP7. Also, the classifier can be considered as part of a real traffic sign assistant operating in vehicles on public roads. This allows to test the developed audit criteria on production-ready systems and not only on academically developed ones.

4.4.2 Use Case Recommendation AP7

After recommending a use case for AP5, we now discuss the recommendation of an additional use case for AP7. This use case should allow to test the transferability of the developed audit criteria in AP5 to other use cases. Also, it is important that the second use case has a higher safety relevance than the first use case, which allows developing audit criteria which are required for practical applications of AD/ADAS systems. The selection is again based on the categorization of the suitability in Table 50. Also, it is important to note that as part of AP7 at the beginning the use case recommendation is revisited. Depending on the results and experience during AP5 it is assessed whether the recommendation is still valid. Otherwise, an alternative use case is selected from the list of alternative use cases provided in Chapter 4.4.2.2. Therefore, the recommendation for the additional use case in AP7 is not final and can change depending on the progress of this project and specifically the information gained during AP5.

4.4.2.1 Main Use Cases

As mentioned previously the first use case of the two use case recommendations for AP7 is the use case Traffic Sign Assistant (12), which is selected for AP5 discussed in Chapter 4.4.1. It is sensible to include this use case in AP7 since it forms the basis of the audit criteria and toolchain development and allows directly expanding on the previous work from AP5. In addition, a second use case needs to be selected which bases on the achieved results and developed components but allows expanding the gained insights to further domains and components. Here, it is preferable that the second use case can reuse most components while still allowing testing the transferability of audit criteria.

For the remaining suitable use cases in Table 50, use case 3 is the only generic use case remaining. All other use cases 15, 16 and 17 are more fundamental use cases which form the basis of an AD system or of other generic use cases. As discussed in Chapter 4.3.2.1, for the initial development and tests of audit criteria it is more sensible to select a fundamental use case which allows extensively testing the IT security of the AI component itself without the increased complexity that arrives when the impact on the entire system and control components is considered. Here, it is more realistic to cover the interplay of the entire system and the concrete interplay with a planning or control component in following projects with an increased duration. For this reason, we do not consider use case 3 for the recommendation of the use cases for AP7 since it would

directly include the control components. Instead, our recommendation is to focus on a fundamental use case in AP7 and then extend this use case to also consider the impact on the control component in following projects for example by the use case combinations presented in Chapter 4.3.2.1.

All remaining fundamental use cases have a very similarly rated suitability in Table 50. The only difference occurs when focusing on the applicability of an attack. Here, it shows that the attack applicability for use case 15 is considered as complex under all conditions. Instead, for use case 16 and 17 the attack applicability is only considered complex when point cloud data is used as input data for the detection. Therefore, we disregard use case 15 for the remaining considerations for an optimal recommendation of a second use case. Summarizing, from all use cases in Table 50 we arrive at the use cases Road Users Detection (16) and Road Elements Detection (17) which are still under consideration for the second use case in AP7. Additionally, use case Free Space Detection (18) can be considered as well because it is complementary to both use cases 16 and 17 as discussed in Chapter 4.2.3.4. It shares all properties with the other two use cases and is therefore not analyzed explicitly in Chapter 4.3 and Table 50. Nevertheless, since use case 16 and 17 remain as candidates for the second use case recommendation, it is sensible to again include use case 18 at this point.

In the following, all three use cases are first narrowed down further before a final decision is made regarding the recommendation. For use case 16 the most important concrete applications are the detection of vehicles and the detection of pedestrians, which are the most relevant and often occurring road users. In both cases the detection is based on outside facing sensors and can be based on camera sensors or on LiDAR sensors. For use case 17 the most important concrete application is the detection of lane markings, which is mainly camera-based and only in specific situations can be based on point cloud data (e.g. construction zones, tunnels, etc.). To detect lane markings, segmentation techniques are typically used as the perception component instead of bounding box detection techniques, which are typically used for use case 16. Similarly, the most important concrete application of use case 18 is the camera-based detection of the drivable area in the front of a vehicle, again using segmentation techniques. As described in Chapter 4.2.3.4 this is complementary to the detection of road users and road elements like lane markings. In total, five narrowed use cases result, which all seem fitting as a recommendation for AP7.

To decide on the best recommendation, we discuss the advantages of each narrowed use case in Table 51. The respective disadvantages arise as the opposite of the advantages. For example, the advantage of all camera-based use cases is that they are most like the use case selected for AP5 and the results and toolchain can likely be reused easiest. At the same time the disadvantage arises that the transferability of audit criteria cannot be tested just as well. This is then an advantage of selecting a LiDAR-based use case because it allows testing the transferability of the proposed criteria to an entirely different sensor setup. Similarly, use cases that have a different perception component have the advantage that the transferability of audit criteria can better be examined than when a similar perception component is used. Regarding the difference between pedestrian and vehicle detection it shows that detecting vehicles is typically easier due to a larger size, but the detection of pedestrians is more critical since they are most vulnerable.

Table 51 Overview of advantages of specific use cases for deriving a second use case recommendation

Use Case		*Advantages*
Camera-based Road User Detection	Pedestrian	• Same sensor -> High similarity • Most vulnerable users
	Vehicle	• Same sensor -> High similarity • Easiest to detect users
LiDAR-based Road User Detection	Pedestrian	• Different sensor -> Test transferability • Most vulnerable users • Different perception techniques
	Vehicle	• Different sensor -> Test transferability • Easiest to detect users • Different perception techniques
Camera-based Lane Marking Detection		• Same sensor -> High similarity

Use Case	*Advantages*
	• Different perception techniques
Camera-based Drivable Area Detection	• Same sensor -> High similarity • Different perception techniques

Based on the presented comparison of the advantages of the different use case we now recommend the use case LiDAR-based vehicle detection for AP7. From all the use cases analyzed in Table 51 this use case has the highest potential to test the transferability of audit criteria to a different parameter space, like different sensors or perception components. Therefore, it is the best supplement of the traffic sign recognition use case selected for AP5 to allow the development of modular audit criteria. However, the advantage of the highest potential to test the transferability comes with the challenge of a low similarity between both use cases. For the recommendation we assume that using a LiDAR-based use case is still manageable in the given timeframe. However, if it shows during the development of AP5 that the effort for the development of the initial audit criteria and toolbox is already higher than expected and it is too time-consuming to expand the development to an entirely different sensor suite, we list camera-based use cases in Chapter 4.4.2.2 which serve as an alternative to still comply with the limited duration of this project. As mentioned at the initial discussion in Chapter 4.4.2 the use case recommendation is revisited as part of work package seven after work package five is concluded. At that point it can be better judged whether using a LiDAR-based use case is possible in the given timeframe of the project.

4.4.2.2 Alternative Use Cases

After we derived the main recommendation for a second use case, we now list alternatives to this recommendation. As discussed previously, the main risk of the recommendation is that it is too time-consuming to expand or adjust the toolbox and the developed audit criteria to a different sensor suite. Therefore, we only list camera-based use cases as alternative, for which it is easier to expand the results from AP5. Hence, the most straightforward alternative is to use the camera-based vehicle detection use case instead of the LiDAR-based use case. This use case is most similar to the use case selected for AP5, which should allow the easiest and fastest adaption of the toolbox. However, this also has the lowest potential to test the transferability of the developed audit criteria due to the high similarity between the use cases. Therefore, our main alternative recommendations are to use either the camera-based lane marking detection use case or the camera-based drivable area detection use case. Both require different perception techniques based on segmentation, which increases the potential to test the transferability of the audit criteria to different use cases. These two use cases are rather similar, meaning it is difficult to provide a recommendation for the ordering of the alternative use cases. It could be argued that detecting only lane markings is easier than detecting the entire drivable space, however the applicability of attacks seems somewhat easier when the entire drivable space is considered. Hence, the final ordering of alternative recommendations is only a suggestion and needs to be discussed depending on the acquired knowledge and performed implementations during AP5 when the case arises that an alternative use case needs to be selected for AP7. Summarizing, the suggested ordering of the alternatively recommended use cases is the following:

1. Camera-based drivable area detection using segmentation
2. Camera-based lane marking detection using segmentation
3. Camera-based vehicle detection using object detection

5 Planning and exemplary Creation of Toolbox (AP5)

This chapter is the final report of work package five “Planung und exemplarische Erstellung Toolbox (Toolchain & Werkzeuge)“ of the project “Vorbereitung der Erprobung und Weiterentwicklung von Anforderungen an KI-Systeme anhand praktischer Use-Cases im Bereich Mobilität (AIMobilityAuditPrep)”. Hence, it contains the results of the toolbox planning and supporting documentation for the initial development of the toolchain and audit tools. At first, we describe a generic toolchain which can be used to develop, simulate and test autonomous driving (AD) or advanced driver assistance system (ADAS) functionalities based on artificial intelligence (AI) and especially deep neural networks (DNNs). Afterwards, we present the exemplary implementation of the toolchain focusing on the selected traffic sign recognition (TSR) use case from Chapter 4.4.1 and discuss the selected interfaces and tools. Finally, based on the selected use case and the developed list of audit criteria in the report of AP3 in Chapter 3 we discuss which safety and security requirements are selected for the use case and how they are evaluated with the help of the toolbox to aid an audit of the system.

As a general disclaimer, it is important to point out that the concrete implementation of the toolbox is only performed in an exemplary manner. It should enable first experiments and allow assessing the suitability and applicability of the proposed audit criteria. This represents a starting point to assess the feasibility of potential audits but is by no means complete. The given resources and timeframe of the project do not allow for an extensive implementation of an audit toolbox, thus only exemplary components are implemented.

5.1 Implementation Concepts

In this chapter we describe the principal concepts behind the toolchain for training a DNN-based AD/ADAS system and the toolbox for auditing such trained systems. These concepts form the basis of the exemplary implementation which is described later in Chapter 4.3 and Chapter 5.3.

5.1.1 Toolchain

Testing the proposed requirements from (2) for auditing AI-based AD/ADAS systems requires that a toolchain exists which allows creating suitable AD/ADAS models. This toolchain can then be used to develop an exemplary system for which the proposed requirements are evaluated. Therefore, the toolchain must conceptually be capable of developing AD/ADAS systems for the different use cases described in (3). Hence, it must be planned generically to allow the application to different use cases which can have different prerequisites.

5.1.1.1 Generic

In Figure 55 an overview of the important elements of such a generic toolchain is shown. The development starts by the general planning of the AD/ADAS system, which is followed by the data collection and preprocessing phase. This is required for DNN-based systems, which rely on high-quality data. Next, the development of the algorithm is performed, which includes the training of DNN models as well as the integration into the overall system with all other hardware components. In this development step the evaluation of the quality of the developed functionality is performed based on a predefined test dataset, which is the standard best practice for developing DNN-based systems. Typically, this quality assessment only includes evaluating the standard performance with respect to some KPI, like accuracy or intersection over union, on this the dataset. However, it can also include more sophisticated evaluations like assessing the robustness on variations of the test data distribution, analyzing whether meaningful data features are used for a prediction or evaluating the IT-Security against an adversary. When all relevant performance aspects on this dataset suffice, the next development phase is reached, which is a more extensive validation based on simulation and importantly also based on physical tests with the real system and complete hardware connections. Once the system passes these validation steps, it is cleared for deployment from the development perspective. During deployment the system is then continuously monitored to ensure the

performance stays inside the assured quality. In practice the described process is often not straightforward and at different steps multiple loops are performed. For example, during the "Test Data Evaluation" component it might turn out that the performance of the system does not reach the required performance even after multiple improvements of the algorithm. Then it might be required to take a step back and collect more or better data or even go back to the beginning to adjust the basic system design.

In parallel to the development steps, audits must be performed at different stages or after some development steps are finished. These audits must ensure that the respective phase or result is according to the requirements from (2) or to requirements from documents issued by standardization bodies like the ISO. If some parts of the development or of the developed system do not fulfil the requirements, it is possible that this development step must be repeated until the audit is successful. Only then the system is cleared for deployment.

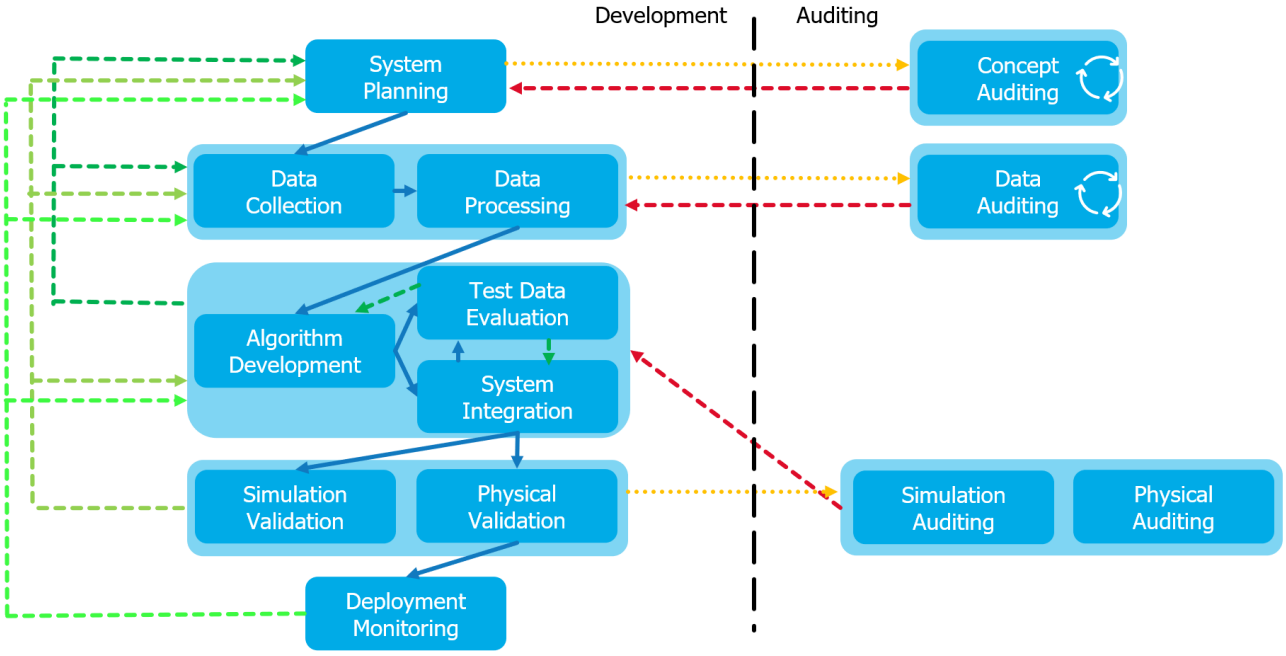


Figure 55 Overview of the generic toolchain for developing a DNN-based AD/ADAS system

5.1.1.2 Project-Specific

The generic toolchain from Figure 55 supports all required steps and tasks at the different stages in the development. It is important to consider the toolchain from this very generic viewpoint since this allows planning the project-specific implementation in accordance with the generic requirements of different use cases. However, due to the limited time of the project it is only possible to implement an exemplary generic toolchain. Therefore, in Figure 56 we show a simplified version of the toolchain which is possible to be implemented in the given timeframe.

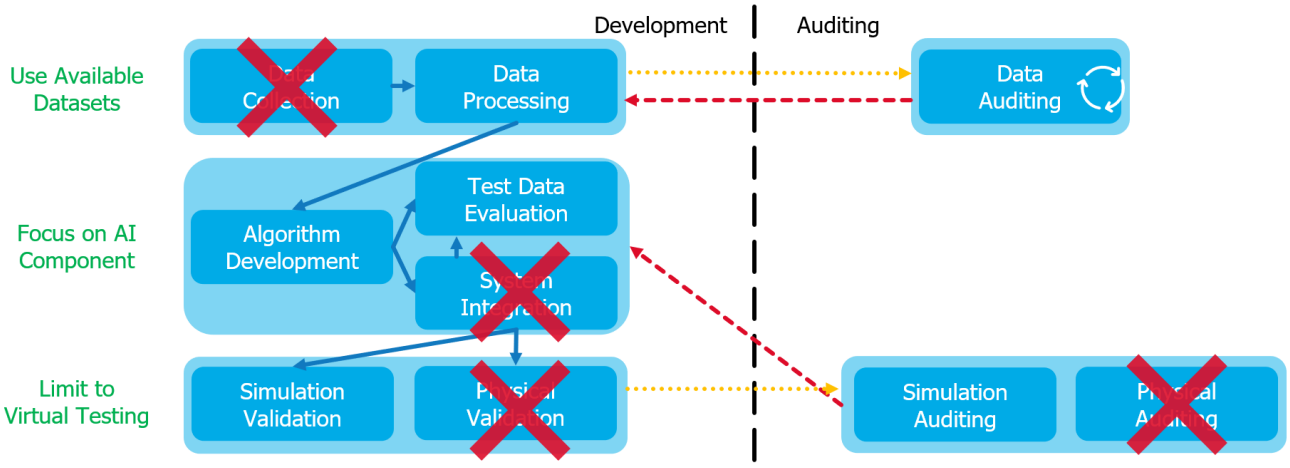


Figure 56 Overview of the toolchain components which are implemented in this AP

In comparison to Figure 55 it is notable that the planning and monitoring stages are dropped. The planning stage is dropped because the planning of the AD/ADAS system is already performed in (3) where the use case is selected for which the toolchain and audit requirements are implemented in an exemplary manner. The monitoring stage is dropped because in this AP the goal is not to deploy a system on public roads but only to show in an exemplary manner the applicability of the toolchain and audit requirements for the selected use case. In the data stage we limit ourselves to using available datasets in this AP. Collecting own data is infeasible given the time and resource constraints. Similarly, we only focus on developing an AI model as AD/ADAS component and do not consider the integration into the entire system with the interplay of different hardware components. It is only feasible to consider the system functionality replicated in software and running virtually. Thus, the final validation and auditing is also only performed in simulation since we do not capture any real data and do not have a physical replica of the AD/ADAS system. The steps that are still included in Figure 56 allow verifying the concept of the toolchain for an exemplary use case and assessing whether all important aspects of development and auditing are supported by the toolchain.

5.1.2 Toolbox

In addition to the toolchain which allows to develop DNN-based AD/ADAS systems it is also relevant to develop the audit tools that check whether certain requirements are fulfilled by the developed system. Similarly to the approach in Chapter 5.1.1 we first discuss a general concept for different auditing tools and then focus on specific examples and discussing what is feasible to implement in this AP.

5.1.2.1 Overview

In Figure 57 a generic overview of a toolbox component is shown. This component can be any requirement from (2) which should be evaluated for a given AD/ADAS system. Thus, the input to the evaluation or audit tool is represented by the model to audit and the associated representative data. In addition, the tool gets a settings file as input, which contains all important information to perform the audit. As output the tool generates a report based on defined metrics that summarize each audit given the requirement that should be tested. Also, all failures are listed explicitly and are described in a report in more detail.

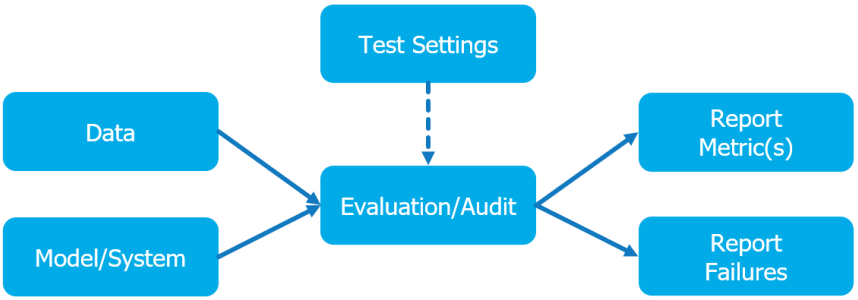


Figure 57 Overview of a generic toolbox component for auditing a DNN-based AD/ADAS system

5.1.2.2 Individual Components

After presenting the overview of a generic evaluation and audit tool we now discuss each component from Figure 57 individually. When suitable, we also present which parts of the individual components are implemented in this AP and where focus points lie.

5.1.2.2.1 Model/System

One of the inputs to the generic evaluation or audit toolbox component in Figure 57 is the model or system for which the audit should be performed. This component is further specified in Figure 58 for the case that an AI model is used as input. As discussed in Chapter 5.1.1.2, we only consider this case in this project and do not discuss the case where entire systems should be audited. In the considered case the model component gets the current data as input. The concrete input type depends on the actual use case but for the use cases considered in this project the input data is typically an image or a point cloud. Then, the model outputs the current prediction, which again depends on the use case and the type of task that should be solved. The output value can then be consumed in concrete implementations of a toolbox component. In addition to the direct input and output, the model has certain parameters or properties that can be set or queried. For example, this includes parameters that influence the behavior of the model like the state of buffers or memory cells. Also, model properties that are relevant for certain audit requirement tests can be queried. This can include whether the model is recurrent or behaves entirely deterministically.

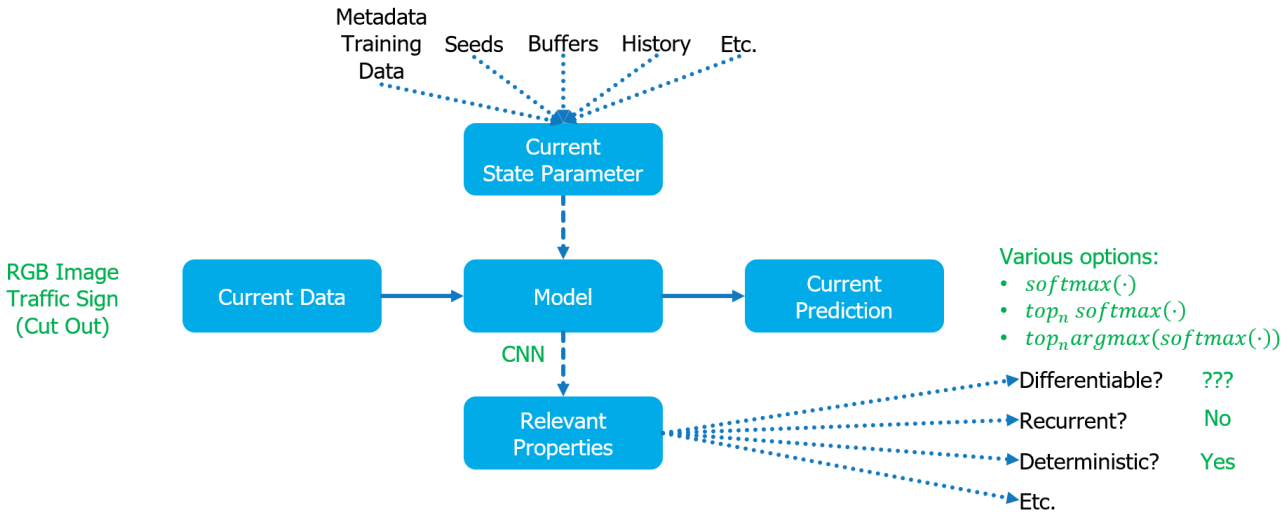


Figure 58 Overview of the model component

In the exemplary implementation in Chapter 4.3 and Chapter 5.3 we only consider a part of this generic model component similar to the approach in Chapter 5.1.1.2. Hence, we select specific values for some of the components of the model component. First, based on the selected TSR use case in (3) we only consider RGB images of traffic signs as our data. In addition, the traffic signs are assumed to be already cut out, meaning a bounding box detection is already performed and only the traffic sign with minimal background is used as input for the model in our considered use case. The model itself is always represented as a CNN without any

stochastic layers, meaning it is entirely deterministic and not recurrent. Regarding the differentiability different options are possible where the auditor either has white-box or black-box access to the model. Finally, the output of the model is typical for a classification problem which fits the selected use case. Hence, as default a probability vector over all available classes is returned by a softmax function. This can be further limited by only returning the top-n classes instead of the entire probability vector or by not returning any probabilities and only a class ranking.

5.1.2.2.2 Data

The second input to the generic toolbox component in Figure 57 is the data component. Here, it is important to point out that three different data sources are possible as indicated in Figure 55. First, it is possible to use a test dataset which is already collected and annotated. Second, it is possible that the data is generated online in a simulator which interacts with the toolbox component. Last, it is possible that a physical validation or audit is performed meaning the data is captured online in reality. All three different data sources are important and play a decisive role at different stages during the development or audit. Therefore, it is required that the interface of the data component with the generic toolbox component is comparable for all three data types. For a toolbox component it should make no difference where the data originates from, meaning the different data types should be formatted and structured in the same way. This ensures that each data source can be processed by a single function without the need for extensive reimplementation specific for a certain data source.

5.1.2.2.2.1 Offline Data Loader

Under this given constraint we show the structure of the data component in case test data is used in Figure 59. Here, the input is represented by the location where the test dataset is located. Then this is consumed by a data loader, which outputs pairs of data samples and the associated labels. Like the model component in Figure 58, the data component has different states and properties which can be set or queried. Most importantly it is possible to set certain conditions that the data loader must fulfil. Some examples include the size of the samples, the exact location where the sample was captured or the brightness in an image-based data sample. These conditions can for example be set by a toolbox component when it requires very specific data samples. Also, it is possible to get information about different properties from the data loader, which can be important for an audit requirement test. For example, this includes the position of the sensor in the vehicle or the frequency at which the data samples are collected.

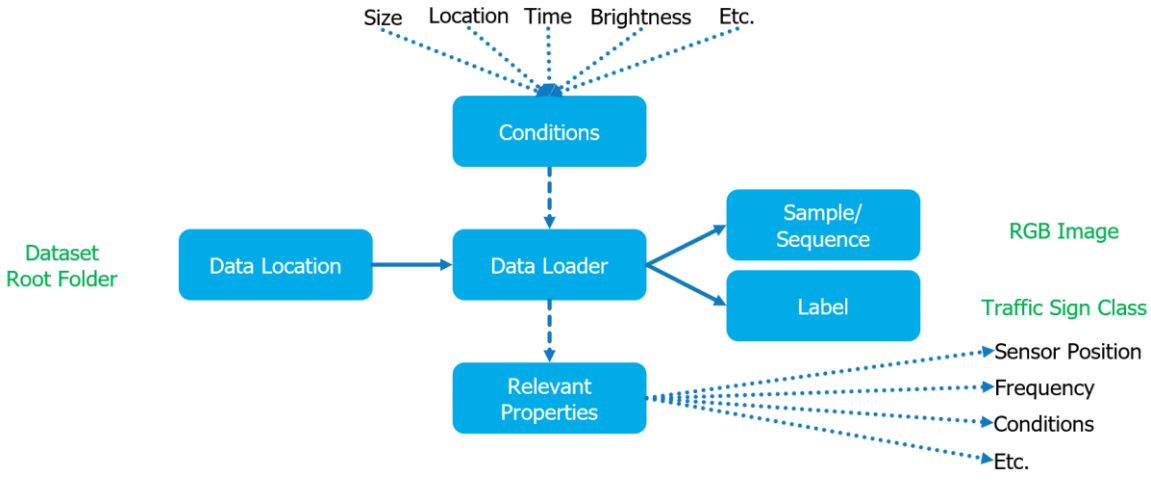


Figure 59 Overview of the test data loader component

In the exemplary implementation, the generic test data loader is again concretized and only specific parts are implemented for the selected exemplary TSR use case. The “Data Location” component is represented by a dataset root folder under which all images are stored using a split in the train, eval and test subsets. For this use case, the sample that is output is a single RGB image and the output label is the class of the traffic sign which is visible in the associated data sample.

5.1.2.2.2 Simulation Data Generator

After discussing the data component when using a test dataset, we now discuss the data component when simulated data is used. The resulting overview is given in Figure 60. Here, it is important to understand that simulated data can be generated in two different ways. First, it is possible to use a simulator which takes a scenario description and then generates new data samples online. The scenario description can include information like the sensor position, road course in the environment or the position of other traffic participants. Alternatively, it is possible to again use an already existing dataset and perform the simulation by performing various augmentations on already existing data samples. For example, in case of image data it is possible to simulate weather effects like rain or snow by augmentation of an existing image. The output of the simulated data generator is then identical to the output of the test data loader in Figure 59. This ensures that a toolbox component can use these data components interchangeably without the need to implement specific data processing methods. Additionally, the data generator has more conditions that can be set by a toolbox component. For example, it can be useful to set the current weather status or specify if a sensor fault should be simulated.

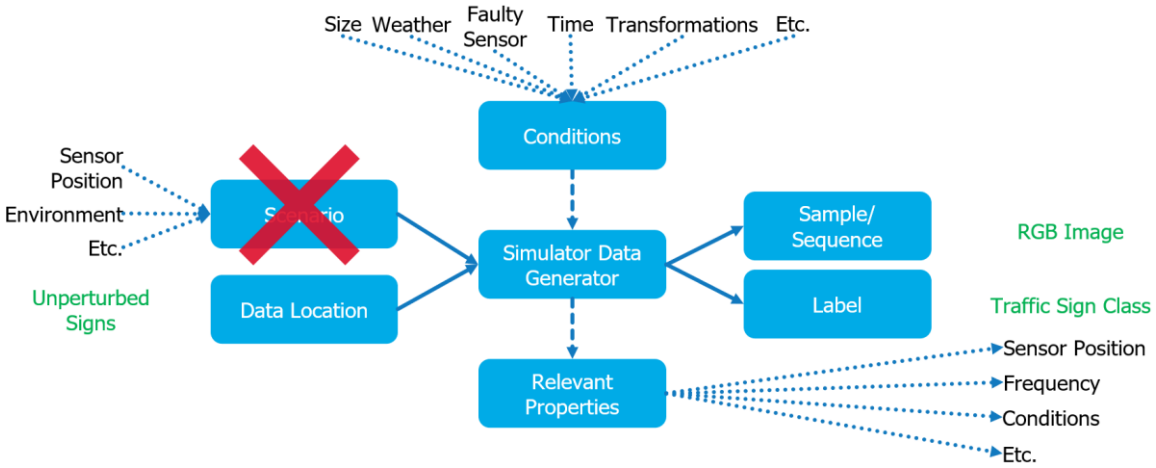


Figure 60 Overview of the simulation data generator

In this project we only consider the case when simulation is performed by augmenting already existing data samples. Setting up and integrating an entire simulator to generate new data samples online is out of scope of this project due to the given time and resource constraints. This can be added at a later stage in potential follow-up projects.

5.1.2.2.3 Sensor Data Generator

Lastly, it is possible that sensor data which is captured online in reality during test drives is used as input of a toolbox component. This is shown in Figure 61 but no exemplary implementation is described later in Chapter 4.3. As discussed in Chapter 5.1.1.2, performing physical tests is not feasible in this project and thus the overview of the sensor data generator is only mentioned for completeness. It can be used in potential follow-up projects to incorporate such a component in the entire toolchain. Similarly to the components in Figure 59 and Figure 60, it has the same output interface, which allows using existing toolbox components without the need to implement specific methods for data handling.

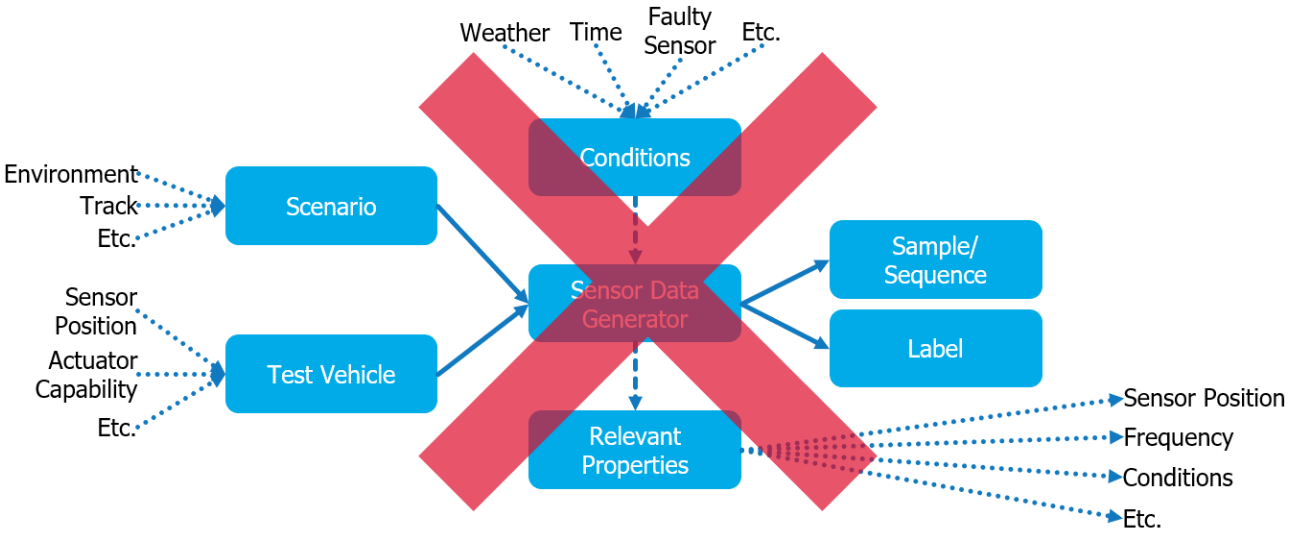


Figure 61 Overview of the sensor data generator

5.1.2.2.3 Test Settings

The last input component to the audit tool in Figure 57 is the "Test Settings" component. Here, all relevant settings for the execution of a respective audit requirement test are included, which allows reproducing the results of this test. In Figure 62 a basic example of such a setting configuration is shown. At the top-level different categories of requirements exist. These are then further specified by moving down in a tree-like structure. For example, in Figure 62 a security test is performed by using an adversarial attack. For this attack the specific parameters must be set, like the "Attack Type" and the "Attack Settings", which contains specific settings for this attack. In a similar fashion, it is possible to build configuration settings for all audit requirements from (2) which focus on testing a trained AD/ADAS system.

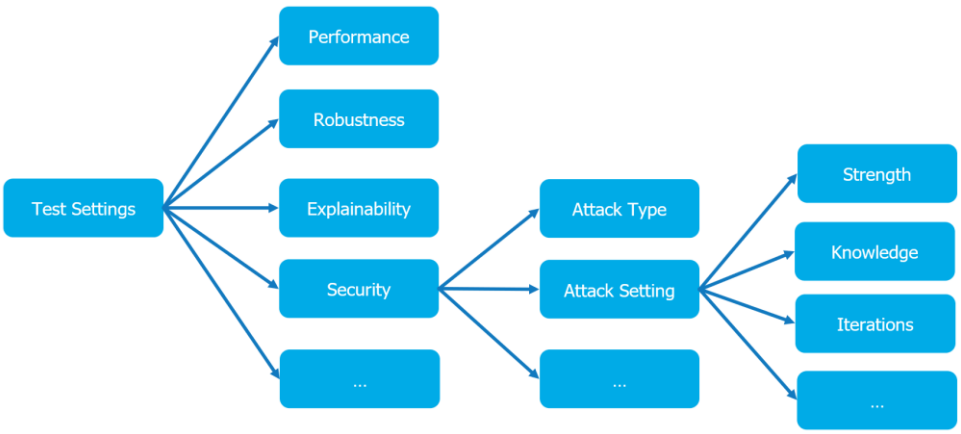


Figure 62 Overview of the setting component

5.1.2.2.4 Report Metrics

The output of the audit tool is represented by a report that summarizes the most important metrics of each requirement test. This report is shown in Figure 63 and is structured in a similar way to the settings configuration in Figure 62. Again, it contains the same tree-like structure having the same high-level categories at the top. Then, for each audit requirement test suitable KPIs are defined, which are listed in the report. The security test example from Figure 62 is again chosen and exemplary KPIs are listed for the case where a test based on adversarial attacks is performed. Here, different success rates of the attack or the average number of iterations until the attack is first successful are shown as exemplary metrics.

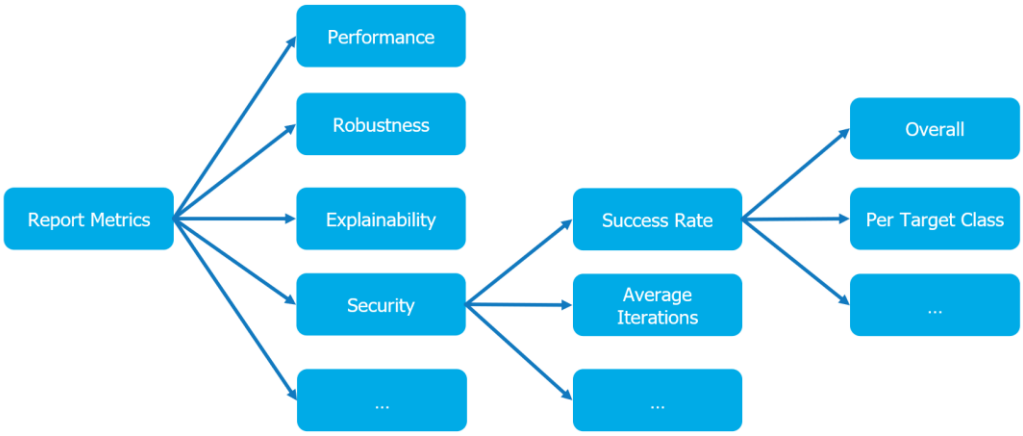


Figure 63 Overview of the report component

5.1.3 Strategies for Comparison of Simulation and Reality

As discussed in Chapter 5.1.2.2.2, it is possible to use data samples from different data sources in the toolchain. This raises the question how it can be assured that the results generated using simulated data are as closely comparable as possible to results generated using real data. To provide answers, in the following we discuss comparison strategies for analyzing the transferability of results from simulation to reality.

In Figure 64 an overview of the different comparison strategies is shown. It is important to note that three different options exist to represent the system functionality. First, it is possible to use a real test vehicle, which includes all system components and is the most realistic system representation. On the other side, there is the option to use a SIL, which represents the functionality of the system but does not include the hardware components of the system. In between, it is possible to use a HIL, which includes all hardware components that are relevant for the direct functionality of the system. Hence, it simulates the behavior of the real system based on captured or simulated data samples: These are either replayed and captured by the sensors in the HIL or are directly injected to the data processing module in the HIL by bypassing the sensor hardware. However, it does not include other vehicle components like actuators. Both the HIL and the SIL can be used on real and simulated data, while the physical test vehicle can only capture the data it observes in its environment.

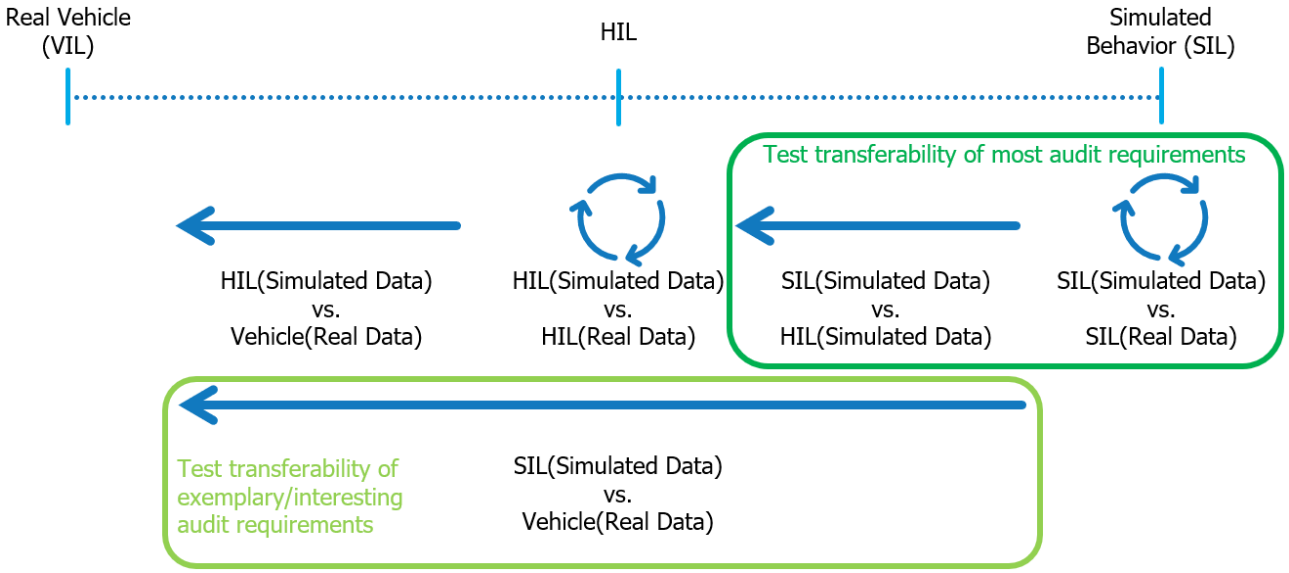


Figure 64 Overview of different options to analyze the transferability of results from simulation to reality

To assess the transferability of results from the simulated world to the physical world it is required to compare the results for audits performed using simulated data samples and for audits performed using physically captured data samples in a structured way. Here, it is most practical to perform most audit requirement tests

using the AD/ADAS system as a SIL. This enables the tightest integration in the remaining toolchain and allows for different audit requirement tests in a quick succession. The fast assessment of the proposed audit criteria allows for an efficient and continuous improvement of the audit criteria. After these initial tests are performed, the next step is to test interesting audit requirements on a HIL and directly compare the results with the tests on the SIL. This should be done for tests where it is identified in the SIL experiments that the audit result is less stable or is more strongly influenced by minor factors of the data samples. For such tests it is more questionable whether the results transfer well from simulated data to real-world data. As the last step, it is required to perform some tests using a real vehicle and perform actual test drives. Only this enables to assess the actual transferability in the most meaningful way. However, it is also the least practical in comparison as real test drives require more time and resource expenses. Therefore, this final comparison should only be done for a strict selection of audit requirement tests to find a balance between using the available resources well and achieving the most meaningful comparison results. As discussed before, it is most interesting to perform this final comparison for audit requirement tests where the results using the SIL or HIL show some variance and are most likely to show a difference when comparing the simulated and real data samples.

5.2 Exemplary Toolchain Implementation

After we presented the generic concepts of the toolchain and toolbox in Chapter 5.1, in the following we describe the exemplary implementation of the toolchain. This includes the selection of the most important software libraries and a discussion on the used datasets and models to develop the exemplary TSR system.

5.2.1 Dataset

As discussed in Chapter 5.1.1.2 we do not perform our own data collection and instead use an already existing dataset. In Table 52 we list an overview of available datasets that focus on the task of traffic sign recognition or detection. For each dataset, we list the country where the dataset is collected, the number of different traffic sign classes included and the number of images in a dataset. This allows comparing the datasets with each other based on the most relevant properties and enables the selection of the most fitting dataset for this project.

Table 52 Overview of some available datasets for the TSR use case selected in AP4 (3)

Name	*Country*	*Number Classes*	*Number Images*	*Link*
German Traffic Sign Recognition Benchmark (GTSRB)	Germany	43	> 50k	https://benchmark.ini.rub.de/gtsrb_news.html
KUL Traffic Sign Dataset	Belgium	62	> 7k	http://people.ee.ethz.ch/~timofter/traffic_signs/
Challenging Unreal and Real Environments for Traffic Sign Detection	Belgium	14	> 2m (video sequences and partly synthetic)	https://github.com/olivesgatech/CURE-TSD
Traffic Signs Dataset	Swedish	19	> 3k	https://www.cvl.isy.liu.se/research/datasets/traffic-signs-dataset/
Mapping and Assessing the State of Traffic Infrastructure	Croatia	97	> 10k	http://www.zemris.fer.hr/~ssegvic/mastif/datasets.shtml
LISA Traffic Sign Dataset	USA	47	> 7k	https://git-disl.github.io/GTDLBench/datasets/lisa_traffic_sign_dataset/

Name	*Country*	*Number Classes*	*Number Images*	*Link*
Chinese Traffic Sign Database	China	58	> 6k	http://www.nlpr.ia.ac.cn/pal/trafficdata/recognition.html
Tsinghua-Tencent 100k	China	221	> 30k	https://cg.cs.tsinghua.edu.cn/traffic-sign/

Based on the presented dataset comparison we select the GTSRB dataset for the exemplary training of an AD/ADAS system for this project. This selection comes for numerous reasons:

1. The GTSRB dataset has the highest number of unique and real images. This is preferable for training DNNs, which require sufficient and good data to achieve strong performances.
2. This dataset is often used in the literature when results are tested for the TSR use case. For example, this includes the test of physical adversarial attacks or the OOD performance of confidence estimation methods.
3. The dataset is based on traffic signs captured in Germany. This means that results on this dataset can most easily be transferred to physical tests that are performed in a potential follow-up project. Since this project is executed in Germany by German partners, German traffic signs are more easily available and the road conditions are more similar. This should minimize the preparation effort for real physical tests.

5.2.2 DNN Models

After selecting the dataset that is used for the exemplary implementation of the toolchain, we now discuss the CNN models that are used as the basis of the TSR system. To minimize the implementation effort and at the same time maximize the generalizability of the toolchain we use publicly available implementations[34] of common CNN backbones. These are then fine-tuned on the GTSRB dataset to learn the specific features of traffic signs and function as AI models for the TSR use case. We use two different exemplary implementations to ensure that the results of the audit requirements tests are not an effect existing for a single DNN architecture. Concretely, we use the ResNet-18 architecture from (292) and the AlexNet architecture from (293). ResNet-18 is selected because this architecture is one of the most successful architectures in DNN history and is often used in literature as a sensible baseline independent of the concrete task and use case. In addition, we select AlexNet because it represents a standard CNN architecture without the use of skip-connections and batch normalization layers, which are both used in the ResNet-18 architecture. Experimenting with two significantly different DNN architectures allows assessing the transferability of the audit requirements to different DNN types.

5.2.3 Toolchain

For the exemplary implementation of the toolchain, we use Python[35] as the main programming language. Its nature of being a high-level scripting language allows a rapid prototyping and the highest possible coverage of implemented components. At the same time, Python is used almost exclusively in research publications in the AI and ML community and has the most open-source tools and utilities available. Again, this should allow for the highest number of implemented components in the toolchain because already existing implementations can be used. This also has the advantage that the resulting toolchain is mainly based on open-source libraries, which should allow making most parts of the implemented toolchain publicly available. Also, most research papers provide code that is written in Python, which allows the easiest integration of recently discovered methods for security, robustness, explainability, etc.

[34] https://pytorch.org/vision/0.8/models.html
[35] https://www.python.org/

Following best practices for program code versioning we use the git[36] tool, which enables the collaboration of different parties and provides a clean version history of the toolchain. Additionally, we follow coding standards like PEP8[37] or the clean code principle from (294) whenever possible.

5.2.3.1 DNN-related Tools

A large part of the toolchain is the handling and training of DNNs. Here, multiple different open-source frameworks exist which all provide specific utilities for this task. We select PyTorch[38] as the main library used for developing DNNs. Concretely, we use the fastai[39] library which allows training DNNs for different use cases with minimal overhead. Since the focus of this project lies on testing the suitability of different audit requirements, it is preferable to use a simple library that removes most of the need for boilerplate code for DNN training while still enabling an easy customization of the training process when needed.

To ensure that the toolchain can also be used with other DL libraries we use the ONNX[40] format to exchange DNNs between different libraries when required. On the one hand, this allows training DNNs with a different library and still use the implemented audit tools in the toolbox. At the same time, this also allows implementing audit tools using a different DL library than PyTorch. This can be useful when some tools are only available for a specific DL library and similar tools are not available for PyTorch. Then, the associated audit tool can be implemented using the other library and the tool can still be incorporated in the overall toolchain. Nevertheless, in this project all audit tools and toolchain components are implemented using PyTorch for simplicity and consistency.

To track experiments for training DNNs we use the MLFlow[41] library. This allows saving and registering trained DNN models, tracking the hyperparameters of different runs for reproducibility or comparing the results of different runs based on defined KPIs. An example for the provided interface to visualize the results of different training runs is given in Figure 65. The image shows the main interface where all different experiments are listed which were logged using MLFLow. In this case it shows the experiments on the GTSRB dataset and the two DNNs selected in Chapter 5.2.2. A more detailed view on different experiments is later given in Chapter 5.3.2.5 and Figure 71.

[36] https://git-scm.com/
[37] https://peps.python.org/pep-0008/
[38] https://pytorch.org/
[39] https://www.fast.ai/
[40] https://onnx.ai/
[41] https://mlflow.org/

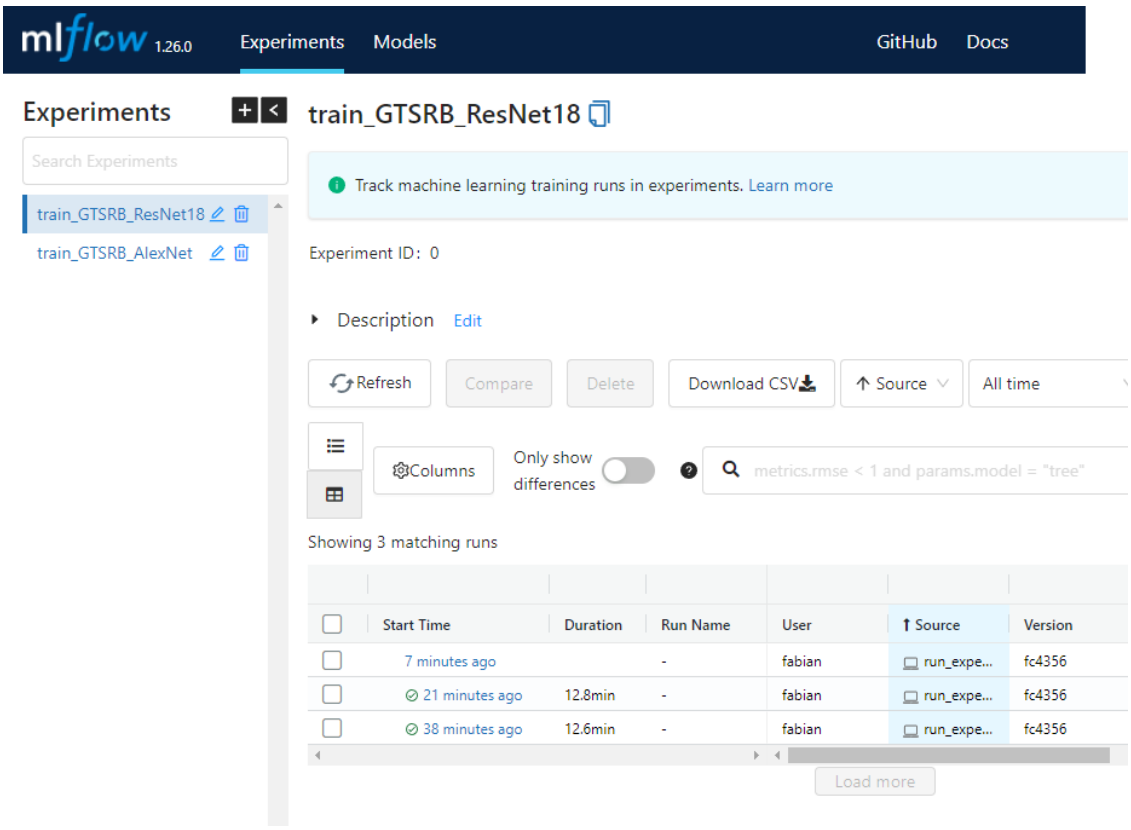


Figure 65 Exemplary excerpt from the experiment and model tracking using MLFlow

5.2.3.2 Simulation-related Tools

As discussed in Chapter 5.1.2.2.2 in this AP we only consider simulation by augmenting already existing images or images of ideal traffic signs that are inserted on a real background image. The integration of a real simulator for driving scenarios like CARLA[42] is possible in a potential follow-up project. For augmentations we use the albumentations[43] library, which provides a rich set of fast augmentations that can be used for different machine learning tasks. The use of this library is not limited to classification tasks, instead albumentations ships with built-in support for segmentation and detection tasks for most provided augmentations. This allows using the same augmentation library and the implemented toolchain components also for other use cases than the TSR use case. In Figure 66 some examples are shown for standard image augmentations that are possible using the albumentations library. Additionally, in Figure 67 further augmentations are shown which focus on the simulation of different weather conditions, which is especially relevant for AD/ADAS.

[42] https://carla.org/
[43] https://albumentations.ai/

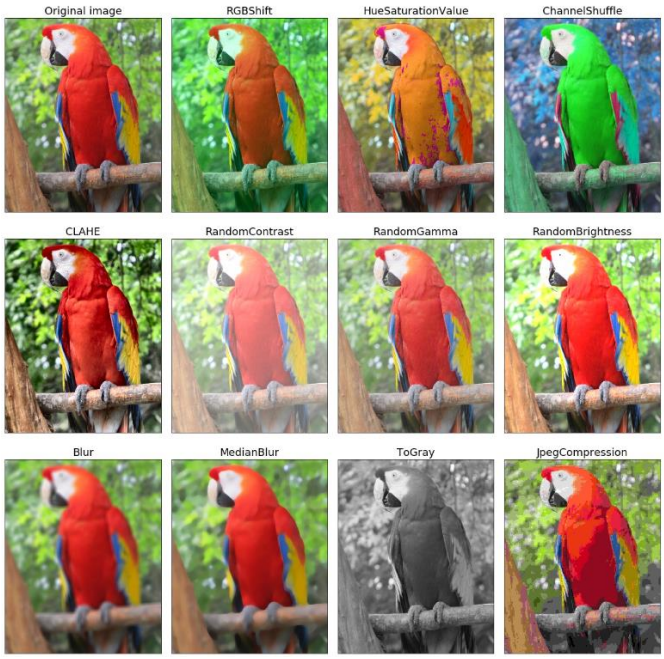


Figure 66 Exemplary overview of standard image augmentations from https://pypi.org/project/albumentations/

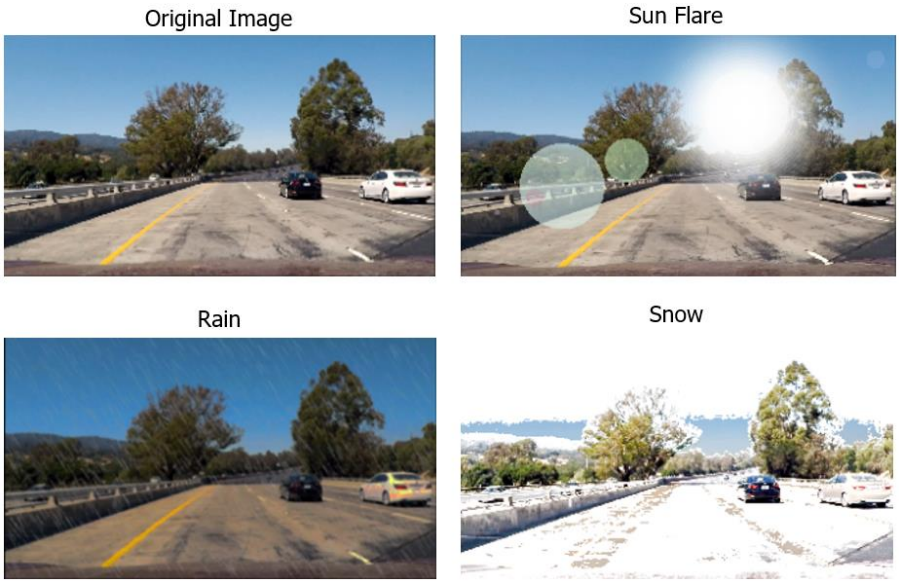


Figure 67 Exemplary overview of weather image augmentations adapted from https://github.com/UjjwalSaxena/Automold--Road-Augmentation-Library

5.2.3.3 Quality-related Tools

After discussing different tools and utilities used for the technical development of the toolchain, we now shortly describe tools which are used to assure a high quality of the toolchain. First, this includes a tool to regularly test the correctness and quality of the toolchain code. For this purpose, we use the pytest[44] library, which is a commonly used tool for testing Python code. We choose this library instead of the built-in testing library unittest[45] because it has a richer feature set and can easier be integrated and expanded. In Figure 68 an example is shown where a test run is performed using the pytest test tool on the implemented tests for different parts of the toolchain. For each test it is possible to specify under which conditions the test is executed and in Figure 68 28 tests are skipped as demonstration.

[44] https://docs.pytest.org/en/7.1.x/
[45] https://docs.python.org/3/library/unittest.html

```
(AIMobilityAudit)                  AIMobilityAudit> python -m pytest .\test\src\
========================================= test session starts =========================================
platform win32 -- Python 3.8.13, pytest-7.1.1, pluggy-1.0.0
rootdir:              AIMobilityAudit\test\src, configfile: pytest.ini
plugins: mock-3.7.0
collected 110 items

test\src\integration\test_run_experiment.py ..sss                                          [  4%]
test\src\integration\test_tools\test_trainer.py ssssssssssss                               [ 16%]
test\src\unit\test_config.py ....                                                          [ 20%]
test\src\unit\test_dataset.py ..............................ssssssssssss                   [ 58%]
test\src\unit\test_models.py ..............                                                [ 70%]
test\src\unit\test_run_experiments.py ......                                               [ 76%]
test\src\unit\test_utils.py ........................                                       [100%]

========================================= warnings summary ==========================================
unit/test_utils.py::TestUtilsGeneral::test_download_and_unzip
                                   AIMobilityAudit\lib\site-packages\urllib3\connectionpool.py:1043: Ins
ecureRequestWarning: Unverified HTTPS request is being made to host 'sid.erda.dk'. Adding certificate ve
rification is strongly advised. See: https://urllib3.readthedocs.io/en/1.26.x/advanced-usage.html#ssl-wa
rnings
    warnings.warn(

-- Docs: https://docs.pytest.org/en/stable/how-to/capture-warnings.html
============================ 82 passed, 28 skipped, 1 warning in 18.79s =============================
```

Figure 68 Exemplary exempt from the implemented test session

Finally, we use the Sphinx[46] library to generate the accompanying documentation of the program code. The Sphinx library is used because it is the most popular choice for documenting Python code, has an optimized integration and features the most options for the resulting documentation. In Figure 69 two exemplary excerpts from the generated documentation are shown. These are generated automatically by Sphinx from properly formatted docstrings that are written in the program code. This allows generating a high-quality documentation with as little effort as possible, because some information like parameter names or types are automatically extracted by Sphinx from the code and do not require any extra effort by the developer.

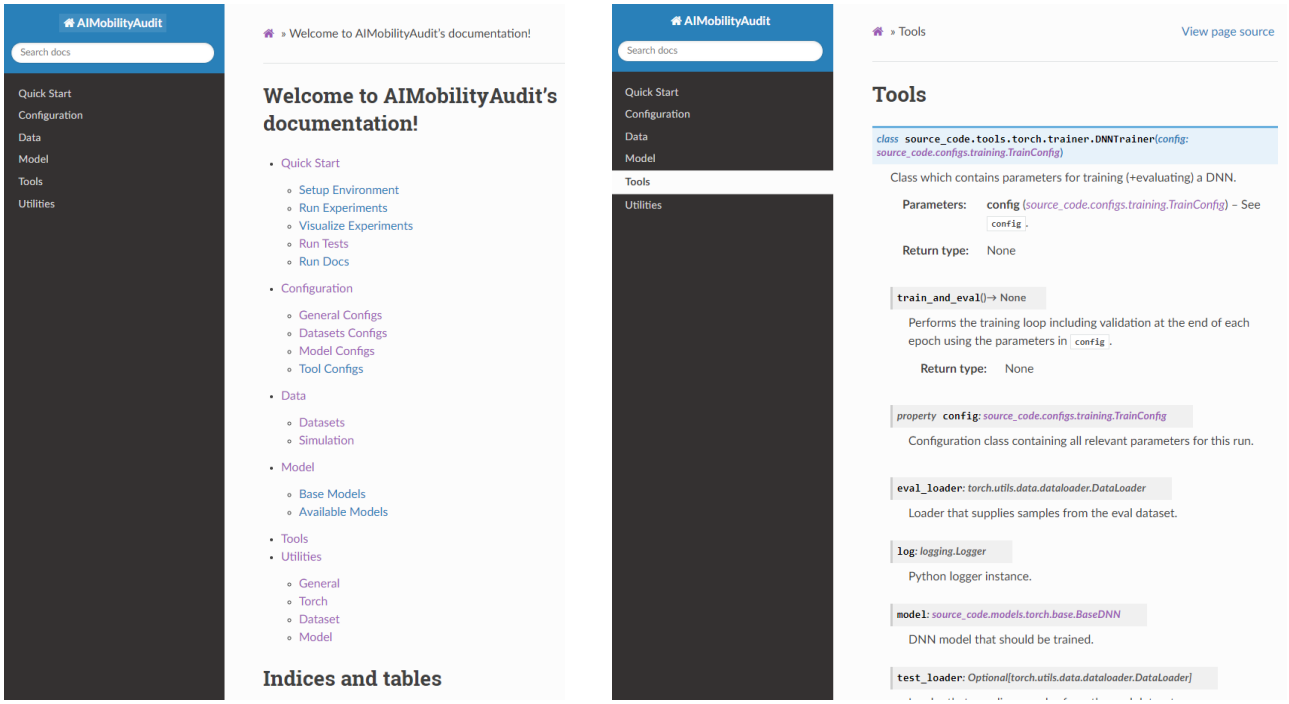


Figure 69 Exemplary excerpts from the code documentation

[46] https://www.sphinx-doc.org/en/master/

5.3 Implementation of Safety/Security Requirements

5.3.1 Selection of Requirements

The following chapter gives an overview of the safety and security requirements selected for auditing the traffic sign use case. As discussed in the AP3 Report (2), safety and security requirements are to be implemented within an ADAS or AD system based on the risk level of the application.

In the case of the traffic sign recognition system used as assistance, accidents and failures are highly controllable and they entail low severity of injuries. Therefore, we assume an ASIL A/low risk for the requirement selection.

5.3.1.1 Generic requirements for the Entire System

Table 53 All ASIL A/low risk generic requirements[7] towards the entire system of a traffic sign recognition system. Requirements highlighted in yellow are highly recommended and therefore mandatory for this use case. Further recommended requirements that are implemented exemplarily are highlighted in red.

Requirement			*Recommendation*
ID	*Description*	*Type*	*ASIL A/ Low*
1	The environmental context shall correspond to the operational design domain (ODD).	ASIL	+
2	The communication, interfaces, signals, etc. between different components shall be coordinated.	ASIL	+
3	The sensor setup shall be similar to the development/training setup.	Additional	+
4	The requirements for AI subsystems shall apply to the entire system (if applicable).	Additional	++
5	The adequate performance shall be guaranteed for a certain timeframe after initial deployment.	ASIL	+
6	The performance on key performance indicators (KPIs) shall be as high as possible	Additional	+
7	The performance shall be compliant to the allowed worst-case error.	ASIL	++
8	The performance shall be reproducible in the real environment for operation.	ASIL	+
9	The feedback of the system shall be tracked while in operation.	ASIL	o
10	The performance shall be corrected when critical errors occur reproducibly after deployment.	ASIL	+
11	The system state shall be tracked in a reproducible way while in operation.	Additional	+
12	The architectural design shall be described explicitly.	ASIL	++
13	The quality & trustworthiness for developers shall be assessed.	ASIL	o
14	The development process shall be tracked.	Additional	+

5.3.1.2 Generic requirements for the AI Subsystem

Table 54 All ASIL A/low risk generic requirements[9] towards the AI subsystems of a traffic sign recognition system. Requirements highlighted in yellow are highly recommended and therefore mandatory for this use case. Further recommended requirements that are implemented exemplarily are highlighted in red.

Requirement			*Recommendation*
ID	*Description*	*Type*	*ASIL A/ Low*
15	The AI model shall be implemented using mitigation strategies against robustness threats.	ASIL	+
16	The AI model shall be verified with formal robustness verification techniques.	ASIL	O
17	The robustness of the AI model shall be verified with empirical robustness estimation techniques.	ASIL	+
18	The AI model shall be tested against out-of-distribution data.	ASIL	++
19	Test cases at the boundary values of the input of the AI model shall be derived.	ASIL	+
20	Test cases based on corner cases of the AI model shall be derived.	ASIL	+
21	Test cases shall be derived through error guessing based on knowledge and experience of the system.	ASIL	+
22	The AI model shall be tested against possible robustness threats.	Additional	+
23	The source of the datasets shall be traceable.	Additional	+
24	The source of the dataset shall be verified.	Additional	O
25	The datasets shall have adequate coverage of the operational input domain.	Additional	+
26	The datasets shall be verified against the safety requirements.	Additional	O
27	The uncertainty of the datasets shall be analyzed and quantified.	Additional	O
28	The datasets used for training, testing and evaluation shall not contain any errors.	Additional	++
29	The training, test and evaluation datasets shall have sufficient size.	Additional	+
30	The training, test and evaluation datasets shall be independent from each other.	Additional	++
31	The training, test and evaluation datasets shall be prepared in an adequate way.	Additional	+
32	The requirements shall be analyzed to derive test cases for interpretable model decisions.	ASIL	++
33	The model's decisions shall be explained to aid the comparison between the modelling of the system and the trained model.	ASIL	++
34	The model's decisions shall be explained to check if the requirements of the system are met.	ASIL	+
35	The model's decision at the boundary values shall be explained.	Additional	+
36	The model's decision on corner cases shall be explained.	Additional	+
37	The model's decision on failed tests shall be explained.	Additional	+

Requirement			*Recommendation*
38	The least complex model architecture needed to solve the task shall be chosen.	Additional	+
39	A model architecture shall be chosen to maximize the interpretability of decisions.	Additional	O
40	The SW unit design shall be described explicitly.	ASIL	++
41	The dataset & model shall be versioned.	Additional	o
42	Standardized methods to record characteristics of datasets, AI models and key processes shall exist and be followed	Additional	o
43	The labelling process of the dataset shall be documented and tracked.	Additional	o
44	The input shall be monitored and checked before it is given into the AI model.	ASIL	+
45	The plausibility of the AI model's output shall be checked.	ASIL	+
46	The AI model shall be monitored during the program execution.	ASIL	o
47	Errors of the model shall be logged.	ASIL	+
48	Damaged or manipulated inputs shall be corrected when it is safely possible.	ASIL	o
49	Fail-safe methods shall be implemented to mitigate entire system failures.	ASIL	+
50	Parallel redundant AI models shall be implemented.	ASIL	o

Since the chosen use case is classified as ASIL A/low risk, the following nine highly recommended requirements are mandatory during the development of this project.

Table 55 Summary of the highly recommended requirements[7] for this use case

Requirement			*Recommendation*
ID	*Description*	*Type*	*ASIL A/ Low*
4	The requirements for AI subsystems shall apply to the entire system (if applicable).	Additional	++
7	The performance shall be compliant to the allowed worst-case error.	ASIL	++
12	The architectural design shall be described explicitly.	ASIL	++
18	The AI model shall be tested against out-of-distribution data.	ASIL	++
28	The datasets used for training, testing and evaluation shall not contain any errors.	Additional	++
30	The training, test and evaluation datasets shall be independent from each other.	Additional	++
32	The requirements shall be analyzed to derive test cases for interpretable model decisions.	ASIL	++
33	The model's decisions shall be explained to aid the comparison between the modelling of the system and the trained model.	ASIL	++
40	The SW unit design shall be described explicitly.	ASIL	++

To obtain further insight into the use of the generic requirements for the system and its auditability, we also selected the following recommended requirements to be demonstrated on a schematic level. This will provide guidance on how to determine and derive specific values such as thresholds or boundaries.

Table 56 Summary of recommended requirements[7] implemented exemplarily for this use case

Requirement			*Recommendation*
ID	*Description*	*Type*	*ASIL A/ Low*
6	The performance on key performance indicators (KPIs) shall be as high as possible	Additional	+
14	The development process shall be tracked.	Additional	+
19	Test cases at the boundary values of the input of the AI model shall be derived.	ASIL	+
20	Test cases based on corner cases of the AI model shall be derived.	ASIL	+

5.3.2 Implementation of Requirements

This chapter describes the implementation of the requirements. The structure starts with the determined test parameters and a corresponding justifying statement. Subsequently, the audit procedure for this requirement in dependence on the considered use case is introduced. The "Implementation" section presents the evidence and findings, which specify to which extent the requirement is covered. A final section concludes the evaluation and gives a final verdict based on the findings from the previous section.

Due to the evaluation of a constructed use case for demonstration purposes, the nature of some of the requirements only allow a schematic evaluation. A restricting factor is, for example, the absence of an overall system, in which an equivalent use case is embedded in real-world applications. Another question impeding the implementation is the definition of meaningful test and evaluation parameters specified for the considered use case without prior knowledge of equivalent projects, available standardization or legal requirements.

5.3.2.1 Requirement 4

Requirement			*Recommendation*
ID	*Description*	*Type*	*ASIL A/ Low*
4	The requirements for AI subsystems shall apply to the entire system (if applicable).	Additional	++

Specification of the parameters and rationale

No parameters have to be set for this requirement.

Audit procedure

One of the most basic requirements from the functional safety point of view is to consider any risk and threat analysis of the components and sub-systems with regard to the entire system. More precisely, any risk consideration of the system will be inherited to the sub-systems and components and vice versa as illustrated in Figure 70. The aim is to have a consistent tracing of any malfunction or flaw which affects the safety and security claim of the system.

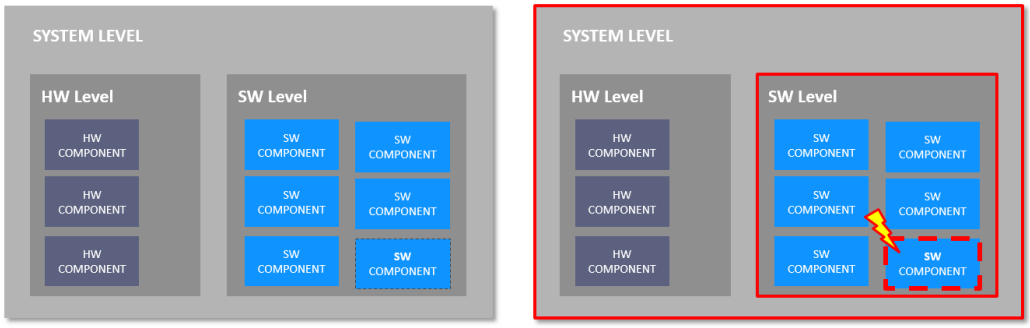


Figure 70 Schematic depiction of the shared risk consideration over several system layers

At each stage across the entire life cycle of the system and the underlying ML component, the risk consideration shall be conducted and examined in terms of the impact of the entire system. For example, the auditor shall examine and describe the impact of data poisoning at the development stage or the impact of an adversarial perturbation within the application stage. The aim of this requirement is to reflect the demand of risk considerations regarding functional safety and corresponding requirements from the homologation body.

Implementation within the use case

The risk consideration is documented and shall cover the threat analysis of the component (e.g. the ML model) in relation to the entire system. Any impact of flaws or malfunctions of the component shall be examined and described with regard to their impact on the system. Thus, risk consideration in terms of threat analysis consists of all potential vulnerabilities which might influence the entire system. The other way around, all vulnerabilities concerning the overall system (including those coming from other sub-components) shall be evaluated regarding their impact on the evaluated component, i.e. the AI subsystem.

Verdict

Due to the nature of the constructed use case and the absence of an entire system (architecture), no verdict is given for this requirement.

5.3.2.2 Requirement 6

Requirement			*Recommendation*
ID	*Description*	*Type*	*ASIL A/ Low*
6	The performance on key performance indicators (KPIs) shall be as high as possible	Additional	+

Specification of the parameters and rationale

The application-specific KPI is chosen as the accuracy of the model's predictions because the accuracy is the most significant measured parameter to quantify the success of the model's purpose, i.e. traffic sign recognition. The performance of the model on the given dataset is directly stated by the accuracy. It is determined by the ratio of correct predictions to the total number of predictions done for the tested dataset.

Audit procedure

For the evaluation of this requirement, first a list of the different model configurations and versions that occurred during development is compiled. Here, especially the resources linked to Requirement 14 in Chapter 5.3.2.5 shall be checked. The model list is examined regarding completeness, i.e. if apparently suitable model configurations had been omitted. If the list is complete, the KPI is evaluated on the different model configurations and/or versions. If the list is incomplete, the missing model configurations are added to the KPI evaluation.

Implementation within the use case

For the development tracking, the version control Git was used. It showed that two different models were considered for usage: a ResNet-18 and an AlexNet architecture. These represent state-of-the-art DNNs/CNNs and a comparison of two networks seems reasonable for the presented use case. The evaluation results of the models were stored in a separate folder using the tracking tool MLflow. The outcome of the specific experiments can be seen in Figure 71. These results could be reproduced and ResNet-18 showed the highest KPI.

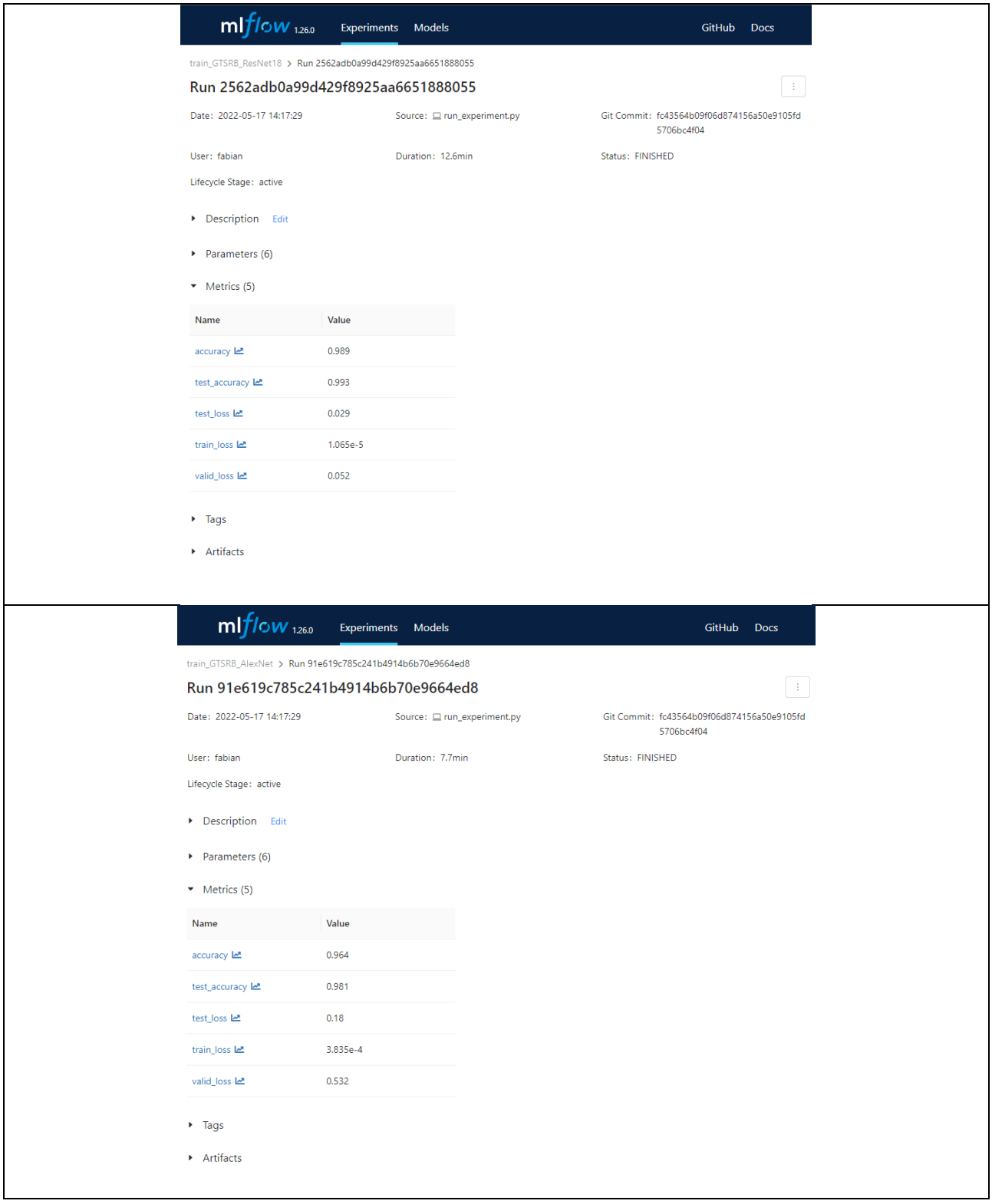


Figure 71 Experiments of the best model training for ResNet-18 (above) and AlexNt (below) tracked via MLflow

Verdict

The evaluation showed reasonable choices for model configurations were made during development. The KPIs were tested and the results could be reproduced. Therefore, it is concluded that Requirement 6 is fulfilled.

5.3.2.3 Requirement 7

Requirement			*Recommendation*
ID	*Description*	*Type*	*ASIL A/ Low*
7	The performance shall be compliant to the allowed worst-case error.	ASIL	++

Specification of the parameters and rationale

This requirement demands the definition of the worst-case error. The worst-case error has to be carefully defined with regard to the intended application of the system, the operational environment and possible threats and hazards stemming from them. In practice, such error definitions are developed after an extensive risk analysis performed by domain experts. Since this project entails a schematic example use case, the worst-case error definition is done in an exemplary way. For this purpose, we assume two different threat and hazard scenarios for the traffic sign recognition system.

The first scenario assumes that the vehicle is operated in Germany, where rain and even heavy rain is common. Therefore, the operational environment requires the system to be reliable and robust within heavy rain conditions. To ensure the reliability, we assume that the assistive traffic sign recognition is still sufficiently reliable for the driver's purpose with an accuracy of at least 90%. It is important to note that as explained above the definition of the hazard or threat as well as the boundary for the accuracy has to be derived by a domain expert and a sound rationale has to be provided. As a result, we define the first example worst-case error as follows: The performance shall be compliant to an accuracy above 90% under heavy rain conditions.

The second scenario assumes that an adversary could alter the input of the model within a specific amount of time. This entails for the security and safety of the system that the model has to be robust against adversarial attacks from an adversary following the threat model of the system. In practice, the threat model for the entire system would be defined by a domain expert. Here, we assume that the attacker has an available time of 1 second to alter the input before it is passed to the model. As the PGD attack is the current state-of-the-art adversarial attack, we chose this attack to give an insight into the robustness of the system. The PGD attack is controlled by a perturbation boundary, which controls the amount of visually perceptible perturbation added to the attacked image. The less perceptible a PGD attack is, the higher the amount of time consumed to produce the perturbation. Due to time available for the attack, we choose the perturbation boundary (epsilon) for this attack as 0.3. As above, we assume a sufficient reliability at an accuracy of above 90%.

As a result, two requirements can be formulated as:

1. The performance shall be compliant to an accuracy above 90% under heavy rain conditions.
2. The performance shall be compliant to an accuracy above 90% under a PGD attack with a perturbation boundary of 0.3.

Audit procedure

If this requirement was adhered to, the model should have been tested against the worst-case errors during its development process. Therefore, the testing documentation and test code should include such tests. An auditor assesses the testing documentation and test code to ensure the compliance with the requirement. In addition, an auditor tests the model against the defined worst-case errors. Within the scope of this project, this is done by adding perturbations to the input of a model and evaluating the model's accuracy. The toolbox enables adding perturbations to images and evaluating the model's accuracy on these images. For the purpose

of this use case, the toolbox implements configurations for the PGD attack from the pytorch attacks library and transformations from the albumentations library. The pytorch attacks library is a Python library that contains a number of state-of-the-art adversarial attacks for image datasets. Finally, the albumentations library contains several commonly used transformations that add some form of perturbation to an image, such as rain, Gaussian noise or motion blur as discussed in Chapter 5.2.3.2.

Implementation within the use case

For the purpose of this use case, the model was tested against both defined worst-case errors using automated unit tests. Equivalent to the auditing process implemented in the toolbox, the model was tested using the pytorch attacks PGD implementation and the heavy rain transformation from the albumentations library with the relevant settings. Figure 72 shows exemplary images for the heavy rain transformation.

GTSRB class 21	GTSRB class 27	GTSRB class 41

Figure 72 Exemplary images for class 21, 27 and 41 transformed by the heavy rain transformation

For the tests, 60 transformed images for every class were included in the test dataset. The results of the tests can be seen in Table 57.

Table 57 Results of heavy rain transformation and PGD attack against the model (accuracies rounded to 4th decimal place)

Worst-Case Error	*Tested Data Samples*	*Correct Predictions*	*Failed Predictions*	*Accuracy*	*Required Accuracy*
Heavy rain transformation	2,580	2,031	549	0.7872	>0.9
PGD attack (epsilon = 0.3)	2,580	552	2,028	0.2140	>0.9

The model had an accuracy of 78.72% on the evaluated dataset transformed with the heavy rain transformation and thus misses the requirement of at least 90% accuracy for this requirement. The model's accuracy on the dataset perturbed via the PGD attack with an epsilon of 0.3 was determined as 21.40%. The set boundary of 90% accuracy was not reached.

Verdict

Based on the evaluation results above, the model fails the defined requirements regarding the acceptable worst-case error. Therefore, Requirement 7 is not fulfilled.

5.3.2.4 Requirement 12

Requirement			*Recommendation*
ID	*Description*	*Type*	*ASIL A/ Low*
12	The architectural design shall be described explicitly.	ASIL	++

Specification of the parameters and rationale

No parameters have to be set for this requirement.

Audit procedure

Since the constructed use case is not embedded within an overall system, this requirement is evaluated schematically. The evaluation of this requirement is twofold. In a first step, the suitability of the measures taken for describing the architectural design adequately is examined. The used methods are reviewed regarding their eligibility to provide a correct and complete description. So, the documentation's form of presentation is assessed.

The basic concept of this requirement is to ensure a broad overview of the overall system to especially demonstrate the integration and the role of the AI subsystem in the entire system. Here, particular importance lies within the presentation of the connections between the sub-components, but also to the system environment, e.g. external (data) sources or interfaces for intervention. Furthermore, functionality of the specific sub-components shall be illustrated as well as the data flow inside the system.

As is stated in the explanatory text corresponding to Requirement 12 in AP3 (2), a description in natural language is mandatory. The extent of further measures is depending on the system's risk level. As the risk is considered low (see Chapter 5.3.1), an additional informal description is sufficient.

In a second step, the correct application of the used methods is verified. For this, it is mandatory to validate correctness and completeness of the description provided by the final documentation. The source code of the SW units and corresponding interfaces is reviewed and it is evaluated if documentation correctly and fully represents the entire system regarding its overall structure and functionality in a comprehensive way.

Implementation within the use case

Text-based documentation describing the design process and the development process is often sufficient to provide an outline of the system's architecture and thus fulfilling the mandatory part of this requirement. Frequently graphical representations, such as UML diagrams, are used by developers as assistance during design and development phase or subsequently for documentation purposes as well.

Regarding functionality, a reference to documentation of the individual SW units is sufficient. The dataflow of the system can be illustrated with automatically created flow charts or diagrams. Further depictions are possible and not limited as long as these are sufficient for and complete in presenting the system in total respectively functionality, dataflow or interfaces of its components.

Verdict

Due to the nature of the constructed use case and the absence of an entire system (architecture), no verdict is given for this requirement.

5.3.2.5 Requirement 14

Requirement			*Recommendation*
ID	*Description*	*Type*	*ASIL A/ Low*
14	The development process shall be tracked.	Additional	+

Specification of the parameters and rationale

No parameters have to be set for this requirement.

Audit procedure

The evaluation of this requirement is twofold. In a first step, the suitability of the measures taken for tracking of the development process is examined. The used methods are reviewed regarding their eligibility to provide the possibility of reproducing and retracing the development of the system to comprehend design decisions and examine the cause of errors.

For Requirement 14, especially a tracking of the training process and architectural changes to the model are relevant. Furthermore, the results of the internal tests and evaluations shall be stored.

In a second step, the correct application of the used methods is verified. For this, the tracked items and affairs are reviewed. It is examined if it is possible to retrace the progression of the training process and the final model decision.

Implementation within the use case

For the overall tracking of the SW development, Git was used. The Git repository enables version control and state control. With the Git status, it is possible to comprehend changes and updates made to the source code. Several Git commits were found assigned to different development states of the source code.

For the tracking of the development of the model, MLflow was used. MLflow is an open-source tool for managing the machine learning lifecycle. The platform consists of four components for experiment tracking, project packing, model packing and model registry.

In this project, the utility of experiment tracking was implemented to record the training process. All started training experiments are listed with timestamp and duration (see Figure 65). For every experiment, the following values and parameters are stored:

- Parameters:
  - Batch size
  - Learning rate
  - Loss type
  - Number of epochs
  - Type of optimizer
  - Decay of weight
- Metrics:
  - Accuracy on validation dataset (relative to epoch)
  - Accuracy on test dataset
  - Loss on training data (relative to epoch)
  - Loss on validation data (relative to epoch)
  - Loss on test data

Additionally, in each case the best model and the final model is stored and a reference to the related Git commit is given. Several training experiments tracked with MLflow were found. The experiments for the final models are illustrated in Figure 71.

Experiments other than training, such as model evaluation on transformed or perturbed data, or methods for explainability are tracked in a separate log folder structure defining its type, model accuracy, loss and the data used for inference.

Verdict

The SW development was tracked in a reproducible way. The development and training processes for the final models could be reproduced and verified. Considering the evaluation results stated, Requirement 14 is fulfilled.

5.3.2.6 Requirement 18

Requirement			*Recommendation*
ID	*Description*	*Type*	*ASIL A/ Low*
18	The AI model shall be tested against out-of-distribution data.	ASIL	++

Specification of the parameters and rationale

For the out-of-distribution data, the China Academic Traffic Sign Amalgamated Dataset (CATSAD) is chosen. The image structure of this dataset is quite similar to the GTSRB dataset , but displays significant differences, both visually and semantically. The images of both datasets share similarities in:

- A diverse background,
- Traffic sign shape (from various angles),
- Different image sections,

Thus, the chosen out-of-distribution data is not completely unrelated to the training data. Obvious differences between the sets are:

- Semantic content,
- Coloration of traffic signs.

Furthermore, the figures on the traffic signs can be quite similar to the German traffic signs, but also completely differ, e.g. when showing Chinese letters.

The model shall not misclassify data from the out-of-distribution dataset. The boundary for an explicit classification shall be set by a domain expert. For this exemplary use case it is set at a model accuracy of 80% and upwards.

Audit procedure

The requirement is evaluated by testing the AI model's performance against suitable out-of-distribution data. First, a suitable dataset containing the out-of-distribution data has to be found. A rationale for the suitability of the chosen dataset representing out-of-distribution data shall be given. Then, the model's performance on the chosen dataset is evaluated and compared to the set boundary in the parameter specification.

Implementation within the use case

For the testing part, three classes of traffic signs from the CATSAD dataset were chosen, that share some characteristics with one or more GTSRB classes. An exemplary image of each chosen class is shown in Figure 73.

Figure 73 Exemplary images of the chosen classes for out-of-distribution data

Evaluation CATSAD class i9:

The CATSAD class i9 contains 75 different images. All images of the class were fed into the model and the model's accuracy on the class set was evaluated. The results are demonstrated below:

Table 58 Evaluation results of CATSAD class i9 (accuracies rounded to 5th decimal place)

GTSRB Class ID	Accuracy		GTSRB Class ID	Accuracy		GTSRB Class ID	Accuracy
00000	0.0		00015	0.0		00030	0.0
00001	0.0		00016	0.0		00031	0.0
00002	0.19697		00017	0.0		00032	0.0
00003	0.0		00018	0.0		00033	0.19697
00004	0.0		00019	0.0		00034	0.0
00005	0.0		00020	0.0		00035	0.0
00006	0.0		00021	0.0		00036	0.0
00007	0.0		00022	0.0		00037	0.0
00008	0.01515		00023	0.0		00038	0.46970
00009	0.03030		00024	0.0		00039	0.0
00010	0.0		00025	0.0		00040	0.0
00011	0.0		00026	0.0		00041	0.0
00012	0.0		00027	0.0		00042	0.0
00013	0.09091		00028	0.0			
00014	0.0		00029	0.0			

The evaluation results for class i9 of the CATSAD datasets are presented in Table 58. For every class, the model showed only low accuracies (<0.5). Therefore, a false prediction of this type of traffic sign is rendered unrealistic.

Evaluation CATSAD class pl5:

The CATSAD class pl5 contains 820 different images. All images of the class were fed into the model and the model's accuracy on the class set was evaluated. The results are demonstrated below:

Table 59 Evaluation results of CATSAD class pl5 (accuracies rounded to 5th decimal place)

GTSRB Class ID	Accuracy		GTSRB Class ID	Accuracy		GTSRB Class ID	Accuracy
00000	0.0		00015	0.00413		00030	0.0
00001	0.04132		00016	0.00551		00031	0.0
00002	0.87741		00017	0.0		00032	0.0
00003	0.0		00018	0.00137		00033	0.00551
00004	0.0		00019	0.0		00034	0.0
00005	0.00964		00020	0.00413		00035	0.0
00006	0.0		00021	0.0		00036	0.0
00007	0.0		00022	0.0		00037	0.00138
00008	0.00275		00023	0.0		00038	0.03719
00009	0.0		00024	0.0		00039	0.0
00010	0.0		00025	0.0		00040	0.0
00011	0.0		00026	0.00138		00041	0.0
00012	0.0		00027	0.0		00042	0.0
00013	0.00826		00028	0.0			
00014	0.0		00029	0.0			

The evaluation results for class pl5 of the CATSAD datasets are presented in Table 59. For class 00002 of the GTSRB dataset, the model showed a relatively high accuracy of around 0.88, thus exceeding the 80% boundary and posing a misclassification. For illustration, Figure 74 shows an exemplary image of this class. Concluding, a false model prediction of a speed limit 5 sign as a speed limit 50 sign might occur.

Figure 74 Exemplary image of class 00002 of the GTSRB

Evaluation CATSAD class w55:

The CATSAD class w55 contains 1,096 different images. All images of the class were fed into the model and the model's accuracy on the class set was evaluated. The results are demonstrated below:

Table 60 Evaluation results of CATSAD class w55 (accuracies rounded to 5th decimal place)

GTSRB Class ID	*Accuracy*		*GTSRB Class ID*	*Accuracy*		*GTSRB Class ID*	*Accuracy*
00000	0.0		00015	0.0		00030	0.08491
00001	0.0		00016	0.0		00031	0.00094
00002	0.0		00017	0.00189		00032	0.0
00003	0.02547		00018	0.00189		00033	0.0
00004	0.0		00019	0.0		00034	0.0
00005	0.00472		00020	0.00283		00035	0.0
00006	0.0		00021	0.00094		00036	0.0
00007	0.0		00022	0.0		00037	0.0
00008	0.0		00023	0.0		00038	0.0
00009	0.0		00024	0.0		00039	0.0
00010	0.0		00025	0.00377		00040	0.0
00011	0.51321		00026	0.00189		00041	0.0
00012	0.17547		00027	0.0		00042	0.0
00013	0.01698		00028	0.02075			
00014	0.0		00029	0.14434			

The evaluation results for class w55 of the CATSAD datasets are presented in Table 60. For every class, the model showed only low accuracies (<0.5 or just above). Therefore, a false prediction of this type of traffic sign is rendered unrealistic.

Verdict

Appropriate out-of-distribution data was chosen using a dataset of Chinese traffic signs. The evaluation showed that the model-under-test predicted 87% of the images of the CATSAD class pl5 as the German traffic sign "Tempolimit 50 km/h". Due to this fact, a false classification cannot be excluded and Requirement 18 is not passed.

5.3.2.7 Requirement 19

Requirement			*Recommendation*
ID	*Description*	*Type*	*ASIL A/ Low*
19	Test cases at the boundary values of the input of the AI model shall be derived.	ASIL	+

Specification of the parameters and rationale

As stated in the introduction of this section, a definition of the test and evaluation parameters is heavily dependent on the considered use case. Regarding this requirement, a challenge is the definition of an acceptable and meaningful input range. For numerical data this can often be trivial, but for other use cases it can be quite complex. For example, the input range for the considered use case of image classification could be every possible combination of different colored pixels, from pitch black to a completely white picture. A more specific approach is to focus on specific image features, i.e. brightness, rotation, contrast or sharpness. But even with the more focused approach, it is hard to define meaningful boundaries that are relevant for the specific use case. For this, domain experts as well as literature or standardization and legal requirements shall be consulted. Due to these reasons, the requirement is evaluated schematically by giving a general procedure.

Audit procedure

This requirement covers the topic of boundary values of the model's input space. For this purpose, the model's behavior for the edge region of the input is analyzed.

As stated above, first a specification of the boundary values is necessary. Either there already is a specified value range or it has to be derived. For the latter, e.g., technical literature can be consulted to establish common value spaces within the use case. The expertise of a domain expert can specify these input spaces even further to derive adequate boundaries for the developed system.

The edge regions of the specified value range are tested as inference input and the resulting model behavior is observed for non-expected or faulty behavior, e.g. out-of-range output values. Then, the defined edge regions can be incorporated in Requirement 20 to craft corner cases, for example, as a combination of several boundary values.

Implementation within the use case

The input value range and thus the boundary values are heavily dependent on the considered use case. Even within the same area of application, diverse input ranges can be reasonable, e.g. due to different models. After specifying the input range or respectively determining the boundary values of the use case and model, a general procedure would be a testing of these inputs against the model. The model's behavior respectively the corresponding output values should be recorded and analyzed. Especially, deviations to normal behavior of the model should be tracked and examined.

5.3.2.8 Requirement 20

Requirement			*Recommendation*
ID	*Description*	*Type*	*ASIL A/ Low*
20	Test cases based on corner cases of the AI model shall be derived.	ASIL	+

Specification of the parameters and rationale

Similar to Requirement 19, it can be quite challenging to determine meaningful corner cases. Again, these are heavily dependent on the specific use case and the entire system. For the definition, domain experts, literature and prior experience shall be taken into account. In the image domain, corner cases can be crafted regarding various features. These can be of technical nature (brightness, contrast, etc.), semantical (unlikely and exceptional scenarios on the image), or combinations of them. Due to this, the requirement is evaluated schematically by giving a general procedure.

Audit procedure

The consideration of corner cases is crucial for checking the system's behavior within unusual and abnormal constellations of input parameters. Functional safety requirements demand such considerations for a certain ASIL level. In contrast to boundary values, where the input parameters are varying along the specified range, corner cases comprise the "worst-case" combination of several input values. Within the audit process,

relevant inputs and values shall be selected and combined to set up a collection of corner cases which can be used as input to the model. The selection process shall take the following guidance into account:

- Combination of the specified min/max values of the inputs (e.g. maximum possible contrast value at the minimum possible brightness value, etc.)
- Selection of the most unlikely combination of inputs or scenarios (e.g. rainy foreground and cloudless background, etc.)
- Analysis of the input parameter distributions and combination of values at edge regions.

The evaluation results are analyzed in terms of non-expected/faulty behavior of the model.

Implementation within the use case

Again, the corner case definition is heavily dependent on the considered use case. A general procedure would be to present pre-selected corner cases crafted from the input signals to the model. An analysis of the results can show unwanted deviations from the expected behavior.

5.3.2.9 Requirement 28

Requirement			*Recommendation*
ID	*Description*	*Type*	*ASIL A/ Low*
28	The datasets used for training, testing and evaluation shall not contain any errors.	Additional	++

Specification of the parameters and rationale

No parameters have to be set for this requirement.

Audit procedure

To evaluate this requirement, the datasets used for training, testing and evaluation have to be reviewed. The procedure how the dataset review is carried out highly depends on the use case and the type of data the dataset contains. In the case of image classification, the following characteristics play an important role and shall be evaluated:

- Size of the images
- Irrelevant images
- Labeling
- Repeated data entries
- Relevance of overall dataset or specific classes to the use case
- Damaged/invalid data samples
- Manipulated data samples

Implementation within the use case

In the considered use case, the GTSRB dataset was used. The dataset was split in a way that the training and test set are independent of each other and do not share a common image (see Requirement 30, Chapter 5.3.2.10). The sets were analyzed regarding image size and no significant outlier was found. All images were analyzed regarding their semantical meaning and relation to the use case without findings. The labeling process was correct and no incorrect labels were identified. Also, there were no repetitive images in the datasets.

Verdict

A review showed no indication of possible errors inside the used datasets. Therefore, it is concluded that Requirement 28 is fulfilled.

5.3.2.10 Requirement 30

Requirement			*Recommendation*
ID	*Description*	*Type*	*ASIL A/ Low*
30	The training, test and evaluation datasets shall be independent from each other.	Additional	++

Specification of the parameters and rationale

No parameters have to be set for this requirement.

Audit procedure

For this requirement, the datasets used for training, evaluation and testing of the ML model have to be assessed.

The training dataset is the data that is used for fitting the model. It is crucial for the model's performance during inference. The evaluation dataset is used for the validation of the model's hyperparameters and training progress during the training procedure. The trained model is evaluated against this dataset and the hyperparameters are adjusted accordingly. The testing dataset is used to verify the model's performance after the training on previously unseen data.

Mixing the datasets is problematic since data that is used for the training is already incorporated in the model and thus is not suitable for the detection of anomalies and errors during training (such as underfitting and overfitting). Further, it is considered best-practice to use unseen data for the final testing of the model, since this verifies the model's performance on new data independent from the training and evaluation data.

Implementation within the use case

The original dataset "German Traffic Sign Recognition Benchmark" was split into sets for training, validation and evaluation. The splitting procedure is described in the toolbox documentation (295) in the section for available datasets. The split is disjunct,such that no dataset shared an element with another. This was confirmed by an examination of the stored datasets within the project.

Verdict

It was confirmed that the used training, validation and evaluation datasets do not share a common item. Therefore, it is concluded that Requirement 30 is fulfilled.

5.3.2.11 Requirement 32

Requirement			*Recommendation*
ID	*Description*	*Type*	*ASIL A/ Low*
32	The requirements shall be analyzed to derive test cases for interpretable model decisions.	ASIL	++

Specification of the parameters and rationale

In order to implement and audit Requirement 32, the functional requirements towards the system of the model have to be analyzed. Since the exemplary use case at hand is not attached to a full system and only consist of the AI model itself, we assume the following exemplary system requirement: *The system shall not be susceptible to background information.* This entails that more insight into model's decisions is needed to be able to reason that this requirement is met.

Audit procedure

The system requirements, testing documentation and test code of the system have to be reviewed by the auditors. The evaluation is based on whether the requirements needed to be explained exist and that the system was tested accordingly and correctly. During the audit process, the auditors need to use an Explainable AI (XAI) method to explain the decisions made by the system. For this purpose, the toolbox contains a module that uses the GradCam method from the pytorch-gradcam Python library. During the audit, the model is tested on a randomly selected number of images for all classes and the decisions are explained using GradCam. The explained decisions are analyzed whether the model considered any background information for its classification. For this use case, the auditors check whether any information around the depicted traffic sign is highlighted by the GradCam method.

Implementation within the use case

A unit test analogous to the toolbox implementation for certification of the system is performed. For the test dataset, the model's decisions are explained via the GradCam method and visually assessed by the developer whether the background information of any image was taken into account by the model for its decision making.

An analysis of the classes via the GradCam module was conducted. Exemplary images of the GradCam results for the "20 km/h" and the "30 km/h" traffic sign are shown in Figure 75. As can be seen, the model's focus lies on the features directly related to the sign, especially the first digit. Background information played little to no role in the decision process.

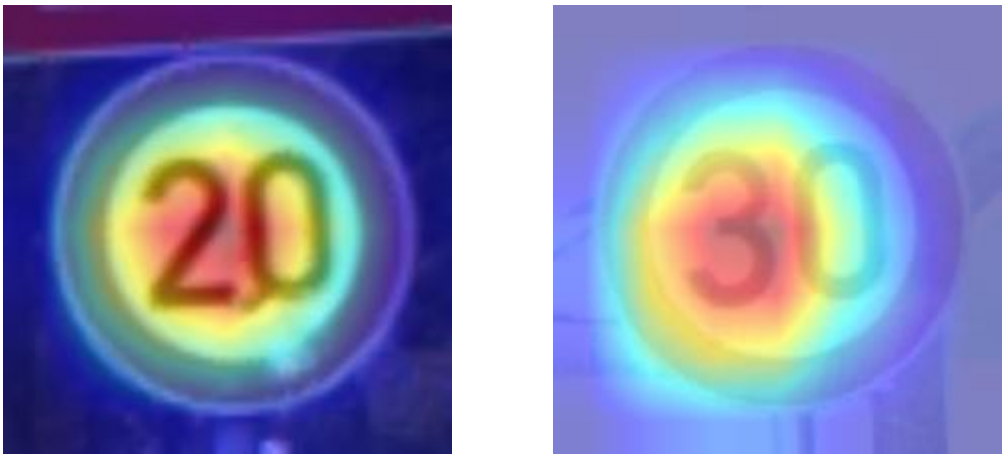


Figure 75 Example toolbox result for GradCam explanations of a 20 km/h traffic sign on the left and a 30 km/h traffic sign on the right

Verdict

A test case for the system's functional requirement ("*The system shall not be susceptible to background information.")* was defined by evaluating the AI model's focus during inference. An analysis via the XAI method GradCam showed that the model did not focus on background information, but rather on the significant features of the particular classes when processing the testing dataset. Therefore, based on the evaluation above, Requirement 32 is fulfilled.

5.3.2.12 Requirement 33

Requirement			*Recommendation*
ID	*Description*	*Type*	*ASIL A/ Low*
33	The model's decisions shall be explained to aid the comparison between the modelling of the system and the trained model.	ASIL	++

Specification of the parameters and rationale

No parameters have to be set for this requirement.

Audit procedure

This requirement reflects the need to explain the model's decisions in order to support the comparison between the model's behavior and the modelled system's behavior. During the software modeling process (SW unit design), all crucial functionality of the system, sub-system and components must be explainable and traceable. Decisions of the modelled parts have to be explainable in the matter of "if-then" taxonomy. The auditor and homologation bodies shall be able to understand and follow the information flow which leads to specific decisions of the system. As decision of ML models cannot be described in a fully comprehensible way. This fact often contradicts the requirements within the SW unit design and the modelling process. Thus, methods from the field of XAI shall be implemented and applied to the model in order to expose the crucial decision processes and to understand how the model led to a certain output.

As a part of the audit, specific traffic scenarios, traffic signs etc. and the corresponding (desired) model reaction shall be described in a formal way, e.g. by definition of the most important features for recognizing a specific traffic sign or a decision rationale for a specific traffic scenario ("if traffic light is green and pedestrian crosses the street, then..."). Subsequently, the actual model's decision for the prior defined scenarios shall be compared to the decision rationales by applying appropriate XAI methods previously. The results shall be checked for consistency and deviations from the desired (modelled) behavior. The evaluation is based on the definition of representative test scenarios during the audit process and the application of XAI methods.

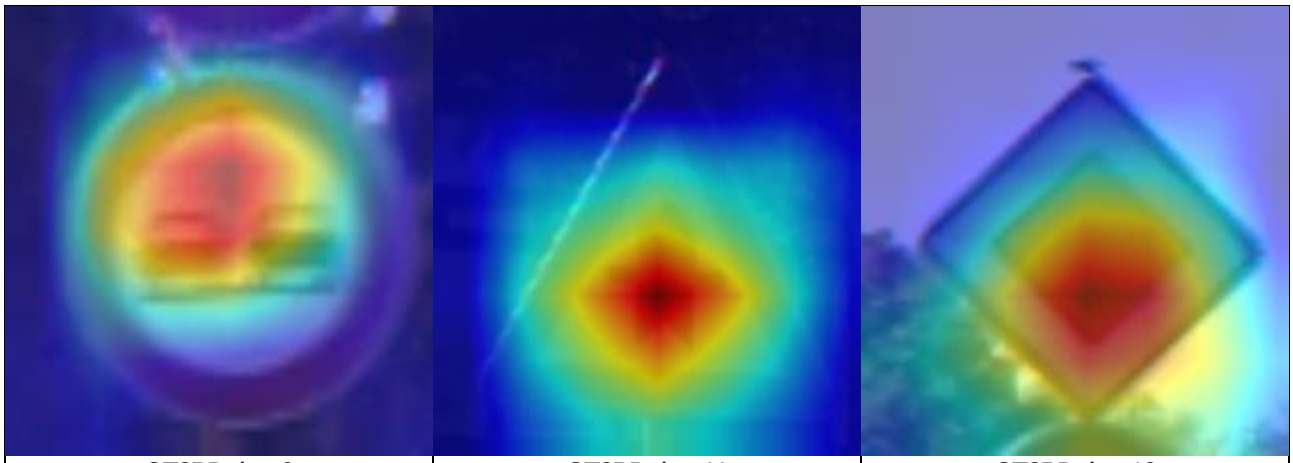
Implementation within the use case

The toolbox contains a module that uses the GradCam method from the pytorch-gradcam Python library. During the audit, the model is tested on previously defined test cases in which a formal decision rationale is given. Then, the model's decisions are explained using GradCam. The results are analyzed and compared to the desired (modelled) behavior with regard to consistency and deviations.

The general formal decision rationale for the considered use case of traffic sign recognition can be formulated as:

"The model decision on a traffic sign image shall depend on the figure displayed by the traffic sign, the sign's coloration and/or the shape of the sign."

A dataset containing different classes was evaluated on the model using GradCam. The dataset contained 60 images for each of the 43 classes of the GTSRB dataset. An analysis of the model's accuracy in combination with the resulting images from GradCam was carried out. The accuracies for the individual classes were all >0.98, meaning the great majority of images was classified correctly.



GTSRB class 9	GTSRB class 11	GTSRB class 12

Figure 76 Exemplary GradCam images from GTSRB

The GradCam images showed that the model's focus was indeed mostly on the traffic sign. Whether the model's decision came from coloration or the depicted figure is not resolved. Exemplary images are presented in Figure 76.

Verdict

The evaluation targeted the suitability of GradCam to help understanding model decisions and thus giving an indication if the modelling objective and goals were successfully implemented by the AI model. It showed that GradCam was able to give an intelligible presentation of the model's focus. Therefore, based on the evaluation above, Requirement 33 is fulfilled.

5.3.2.13 Requirement 40

Requirement			*Recommendation*
ID	*Description*	*Type*	*ASIL A/ Low*
40	The SW unit design shall be described explicitly.	ASIL	++

Specification of the parameters and rationale

No parameters have to be set for this requirement.

Audit procedure

The evaluation of this requirement is twofold. In a first step, the suitability of the measures taken for describing the SW unit are adequately examined. The used methods are reviewed regarding their eligibility to provide a correct and complete description. So, the documentation's form of presentation is assessed.

The machine learning part of the SW unit is primarily defined by the training process and the model's architecture. Therefore, the focus lies on the description of the SW parts responsible for the creation and the training of the model. The documentation of potential additional modules of the system, e.g. for pre- or post-processing, is considered as well.

As is stated in the explanatory text corresponding to Requirement 40 in AP3 (2), a description in natural language is mandatory. The extent of further measures is depending on the system's risk level. As the risk is considered low (see Chapter 5.3.1), an additional informal description is sufficient.

In a second step, the correct application of the used methods is verified. For this, it is mandatory to validate correctness and completeness of the description provided by the final documentation. The source code of the SW units is reviewed and it is evaluated if documentation correctly and fully represents the SW regarding the overall structure and functionality in a comprehensive way.

Implementation within the use case

The documentation of the software unit design is implemented using Sphinx as discussed in Chapter 5.2.3.3. Sphinx extracts information from the python source code to create documentation files. More accurately, for every SW module reStructuredText files are created from docstrings in the source code, which then again can be converted, for example, into a HTML-based documentation.

The architecture of the used models is given in form of a graphical representation in Figure 77. Here, the graphs describing the models are unambiguous and it was verified that the illustration matches the model architecture.

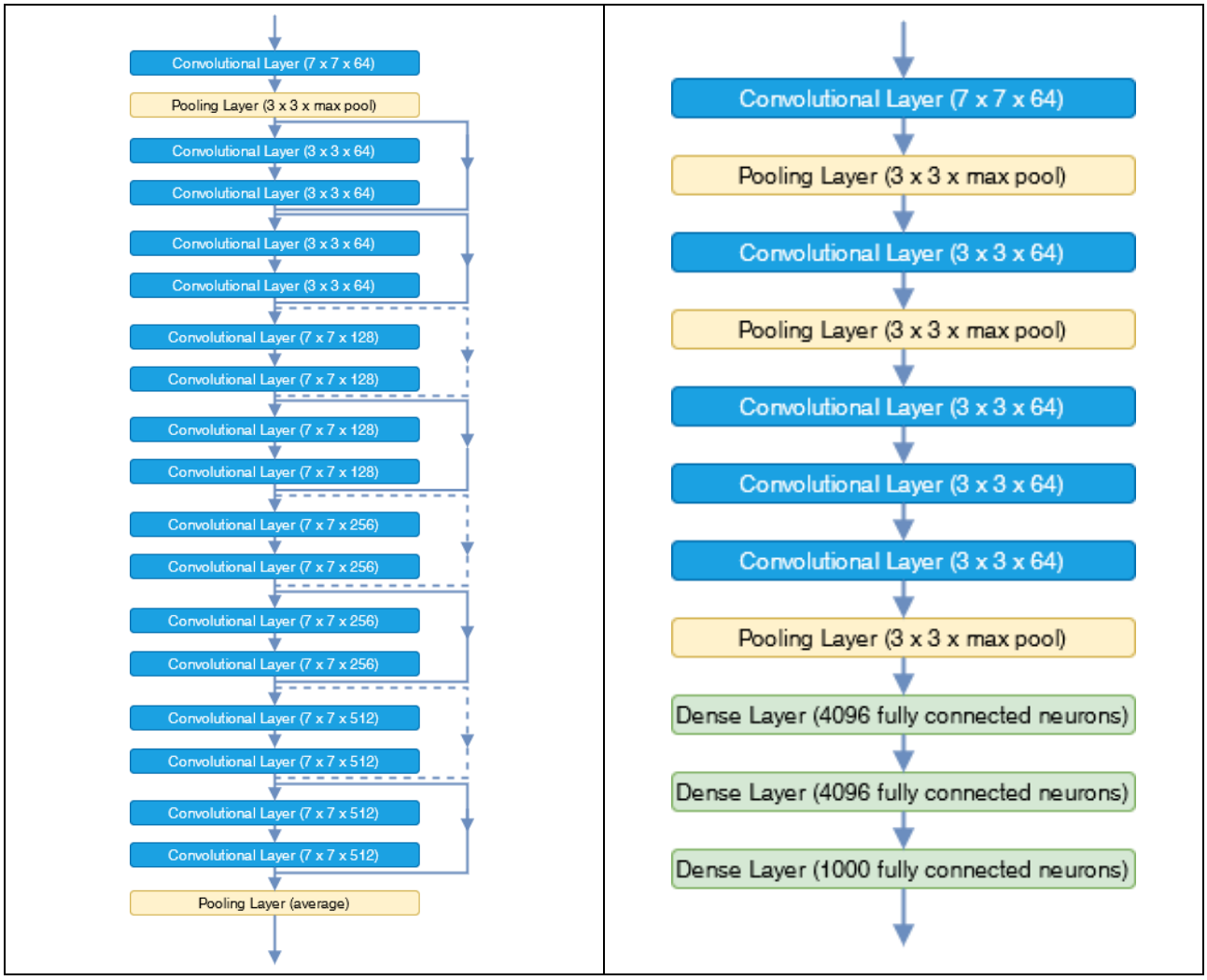


Figure 77 Model architectures of ResNet-18 (left) and AlexNet (right)

The target of the evaluation consists solely of the machine learning model. Additional modules are not present. For every module concerning the training process, clear and complete docstrings for classes, functions and methods are present. Figure 78 illustrates a source code excerpt containing exemplary docstrings.

```python
class DataSetGTSRB(ImageClassificationDataSet):
    """
    Concrete class which contains the German Traffic Sign Recognition Benchmark (GTSRB)
    dataset from https://benchmark.ini.rub.de/gtsrb_news.html.

    Args:
        *args: Passed to :class:`.ImageClassificationDataSet`.
        **kwargs: Passed to :class:`.ImageClassificationDataSet`.

    Notes:
        The eval dataset is created by separating the images from the first two drives
        for each class in the train dataset.
    """
    def __init__(self, *args, **kwargs) -> None:
        super().__init__(*args, **kwargs)

    @classmethod
    def get_class_label(cls, image_path: Path) -> int:
        return int(image_path.parents[0].name)

    def _build_train_and_eval_data(self) -> None:
        """
        Creates the correctly formatted train and eval datasets from the structure of
        the downloaded and extracted .zip file containing the training dataset. The
        first two drives of each class are used as a eval dataset and are removed
        from the train dataset.
        """
        internal_path = self.path / PATH_TRAIN_DATASET
        if self.split == 'train':
            train_path = self.path
            eval_path = self.path.parent / 'eval'
            create_dir_if_missing(eval_path)
```

Figure 78 Exemplary exempt for class and method describing docstrings

Verdict

The chosen form of documentation, i.e. the docstrings in the source code respectively their transformation into a HTML-based documentation, generally fulfills the requirement regarding the description in natural language. The graphical illustration of the model architecture is correct and fulfills the requirement regarding an informal description of the SW unit. Therefore, based on the evaluation above, Requirement 40 is fulfilled.

5.3.3 Summary of Requirements

This chapter summarizes the evaluation results of the implemented requirements from Chapter 5.3.2. An overview of the requirements and their verdicts is given in Table 61.

Table 61 Summary of the implemented requirements[7] and their corresponding verdicts

Requirement		*Evaluation*	
ID	*Description*	*Chapter*	*Verdict*
4	The requirements for AI subsystems shall apply to the entire system (if applicable).	5.3.2.1	-
6	The performance on key performance indicators (KPIs) shall be as high as possible	5.3.2.2	Pass
7	The performance shall be compliant to the allowed worst-case error.	5.3.2.3	Fail
12	The architectural design shall be described explicitly.	5.3.2.4	-
14	The development process shall be tracked.	5.3.2.5	Pass

Requirement		*Evaluation*	
18	The AI model shall be tested against out-of-distribution data.	5.3.2.6	Fail
19	Test cases at the boundary values of the input of the AI model shall be derived.	5.3.2.7	-
20	Test cases based on corner cases of the AI model shall be derived.	5.3.2.8	-
28	The datasets used for training, testing and evaluation shall not contain any errors.	5.3.2.9	Pass
30	The training, test and evaluation datasets shall be independent from each other.	5.3.2.10	Pass
32	The requirements shall be analyzed to derive test cases for interpretable model decisions.	5.3.2.11	Pass
33	The model's decisions shall be explained to aid the comparison between the modelling of the system and the trained model.	5.3.2.12	Pass
40	The SW unit design shall be described explicitly.	5.3.2.13	Pass

Overall, 13 requirements recommended for the evaluation of the considered use case are introduced. As stated before, four requirements are covered schematically. From the remaining requirements, 7 were fulfilled, while for Requirement 7 and Requirement 18 the conditions were not met. The failed requirements indicate a lack of robustness of the model's prediction. Further examinations should follow, e.g., if the lack of robustness is limited to certain classes or affects all classes. Potential measures to enhance the model for fulfilling the failed requirements can include a general expansion of the training set with more data samples as well as an integration of perturbed images for an adversarial training.

6 Conclusion

6.1 Summary

The work and results described in this report build the foundation for the follow-up project AIMobilityAudit. True to its name AIMobilityAudit**Prep**, the current project enables an extensive testing of different auditing requirements for AI-based AD/ADAS systems in the future. For this, the first major contribution of this work is a list of 50 requirements which are technically relevant to assure the IT-Security, robustness, explainability, etc. of AI-based AD/ADAS systems. These requirements evolved based on a continuous discussion of the different stakeholders that participated in this project und under attention to existing regulations, norms, or guidelines and an extensive SOTA literature review. The literature review ensures that the proposed requirements remain feasible and that current findings from recent research are incorporated to provide up to date requirements.

The second main contribution of this work are the developed toolchain and toolbox components that are useful to audit AI-based systems. On the one hand, with the developed software it is possible to train an exemplary AD/ADAS systems for a chosen use case. In addition, tools are provided that allow to audit the system whether it is compliant to selected audit requirements. Here, we exemplary implement 13 requirements which are supported in the developed toolchain as proof-of-concept. To demonstrate that the toolchain concept is suitable and that the developed tools can be used to audit actual AI-based AD/ADAS systems we demonstrate the use on a selected use case. First, to select the most fitting AD/ADAS use case we perform an extensive comparison of many possible AD/ADAS use cases, based on different categories like complexity, auditability, available resources, etc. In this analysis it showed that the TSR use case is best suited for the initial testing of the toolchain and toolbox. Therefore, we use the developed toolchain to create and train two exemplary TSR systems based on publicly available datasets and DNN architectures which serve as exemplary AD/ADAS systems. These systems are then examined using the 13 implemented audit requirements in the toolbox. Thereby, we use a simulation strategy which is based on the augmentation of existing images to simulate different environmental conditions.

During the proof-of-concept implementation, we find that the toolbox is easily extendable to include further audit requirements. Also, we find that the selected requirements can be specified to provide meaningful results for a DNN-based TSR system which serves as a placeholder of a general AD/ADAS system. All in all, the achieved results of this project serve as a good starting point for following projects which assess the suitability of different audit requirements in more depth.

6.2 Outlook

As discussed in the last chapter we implemented only 13 of the 50 proposed audit requirements as proof-of-concept. A natural step in a follow-up project is to extend the existing toolbox to include all 50 requirements. This allows to assess for all requirements whether they are suitable to audit AI-based AD/ADAS systems or whether some requirements are challenging or infeasible to implement for audits in praxis. Additionally, one can expand the extent of the already implemented requirements. Some of these requirements are quite extensive and can be implemented for practical tests in different ways. For example, the security can be evaluated against different adversarial attacks and the robustness can be evaluated in different weather conditions. In the current implementation we chose a single exemplary implementation for each requirement to show the principled applicability of the toolbox. In a follow-up project this exemplary implementation can be expanded to cover further aspects of the associated audit requirement. This enables more extensive audits and increases the meaningfulness of the obtained results.

Furthermore, it is especially interesting to test some audit requirements using actual hardware and test facilities. Instead of performing all tests in a simulation environment, the most interesting audit requirements should also be tested in reality. Only these tests enable to properly assess the feasibility and expressiveness of the proposed audit requirements. To maximize the gained information from practical tests, it is most

important to perform the physical tests with audit requirements where one might suspect a difference between the results obtained by performing the test in a simulation environment. Lastly, we advise to revisit all proposed audit requirements based on the obtained results in the extensive tests and based on new insights from the literature. This allows to continuously optimize the audit requirements and be best prepared to use the results as part of a “technische Richtlinie”.

List of Figures

List of Tables

Acronyms

Acronym	*Meaning*
AD	Autonomous Driving
ADAS	Advanced Driver Assistance System
ADP	Adaptive Diversity Promoter
AI	Artificial Intelligence
ANSI	American National Standards Institute
AP	Work Package (Arbeitspaket)
AR	Application Rule
ASIL	Automotive Safety Integrity Level
AV	Autonomous Vehicle
AWI	Approved Work Item
BIM	Basic Iterative Method
BNN	Bayesian Neural Network
BSI	Bundesamt für Sicherheit in der Informationstechnik
CAL	Cybersecurity Assurance Level
CATSAD	China Academic Traffic Sign Amalgamated Dataset
CD	Committee Draft
CI	Correct implementation of external and internal interface
CNN	Convolutional Neural Network
CU	Chinese University of Hong Kong
DE	Deriving test cases for the embedded software testing
DI	Deriving test cases for integration testing
DIN	German Institute for Standardization
DL	Deep Learning
DMD	Driver Monitoring Dataset
DNN	Deep Neural Network
DP-SGC	Differentially Private Stochastic Gradient Descent
DU	Deriving test cases for the software unit testing
ED	Error Detection
EH	Error Handling
EU	European Union
FDIS	Final Draft International Standard
FGSM	Fast Gradient Sign Method
FP	Functional Performance
GAN	Generative Adversarial Network
GPS	Global Positioning System
GTSRB	German Traffic Sign Recognition Benchmark
HD	High-Definition
HIL	Hardware in the Loop
HMI	Human Machine Interface
IARA	Intelligent Autonomous Robotics Automobile
IEC	International Electrotechnical Commission
IEEE	Institute of Electrical and Electronics Engineers
IMU	Inertial Measurement Unit
ISO	International Organization for Standardization
IT	Information Technology
IV	Software Integration Verification

Acronym	*Meaning*
KITTI	Karlsruhe Institute of Technology/Toyota Technological Institute
KPI	Key Performance Indicator
LiDAR	Light Detection and Ranging
LSTM	Long Short-Term Memory
MC	Modelling and Coding Guidelines
MILP	Mixed Integer Linear Programming
ML	Machine Learning
NA	Notations for the Architectural Design
NU	Notations for the Software Unit Design
ODD	Operational Design Domain
ONNX	Open Neural Network Exchange
PAS	Publicly Available Specification
PCI	Peripheral Component Interconnect
PGD	Projected Gradient Descent
PIE	Pedestrian Intention Estimation
QM	Quality Management
RADAR	Radio Detection and Ranging
ReLU	Rectified Linear Unit
RGB	Red Green Blue
SAE	Society of Automotive Engineers
SIL	Software in the Loop
SLAM	Simultaneous Localization and Mapping
SMT	Satisfiability Modulo Theories
SOTA	State-Of-The-Art
SOTIF	Safety of the Intended Functionality
ST	Software Testing
TEE	Trusted Execution Environment
TR	Technical Report
TSR	Traffic Sign Recognition
UL	Underwriters Laboratories
UNECE	United Nations Economic Commission for Europe
UV	Software Unit Verification
VDE	Association for Electrical, Electronic & Information Technologies
XAI	Explainable AI

Bibliography

1. AIMobilityAuditPrep. *Final Results AP2 - State-of-the-Art Report.* 2022.

2. AIMobilityAuditPrep. *Final Results AP3 - Generic Requirements.* 2022.

3. AIMobilityAuditPrep. *Final Results AP4 - Use Case Comparison for Audit Criteria Development.* 2022.

4. AIMobilityAuditPrep. *Final Results AP5 - Planning and exemplary Creation of Toolbox.* 2022.

5. *AI Enabling Technologies: A Survey.* Gadepally, Vijay, et al. Ithaca, USA : arXiv, 2019, Vol. abs/1905.03592.

6. *Characterizing Machine Learning Process: A Maturity Framework.* Akkiraju, Rama, et al. Business Process Management: Lecture Notes in Computer Science, Berlin, Germany : Springer, Cham, 2020, Vol. 12168.

7. *Characterizing the Software Process: A Maturity Framework.* Humphrey, Watts. New York, USA : IEEE Software, 1988, Vol. 5.

8. *Trustworthy AI: From Principles to Practices.* Li, Bo, et al. New York, USA : Association for Computing Machinery, 2021.

9. *Vulnerabilities of Connectionist AI Applications: Evaluation and Defence.* Berghoff, Christian, Neu, Matthias and von Twickel, Arndt. Lausanne, Switzerland : Frontiers in Big Data, 2020, Vol. 3.

10. *ModelOps: Cloud-Based Lifecycle Management for Reliable and Trusted AI.* Hummer, Waldemar, et al. San Francisco, USA : IEEE International Conference on Cloud Engineering, 2019.

11. *ABOUT ML: Annotation and Benchmarking on Understanding and Transparency of Machine Learning Lifecycles.* Raji, Inioluwa Deborah and Yang, Jingying. Vancouver, Canada : Conference on Neural Information Processing Systems, 2019.

12. *Closing the AI Accountability Gap: Defining an End-to-End Framework for Internal Algorithmic Auditing.* Raji, Inioluwa Deborah, et al. Barcelona, Spain : ACM Conference on Fairness, Accountability, and Transparency, 2020.

13. *Towards the Science of Security and Privacy in Machine Learning.* Papernot, N., et al. s.l. : arXiv preprint, 2016. arXiv:1611.03814.

14. *Delving into Transferable Adversarial Examples and Black-Box Attacks.* Liu, Yanpei, et al. s.l. : 5th International Conference on Learning Representations, {ICLR} 2017, 2017.

15. *DeepSniffer: A DNN Model Extraction Framework Based on Learning Architectural Hints.* Hu, Xing, et al. New York, USA : Proceedings of the Twenty-Fifth International Conference on Architectural Support for Programming Languages and Operating Systems, 2020.

16. *InverseNet: Augmenting Model Extraction Attacks with Training Data Inversion.* Xueluan, Gong, et al. Montreal, Canada : Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence, 2021.

17. *Autoencoders, Unsupervised Learning, and Deep Architectures.* Baldi, Pierre. s.l. : ICML Workshop on Unsupervised and Transfer Learning, 2012.

18. *PRADA: Protecting against DNN Model Stealing Attacks.* Juuti, Mika, Szyller, Sebastian and Asokan, N. s.l. : IEEE European Symposium on Security and Privacy, 2019.

19. *DeepBillboard: Systematic Physical-World Testing of Autonomous Driving Systems.* Zhou, Husheng, et al. Seoul, South Korea : International Conference on Software Engineering, 2020.

20. *Deep Learning-Based Autonomous Driving Systems: A Survey of Attacks and Defenses.* Deng, Yao, et al. s.l. : IEEE Transactions on Industrial Informatics, 2021.

21. *Explaining and Harnessing Adversarial Examples.* Goodfellow, Ian J., Shlens, Jonathon and Szegedy, Christian. s.l. : 3rd International Conference on Learning Representations, {ICLR} 2015, 2015.

22. *Towards Deep Learning Models Resistant to Adversarial Attacks.* Madry, Aleksander, et al. Vancouver, Canada : International Conference on Learning Representations, 2018.

23. *Counteracting Adversarial Attacks in Autonomous Driving.* Sun, Qi, et al. s.l. : 2020 IEEE/ACM International Conference On Computer Aided Design (ICCAD), 2020.

24. *An Analysis of Adversarial Attacks and Defenses on Autonomous Driving Models.* Deng, Yao, et al. s.l. : 2020 IEEE International Conference on Pervasive Computing and Communications (PerCom), 2020.

25. *DeepFool: A Simple and Accurate Method to Fool Deep Neural Networks.* Moosavi-Dezfooli, Seyed-Mohsen, Fawzi, Alhussein and Frossard, Pascal. Las Vegas, USA : IEEE Conference on Computer Vision and Pattern Recognition, 2016.

26. *Towards Evaluating the Robustness of Neural Networks.* Carlini, Nicholas and Wagner, David. s.l. : 2017 IEEE Symposium on Security and Privacy (SP), 2017.

27. *Generative Adversarial Nets.* Goodfellow, Ian, et al. Montreal, Canada : Conference on Neural Information Processing Systems, 2014.

28. *Generating Adversarial Examples with Adversarial Networks.* Xiao, C., et al. s.l. : IJCAI, 2018.

29. *Learning Multiple Layers of Features from Tiny Images.* Krizhevsky, Alex. Toronto, Canada : University of Toronto, 2009.

30. *AdvGAN++ : Harnessing Latent Layers for Adversary Generation.* Mangla, Puneet, et al. Seoul, South Korea : International Conference on Computer Vision, 2019.

31. *Generative Adversarial Perturbations.* Poursaeed, Omid, et al. s.l. : CVPR, 2018.

32. Udacity. Udacity challenge 2: Steering angle prediction. [Online] 2017. https://github.com/udacity/self-driving-car/tree/master/datasets/CH2.

33. *Perceptual-Sensitive GAN for Generating Adversarial Patches.* Liu, A., et al. s.l. : Proceedings of the AAAI Conference on Artificial Intelligence, 2019.

34. *Seeing Isn't Believing: Towards More Robust Adversarial Attack Against Real World Object Detectors.* Zhao, Yue, et al. s.l. : Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security, 2019.

35. *PhysGAN: Generating Physical-World-Resilient Adversarial Examples for Autonomous Driving.* Kong, Zelun, et al. s.l. : 2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 2020.

36. *Physical Adversarial Attacks on Deep Neural Networks for Traffic Sign Recognition: A Feasibility Study.* Woitschek, Fabian and Schneider, Georg. Nagoya, Japan : IEEE Intelligent Vehicles Symposium, 2021.

37. *Adversarial Examples for Semantic Segmentation and Object Detection.* Xie, Cihang, et al. Venice, Italy : IEEE International Conference on Computer Vision, 2017.

38. *Physical Adversarial Examples for Object Detectors.* Eykholt, Kevin, et al. Baltimore, USA : USENIX Workshop on Offensive Technologies, 2018.

39. *The Vulnerability of Semantic Segmentation Networks to Adversarial Attacks in Autonomous Driving: Enhancing Extensive Environment Sensing.* Bär, Andreas, et al. New York, USA : IEEE Signal Processing Magazine, 2021, Vol. 38.

40. *Indirect Local Attacks for Context-aware Semantic Segmentation Networks.* Nakka, Krishna Kanth and Salzmann, Mathieu. Glasgow, United Kingdom : European Conference on Computer Vision, 2020.

41. *Consistent Semantic Attacks on Optical Flow.* Koren, Tom, et al. Ithaca, USA : arXiv, 2021, Vol. abs/2111.08485.

42. *What Causes Optical Flow Networks to be Vulnerable to Physical Adversarial Attacks.* Schrodi, Simon, Saikia, Tonmoy and Brox, Thomas. Ithaca, USA : arXiv, 2021, Vol. abs/2103.16255.

43. *Targeted Adversarial Perturbations for Monocular Depth Prediction.* Wong, Alex, Cicek, Safa and Soatto, Stefano. Vancouver, Canada : Conference on Neural Information Processing Systems, 2020.

44. *Monocular Depth Estimators: Vulnerabilities and Attacks.* Mathew, Alwyn, Patra, Aditya Prakash and Mathew, Jimson. Ithaca, USA : arXiv, 2020, Vol. abs/2005.14302.

45. *AdvPC: Transferable Adversarial Perturbations on 3D Point Clouds.* Hamdi, Abdullah, et al. Glasgow, United Kingdom : European Conference on Computer Vision, 2020.

46. *Adversarial Objects Against LiDAR-Based Autonomous Driving Systems.* Cao, Yulong, et al. Ithaca, USA : arXiv, 2019, Vol. abs/1907.05418.

47. *Exploring Adversarial Robustness of Multi-sensor Perception Systems in Self Driving.* Tu, James, et al. London, United Kingdom : Conference on Robot Learning, 2021.

48. *Towards Robust LiDAR-based Perception in Autonomous Driving: General Black-box Adversarial Sensor Attack and Countermeasures.* Sun, Jiachen, et al. Berkeley, USA : USENIX Security Symposium, 2020.

49. *Dataset Security for Machine Learning: Data Poisoning, Backdoor Attacks, and Defenses.* Goldblum, Micah, et al. Ithaca, USA : arXiv, 2020, Vol. abs/2012.10544.

50. *Just How Toxic is Data Poisoning? A Unified Benchmark for Backdoor and Data Poisoning Attacks.* Schwarzschild, Avi, et al. Vienna, Austria : International Conference on Machine Learning, 2021.

51. *BadNets: Identifying Vulnerabilities in the Machine Learning Model Supply Chain.* Gu, Tianyu, Dolan-Gavitt, Brendan and Garg, Siddharth. New York, USA : IEEE Access, 2017, Vol. 7.

52. *LIRA: Learnable, Imperceptible and Robust Backdoor Attacks.* Doan, Khoa, et al. Montreal, Canada : International Conference on Computer VIsion, 2021.

53. *Label-Consistent Backdoor Attacks.* Turner, Alexander, Tsipras, Dimitris and Madry, Aleksander. Ithaca, USA : arXiv, 2019, Vol. abs/1912.02771.

54. *Hidden Trigger Backdoor Attacks.* Saha, Aniruddha, Subramanya, Akshayvarun and Pirsiavash, Hamed. New York, USA : AAAI Conference on Artificial Intelligence, 2020.

55. *Backdoor Attack with Imperceptible Input and Latent Modification.* Doan, Khoa, Lao, Yingjie and Li, Ping. San Diego, USA : Conference on Neural Information Processing Systems, 2021.

56. *FixMatch: Simplifying Semi-Supervised Learning with Consistency and Confidence.* Sohn, Kihyuk, et al. Vancouder, Canada : Conference on Neural Information Processing Systems, 2020.

57. *Poisoning and Backdooring Contrastive Learning.* Carlini, Nicholas and Terzis, Andreas. Lisbon, Portugal : International Conference on Learning Representations, 2022.

58. *Witches' Brew: Industrial Scale Data Poisoning via Gradient Matching.* Geiping, Jonas, et al. Vienna, Austria : International Conference on Learning Representations, 2021.

59. *Poison Frogs! Targeted Clean-Label Poisoning Attacks on Neural Networks.* Shafahi, Ali, et al. Montreal, Canada : Conference on Neural Information Processing Systems, 2018.

60. *MetaPoison: Practical General-purpose Clean-label Data Poisoning.* Huang, Ronny, et al. Vancouver, Canada : Conference on Neural Information Processing Systems, 2020.

61. *ImageNet Large Scale Visual Recognition Challenge.* Russakovsky, Olga, et al. Amsterdam, Netherlands : International Journal of Computer Vision, 2015, Vol. 115.

62. *Poisoning the Unlabeled Dataset of Semi-Supervised Learning.* Carlini, Nicholas. Berkeley, USA : USENIX Security Symposium, 2021.

63. *Unlearnable Examples: Making Personal Data Unexploitable.* Huang, Hanxun, et al. Vienna, Austria : International Conference on Learning Representations, 2021.

64. *Adversarial Examples Make Strong Poisons.* Fowl, Liam, et al. Sydney, Australia : Conference on Neural Information Processing Systems, 2021.

65. *Benchmarking Neural Network Robustness to Common Corruptions and Perturbations.* Hendrycks, Dan and Dietterich, Thomas. New Orleans, USA : International Conference on Learning Representations, 2019.

66. *On Interaction Between Augmentations and Corruptions in Natural Corruption Robustness.* Mintun, Eric, Kirillov, Alexander and Xie, Saining. Ithaca, USA : arXiv, 2021, Vol. abs/2102.11273.

67. *The Many Faces of Robustness: A Critical Analysis of Out-of-Distribution Generalization.* Hendrycks, Dan, et al. Montreal, Canada : International Conference on Computer Vision, 2021.

68. *Shortcut Learning in Deep Neural Networks.* Geirhos, Robert, et al. Berlin, Germany : Nature Machine Intelligence, 2020, Vol. 2.

69. *Generalisation in Humans and Deep Neural Networks.* Geirhos, Robert, et al. Montreal, Canada : Conference on Neural Information Processing Systems, 2018.

70. *Explaining Explanations: An Overview of Interpretability of Machine Learning.* Gilpin, Leilani, et al. Turin, Italy : International Conference on Data Science and Advanced Analytics, 2018.

71. *Visual Interpretability for Deep Learning: a Survey.* Zhang, Quanshi and Zhu, Song-Chun. Hangzhou, China : Frontiers of Information Technology & Electronic Engineering, 2018, Vol. 19.

72. *A Survey Of Methods for Explaining Black Box Models.* Guidotti, Riccarda, et al. New York, USA : ACM Computing Surveys, 2019, Vol. 51.

73. *Machine Learning Explainability for External Stakeholders.* Bhatt, Umang, et al. Barcelona, Spain : Conference on Fairness, Accountability, and Transparency, 2020.

74. European Commission. *Regulation of the European Parliament and of the Council Laying Down Harmonised Rules on Artificial Intelligence and Amending Certain Union Legislative Acts.* Brussels, Belgium : EUR-Lex, 2021.

75. *Broadening Differential Privacy for Deep Learning Against Model Inversion Attacks.* Zhang, Qiuchen, et al. Atlanta, USA : IEEE Conference on Big Data, 2020.

76. *Towards a Robust and Trustworthy Machine Learning System Development: An Engineering Perspective.* Xiong, Pulei, et al. s.l. : arXiv, 2022.

77. *Developing Privacy-preserving AI Systems: The Lessons learned.* Chen, Huili, et al. s.l. : 57th ACM/IEEE Design Automation Conference (DAC), 2020.

78. *Trustworthy AI Inference Systems: An Industry Research View.* Cammarota, Rosario, et al. s.l. : arXiv, 2020.

79. *Better Together: Privacy-Preserving Machine Learning Powered by Intel® SGX and Intel® DL Boost.* Intel. Santa Clara, USA : Intel Artificial Intelligence Blog, 2021.

80. *On Evaluating Adversarial Robustness.* Carlini, Nicholas, et al. Ithaca, USA : arXiv, 2019, Vol. abs/1902.06705.

81. *Reliable Evaluation of Adversarial Robustness with an Ensemble of Diverse Parameter-free Attacks.* Croce, Francesco and Hein, Matthias. Vienna, Austria : International Conference on Machine Learning, 2020.

82. *Adversarial Training and Robustness for Multiple Perturbations.* Tramèr, Florian and Boneh, Dan. s.l. : Conference on Neural Information Processing Systems, 2019.

83. *Ensemble Adversarial Training: Attacks and Defenses.* Tramèr, Florian, et al. s.l. : International Conference on Learning Representations, 2018.

84. *Distillation as a Defense to Adversarial Perturbations against Deep Neural Networks.* Papernot, Nicolas, et al. s.l. : 37th IEEE Symposium on Security & Privacy, 2016.

85. *Towards Robust Neural Networks via Random Self-Ensemble.* Liu, Xuanqing, et al. s.l. : European Conference on Computer Vision (ECCV), 2018.

86. *Improving Adversarial Robustness via Promoting Ensemble Diversity.* Pang, Tianyu, et al. s.l. : International Conference on Machine Learning (ICML), 2019.

87. *Evading Adversarial Example Detection Defenses with Orthogonal Projected Gradient Descent.* Bryniarski, Oliver, et al. s.l. : International Conference on Learning Representations, 2022.

88. *Feature Squeezing: Detecting Adversarial Examples in Deep Neural Networks.* Xu, Weilin, Evans, David and Qi, Yanjun. s.l. : 25th Annual Network and Distributed System Security Symposium, 2018.

89. *A simple unified Framework for Detecting Out-of-Distribution Samples and Adversarial Attacks.* Lee, Kimin, et al. s.l. : Neural Information Processing Systems (NeurIPS), 2018.

90. *Defense-GAN: Protecting Classifiers against Adversarial Attacks using Generative Models.* Samangouei, Pouya, Kabkab, Maya and Chellappa, Rama. s.l. : International Conference on Learning Representations, 2018.

91. *Defense against Adversarial Attacks using High-Level Representation guided Denoiser.* Liao, Fangzhou, et al. s.l. : IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 2018.

92. *On Evaluating Neural Network Backdoor Defenses.* Veldanda, Akshaj and Garg, Siddharth. Vancouver, Canada : Conference on Neural Information Processing Systems: Workshop on Dataset Curation and Security, 2020.

93. *Spectral Signatures in Backdoor Attacks.* Tran, Brandon, Li, Jerry and Madry, Aleksander. Montreal, Canada : Conference on Neural Information Processing Systems, 2018.

94. *Detecting Backdoor Attacks on Deep Neural Networks by Activation Clustering.* Chen, Bryant, et al. Honululu, USA : AAAI Conference on Artificial Intelligence: SafeAI Workshop, 2019.

95. *Bypassing Backdoor Detection Algorithms in Deep Learning.* Tan, Te Juin Lester and Shokri, Reza. Genoa, Italy : IEEE European Symposium on Security and Privacy, 2020.

96. *Stronger Data Poisoning Attacks Break Data Sanitization Defenses.* Koh, Pang Wei, Steinhardt, Jacob and Liang, Percy. Cham, Switzerland : Machine Learning, 2021, Vol. 111.

97. *What Doesn't Kill You Makes You Robust(er): Adversarial Training against Poisons and Backdoors.* Geiping, Jonas, et al. Vienna, Austria : Internationcal Conference on Computer Vision. Workshop on Security and Safety in Machine Learning Systems, 2021.

98. *Strong Data Augmentation Sanitizes Poisoning and Backdoor Attacks without an Accuracy Tradeoff.* Borgnia, Eitan, et al. Toronto, Canada : IEEE International Conference on Acoustics, Speech, and Signal Processing, 2021.

99. *MaxUp: Lightweight Adversarial Training with Data Augmentation Improves Neural Network Training.* Gong, Chengyue, et al. Nashville, USA : IEEE Conference on Computer Vision and Learning Representations, 2021.

100. *Deep Learning with Differential Privacy.* Abadi, Martin, et al. Vienna, Austria : ACM Conference on Computer and Communications Security, 2016.

101. *On the Effectiveness of Mitigating Data Poisoning Attacks with Gradient Shaping.* Hong, Sanghyun, et al. Ithaca, USA : arXiv, 2020, Vol. abs/2002.11497.

102. *RAB: Provable Robustness Against Backdoor Attacks.* Weber, Maurice, et al. Ithaca, USA : arXiv, 2020, Vol. abs/2003.08904.

103. *Certified Adversarial Robustness via Randomized Smoothing.* Cohen, Jeremy, Rosenfeld, Elan and Kolter, Zico. Long Beach, USA : Internatioanl Conference on Machine Learning, 2019.

104. *Fine-Pruning: Defending Against Backdooring Attacks on Deep Neural Networks.* Liu, Kang, Dolan-Gavitt, Brendan and Garg, Siddharth. Heraklion, Greece : International Symposium on Research in Attacks, Intrusions, and Defenses, 2018.

105. *Neural Cleanse: Identifying and Mitigating Backdoor Attacks in Neural Networks.* Wang, Bolun, et al. San Franscisco, USA : IEEE Symposium on Security and Privacy, 2019.

106. *AugMix: A Simple Data Processing Method to Improve Robustness and Uncertainty.* Hendrycks, Dan, et al. Addis Ababa, Ethiopia : International Conference on Learning Representations, 2020.

107. *Mixup: Beyond Empirical Risk Minimization.* Zhang, Hongyi, et al. Vancouver, Canada : International Conference on Learning Representations, 2018.

108. *Increasing the Robustness of DNNs against Image Corruptions by Playing the Game of Noise.* Rusak, Evgenia, et al. Addisa Ababa, Ethiopia : International Conference on Learning Representations, 2020.

109. *Making Convolutional Networks Shift-Invariant Again.* Zhang, Richard. Long Beach, USA : International Conference on Machine Learning, 2019.

110. *Improving Robustness against Common Corruptions by Covariate Shift Adaptation.* Schneider, Steffen, et al. Vancouver, Canada : Conference on Neural Information Processing Systems, 2020.

111. *Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift.* Ioffe, Sergey and Szegedy, Christian. Lille, France : International Conference on Machine Learning, 2015.

112. *A Less Biased Evaluation of Out-of-Distribution Sample Detectors.* Shafaei, Alireza, Schmidt, Marc and Little, James. Cardiff, United Kingdom : British Machine Vision Conference, 2019.

113. *A Baseline for Detecting Misclassified and Out-of-Distribution Examples in Neural Networks.* Hendrycks, Dan and Gimpel, Kevin. Toulon, France : International Conference on Learning Representations, 2017.

114. *Generalized ODIN: Detecting Out-of-Distribution Image without Learning from Out-of-Distribution Data.* Hsu, Yen-Chang, et al. Seatlle, USA : IEEE Conference on Computer Vision and Pattern Recognition, 2020.

115. *Dropout as a Bayesian Approximation: Representing Model Uncertainty in Deep Learning.* Gal, Yarin and Ghahramani, Zoubin. New York, USA : International Conference on Machine Learning, 2016.

116. *Dropout: A Simple Way to Prevent Neural Networks from Overfitting.* Srivastava, Nitish, et al. New Yotk, USA : The Journal of Machine Learning Research, 2014, Vol. 15.

117. *Simple and Scalable Predictive Uncertainty Estimation using Deep Ensembles.* Lakshminarayanan, Balaji, Pritzel, Alexander and Blundell, Charles. Long Beach, USA : Conference on Neural Information Processing Systems, 2017.

118. *BatchEnsemble: An Alternative Approach to Efficient Ensemble and Lifelong Learning.* Wen, Yeming, Tran, Dustin and Ba, Jimmy. Addis Ababa, Ethiopia : International Conference on Learning Representations, 2020.

119. *Evidential Deep Learning to Quantify Classification Uncertainty.* Sensoy, Murat, Kaplan, Lance and Kandemir, Melih. Montreal, Canada : Conference on Neural Information Processing Systems, 2018.

120. *Cyclical Stochastic Gradient MCMC for Bayesian Deep Learning.* Zhang, Ruqi, et al. Addis Abab, Ethiopia : International Conference on Learning Representations, 2020.

121. *Efficient and Scalable Bayesian Neural Nets with Rank-1 Factors.* Dusenberry, Michael, et al. Vienna, Austria : International Conference on Machine Learning, 2020.

122. *Deep k-Nearest Neighbors: Towards Confident, Interpretable and Robust Deep Learning.* Papernot, Nicolas and McDaniel, Patrick. Ithaca, USA : arXiv, 2018, Vol. abs/1803.04765.

123. *On Calibration of Modern Neural Networks.* Guo, Chuan, et al. Sydney, Australia : International Conference on Machine Learning, 2017.

124. *Learning Confidence for Out-of-Distribution Detection in Neural Networks.* DeVries, Terrance and Taylor, Graham. Ithaca, USA : arXiv, 2018, Vol. abs/1802.04865.

125. *Attribution-Based Confidence Metric For Deep Neural Networks.* Jha, Susmit, et al. Vancouver, Canada : Conference on Neural Information Processing Systems, 2019.

126. *Online Black-Box Confidence Estimation of Deep Neural Networks.* Woitschek, Fabian and Schneider, Georg. Aachen, Germany : IEEE Intelligent Vehicles Symposium (Under Review), 2022.

127. *Causal Interpretability for Machine Learning - Problems, Methods and Evaluation.* Moraffah, Raha, et al. New York, USA : ACM SIGKDD Explorations Newsletter, 2020, Vol. 22.

128. *Explainability of Vision-Based Autonomous Driving Systems: Review and Challenges.* Zablocki, Eloi, et al. Toulouse, France : Dependable & Explainable Learning Workshop on Machine Learning in Certified Systems, 2021.

129. *Explanations in Autonomous Driving: A Survey.* Omeiza, Daniel, et al. New York, USA : IEEE Transactions on Intelligent Transportation Systems, 2021.

130. *Score-CAM: Score-Weighted Visual Explanations for Convolutional Neural Networks.* Wang, Haofan, Wang, Zifan and Mardziel, Piotr. Seattle, USA : IEEE Conference on Computer Vision and Pattern Recognition Workshops, 2020.

131. *Axiomatic Attribution for Deep Networks.* Sundararajan, Mukund, Taly, Ankur and Yan, Qiqi. Sydney, Australia : International Conference on Machine Learning, 2017.

132. *Attribution in Scale and Space.* Xu, Shawn, Venugopalan, Subhashini and Sundararajan, Mukund. Seattle, USA : IEEE Conference on Computer Vision and Pattern Recognition, 2020.

133. *Grad-CAM++: Improved Visual Explanations for Deep Convolutional Networks.* Chattopadhyay, Aditya, et al. Lake Tahoe, USA : IEEE Winter Conference on Applications of Computer Vision, 2018.

134. *Striving for Simplicity: The All Convolutional Net.* Springenberg, Jost Tobias, et al. San Diego, USA : International Conference on Learning Representations Workshops, 2015.

135. *Layer-wise Relevance Propagation for Neural Networks with Local Renormalization Layers.* Binder, Alexander, et al. Barcelona, Spain : International Conference on Artificial Networks, 2016.

136. *Interpretable Explanations of Black Boxes by Meaningful Perturbation.* Fong, Ruth and Vedaldi, Andrea. Venice, Italy : International Conference on Computer Vision, 2017.

137. *Real Time Image Saliency for Black Box Classifiers.* Dabkowski, Piotr and Gal, Yarin. Long Beach, USA : Conference on Neural Information Processing Systems, 2017.

138. *Sanity Checks for Saliency Maps.* Adebayo, Julius, et al. Montreal, Canada : Conference on Neural Information Processing Systems, 2018.

139. *Quick Shift and Kernel Methods for Mode Seeking.* Vedaldi, Andrea and Soatto, Stefano. Marseille, France : European Conference on Computer Vision, 2008.

140. *"Why Should I Trust You?" - Explaining the Predictions of Any Classifier.* Ribeiro, Marco Tulio, Singh, Sameer and Guestrin, Carlos. San Francisco, USA : ACM International Conference on Knowledge Discovery and Data Mining, 2016.

141. *Anchors: High-Precision Model-Agnostic Explanations.* Ribeiro, Marco Tulio, Singh, Sameer and Guestrin, Carlos. New Orleans, USA : AAAI Conference on Artificial Intelligence, 2018.

142. *A Unified Approach to Interpreting Model Predictions.* Lundberg, Scott and Lee, Su-In. Long Beach, USA : Conference on Neural Information Processing Systems, 2017.

143. *Explaining in Style: Training a GAN to explain a classifier in StyleSpace.* Lang, Oran, et al. Montreal, Canada : International Conference on Computer Vision, 2021.

144. *Counterfactual Visual Explanations.* Goyal, Yash, et al. Long Beach, USA : International Conference on Machine Learning, 2019.

145. *GANalyze: Toward Visual Definitions of Cognitive Image Properties.* Goetschalckx, Lore, et al. Seoul, South Korea : International Conference on Computer Vision, 2019.

146. *Understanding Black-box Predictions via Influence Functions.* Koh, Pang Wei and Liang, Percy. Sydney, Australia : International Conference on Machine Learning, 2017.

147. *Model-Agnostic Interpretability with Shapley Values.* Messalas, Andreas, Kanellopoulos, Yiannis and Makris, Christos. Patras, Greece : International Conference on Information, Intelligence, Systems and Applications, 2019.

148. *Causal Learning and Explanation of Deep Neural Networks via Autoencoded Activations.* Harradon, Michael, Druce, Jeff and Ruttenberg, Brian. Ithaca, USA : arXiv, 2018, Vol. abs/1802.00541.

149. *Interpreting CNN Knowledge Via An Explanatory Graph.* Zhang, Quanshi, et al. New Orleans, USA : AAAI Conference on Artificial Intelligence, 2018.

150. *Rule Extraction Algorithm for Deep Neural Networks: A Review.* Hailesilassie, Tameru. New York, USA : International Journal of Computer Science and Information Security, 2016, Vol. 14.

151. *DeepRED – Rule Extraction from Deep Neural Networks.* Zilke, Jan Ruben, Loza, Eneldo and Janssen, Frederik. Bari, Italy : International Conference on Discovery Science, 2016.

152. *Gradient-based Learning Applied to Document Recognition.* LeCun, Yann, et al. New York, USA : Proceedings of the IEEE, 1998, Vol. 86.

153. *Interpretable Learning for Self-Driving Cars by Visualizing Causal Attention.* Kim, Jinkyu and Canny, John. Venice, Italy : International Conference on Computer Vision, 2017.

154. *Neural Additive Models: Interpretable Machine Learning with Neural Nets.* Agarwal, Rishabh, et al. San Diego, USA : Conference on Neural Information Processing Systems, 2021.

155. *Datasheets for Datasets.* Gebru, Timnit, et al. New York, USA : Communication of the ACM, 2021, Vol. 64.

156. *Ensuring Dataset Quality for Machine Learning Certification.* Picard, Sylvaine, et al. Coimbra, Portugal : IEEE International Workshop on Software Certification, 2020.

157. *Model Cards for Model Reporting.* Mitchell, Margaret, et al. Atlanta, USA : ACM Conference on Fairness, Accountability, and Transparency, 2019.

158. *FactSheets: Increasing Trust in AI Services through Supplier's Declarations of Conformity.* Arnold, Matthew, et al. Yorktown Heights, USA : IBM Journal of Research and Development, 2019, Vol. 1.

159. *A Methodology for Creating AI FactSheets.* Richards, John, et al. Ithaca, USA : arXiv, 2020, Vol. abs/1810.03993.

160. Weights & Biases. *https://wandb.ai/site.* [Online] February 2022.

161. DataRobot. *https://www.datarobot.com/.* [Online] February 2022.

162. Neptune Labs. *https://neptune.ai/.* [Online] February 2022.

163. MLflow Project. *https://mlflow.org/.* [Online] February 2022.

164. Hopsworks. *https://www.hopsworks.ai/.* [Online] February 2022.

165. ISO 26262-3:2018 - Road vehicles - Functional safety. *Concept phase.* Geneva, Switzerland : International Organization for Standardization (ISO), 2018.

166. *Practical Solutions for Machine Learning Safety in Autonomous Vehicles.* Mohseni, Sina, et al. New York, USA : AAAI Conference on Artificial Intelligence: Workshop on Safe AI, 2020.

167. *Practical Machine Learning Safety: A Survey and Primer.* Mohseni, Sina, et al. New York, USA : Association for Computing Machinery, 2021, Vol. 1.

168. *Towards Fast Computation of Certified Robustness for ReLU Networks.* Weng, Tsui-Wei, et al. s.l. : CoRR, 2018. 1804.09699.

169. *Rectified Linear Units Improve Restricted Boltzmann Machines.* Nair, Vinod and Hinton, Geoffrey. Haifa, Israel : International Conference on Machine Learning, 2010.

170. *Reluplex: An Efficient SMT Solver for Verifying Deep Neural Networks.* Katz, Guy, et al. abs/1702.01135, s.l. : CoRR, 2017.

171. *Evaluating Robustness of Neural Networks with Mixed Integer Programming.* Tjeng, Vincent, Xiao, Kai and Tedrake, Russ. s.l. : International Conference on Learning Representations, 2019.

172. *Branch and Bound for Piecewise Linear Neural Network Verification.* Bunel, Rudy, et al. abs/1909.06588, s.l. : CoRR, 2019.

173. *SoK: Certified Robustness for Deep Neural Networks.* Li, Linyi, Xie, Tao und Li, Bo. s.l. : ArXIv, 2020.

174. *Optimization and Abstraction: A Synergistic Approach for Analyzing Neural Network Robustness.* Anderson, Greg, et al. abs/1904.09959, s.l. : CoRR, 2019.

175. *A Dual Approach to Scalable Verification of Deep Networks.* Dvijotham, Krishnamurthy, et al. s.l. : CoRR, 2018, Vol. abs/1803.06567. 1803.06567.

176. *Certified Adversarial Robustness with Additive Noise.* Li, Bai, et al. s.l. : CoRR, 2018, Vol. abs/1809.03113. 1809.03113.

177. *Certified Robustness to Adversarial Examples with Differential Privacy.* Lecuyer, Mathias, et al. San Francisco, USA : IEEE Symposium on Security and Privacy, 2019.

178. *PROVEN: Certifying Robustness of Neural Networks with a Probabilistic Approach.* Weng, Tsui-Wei, et al. CoRR : s.n., 2018. 1812.08329.

179. *Randomized Smoothing of All Shapes and Sizes.* Yang, Greg, et al. s.l. : CoRR, 2020. 2002.08118.

180. *Black-Box Certification with Randomized Smoothing: A Functional Optimization Based Framework.* Zhang, Dinghuai, et al. CoRR : s.n., 2020. 2002.09169.

181. *A Framework for Robustness Certification of Smoothed Classifiers using F-Divergences.* Dvijotham, Krishnamurthy, et al. s.l. : ICLR 2020 Conference Blind Submission, 2019.

182. *Boosting Robustness Certification of Neural Networks.* Singh, Gagandeep, et al. s.l. : ICLR 2019 Conference Blind Submission, 2018.

183. *Fast Geometric Projections for Local Robustness Certification.* Fromherz, Aymeric, et al. CoRR : s.n., 2020. 2002.04742.

184. *A Survey of Autonomous Driving: Common Practices and Emerging Technologies.* Yurtsever, Ekim, et al. New York, USA : IEEE Access, 2020, Vol. 8.

185. *Self-Driving Cars: A Survey.* Badue, Claudine, et al. Amsterdam, Netherlands : Expert Systems with Applications, 2021, Vol. 165.

186. *A Survey of Deep Learning Techniques for Autonomous Driving.* Grigorescu, Sorin, et al. Hoboken, USA : Journal of Field Robotics, 2020, Vol. 37.

187. *Trends in camera based Automotive Driver Assistance Systems (ADAS).* Dabral, Shashank, et al. College Station : IEEE International Midwest Symposium on Circuits and Systems, 2014.

188. *A Survey of ADAS Technologies for the Future Perspective of Sensor Fusion.* Ziebinski, Adam, et al. Halkidiki, Greece : International Conference on Computational Collective Intelligence, 2016.

189. *Computer Vision for Autonomous Vehicles: Problems, Datasets and State of the Art.* Janai, Joel, et al. Boston, USA : Foundations and Trends in Computer Graphics and Vision, 2020, Vol. 12.

190. *Experimental Security Research of Tesla Autopilot.* Tencent Keen Security Lab. Shenzhen, China : Tencent, 2019.

191. *Workshop on Autonomous Driving - Tesla Keynote.* Karpathy, Andrej. Nashville, USA : IEEE Conference on Computer Vision and Pattern Recognition, 2021.

192. *How we've built the World's Most Experienced Urban Driver.* Waymo. Mountain View, USA : Waypoint, 2021.

193. *Is it Safe to Drive? An Overview of Factors, Metrics, and Datasets for Driveability Assessment in Autonomous Driving.* Guo, Junyao, Kurup, Unmesh and Shah, Mohak. New York, USA : IEEE Transactions on Intelligent Transportation Systems, 2018, Vol. 21.

194. *Deep Learning for Generic Object Detection: A Survey.* Liu, Li, et al. Amsterdam, Nethrelands : International Journal of Computer Vision, 2019, Vol. 128.

195. *A Survey of Deep Learning Based Object Detection.* Jiao, Licheng, et al. New York, USA : IEEE Access, 2019, Vol. 9.

196. *3D Object Detection for Autonomous Driving: A Survey.* Qian, Rui, Lai, Xin and Li, Xirong. Ithaca, USA : arXiv, 2021, Vol. abs/2106.10823.

197. *Image Segmentation Using Deep Learning: A Survey.* Minaee, Shervin, et al. New York, USA : IEEE Transactions on Pattern Analysis and Machine Intelligence, 2021.

198. *A Survey on Deep Learning Based Approaches for Scene Understanding in Autonomous Driving.* Guo, Zhiyang, et al. Basel, Switzerland : Electronics, 2021, Vol. 10.

199. *Object Scene Flow for Autonomous Vehicles.* Menze, Moritz and Geiger, Andreas. Boston, USA : IEEE Conference on Computer Vision and Pattern Recognition, 2015.

200. *A Survey of Variational and CNN-based Optical Flow Techniques.* Tu, Zhigang, et al. Amsterdam, Netherlands : Signal Processing: Image Communication, 2019, Vol. 72.

201. *Monocular Depth Estimation Based On Deep Learning: An Overview.* Xiaxogang, Ruan, et al. Shanghai, China : Chinese Automation Congress, 2020.

202. *A Survey on Deep Learning Techniques for Stereo-based Depth Estimation.* Laga, Hamid, et al. New York, USA : IEEE Transactions on Pattern Analysis and Machine Intelligence, 2020, Vol. 10.

203. *Survey on Localization Methods for Autonomous Vehicles in Smart Cities.* Chehri, Abdellah, Quadar, Nordine and Rachid, Saadane. Casablanca, Morocco : International Conference on Smart City Applications, 2019.

204. *A Survey of Localization Methods for Autonomous Vehicles in Highway Scenarios.* Laconte, Johann, et al. Basel, Switzerland : Sensors (Under Review), 2022, Vol. 22.

205. *Survey on Vision-based Path Prediction.* Hirakawa, Tsubasa, et al. Las Vegas, USA : Distributed, Ambient and Pervasive Interations, 2018.

206. *Human Motion Trajectory Prediction: A Survey.* Rudenko, Andrey, et al. New York, USA : The International Journal of Robotics Research, 2020, Vol. 39.

207. *Deep Learning for Vision-based Prediction: A Survey.* Rasouli, Amir. Ithaca, USA : arXiv, 2020, Vol. abs/2007.00095.

208. *A Survey of Motion Planning and Control Techniques for Self-driving Urban Vehicles.* Paden, Brian, et al. New York, USA : IEEE Transactions on Intelligent Vehicles, 2016, Vol. 1.

209. *Planning and Decision-Making for Autonomous Vehicles.* Schwarting, Wilko, Alonso-Mora, Javier and Rus, Daniela. San Mateo, USA : Annual Review of Control, Robotics, and Autonomous Systems, 2018, Vol. 1.

210. *A Survey of Deep RL and IL for Autonomous Driving Policy Learning.* Zhu, Zeyu and Zhao, Huijing. Ithaca, USA : arXiv, 2021, Vol. abs/2101.01993.

211. *Deep Reinforcement Learning for Autonomous Driving: A Survey.* Kiran, Bangalore Ravi, et al. New York, USA : IEEE Transactions on Intelligent Transportation Systems, 2021, Vol. 2.

212. *A Survey of Deep Learning Applications to Autonomous Vehicle Control.* Kuutti, Sampo, et al. New York, USA : IEEE Transactions on Intelligent Transportation Systems, 2019, Vol. 12.

213. *End to End Learning for Self-Driving Cars.* Bojarski, Mariusz, et al. Holmdel, USA : NVIDIA Developer Blog, 2016.

214. *ChauffeurNet: Learning to Drive by Imitating the Best and Synthesizing the Worst.* Bansal, Mayank, Krizhevsky, Alex and Ogale, Abhijit. Freiburg, Germany : Robotics: Science and Systems, 2019.

215. *On Offline Evaluation of Vision-based Driving Models.* Codevilla, Felipe, et al. Munich, Germany : European Conference on Computer Vision, 2018.

216. *Towards Fully Autonomous Driving: Systems and Algorithms.* Levinson, Jesse, et al. Baden-Baden, Germany : IEEE Intelligent Vehicles Symposium, 2011.

217. *Making Bertha Drive - An Autonomous Journey on a Historic Route.* Ziegler, Julius, et al. New York, USA : IEEE Intelligent Transportation Systems Magazine, 2014, Vol. 6.

218. *Experience, Results and Lessons Learned from Automated Driving on Germany's Highways.* Aeberhard, Michael, et al. New York, USA : IEEE Intelligent Transportation Systems Magazine, 2015, Vol. 7.

219. *Computing Systems for Autonomous Driving: State of the Art and Challenges.* Liu, Liankai, et al. New York, USA : IEEE Internet of Things Journal, 2021, Vol. 8.

220. Baidu. *https://apollo.auto/developer.html.* [Online] 2022.

221. *All-In-One Drive: A Comprehensive Perception Dataset with High-Density Long-Range Point Clouds.* Weng, Xinshao, et al. Nagoya, Japan : IEEE Intelligent Vehicles Symposium: Workshop on 3D Deep Learning for Automated Driving, 2021.

222. *Man vs. Computer: Benchmarking Machine Learning Algorithms for Traffic Sign Recognition.* Stallkamp, Johannes, et al. San Jose, USA : International Joint Conference on Neural Networks, 2011.

223. *Chinese Traffic Sign Recognition Database.* Huang, Linlin. Beijing, China : School of Electronic and Information Engingeering, 2019.

224. *Vision-based Traffic Sign Detection and Analysis for Intelligent Driver Assistance Systems: Perspectives and Survey.* Mogelmose, Andreas, Trivedi, Mohan Manubhai and Moeslund, Thomas. New York, USA : IEEE Transactions on Intelligent Transportation Systems, 2012, Vol. 13.

225. *Histograms of Oriented Gradients for Human Detection.* Dalal, Navneet and Triggs, Bill. San Diego, USA : IEEE Conference on Computer Vision and Pattern Recognition, 2005.

226. *Pedestrian Detection: A Benchmark.* Dollar, Piotr, et al. Miami, USA : IEEE Conference on Computer Vision and Pattern Recognition, 2009.

227. *Benchmarking Robustness in Object Detection: Autonomous Driving when Winter is Coming.* Michaelis, Claudio, et al. Vancouver, Canada : Conference on Neural Information Processing Systems: Machine Learning for Autonomous Driving Workshop, 2019.

228. *Benchmarking the Robustness of Semantic Segmentation Models.* Kamann, Christoph and Rother, Carsten. Seattle, USA : IEEE Conference on Computer Vision and Pattern Recognition, 2020.

229. *CARLA: An Open Urban Driving Simulator.* Dosovitskiy, Alexey, et al. Mountain View, USA : Conference on Robot Learning, 2017.

230. *AirSim: High-Fidelity Visual and Physical Simulation for Autonomous Vehicles.* Shah, Shital, et al. Zürich, Switzerland : Conference on Field and Service Robotics, 2017.

231. IPG Automotive. *https://ipg-automotive.com/de/produkte-loesungen/software/carmaker/.* [Online] February 2022.

232. NVIDIA. *https://www.nvidia.com/en-us/self-driving-cars/simulation/.* [Online] February 2022.

233. *Learning to Drive from Simulation without Real World Labels.* Bewley, Alex, et al. Montreal, Canada : IEEE International Conference on Robotics and Automation, 2019.

234. *Enhancing Photorealism Enhancement.* Richter, Stephan, AlHaija, Hassan Abu and Koltun, Vladlen. Vienna, Austria : Eurographics, 2021.

235. *Learning from Simulated and Unsupervised Images through Adversarial Training.* Shrivastava, Ashish, et al. Honolulu, USA : IEEE Conference on Computer Vision and Pattern Recognition, 2016.

236. *Unpaired Image-to-Image Translation using Cycle-Consistent Adversarial Networks.* Zhu, Jun-Yan, et al. Venedig, Italy : International Conference on Computer Vision, 2017.

237. *Unpaired Image-to-Image Translation using Adversarial Consistency Loss.* Zhao, Yihao, Wu, Ruihai and Dong, Hao. Glasgow, United Kingdom : European Conference on Computer Vision, 2020.

238. *Image Super-Resolution via Iterative Refinement.* Saharia, Chitwan, et al. Ithaca, USA : arXiv, 2021, Vol. abs/2104.07636.

239. *Palette: Image-to-Image Diffusion Models.* Saharia, Chitwan, et al. Ithaca, USA : arXiv, 2021, Vol. abs/2111.05826.

240. *Deep Unsupervised Learning using Nonequilibrium Thermodynamics.* Sohl-Dickstein, Jascha, et al. Lille, France : International Conference on Machine Learning, 2015.

241. *Automotive Safety and Machine Learning: Initial Results from a Study on How to Adapt the ISO 26262 Safety Standard.* Henriksson, Jens, Borg, Markus and Englund, Cristofer. s.l. : ACM/IEEE 1st International Workshop on Software Engineering for AI in Autonomous Systems, 2018.

242. ISO/PAS 21448. *Road vehicles - Safety of the intended funtionality.* Geneva, Switzerland : International Organization for Standardization (ISO), 2019.

243. Aptiv, et al. Safety First for Automated Driving. [Online] 2019. https://www.press.bmwgroup.com/global/article/attachment/T0298103EN/434404.

244. *Organization of Machine Learning based Product Development as per ISO 26262 and ISO/PAS 21448.* Radlak, Krystian, et al. s.l. : 25th {IEEE} Pacific Rim International Symposium on Dependable Computing PRCD, 2020.

245. ISO. ISO/FDIS 21448. [Online] 2022. https://www.iso.org/standard/77490.html.

246. ISO 26262-4:2018 - Road vehicles - Functional safety. *Product development at the system level.* Geneva, Switzerland : International Organization for Standardization (ISO), 2018.

247. ISO 26262-6:2018 - Road vehicles - Functional safety. *Product development at the software level.* Geneva, Switzerland : International Organization for Standardization (ISO), 2018.

248. ANSI/UL 4600. *Standard for Safety for the Evaluation of Autonomous Products.* Washington, D.C. : American National Standards Institute (ANSI), Underwriters Laboratories (UL), 2022. 2.

249. ISO/SAE 21434:2021. *Road vehicles - Cybersecurity engineering.* Geneva, Switzerland : International Organization for Standardization (ISO), SAE, 2021. 1.

250. UNECE R 155. *Uniform provisions concerning the approval of vehicles with regards to cyber security and cyber security management system.* Geneva, Switzerland : United Nations Economic Commission for Europe (UNECE), 2021.

251. Kaur, Prabhjot, et al. A Survey on Simulators for Testing Self-Driving Cars. *CoRR.* 2021, abs/2101.05337.

252. ISO 17387. *Intelligent Transport Systems –Lane Change Decision Aid Systems (LCDAS) – Performance Requirements and Test Procedures.* Geneva, Switzerland : International Organization for Standardization (ISO), 2008.

253. ISO 19377. *Heavy commercial vehicles and buses — Emergency braking on a defined path — Test method for trajectory measurement.* Geneva, Switzerland : International Organization for Standardization (ISO), 2017.

254. ISO 19237. *Intelligent transport systems — Pedestrian detection and collision mitigation systems (PDCMS) — Performance requirements and test procedures.* Geneva, Switzerland : International Organization for Standardization (ISO), 2017.

255. ISO 22078. *Intelligent Transport Systems — Bicyclist Detection And Collision Mitigation Systems (BDCMS) — Performance requirements and test procedures.* Geneva, Switzerland : International Organization for Standardization (ISO), 2020.

256. ISO 3888-2. *Passenger Cars— Test Track For A Severe Lane-Change manoeuvre — Part 2: Obstacle Avoidance.* Geneva, Switzerland : International Organization for Standardization (ISO), 2011.

257. ISO 22735. *Road vehicles — Test method to evaluate the performance of lane-keeping assistance systems.* Geneva, Switzerland : International Organization for Standardization (ISO), 2021.

258. ISO 11270. *Intelligent transport systems — Lane keeping assistance systems (LKAS) — Performance requirements and test procedures.* Geneva, Switzerland : International Organization for Standardization (ISO), 2014.

259. ISO/SAE PAS 22736. *Taxonomy and definitions for terms related to driving automation systems for on-road motor vehicles.* Geneva, Switzerland : International Organization for Standardization (ISO), 2021.

260. ISO 19638. *Intelligent Transport Systems — Road Boundary Departure Prevention Systems (RBDPS) — Performance Requirements And Test Procedures.* Geneva, Switzerland : International Organization for Standardization (ISO), 2018.

261. ISO 21717. *Intelligent Transport Systems — Partially Automated In-Lane Driving Systems (PADS) — Performance Requirements And Test Procedures.* Geneva, Switzerland : International Organization for Standardization (ISO), 2018.

262. ISO 21202. *Intelligent Transport Systems — Partially Automated Lane Change Systems (PALS) — Functional / Operational Requirements And Test Procedures.* Geneva, Switzerland : International Organization for Standardization (ISO), 2020.

263. ISO 15622. *Intelligent Transport Systems — Adaptive Cruise Control Systems — Performance Requirements And Test Procedures.* Geneva, Switzerland : International Organization for Standardization (ISO), 2018.

264. ISO 20035. *Intelligent Transport Systems — Cooperative Adaptive Cruise Control Systems (CACC) — Performance Requirements And Test Procedures.* Geneva, Switzerland : International Organization for Standardization (ISO), 2019.

265. ISO 15622. *Transport Information And Control Systems — Adaptive Cruise Control Systems — Performance Requirements And Test Procedures.* Geneva, Switzerland : International Organization for Standardization (ISO), 2002.

266. ISO/TR 22086-1. *Intelligent Transport Systems (ITS) — Network Based Precise Positioning Infrastructure For Land Transportation — Part 1: General Information And Use Case Definitions.* Geneva, Switzerland : International Organization for Standardization (ISO), 2019.

267. ISO/TS 21176. *Cooperative Intelligent Transport Systems (C-ITS) — Position, Velocity And Time Functionality In The Its Station.* Geneva, Switzerland : International Organization for Standardization (ISO), 2020.

268. ISO/TR 16786. *Intelligent transport systems — The use of simulation models for evaluation of traffic management systems — input parameters and reporting template for simulation of traffic signal control systems.* Geneva, Switzerland : International Organization for Standardization (ISO), 2015.

269. ISO 22741. *Intelligent transport systems — Roadside modules AP-DATEX data interface.* Geneva, Switzerland : International Organization for Standardization (ISO), 2022.

270. ISO/AWI TS 5283. *Road Vehicles — Ergonomic aspects of driver monitoring and system interventions in the context of automated driving.* Geneva, Switzerland : International Organization for Standardization (ISO), Under development.

271. ISO 19206-3. *Road Vehicles — Test devices for target vehicles, vulnerable road users and other objects, for assessment of active safety functions — Part 3: Requirements for passenger vehicle 3D targets.* Geneva, Switzerland : International Organization for Standardization (ISO), 2021.

272. ISO/TS 18506. *Procedure to construct injury risk curves for the evaluation of road user protection in crash tests.* Geneva, Switzerland : International Organization for Standardization (ISO), 2014.

273. SAE J2400_200308. *Human Factors in Forward Collision Warning Systems: Operating Characteristics and User Interface Requirements.* Warrendale, USA : SAE, 2003.

274. SAE J3029_201510. *Forward Collision Warning and Mitigation Vehicle Test Procedure - Truck and Bus.* Warrendale, USA : SAE, 2015.

275. SAE J3048_201602. *Driver-Vehicle Interface Considerations for Lane Keeping Assistance Systems.* Warrendale, USA : SAE, 2016.

276. SAE J2808_201701. *Lane Departure Warning Systems: Information for the Human Interface.* Warrendale, USA : SAE, 2017.

277. SAE J3240. *Passenger Vehicle Lane Departure Warning and Lane Keeping Assistance Systems Test Procedure.* Warrendale, USA : SAE, Under development.

278. SAE: J2399_202110. *Adaptive Cruise Control (ACC) Operating Characteristics and User Interface.* Warrendale, USA : SAE, 2021.

279. SAE J2365. *Calculation and Measurement of the Time to Complete In-Vehicle Navigation and Route Guidance Tasks.* Warrendale, USA : SAE, Under development.

280. SAE J2678_201609. *Navigation and Route Guidance Function Accessibility While Driving Rationale.* Warrendale, USA : SAE, 2016.

281. SAE J3114_201612. *Human Factors Definitions for Automated Driving and Related Research Topics.* Warrendale, USA : SAE, 2016.

282. SAE J2396_201705. *Definitions and Experimental Measures Related to the Specification of Driver Visual Behavior Using Video Based Techniques.* Warrendale, USA : SAE, 2017.

283. SAE J2944_201506. *Operational Definitions of Driving Performance Measures and Statistics.* Warrendale, USA : SAE, 2015.

284. SAE J2945/A. *Standard for Lane-Level and Road Furniture Mapping for Infrastructure-based V2X Applications.* Warrendale, USA : SAE, Under development.

285. SAE J2945/9. *Vulnerable Road User Safety Message Minimum Performance Requirements.* Warrendale, USA : SAE, Under development.

286. SAE J3134_201905. *Automated Driving System (ADS) Marker Lamp.* Warrendale, USA : SAE, 2019.

287. UNECE R 79. *Steering Equipment.* Geneva, Switzerland : United Nations Economic Commission for Europe (UNECE), 2008.

288. UNECE R 157. *UN Automated Lane Keeping Systems (ALKS).* Geneva, Switzerland : United Nations Economic Commission for Europe (UNECE), 2021.

289. UNECE GTR 9. *Pedestrian Safety.* Geneva, Switzerland : United Nations Economic Commission for Europe (UNECE), 2008.

290. Bielik, Pavol, et al. *Reliability Assessment of Traffic Sign Classifiers.* Bonn, Germany : Federal Office for Information Security, 2020.

291. Indaheng, Francis, et al. A Scenario-Based Platform for Testing Autonomous Vehicle Behavior Prediction Models. *CoRR.* 2021, abs/2110.14870.

292. *Deep Residual Learning for Image Recognition.* He, Kaiming, et al. Boston, USA : IEEE Conference on Computer Vision and Pattern Recognition, 2015.

293. *ImageNet Classification with Deep Convolutional Neural Networks.* Krizhevsky, Alex, Sutskever, Ilya und Hinton, Geoffrey. Lake Tahoe, USA : Conference on Neural Information Processing Systems, 2012.

294. Martin, Robert Cecil. *Clean Code: A Handbook of Agile Software Craftsmanship.* New Jersey, USA : Prentice Hall, 2008.

295. Woitschek, Fabian. AIMobilityAuditPrep Documentation. *Sphinx Documentation of the Toolbox.* 2022.