AI Perspectives

RESEARCH

Open Access

Corner cases in machine learning processes

Florian Heidecker[1*], Maarten Bieshaar[2] and Bernhard Sick[1]

Abstract

Applications using machine learning (ML), such as highly autonomous driving, depend highly on the performance of the ML model. The data amount and quality used for model training and validation are crucial. If the model cannot detect and interpret a new, rare, or perhaps dangerous situation, often referred to as a corner case, we will likely blame the data for not being good enough or too small in number. However, the implemented ML model and its associated architecture also influence the behavior. Therefore, the occurrence of prediction errors resulting from the ML model itself is not surprising. This work addresses a corner case definition from an ML model's perspective to determine which aspects must be considered. To achieve this goal, we present an overview of properties for corner cases that are beneficial for the description, explanation, reproduction, or synthetic generation of corner cases. To define ML corner cases, we review different considerations in the literature and summarize them in a general description and mathematical formulation, whereby the expected relevance-weighted loss is the key to distinguishing corner cases from common data. Moreover, we show how to operationalize the corner case characteristics to determine the value of a corner case. To conclude, we present the extended taxonomy for ML corner cases by adding the input, model, and deployment levels, considering the influence of the corner case properties.

Keywords Nature of corner case, Corner cases definition, Machine learning, Taxonomy

Introduction

With the spread of machine learning (ML) methods, the number of critical systems, such as highly automated driving, medicine, and aviation, relying on artificial intelligence is increasing [1]. The influence of ML methods is not surprising considering their impressive performance, e.g., object detection in images [2, 3], speech recognition [4], large language models [5], and other applications. Nevertheless, all these tasks have something in common as they rely on data, which is often imbalanced or incomplete, and the labels can be inaccurate or inconsistent [6, 7]. Interpreting and modeling the epistemic and aleatoric uncertainty [8] of an ML model with techniques such as MC-Dropout [9], Bayes by Backprop [10], Prior Networks [11], and Deep Ensemble [12] to acquire a trusted and valid decision is a key challenge in ML applications.

If we want to develop an application using ML methods, data is always essential for model training, validation, and testing within the development cycle. Databases such as [13–15] provide many different datasets for various applications. For many smaller application areas and less noticed niches, a dataset with an appropriate quantity of labels is not available, and it is necessary to record data or generate a synthetic dataset. Both approaches are legitimate because there is no other way to obtain data for developing ML models. However, no matter where the data come from, they usually have the same problem. The collected data only represents a part of reality and has a particular view. The data variability in real-world scenarios is enormous, and the collected dataset contains some or many, but usually not all. In highly automated driving scenarios, for example, location- and country-specific factors such as regulations, local behavior, and visual differences, e.g., signs and symbols, limit the completeness of a single dataset.

*Correspondence:
Florian Heidecker
florian.heidecker@uni-kassel.de
[1] Intelligent Embedded Systems, University of Kassel, Wilhelmshöher Allee 71 - 73, 34121 Kassel, Germany
[2] Bosch Center for Artificial Intelligence, Robert Bosch GmbH, Robert-Bosch-Straße 200, 31132 Hildesheim, Germany

Automotive datasets reveal this problem very well. *BDD100k* [16], *nuScenes* [17], and *Waymo Open* [18], for example, were recorded in the USA, *KITTI* [19] and *A2D2* [20] are from Germany, and *ApolloScape* [21] is from China. *Mapillary* [22] has a slightly more global footprint, with a small amount of footage from South America, Asia, and Africa and a higher proportion from the USA, Europe, and Japan. Data with rainy weather, bright daylight, or darkness during nighttime vary widely, with more and more datasets covering these conditions. Other conditions, such as winter scenes, are far less frequently represented in the datasets such as *BDD100K* [16]. The *Canadian Adverse Driving Conditions* [23] dataset has, for example, a strong focus on winter scenes but again has other deficiencies. This problem can also be found in other domains and ML applications, which may be more or less dominant.

If the trained ML model is then deployed in an application, an erroneous behavior may occur sooner or later, resulting from corner cases that were not considered or samples wrongly labeled [6, 7] during development and testing for whatever reason. Due to the need to improve the ML model's performance, detecting corner cases is becoming increasingly important for safety reasons. Besides, safety-related aspects in critical systems, such as highly automated driving, significantly influence how and where ML models are used and whether the user accepts them.

As we have a strong relationship with highly automated driving, many of the examples we have chosen come from this area but are not limited to it. Consequently, relationships, comparable examples, or the transferability of content to other areas are possible. However, what do we actually mean by a corner case? – We have noticed that everyone has their interpretation and understanding [24–28]. This includes corner case examples, associated properties, and definitions regarding a corner case, some of which can be very different. In highly automated driving, we consider cases as corner cases where the model exhibits erroneous behavior. In Fig. 1, for example, a detection model for vulnerable road users detects the image of a person on the car's advertising banner as a real person. Technically speaking, the ML model did nothing wrong by detecting the person in the advertisement. Yet, the person is not real. Figure 2 provides some more ML corner case examples, which occur in the applied model itself due to a lack of knowledge or because the model has never encountered a comparable sample before, which refers to epistemic uncertainty. Otherwise, adversarial samples are also a type of ML corner case, as a small change in the input can change the result even if it is invisible to humans. These aspects are essential and should be included in an ML corner case definition. How to define the term corner cases with respect to ML is not yet conclusively clarified. In the literature, there are some starting points from software and hardware testing [24, 25] and some definitions for ML perspective [26–35].

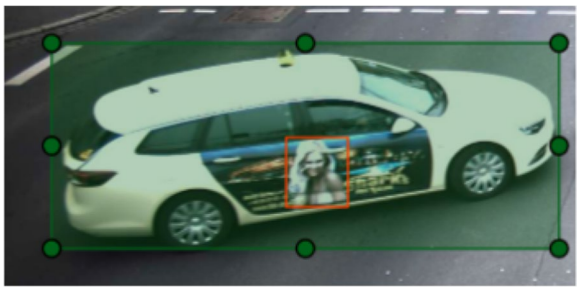

Fig. 1 Corner case example: The image shows an advertisement with a person printed on a cab. The person in the advertisement is wrongly classified and represents a corner case for the ML model because the person is not real

In this article, we approach the topic differently and first discuss the nature, i.e., peculiarities and properties, of ML corner cases, whereby we have not yet seen the bandwidth of properties in any other article. On the one hand, we see the human view on corner cases. However, in particular, we also look at each of the listed properties from the perspective of an ML model. Definitions of corner cases are not new but usually have a strong connection to an application [26–28]. Instead, we aim for a more general corner case description, which is mainly based on the particularities and properties of a corner case. To support this verbal description, we present our mathematical definition of a corner case and recap it on a toy example to show the influence regarding training, testing, deployment, and the importance of model calibration. To get an overview of where corner cases can occur, we provide an extended and, in our view, comprehensive taxonomy of corner cases, including hardware, physics, data, methods, and much more.

To put it in a nutshell, the main contributions of this article are:

- Peculiarities and properties to describe the nature of ML corner cases.
- General corner case description and mathematical corner case definition illustrated on a toy example.
- Providing various corner case examples during the discussion of their nature, description, definition, and taxonomy.

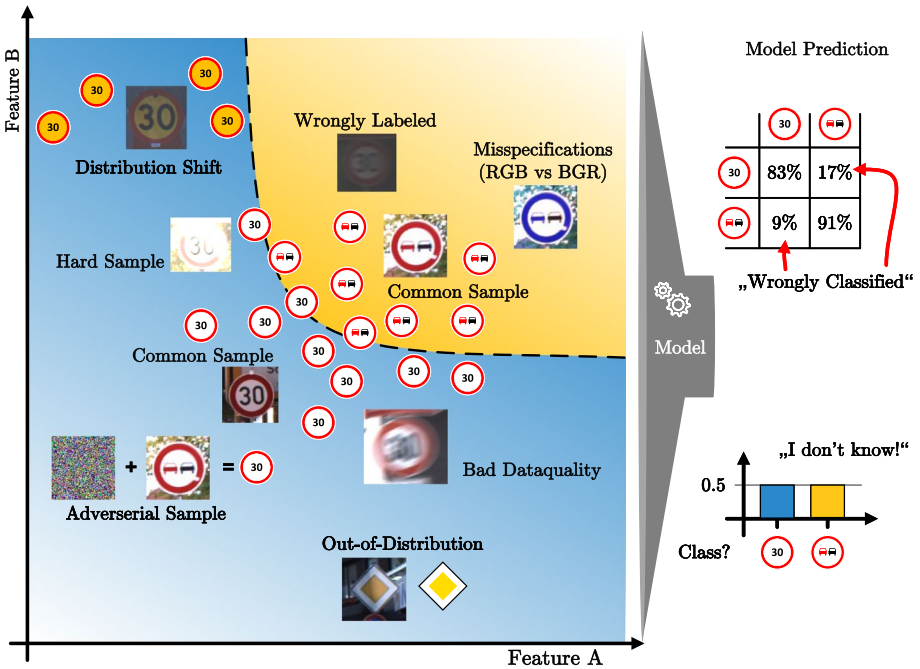

Fig. 2 Corner cases of an ML model that result from the model itself or the model architecture are currently not considered much. However, to use an ML model in a critical application, the corner cases that cause incorrect behavior are needed for training and validation. This graphic shows a classification problem with two classes (speed limit 30 and no passing sign) highlighted by a blue and yellow area and divided by a decision boundary (dashed line). Besides the abstract traffic sign symbol, we have added some real samples of German and Scandinavian traffic signs [36, 37] to have a better impression. Some are common samples, and a few show different ML "corner cases" that could appear in the data. However, the question arises, which samples are a corner case from the point of view of the ML model and deserve the term corner case

The remainder of this article is structured as follows: Section "Nature of corner cases" provides an overview of peculiarities or properties that represent and characterize corner cases. In section "Corner case definition", we present our definition of corner cases in an ML process together with a mathematical formulation and suitable example. Section "Quantitative assessment of corner cases" covers the quantitative assessment of corner cases and presents an answer to possible metrics to measure the significance. The taxonomy for corner cases follows in section "Taxonomy of corner cases". Finally, section "Conclusion" summarizes the article's key message.

Nature of corner cases

Corner cases are by nature not directly tangible, and no one would consider them ordinary. However, any developer of ML algorithms could give several examples representing a corner case related to their application in no time. Describing a corner case is far more challenging [38] as they are difficult to describe precisely, but based on their peculiarities and properties, they can be described more efficiently. For this purpose, we have started to collect peculiarities and properties of how corner cases are described and sorted them as they refer to different characteristics of corner cases. While collecting, we also noticed that in discussions, a corner case was often described as a novelty or unknown data sample, with frequent references to uncertainty.

To get an overview of the different peculiarities and properties of corner cases, we would like to take a closer look at the nature of corner cases and discuss their properties. In Fig. 3, we provide an overview of all the properties we identified and discuss further below. As we sorted and cataloged the various corner case properties, it became apparent that some properties generally describe the corner case data sample, and others require a related ML model or application. We divide corner case properties into three groups:

- Yellow visualizes the first group and represents corner case properties at the level of data samples, where the properties refer to characteristics of the corner case data.

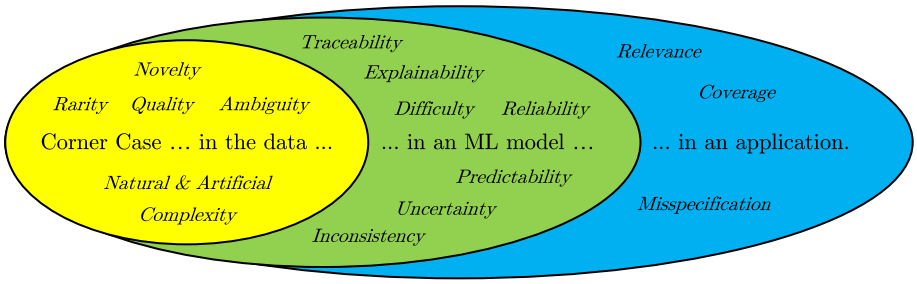

Fig. 3 Peculiarities and properties of corner cases in the data, ML model, and application

- Green is the second group of properties, which is more related to ML methods and indirectly describes the corner case character, i.e., a property of the model that results from the corner case.
- Blue represents the last group of properties. These properties only make sense if the corner case is considered together with an ML model and an application task.

Furthermore, the property under consideration can always be examined from two perspectives: (1) The human point of view and (2) the perspective of the ML model.

Even if the following examples come almost exclusively from the automotive sector and represent mostly image-related corner cases for better comprehensibility, the peculiarities and properties are also generalizable to other domains. In highly autonomous driving, there is a wide range of sensor modalities such as LiDAR, RADAR, ultrasonic or motion data, e.g., GPS, besides the camera, which have different types of data and thus corner cases. From our point of view, the peculiarities and properties listed here can easily be transferred to other sensor modalities and applications.

Data corner case properties

Data corner case properties can be assigned to a single data sample and describe why the sample is a corner case and how it behaves or is characterized. The properties can be viewed from both the human and the ML model perspective.

Novelty

From a human point of view, novelty is a fascinating property because it is primarily associated with something new or innovative, but actually, it depends on the person's knowledge, and even something unknown but ancient can be perceived as a novelty.

For ML, novelty is also essential, especially novelty detection is a crucial task [39, 40]. The novelty property describes whether the data sample can be classified as, e.g., common, rare, exotic, or even unknown for a given ML model. Common data is already used many times during the model training and represents no longer any value for the model. Rare or exotic data samples are partly known to the model, as the amount in the training dataset is relatively low, leading to problems and errors when recognizing these or similar data samples. Some caution is needed with unknown data samples because the model has never processed this data sample, and it is impossible to determine what will happen. Therefore, novel data samples and the analysis of whether they are part of the already known data (in-distribution) or not (out-of-distribution) [41, 42] are of great interest for model training and the validation of ML models.

From the human point of view, this corresponds to a person's knowledge that is learning something new. However, the ML and human points of view can be contradictory, for example, in the case of an adversarial sample [43], where the content is still evident for the human but not for the ML model.

Ambiguity

Another property that characterizes a corner case is ambiguity or indefinability. Ambiguities are omnipresent from a human point of view, whether in language, jokes, symbolism, or even in things composed of two unique parts, e.g., trikes.

Nevertheless, from the point of view of ML models, e.g., classification, this can lead to major issues. While many data samples are unique and distinguishable, and recognized by the ML model, a clear assignment or classification of an ambiguous data sample is nearly impossible for the ML model. For example, ML models trained to classify objects can easily distinguish

a four-wheeler from a motorbike but have difficulties assigning a trike or reverse trike to one of their known classes.

However, ambiguities can also be found in almost any data type, including time series and motion patterns. The critical thing about this property is that the ML model prediction is based on a false assumption that leads to a fateful decision in the worst case. From a human point of view, this problem is much more minor since we incorporate or transfer additional knowledge of the original components, and the decision is not alone based on the shape or same condition.

Natural and artificial

Corner cases can be assigned the property "natural" if the origin is due to a random event or progressive development. Thus, objects with changed design, e.g., a historical car from the 20th century vs. a car today or products that did not exist at the time of the development of the ML model, all naturally emerged and existed in reality. This circumstance is also true for other data samples affected by constant change. In addition, all possible cases appear in highly automated driving and other areas. The question is only when and how critical they are for the involved people.

On the other hand, corner cases can be described as artificial, where the data is synthesized, e.g., synthetic data generation [44, 45], or deliberately manipulated by a human hand. The influence on the ML model and prediction can be positive or negative, whereby adverse effects are of considerably higher importance. In the literature, influencing the model by artificially altering the data examples is called an adversarial attack [43]. This attack can be done intentionally for testing purposes or maliciously to achieve a specific goal. Due to the targeted manipulation of the data and thus the influence on the model, an adversarial attack can also be understood in such a way that it does not count as a classic corner case.

From an ML point of view, there is no distinction between natural and artificial because the ML model receives data, and most detection models do not distinguish between real and synthetic data. However, from a human perspective, we always distinguish on the visual level. As tools become more advanced to generate synthetic data [46], the distinction becomes more complicated for humans, comparable to the ML perspective, as we can not separate real and synthetic without help. Besides, specific artificial effects like adversarial samples are not directly evident to a human but, as mentioned before, could have a high impact on ML models.

Quality

Quality is a widely used property for all kinds of things from a human point of view and is often decisive for how something can be further used. That can also be observed in ML models where the quality of the data and the available annotations are of essential importance for model training. In terms of the data quality [47–49], these can be outliers that may have nothing to do with the actual dataset or external influences that permanently degrade the data, e.g., overlays by other signals, or overexposure and motion blur in case of cameras. Another source of corner cases in ML is incorrect, noisy, or erroneous labeled data, which can lead to devastating performance drops of the ML model [6, 7, 50]

Complexity

Complexity is seen as something that is multi-layered from a human perspective and includes many factors that cannot be captured unambiguously and clearly in their entirety as many interactions come together. The same view accounts for ML because the more factors influence the data, the more difficult it can be for an ML model to extract specific information from all the available information. Often we find more possible corner cases in highly complex situations as multiple influences come together. A similar connection can be found in section "Corner case definition", where Houben et al. [24] defines corner cases as the result of a combination of several influences, which increases complexity.

Rarity

Rarity describes something of low number, quantity, or frequency that is sought after and is usually reflected in many collectibles. The human and ML perspective is quite similar, where two points come together. First, the frequency with which an effect/situation occurs in reality, e.g., a corner case caused by a sun glare in the windshield during sunrise, is more frequent than a lightning strike, and second, the difficulty of recording the required data to train and validate ML models. However, compared to the novelty property (cf. section "Novelty") a rare data sample is not necessarily new or unknown but rather something well-known that is not easy or expensive to obtain.

ML model corner case properties

ML model corner case properties describe the behavior of the ML model. Corner cases influence these properties and can be described using the ML model properties.

Traceability

The traceability of a corner case is an essential property that contributes significantly to the human view of corner cases in ML models. The property indicates whether the cause of an occurred corner case is traceable. For example, if the ML model does not recognize a pedestrian with a green shirt in front of a green hedge, it is evident that the ML model has problems separating the two objects. Therefore, traceability is an essential property for humans to understand and describe the connection of data, specifically corner cases and ML models. In terms of reproducing, synthetically generating, or solving the corner case, it also indicates that the connection can be observed in some way by a human. At the same time, the explainability of the corner case may still be unresolved.

Explainability

Closely related to traceability is explainability. The explainability of a corner case is also an essential property from the viewpoint of a human. It enables the developer to understand and explain how the corner case occurs and which side effects have caused it [38]. The earlier mentioned corner case example of a green hedge and a pedestrian with a green shirt in front (cf. section "Traceability") may be traceable, but it is not necessarily explainable how the ML model comes to this decision. Patterns and situations representing a corner case that cannot be explained or retraced are hard to solve because the ML model's problem with the data is not apparent to a human.

Difficulty

Besides the complexity in the data, there is also the property difficulty for an ML model of learning certain things [51], so recognizing something in complex sequences can be challenging (cf. section "Complexity"). The property difficulty also applies to other data corner case properties, such as difficult because of poor quality (cf. section "Quality") or ambiguity (cf. section "Ambiguity"). In all these cases, difficult refers to an ML model corner case, as the ML model has difficulties with the data. The task to be learned can also be difficult because, e.g., an extreme imbalance of the classes in the data or extreme noise makes learning more challenging. Therefore, the difficulty property is closer to the ML Model than the data. Hence we are listing it as an ML corner case property.

Predictability

From human and ML perspectives, corner cases are equivalent in terms of predictability, with corner cases being characterized as predictable, challenging to predict, or completely unpredictable. The predictability is based on the forecast that the next state is predictable by the previous state. That means the lower the entropy, the higher the predictability of the ML model output [52]. An example is a pedestrian who suddenly runs onto the street, although the pedestrian was on the way into a building shortly before and has not shown any signs. Compared to another pedestrian who looks around to the right and left and then steps onto the street, which is considered easy in terms of predictability.

Besides, predictability also has limits, e.g., corner cases caused by hardware failure are mostly impossible to predict as the ML model is confronted suddenly, and only damage limitation can be done if possible. Comparable to a human.

Uncertainty

Insufficient or incomplete knowledge about a process or procedure causes uncertainty and represents another characteristic of corner cases. For example, a pedestrian walking along the side of the road gets scared. For an ML model, estimating what will happen, whether a pedestrian stops, crouches, jumps onto the road, or where the pedestrian will walk, is impossible. In this example, the ML model faces two types of uncertainty which could be part of corner cases: Aleatoric uncertainty [8], which describes the randomness in the data about which one of the mentioned activities will be performed by the pedestrian, and epistemic uncertainty [8] in the ML model due to incomplete knowledge. However, if we had all-encompassing knowledge, i.e., if we could look into the pedestrian's head, there should be theoretically no corner cases. Nevertheless, as this is not the case and we always look only at the visible behavior, there will always be corner cases. Therefore, knowing in which environment the ML model is deployed and which influences/knowledge are included or excluded in the consideration is essential.

Reliability

Reliability [53, 54] is another property that characterizes an ML model and leads to a decrease in end-user confidence if the ML model fails on a corner case sample. This is also the human perspective because corner cases may occur, and the reliability of an ML model may be reduced as a consequence. For example, an ML model continuously detects, classifies, or predicts the accurate result even on a slightly varying data sample. Eventually, a new data sample variation occurs, leading to the wrong prediction, i.e., the ML Model reliability

decreases as it can not handle all data variations. The interesting thing is to know with which frequency this happens in order to make a statement about the reliability of the ML model.

Inconsistency

Inconsistency is a unique property requiring an ML model to receive data pairs or sequences. The expectation is that similar model inputs will follow similar model outputs. For example, the model receives similar data from two points in time, e.g., two consecutive images where a vehicle is displayed with a slight shift. The response of the ML model should be consistent and not give inconsistent prediction results. This behavior can also be applied to the feature space, e.g., a variational autoencoder [41] that has learned a data representation, where similar inputs should also be mapped to similar representations.

Another inconsistency appears from multiple data sources, such as several sensors observing the same situation simultaneously. For example, a camera and a LiDAR sensor observe the same situation. A pedestrian in the field of view of both sensors should also be present in the data of both sensors. However, in case of inconsistencies, the ML model receives useless data or inconsistent data pairs, where for example, the pedestrian is only recognized in the image but not in the LiDAR data. If the model detects these inconsistencies, the question arises regarding which sensor we can trust and how reliable the model's results are. This problem occurs when multiple data sources provide contradictory information, and several models produce different predictions for the same data. However, comparing or fusing and thereby detecting inconsistency is an excellent method to detect these corner cases [33].

Application corner case properties

At last, we consider corner case characterization at the application level, including an ML model and the data.

Relevance

Relevance represents the importance or significance of something to achieve a specific goal. In terms of ML, it indicates the importance or significance of a data sample, e.g., corner case, for an application task at hand [28]. A cyclist, only a few meters in front of the car, is more relevant for object detection from the safety point of view than a cyclist far away. Besides describing the relevance of a corner case for an application task, specific events or situations can also be classified as irrelevant to the current task.

Another term used in software validation and assurance is criticality [55], where different rare and life-threatening situations are used to test the ML model concerning safety-critical aspects. However, in our opinion, the term critical, e.g., critical situation, is covered by property relevance already as they are relevant for the application task.

Coverage

When running an ML model in an open-world scenario, events or situations may occur that were only partially or not considered while validating and testing the ML model. The property coverage describes this relationship and is usually specified by the Operational Design Domain [56]. That formalizes the scenarios covered by the ML model, e.g., via systematic testing [57]. It also represents the perspective of a human as it refers mainly to the semantics of the data. However, from an ML perspective, we could also abstractly relate it to the distribution in the input space, which comprises the coverage of the input features by training and testing the ML model.

In the software validation use case, predefined scenario samples are used to test the ML model. These predefined scenarios often contain critical events or situations the ML model must pass successfully. The degree of coverage [57] within these test sets is relevant because the test cases should cover a spectrum of critical cases as wide as possible.

Misspecification

Some corner cases can already be derived from the model specification or result from misspecification [34]. For example, a model trained and validated on a particular dataset can only be applied to other data to a limited extent. There is a whole research area on domain adaptation [58, 59], which tries to close this gap and make the model transferable. However, it remains to be seen to what extent this will lead to further corner cases. In addition, the transferability of the test results is impossible and must be repeated each time based on the appropriate specification. Therefore, it is crucial to know for which application the ML model was developed and under which conditions it will operate.

Besides the change from one dataset to another, a significant change within a dataset can occur, e.g., when the road is suddenly covered with snow, which is comparable and called a concept shift [39]. The same applies for out-of-distribution samples when an unknown object appears in the data, e.g., a new class such as an e-scooter, which is not considered by the previous specification and thus represents a corner case for the ML model.

Corner case definition

In this section, we review the corner case definitions found in the literature. Based on this, we present a new and encompassing definition of corner cases in ML models. We also provide a mathematical description of our definition together with an example.

Overview of corner case definitions

Edge cases, rarely also called extreme or boundary cases, represent rare situations or parameters already considered during development [25, 33]. Therefore, they have already lost their edge case status [25] in the running system after they have been resolved. The term corner case has a slightly different meaning depending on its origin and application domain. Houben et al. [24] picks up two essential meanings of corner cases: (1) corner cases represent situations that result from a rare combination of input parameters [25, 60], (2) corner cases are situations caused by an erroneous, malfunction, or incorrect behavior of the ML model [26, 29–32, 34, 35]. In addition, a list of systematic errors in ML is given in [34], such as Monte Carlo approximation, data issues, and sampling class-conditional, which may well represent corner cases. [35] goes even further and provides a total of five error factors covering variability in real situations, errors and noise in measurement systems, errors in the model structure, errors in the training procedure, and errors caused by unknown data, which corresponds well with our taxonomy of corner cases (cf. section "Taxonomy of corner cases"). Bolte et al. [27] is more specific and states related to image-based corner cases detection: "A corner case is given, if there is a non-predictable relevant object/class in a relevant location." But corner cases do not result from a combination of parameters or incorrect behavior alone, as mentioned in [32, 33], whereby completely new data samples are also considered as corner cases.

Ouyang et al. defines a corner case as a perturbation of the input sample, which no longer corresponds to the label [31]. The classification is specifically addressed, and the decision boundary region, i.e., reject region [52], is described as sensitive for corner cases.

A corner case definition is given in [28, 32] for trajectory data. Their definitions are not restricted to a single trajectory alone. They also include corner cases due to interactions with other traffic participants, violations of norms and rules, the (driving) environment, and the model's task. The consideration of further factors in [28] is valuable since violations of traffic rules or driving maneuvers can influence the counterpart and can cause corner cases even without an accident.

However, to our knowledge, no commonly accepted definition or one that unifies the various definitions of corner cases exists in the literature, especially not for corner cases in the context of ML. So far, we discovered four separable groups for corner case definition in ML: (1) erroneous behavior, (2) relevance-centered definition, (3) anomaly and novelty point of view, and (4) information gain.

1. It becomes apparent from the different interpretations of the term corner case that in the context of ML, the ML model is considered as a unit with dozens of parameters, which cause an erroneous behavior [26, 29–32, 34, 35]. Individual input parameters and rare combinations are far more important in software and hardware testing to determine a corner case [25, 60] and do not fit in an ML corner case definition. Bolte et al. [27] addresses the erroneous behavior of the ML model and could be counted to the same group.
2. However, the definition of Bolte et al. [27] falls more on the side of a task-related definition because of the relevance aspect together with [28, 32] and behaves differently depending on the application task and goal. Rösch et al. [28] also expresses another point, namely that a corner case depends strongly on the importance or relevance with respect to the model task. In the case of driving trajectories, a corner case that happens in immediate proximity to the car is more important than an event that happens very far away. Even if it is the same corner case.
3. Another point is to consider terms such as outlier, anomaly, and novelty, which are well-known in the ML community. A distinction and definition of these three terms are available in [39]. These terms are highly related to corner cases [61] and have a considerable overlap [33] from a methodological point of view because methods such as outlier, anomaly, and novelty detection are used in corner case detection. Corner cases that arise, for example, due to new, novel, or unknown data [33] would be treated equally as erroneous behavior as long as they contribute to the improvement of the model.
4. Let's think about corner cases from the perspective of model training and retraining instead of erroneous or incorrect behavior, as in active learning [62]. A corner case is a data sample that helps (best) with learning the task at hand. In such a case, a corner case would not be particularly rare but would generally be a diverse, representative, or informative sample that reduces the overall expected error [63]. The mathematical definition of a corner case for classification from [31] fits well in this fraction and is easily applicable to other domains in ML.

Based on these four definition groups, we describe ML corner cases in general as follows:

Description (ML Corner Case): Corner cases in ML are cases (characterized by the peculiarities and properties from section "Nature of corner cases") relevant for the task} with a high predictive uncertainty, for which we would expect to improve the overall performance of an ML model (e.g., in terms of loss) if they were more abundant for training. Moreover, a corner case sample is a corner case for the ML model and task only as long as the constraints of the previous sentence hold. Otherwise, the corner case becomes a common sample.

Mathematical corner case definition

With the corner case definitions and descriptions from section "Overview of corner case definitions" in mind, the question is how to implement them efficiently. For this reason, we present a mathematical corner case definition based on the ML corner case description. We base our definition on the mathematical description of a supervised learner from [8, 52]. We define the (training) data for a supervised learner as

$$\mathcal{D} = \{ (x_1, y_1), ..., (x_N, y_N) \} \subset \mathcal{X} \times \mathcal{Y}, \tag{1}$$

with an instance space $\mathcal{X}$ and an associated output space $\mathcal{Y}$. A training sample consists of (x_i, y_i). The hypothesis space $\mathcal{H}$ provides mappings $h : \mathcal{X} \to \mathcal{Y}$ from an instance x to an outcome y and a loss function $l : \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}$. The risk, i.e., expected loss, associated with a given hypothesis h is given by

$$R(h) = \mathbb{E}_p[l(h(x), y)] = \iint l(h(x), y)\, p(x, y)\, dxdy, \tag{2}$$

where $\mathbb{E}_p$ denotes the expectation with respect to the joint distribution $p(x, y)$ of input $x \in \mathcal{X}$ and output variables $y \in \mathcal{Y}$. The learning algorithm aims to find a hypothesis (or model) $h^* \in \mathcal{H}$ with minimal risk (expected loss)

$$h^* = \arg\min_{h \in \mathcal{H}} R(h). \tag{3}$$

However, the integral is not solvable in practice, so it is approximated using the empirical risk

$$R_{emp}(h) = \frac{1}{N} \sum_{i=1}^{N} l(h(x_i), y_i), \tag{4}$$

which is an estimation of the true risk $R(h)$ that gets minimized by the empirical risk minimizer

$$\hat{h} = \arg\min_{h \in \mathcal{H}} R_{emp}(h). \tag{5}$$

The dependency between $\mathcal{X}$ and $\mathcal{Y}$ is usually non-deterministic, which results in a conditional probability distribution and is accompanied by aleatoric uncertainty [8]. Therefore, a pointwise Bayes predictor f^* is used. This predictor outputs the prediction $\hat{y} \in \mathcal{Y}$ that minimizes the expected loss (cf. [52]) and is given by

$$f^*(x) = \arg\min_{\hat{y} \in \mathcal{Y}} \int l(y, \hat{y})\, p(y|x)\, dy. \tag{6}$$

Considering $\hat{h}$ is an approximation of h^* (approximation uncertainty) and the discrepancy between the Bayes predictor h^* and the pointwise Bayes predictor f^* of fitting the best hypothesis (model uncertainty), in summary, it is referred to as epistemic uncertainty [8] due to lack of knowledge.

The loss or task-specific evaluation metric (cf. [64]) provides information about the relevancy of detected samples in the dataset in conjunction with the model performance and the prediction quality of the respective sample. However, the loss in its bare form $l : \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}$ does not say something about the relevancy of a sample in general. For example, in highly automated driving, detecting pedestrians directly in front of a car is far more important for safety argumentation than detecting pedestrians far away. In the literature, a weighted loss function is often used to increase the influence of specific features [65, 66]. We propose to use a sample-specific weight to model such sample relevance. The weight is highly task-specific and may depend on features $\mathcal{X}$, the target $\mathcal{Y}$, and the task context $c_{task} \in \mathcal{C}$ (e.g., the distance to the object, or object class). $\mathcal{C}$ is referred to as the context space. It comprises everything besides the instance space $\mathcal{X}$ and output space $\mathcal{Y}$, which is necessary to describe the importance of a sample for the task at hand. This may be the task goal, e.g., object detection, semantic segmentation, and the operational design domain [24] via meta attributes, e.g., weather and street conditions.

To model the sample-specific weight we use a weight function $w : \mathcal{X} \times \mathcal{Y} \times \mathcal{C} \to \mathbb{R}^+$. Given the relevancy, i.e., a weight, the expected relevancy-weighted loss is

$$\mathbb{E}_p^w[l(h(x), y)] = \iint w(x, y, c_{task})\, l(h(x), y)\, p(x, y)\, dxdy, \tag{7}$$

where $w(x, y, c_{task})$ represents the weight function. It determines the weight of sample x in conjunction with the present target y and task context c_{task}. The expected relevancy-weighted loss thereby represents the basis of our corner case definition for ML models.

Definition (ML Corner Case): A corner case is a relevant sample x for which the expected relevancy-weighted loss of the best pointwise Bayer predictor (i.e., the loss minimizer) is higher than a predefined threshold $\tau \in \mathbb{R}$.

$$\arg\min_{\hat{y} \in \mathcal{Y}} \int w(x, y, c_{task})\, l(y, \hat{y})\, p(y|x)\, dy > \tau. \tag{8}$$

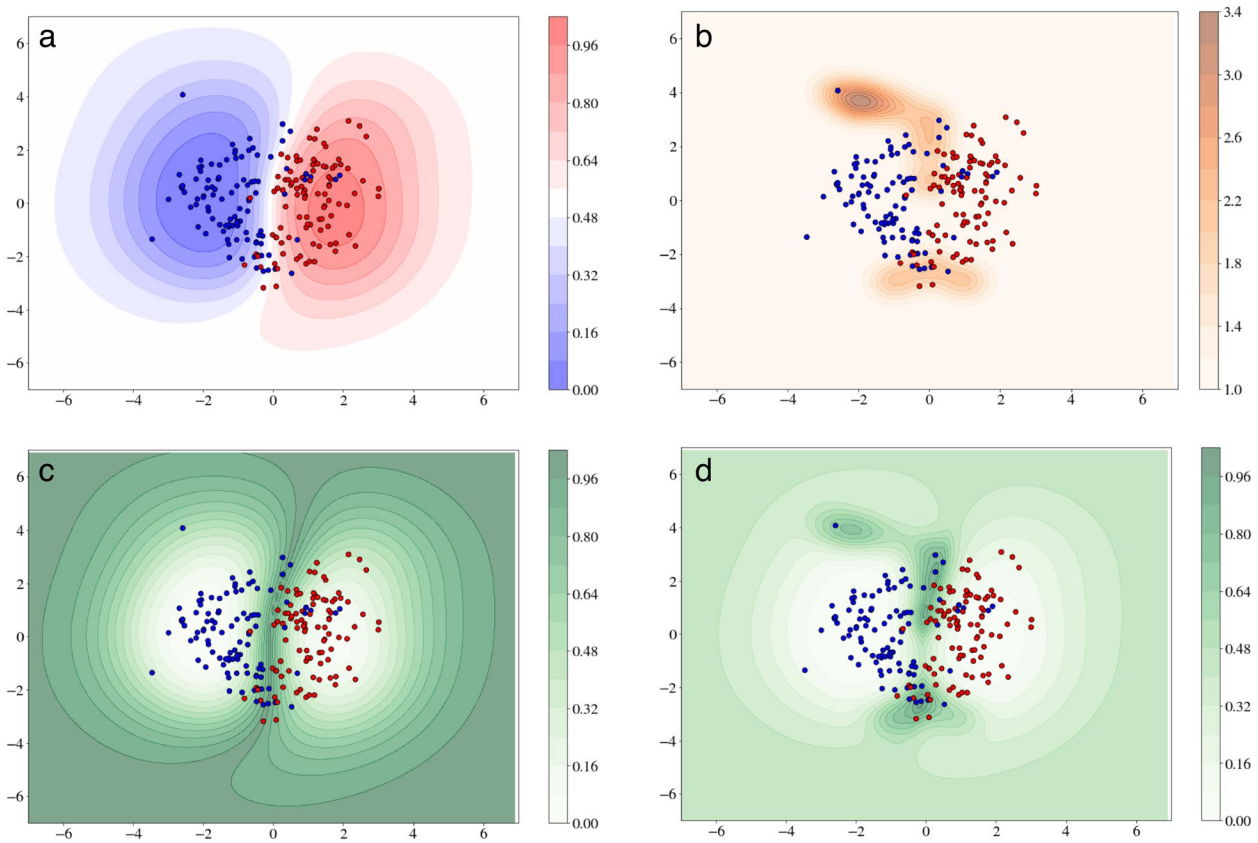

Fig. 4 The four diagrams illustrate our definition of corner cases using a toy example with two classes (red and blue). (**a**) Prediction probability. (**b**) Task-specifc weighting. (**c**) Minimal expected loss (no weighting). (**d**) Minimal expected relevancy-weighted loss (without threshold), see Eq. 8

Connection to Corner Case Description: In summary, we would like to briefly discuss the connection between the ML corner case definition and the earlier presented corner case description. The crucial point in the corner case description and definition is the goal to improve the model with respect to the task at hand. In the definition, the expected relevance-weighted loss and the introduced parameter τ are used to make this decision and to separate corner cases from the remaining samples. Hence, as with the corner case description, whether a sample is a corner case depends on the predictive uncertainty (i.e., the aleatory and epistemic uncertainty) and relevance for the task at hand, with the aim of improving model performance.

Implementation of our corner case definition

Next, we illustrate the definition we introduced earlier with a toy example. For this purpose, we create a synthetic two-class problem inspired by Fig. 2 with two features and the classes blue and red. This toy example and the resulting prediction probability obtained by Gaussian process classifier [52] are depicted in Fig. 4a

The minimal expected loss for our toy example is depicted in Fig. 4c. The minimal expected loss is high in the area around the decision boundary and areas further away from the class centers. Because of the underlying classification problem, many samples close to the decision boundary show a high expected loss, but they are not corner cases. In addition, potentially relevant corner cases that lie further away from the decision boundary and show a low expected loss value are suppressed, and it is precisely these samples that are important.

Figure 4b shows our relevancy-weighting function, where the samples in the darker areas are weighted higher than those in the brighter areas. By combining the weight function with the minimum expected loss, we get Fig. 4d. With the weighting, we indicate the corner case samples essential for the task using the threshold τ. The key is to create an appropriate weight function that weights the needs of the task higher so that the required samples can be identified. The quantitative assessment of the corner cases based on the introduced expected relevancy-weighted loss in section "Quantitative assessment of corner cases" provides a more detailed overview.

Benefits of the corner case definition

According to our Corner Cases Definition (cf. section "Mathematical corner case definition"), it is possible to divide data samples into common and corner cases whereby the threshold of when a corner case is to be categorized as such is freely adjustable via the parameters τ and a task-specific weighting function w. Next, we show how the corner case definition behaves during model training and testing, as well as after the deployment of the model.

Training

During the learning process, the expected relevance-weighted loss $\mathbb{E}_p^w[l(h(x), y)]$ and the threshold τ applied to it show us whether a minimization of the empirical risk is given. From the learning perspective, we are interested in those corner cases for which the relevance-weighted risk can be reduced. Hence, these are cases where the epistemic uncertainty (i.e., reducible uncertainty) is high, and the task relevancy is still given. This directly links our definition and active learning [62].

Testing

In the case of testing, we use the expected relevancy-weighted loss. Regarding classification, the integral in Eq. 8 becomes a summation, which can be easily solved. However, for regression tasks, we might need to rely on approximations of the expected loss, e.g., a Monte Carlo approximation [52]. From a testing perspective, the expected relevancy-weighted loss uncovers which samples are considered corner cases by the model, i.e., relevant samples for which the model has a high loss.

Deployment

When the model is deployed, it is crucial to model the distribution $p(y|x)$ with no or only little (human) feedback. This is a difficult task, as presented in Hüllermeier et al. [8], as it involves the potentially error-prone approximation of the aleatory and epistemic uncertainty. In addition, the distribution can also change at runtime (e.g., due to data shift or drift). Hence, we must detect this drift and update the ML model accordingly. In order to detect corner cases at runtime, i.e., after deployment, we can compute the expected relevance-weighted loss. This can be done without ground truth data. Moreover, the weighting function helps to determine the weighted relevance of each sample, which reveals the value of the sample and at the same time helps to acquire new data specifically for further model training.

On the importance of calibration

Consideration of model calibration is essential because it affects predicted probabilities. The prediction confidence score in an uncalibrated ML model mostly does not correspond to the actual quality of the predicted probabilities. In an ideally calibrated ML model, on the other hand, the confidence score matches the actual probability of correctness of the predicted class, i.e., accuracy. To close the ideal and predicted confidence value gap, the model must be calibrated using, for example, Platt Scaling or Temperature Scaling [67] whereby a single parameter T, called temperature, is optimized to the negative log-likelihood on the validation set.

Calibrating the ML model affects the probability as the prediction confidence scores correspond to the input data and thus produce more reliable probabilities. As the probabilities are required for minimizing the expected relevance-weighted loss, a good calibration boosts the accuracy of the expected relevance-weighted loss to identify corner cases. However, the calibration is useless if the data distribution between the validation set used for calibration and the data that the ML model has to process after deployment changes (cf. section "Misspecification" and [8]). Therefore, ensuring that the data distribution is equivalent is important.

Quantitative assessment of corner cases

The determination of sample importance is challenging, i.e., to obtain a task-dependent weighting function. Therefore, the chosen metric is essential to quantify corner cases since the value of a corner case gains or loses significance depending on the selected metric. Classical metrics like accuracy, F1-score, mean squared error, intersection over union, and mean average precision provide the performance of the ML model in terms of dedicated error measure (e.g., squared-loss) on entire datasets. However, most metrics treat each sample equally and do not consider the task-dependent relevance of corner cases.

Not only should the metric be considered for the quantitative assessment of corner cases, but also task-specific measures. These measures assess the relevance of corner cases (cf. section "Relevance"). How these relevance measures are interpreted for the individual task depends mainly on the task. Lyssenko et al. [64] presents a task-oriented relevance metric for pedestrian detection. Not all predictions or ground truth objects are treated equally because the relevant objects for task validation, e.g., highly automated driving, receive preferential treatment. In [64], the distance to the object is, for example, used as a proxy measure. Bolte et al. [27] describes corner cases as objects of relevant class in the relevant location, and both are also suitable relevance measures. Also, a complexity measure as in [68, 69] might be suitable to describe the relevance. In addition, other contextual information can be included if it is relevant to the task.

As we can see, many measures can be taken into account depending on the application task. By quantifying the corner case using various measures in a well-chosen and task-depended weight function (cf. section "Corner case definition"), a corner case scoring based on the expected relevancy-weighted loss can be achieved and individually adapted to the application's requirements. In this regard, the corner case properties introduced in section "Nature of corner cases" are reflected in the relevance measures. Ultimately, the scoring determines the importance or value of a corner case for the ML model and the associated application task.

Taxonomy of corner cases

In this section, we present our extension to an existing corner cases systematization [33], where we incorporate the corner case definition (cf. section "Corner case definition"). The extension is oriented on the development stages of an ML system and uses the introduced method layer from [33] as a starting point. Based on the existing structure, we introduce new levels, such as input, model, and deployment, to show the different types and areas where corner cases could occur within the method layer.

Existing layer & level systematisation

The basis for systematizing corner cases introduced in [61, 70] consists of five levels: pixels, domain, object, scene, and scenario, which are ordered by complexity and represent the core of the structure. This first systematization is vision-oriented and considers only images and no other sensor technology used in highly automated driving. In [33], the authors enhance the systematization by adding missing elements and provide a comprehensive systematization of corner cases, including other sensors such as camera, LiDAR, and RADAR sensors. Table 1 illustrates an abbreviation of the corner cases systematization.

The structure of the corner case systematization [33] in Table 1 consists of three successive layers: (1) the sensor layer, (2) the content layer, and (3) the temporal layer. These layers are further split into other sub-levels and can be differentiated into:

1. **Sensor Layer:** Comprises corner cases that result from the used sensor.
   (a) *Hardware Level:* Corner case due to damaged or broken hardware.
   (b) *Physical Level:* Corner cases caused by physical or environmental influences.
2. **Content Layer:** Corner cases that result from the data and have no temporal relation.
   (a) *Domain Level:* Corner cases that arise due to location or environmental changes.
   (b) *Object Level:* Corner cases resulting from unknown objects.
   (c) *Scene Level:* Corner cases caused by known objects appearing in unexpected quantities or locations.
3. **Temporal Layer:** Covers cases that result from multiple data points in time and thus have a temporal context.
   (a) *Scenario Level:* Corner Cases that result from an unexpected movement or behavior.

According to [33], the method layer comprises corner cases resulting from the used ML method, model, and architecture. For this purpose, the model uncertainty, also known as epistemic uncertainty [8], and adversarial examples [71] as the cause of these corner cases is discussed. In [33] is also mentioned that with the selection of the

Table 1 Corner case systematization from [33], with examples corner cases for camera and LiDAR. (reworked, original from [33])

	Sensor Layer		Content Layer			Temporal Layer
	Hardware Level	**Physical Level**	**Domain Level**	**Object Level**	**Scene Level**	**Scenario Level**
Camera	Pixel Error: *Dead pixel*	Pixel-based: *overexposure*	Single Frame Domain Shift: *Location (Germany - U.S.A.)*	Anomaly: *Garbage on street*	Contextual Anomaly: *People on billboards*	Anomaly Represented by Multiple Frames or Point Clouds: *Car drives snake lines*
LiDAR	Laser Error: *Broken mirror*	Beam-based: *Surface does not reflect the beam*	Single Point Cloud Domain Shift: *Shape of road markings*	Anomaly: *Dust cloud*	Contextual Anomaly: *Sweeper cleaning the sidewalk*	

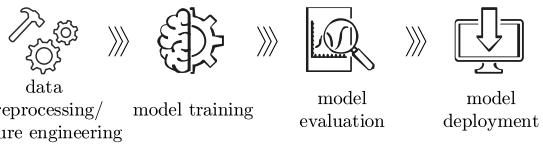

Fig. 5 ML development process

ML model and its architecture, the (inductive) bias influences which method layer corner cases are favored and which are not. In addition to the three layers above, which are readily observable by humans, the introduced method layer describes corner cases from the model's point of view, which makes it harder for humans to comprehend.

However, a clear structure of the method layer into different levels, e.g., compared to the content layer, was not carried out before. This article closes this gap and elaborates a clear structure for systematizing method layer corner cases.

ML-development stages

Before going into more detail about our extension of the corner cases systematization for the method layer, we briefly introduce the four basic stages of development that each ML model typically undergoes. These are the (1) data preprocessing and feature engineering, (2) model training, (3) model evaluation, and (4) model deployment stages. The four stages are depicted in Fig. 5. The process stages are relevant for our extension because each stage has a specific purpose within the ML model development.

1. The first stage is about preprocessing, cleaning the data, and generating features with high informational value to improve the subsequent algorithm's performance. Note that many deep learning algorithms skip feature engineering and instead learn the features themselves. Still, data preprocessing is generally inevitable and necessary in every ML model development.
2. The model training stage is about choosing the suitable ML model architecture and training the ML model.
3. The model evaluation stage is needed to measure the model performance by testing and validating the model if it behaves as expected. On top of the ML model, performance is used to rate prediction quality, accuracy, and speed to compare it with other existing ML models.
4. The last stage is the model deployment stage, which integrates the trained and validated model into a running system or application.

Be aware that multiple iterations may be required between the different stages (e.g., continuous integration) and slightly modified development processes [72]. However, the essential steps are similar or the same.

Method layer extensions

Based on the development process presented before, we defined three new corner case levels for the method layer: the *input*, *model*, and *deployment level*. The three levels are illustrated in Table 2. The following discusses the relevance and provides examples for each defined level.

Input level

The input level addresses corner cases caused by the data itself and relates to data quality, complexity, novelty, and ambiguity. This includes biased data [47, 73] as well as an imbalanced class distribution [65], data distribution [74], or label errors [6, 7] in the ground truth. There is a risk that the model cannot compensate for biased data [47] and labeling errors [6, 7], which can lead to corner cases. However, methods such as focal loss [75] or data argumentation [73] of the underrepresented classes can compensate for the imbalance to a certain degree.

The uncertainty in the input data, also called aleatoric uncertainty [8], is also part of the input level and can be modeled with different methods [76], such as Prior networks [11], or Mixture Density Networks [77].

We also assigned data with noise, e.g., adversarial data samples [71], to the input level. In the case of adversarial samples, even slight changes in the input data, e.g., adding a specific noise pattern, can dramatically change the model output.

Table 2 Extention of the corner case systematization

	Method Layer		
	Input Level	**Model Level**	**Deployment Level**
Sensor Independent	Database Issue: *bad labels or underrepresented classes*	Model Issue: *bad calibration or high epistemic uncertainty*	Environment Issue: *concept shift due to misspecification between train and real data*

Model level

The used design and architecture of the ML model significantly influence the traceability and explainability of the ML model and the appearing corner cases. Depending on the selected ML model, an inductive model bias is introduced into the ML model, which can result in a corner case.

Another source of corner cases is model uncertainty, referred to in the literature as epistemic uncertainty [8, 78]. Among others, methods like Monte Carlo dropout [9, 79], deep ensemble [12, 80], Bayes-by-Backprop [10], or Prior networks [11] aim to approximate the epistemic uncertainty. A high epistemic uncertainty of detected objects, classes, or other predictions is considered as an indicator for corner cases [78, 81, 82] because the ML model cannot reliably handle these data samples. In this respect, we have to consider the approximation quality of the predicted uncertainties [83].

In summary, the model level includes all those corner cases that result from a model's uncertainty, architecture, and reliability.

Deployment level

When the ML model is installed in a production chain or application in the commercial or end customer segment, corner cases on the deployment level can occur. The most prominent cause that can lead to corner cases is a misspecification issue (cf. section "Misspecification") or inaccuracy between the ML model specification and the actual model-working environment [84]. Therefore, calibrating the confidence level [67, 85] is an important aspect. A wrong calibration based on a dataset with a different distribution or a concept shift [39] between training and runtime can lead to devastating model predictions [84]. Calibration can be considered part of the model level, but we did not do it regarding the misspecification. In addition, the deployment level and domain level of the content layer are similar in terms of concept shift but have a decisive difference. At the domain level, we see a change in the real world (before and after). The deployment level results from the misspecification between training and reality, as the training only covers a portion of the reality and most likely contains a distribution difference. The effect is thus identical, but the origin is not necessarily the same.

Another possible cause of corner cases relates to the setup of the ML model and, thus, whether the ML model is fully operational, e.g., receiving all required sensor signals and expected data quality. Related topics like diagnostics or model self-awareness [86, 87] may help to explain possible model output or prediction errors.

Conclusion

In this article, we analyzed the nature of a corner case and presented various properties and particularities of corner cases. The associated diversity of properties that can make up a corner case cannot be underestimated, with some properties of a corner case only becoming apparent when considering an ML model or application. The particularities and properties introduced are useful for describing corner case data or scenarios, e.g., in highly automated driving, so that the corner cases can be defined more clearly, generated virtually, or captured specifically in the real world. The properties can be used to extend existing ontologies [88, 89] to formalize corner cases better. This circumstance is innovative to our knowledge when describing driving situations since the data is described in an experimental setup, and the ML model and application are included.

We also present a novel generalizing (verbal) description and mathematical definition of corner cases in this article. Both are based on the listed properties and an extensive literature study about corner cases in ML. Our definition of a corner case is based on the expected relevance-weighted loss and offers the possibility to make corner cases tangible and thus measurable. Furthermore, considering the expected relevance-weighted loss allows us to understand how the model behaves, how it can learn further with the corner case samples, and whether risk reduction is reached. The definition of the expected relevance-weighted loss can also be directly operationalized: (1) The relevance of data samples is determined by means of the weighting, and (2) by choosing an application-specific threshold value, corner cases can be identified directly at runtime. In the future, we aim to implement and evaluate the approach described in this article in an application use case such as object detection or semantic instance segmentation. Initial work on creating a task-specific weighting has already been carried out in [90].

As mentioned in the introduction, our corner case definition focuses on data and ML corner cases to improve the model performance or the input space coverage (Operational Design Domain [56]). Besides, there are other views on highly automated driving, such as legal (e.g., traffic rules, contract, liability), data privacy, economic, ecological, ethnic/social perspective, or the hardware-oriented view aiming to improve the sensor. The different perspectives, of course, also introduce varying corner case definitions that go beyond our ML-centric definition's scope. However, some of these perspectives extend far into other (highly relevant) research areas but should be considered in the future.

Subsequently, we present an extension of a pre-existing corner case taxonomy, incorporating our novel corner

case definition and grouping different model-specific corner cases. The taxonomy helps to get an overview of which aspects are crucial for ML model testing and validation.

Finally, we discuss the quantitative assessment of corner cases, focusing on the relevance of samples and how initial work has attempted to measure relevance. However, investigating how well the quantitative assessment of corner cases correlates with the introduced definition of corner cases remains open for future work. Regardless, we think there is still much research to be done on metrics that consider the value or importance of an corner case. From a human point of view, we need to enable the ML model during training to focus on extracting the task's important factors and pay less attention to minor details.

Abbreviation
ML Machine Learning

Acknowledgements
Thanks to our Intelligent Embedded Systems group colleagues for their helpful comments, suggestions, and participation in the corner case property poll.

Authors' contributions
F. Heidecker wrote the majority of the manuscript and created the illustrations. M. Bieshaar provided suggestions and concepts, which became the basis for countless discussions, and was responsible for double-checking the article. B. Sick provided suggestions and reviewed the manuscript. All authors read and approved the final manuscript.

Funding
Open Access funding enabled and organized by Projekt DEAL. This work results from the project KI Data Tooling (19A20001O), funded by the German Federal Ministry for Economic Affairs and Climate Action (BMWK).

Availability of data and materials
Not applicable.

Declarations

Consent for publication
Not applicable.

Competing interests
The authors declare no competing interests.

Received: 26 June 2023 Accepted: 14 November 2023
Published online: 02 January 2024

References

1. Laplante P, Milojicic D, Serebryakov S, Bennett D (2020) Artificial Intelligence and Critical Systems: From Hype to Reality. Computer 53(11):45–52
2. He K, Gkioxari G, Dollár P, Girshick R (2017) Mask R-CNN. In: Proc. of the International Conference on Computer Vision, Venice, pp 2980–2988
3. Wu Y, Kirillov A, Massa F, Lo WY, Girshick R (2019) Detectron2. https://github.com/facebookresearch/detectron2. Accessed 15 July 2022.
4. Baevski A, Zhou Y, Mohamed A, Auli M (2020) wav2vec 2.0: A Framework for Self-Supervised Learning of Speech Representations. In: Proc. of the Advances in Neural Information Processing Systems, Vancouver, pp 12449–12460
5. Zhao WX, Zhou K, Li J, Tang T, Wang X, Hou Y, Min Y, Zhang B, Zhang J, Dong Z, Du Y, Yang C, Chen Y, Chen Z, Jiang J, Ren R, Li Y, Tang X, Liu Z, Liu P, Nie JY, Wen JR (2023) A Survey of Large Language Models. arXiv preprint arXiv:2303.18223
6. Frenay B, Verleysen M (2014) Classification in the Presence of Label Noise: A Survey. IEEE Trans Neural Netw Learn Syst 25(5):845–869
7. Herde M, Huseljic D, Sick B, Calma A (2021) A Survey on Cost Types, Interaction Schemes, and Annotator Performance Models in Selection Algorithms for Active Learning in Classification. IEEE Access 9:166970–166989
8. Hüllermeier E, Waegeman W (2021) Aleatoric and Epistemic Uncertainty in Machine Learning: An Introduction to Concepts and Methods. Mach Learn 110(3):457–506
9. Gal Y, Ghahramani Z (2016) Dropout as a Bayesian Approximation: Representing Model Uncertainty in Deep Learning. In: Proc. of the International Conference on Machine Learning, New York, pp 1050–1059
10. Blundell C, Cornebise J, Kavukcuoglu K, Wierstra D (2015) Weight Uncertainty in Neural Network. In: Proc. of the International Conference on Machine Learning, Lille, pp 1613–1622
11. Malinin A, Gales M (2018) Predictive Uncertainty Estimation via Prior Networks. In: Proc. of the Advances in Neural Information Processing Systems, Montréal, Canada, pp 7047–7058
12. Lakshminarayanan B, Pritzel A, Blundell C (2017) Simple and Scalable Predictive Uncertainty Estimation Using Deep Ensembles. In: Proc. of the Advances in Neural Information Processing Systems, Long Beach, pp 6402–6413
13. Google LLC (2022) Dataset Search. https://datasetsearch.research.google.com/. Accessed 24 Oct 2022
14. Kaggle Inc (2022) Kaggle Datasets. https://www.kaggle.com/datasets. Accessed 24 Oct 2022
15. VisualData (2022) VisualData Discovery. https://www.visualdata.io/discovery. Accessed 24 Oct 2022
16. Yu F, Chen H, Wang X, Xian W, Chen Y, Liu F, Madhavan V, Darrell T (2020) BDD100K: A Diverse Driving Dataset for Heterogeneous Multitask Learning. In: Proc. of the Conference on Computer Vision and Pattern Recognition, Seattle, pp 2636–2645
17. Caesar H, Bankiti V, Lang AH, Vora S, Liong VE, Xu Q, Krishnan A, Pan Y, Baldan G, Beijbom O (2020) nuScenes: A Multimodal Dataset for Autonomous Driving. In: Proc. of the Conference on Computer Vision and Pattern Recognition, Seattle, pp 11618–11628
18. Waymo (2019) Waymo Open Dataset: An Autonomous Driving Dataset. https://www.waymo.com/open. Accessed 24 Oct 2022
19. Geiger A, Lenz P, Stiller C, Urtasun R (2013) Vision Meets Robotics: The KITTI Dataset. Int J Robot Res 32(11):1231–1237
20. Geyer J, Kassahun Y, Mahmudi M, Ricou X, Durgesh R, Chung AS, Hauswald L, Pham VH, Mühlegg M, Dorn S, Fernandez T, Jänicke M, Mirashi S, Savani C, Sturm M, Vorobiov O, Oelker M, Garreis S, Schuberth P (2020) A2D2: Audi Autonomous Driving Dataset. arXiv preprint arXiv:2004.06320, https://www.a2d2.audi
21. Huang X, Cheng X, Geng Q, Cao B, Zhou D, Wang P, Lin Y, Yang R (2018) The ApolloScape Dataset for Autonomous Driving. In: Proc. of the Conference on Computer Vision and Pattern Recognition, Workshop, Salt Lake City, pp 1067–1073
22. Neuhold G, Ollmann T, Bulò SR, Kontschieder P (2017) The Mapillary Vistas Dataset for Semantic Understanding of Street Scenes. In: Proc. of the International Conference on Computer Vision, Venice, pp 5000–5009
23. Pitropov M, Garcia DE, Rebello J, Smart M, Wang C, Czarnecki K, Waslander S (2020) Canadian Adverse Driving Conditions Dataset. Int J Robot Res 40(4-5):681–90
24. Houben S, Abrecht S, Akila M, Bär A, Brockherde F, Feifel P, Fingscheidt T, Gannamaneni SS, Ghobadi SE, Hammam A, Haselhoff A, Hauser F, Heinzemann C, Hoffmann M, Kapoor N, Kappel F, Klingner M, Kronenberger J, Küppers F, Löhdefink J, Mlynarski M, Mock M, Mualla F, Pavlitskaya S, Poretschkin M, Pohl A, Ravi-Kumar V, Rosenzweig J, Rottmann M, Rüping S, Sämann T, Schneider JD, Schulz E, Schwalbe G, Sicking J, Srivastava T, Varghese S, Weber M, Wirkert S, Wirtz T, Woehrle M (2021) Inspect, Understand, Overcome: A Survey of Practical Methods for AI Safety. arXiv preprint arXiv:2104.14235

25. Koopman P, Kane A, Black J (2019) Credible Autonomy Safety Argumentation. Proc of the Safety-Critical Systems Symposium, (preprint). Bristol; pp 1–27
26. Pei K, Cao Y, Yang J, Jana S (2017) DeepXplore: Automated Whitebox Testing of Deep Learning Systems. In: Proc. of the Symposium on Operating Systems Principles, Shanghai, pp 1–18
27. Bolte JA, Bär A, Lipinski D, Fingscheidt T (2019) Towards Corner Case Detection for Autonomous Driving. In: Proc. of the Intelligent Vehicles Symposium, Paris, pp 438–445
28. Rösch K, Heidecker F, Truetsch J, Kowol K, Schicktanz C, Bieshaar M, Sick B, Stiller C (2022) Space, Time, and Interaction: A Taxonomy of Corner Cases in Trajectory Datasets for Automated Driving. In: Proc. of the Symposium Series on Computational Intelligence, IEEE CIVTS, Singapore, pp 1–8
29. Tian Y, Pei K, Jana S, Ray B (2018) DeepTest: Automated Testing of Deep-Neural-Network-Driven Autonomous Cars. In: Proc. of the International Conference on Software Engineering, New York, pp 303–314
30. Zhang JM, Harman M, Ma L, Liu Y (2022) Machine Learning Testing: Survey, Landscapes and Horizons. IEEE Trans Softw Eng 48(1):1–36
31. Ouyang T, Marco VS, Isobe Y, Asoh H, Oiwa Y, Seo Y (2021) Corner Case Data Description and Detection. In: Proc. of the International Conference on Software Engineering Workshop, Madrid, pp 19–26
32. Pfeil J, Wieland J, Michalke T, Theissler A (2022) On Why the System Makes the Corner Case: AI-based HolisticAnomaly Detection for Autonomous Driving. In: Proc. of the Intelligent Vehicles Symposium, Achen, pp 337–344
33. Heidecker F, Breitenstein J, Rösch K, Löhdefink J, Bieshaar M, Stiller C, Fingscheidt T, Sick B (2021) An Application-Driven Conceptualization of Corner Cases for Perception in Highly Automated Driving. In: Proc. of the the Intelligent Vehicles Symposium, Nagoya, pp 1–8
34. Metzen JH, Hutmacher R, Hua NG, Boreiko V, Zhang D (2023) Identification of Systematic Errors of Image Classifiers on Rare Subgroups. In: Proc. of the International Conference on Computer Vision, Paris, pp 5064–5073
35. Gawlikowski J, Tassi CRN, Ali M, Lee J, Humt M, Feng J, Kruspe A, Triebel R, Jung P, Roscher R, Shahzad M, Yang W, Bamler R, Zhu XX (2022) A Survey of Uncertainty in Deep Neural Networks. arXiv preprint arXiv:2107.03342
36. Stallkamp J, Schlipsing M, Salmen J, Igel C (2011) The German Traffic Sign Recognition Benchmark: A multi-class classification competition. In: Proc. of the International Joint Conference on Neural Networks, San Jose, pp 1453–1460
37. Larsson F, Felsberg M (2011) Using Fourier Descriptors and Spatial Models for Traffic Sign Recognition. In: Proc. of the Scandinavian Conference on Image Analysis, Ystad Saltsjöbad, pp 238–249
38. Bogdoll D, Breitenstein J, Heidecker F, Bieshaar M, Sick B, Fingscheidt T, Zöllner JM (2021) Description of Corner Cases in Automated Driving: Goals and Challenges. In: Proc. of the International Conference on Computer Vision, ERCVAD Workshop, Virtual, Montreal; pp 1023–1028
39. Gruhl C, Sick B, Tomforde S (2021) Novelty Detection in Continuously Changing Environments. Futur Gener Comput Syst 114:138–154
40. Pimentel MAF, Clifton DA, Clifton L, Tarassenko L (2014) A Review of Novelty Detection. Signal Process 99:215–249
41. Möller F, Botache D, Huseljic D, Heidecker F, Bieshaar M, Sick B (2021) Out-of-distribution Detection and Generation using Soft Brownian Offset Sampling and Autoencoders. In: Proc. of the International Conference on Computer Vision and Pattern Recognition, SAIAD Workshop, virtual, Nashville; pp 46–55
42. Cen J, Yun P, Cai J, Wang MY, Liu M (2021) Deep Metric Learning for Open World Semantic Segmentation. In: Proc. of the International Conference on Computer Vision, virtual, Montreal; pp 15,333–15,342
43. Goodfellow I, Bengio Y, Courville A (2016) Deep Learning. MIT Press, Cambridge
44. Richter SR, Vineet V, Roth S, Koltun V (2016) Playing for Data: Ground Truth from Computer Games. In: Proc. of the European Conference Computer Vision, vol 9906, Amsterdam, pp 102–118
45. Dosovitskiy A, Ros G, Codevilla F, Lopez A, Koltun V (2017) CARLA: An Open Urban Driving Simulator. In: Proc. of the Conference on Robot Learning, Mountain View, pp 1–16
46. Corvi R, Cozzolino D, Poggi G, Nagano K, Verdoliva L (2023) Intriguing Properties of Synthetic Images: From Generative Adversarial Networks to Diffusion Models. In: Proc. of the Conference on Computer Vision and Pattern Recognition Workshops, Vancouver, pp 973–982
47. Gudivada VN, Apon AW, Ding J (2017) Data Quality Considerations for Big Data and Machine Learning: Going Beyond Data Cleaning and Transformations. Int J Adv Softw 10(1 &2):1–20
48. Pradhan SK, Heyn HM, Knauss E (2023) Identifying and Managing Data Quality Requirements: a Design Science Study in the Field of Automated Driving. Softw Qual J
49. Challa H, Niu N, Johnson R (2020) Faulty Requirements Made Valuable: On the Role of Data Quality in Deep Learning. In: In Proc. of the International Workshop on Artificial Intelligence for Requirements Engineering, Zurich, pp 61–69
50. Algan G, Ulusoy I (2021) Image classification with deep learning in the presence of noisy labels: A survey. Knowl-Based Syst 215:106771
51. Zhou X, Wu O (2022) Which Samples Should be Learned First: Easy or Hard? arXiv preprint arXiv:2110.05481
52. Bishop CM (2006) Pattern Recognitionand Machine Learning (Information Science and Statistics). Springer-Verlag, New York
53. Xu H, Mannor S (2012) Robustness and Generalization. Mach Learn 86(3):391–423
54. Rauber J, Brendel W, Bethge M (2018) Foolbox: A Python Toolbox to Benchmark the Robustness of Machine Learning Models. arXiv preprint arXiv:1707.04131
55. Abrecht S, Gauerhof L, Gladisch C, Groh K, Heinzemann C, Woehrle M (2021) Testing Deep Learning-Based Visual Perception for Automated Driving. ACM Trans Cyber-Phys Syst 5(4):28
56. Fingscheidt T, Gottschalk H, Houben S (eds) (2022) Deep Neural Networks and Data for AutomatedDriving: Robustness, Uncertainty Quantification, and In-sights Towards Safety. Springer International Publishing, Cham. https://link.springer.com/book/10.1007/978-3-031-01233-4
57. Gladisch C, Heinzemann C, Herrmann M, Woehrle M (2020) Leveraging Combinatorial Testing for Safety-Critical Computer Vision Datasets. In: Proc. of the International Conference on Computer Vision and Pattern Recognition, virtual, Seattle; pp 324–325
58. Glorot X, Bordes A, Bengio Y (2011) Domain Adaptation for Large-Scale Sentiment Classification: A Deep Learning Approach. In: Proc. of the International Conference on Machine Learning, Bellevue, pp 513–520
59. Tobin J, Fong R, Ray A, Schneider J, Zaremba W, Abbeel P (2017) Domain Randomization for Transferring Deep Neural Networks from Simulation to the Real World. In: Proc. of the International Conference on Intelligent Robots and Systems, Vancouver, pp 23–30
60. Hanhirova J, Debner A, Hyyppä M, Hirvisalo V (2020) A Machine Learning Environment for Evaluating Autonomous Driving Software. arXiv preprint arXiv:2003.03576
61. Breitenstein J, Termöhlen JA, Lipinski D, Fingscheidt T (2020) Systematization of Corner Cases for Visual Perception in Automated Driving. In: Proc. of the Intelligent Vehicles Symposium, Las Vegas, pp 1257–1264
62. Kottke D, Herde M, Sandrock C, Huseljic D, Krempl G, Sick B (2021) Toward Optimal Probabilistic Active Learning Using a Bayesian Approach. Mach Learn 110(6):1199–1231
63. Wu D (2018) Pool-Based Sequential Active Learning for Regression. Trans Neural Netw Learn Syst 30:1348–1359
64. Lyssenko M, Gladisch C, Heinzemann C, Woehrle M, Triebel R (2021) From Evaluation to Verification: Towards Task-Oriented Relevance Metrics for Pedestrian Detection in Safety-Critical Domains. In: Proc. of the International Conference on Computer Vision and Pattern Recognition, Workshops, virtual, Nashville; pp 38–45
65. Johnson JM, Khoshgoftaar TM (2019) Survey on Deep Learning with Class Imbalance. J Big Data 6(1):1–27
66. Philion J, Kar A, Fidler S (2020) Learning to Evaluate Perception Models using Planner-Centric Metrics. In: Proc. of Conference on Computer Vision and Pattern Recognition, Seattle, pp 14052–14061
67. Guo C, Pleiss G, Sun Y, Weinberger KQ (2017) On Calibration of Modern Neural Networks. In: Proc. of the International Conference on Machine Learning, Sydney, pp 1321–1330
68. Sadat A, Segal S, Casas S, Tu J, Yang B, Urtasun R, Yumer E (2021) Diverse Complexity Measures for Dataset Curation in Self-driving. arXiv preprint arXiv:2101.06554
69. Rahane AA, Subramanian A (2020) Measures of Complexity for Large Scale Image Datasets. In: Proc. of the International Conference on Artificial Intelligence in Information and Communication, Fukuoka, pp 282–287

70. Breitenstein J, Termöhlen JA, Lipinski D, Fingscheidt T (2021) Corner Cases for Visual Perception in Automated Driving: Some Guidance on Detection Approaches. arXiv preprint arXiv:2102.05897
71. Goodfellow I, Shlens J, Szegedy C (2015) Explaining and Harnessing Adversarial Examples. In: Proc. of the International Conference on Learning Representations, San Diego, pp 1–10
72. Zhang R, Albrecht A, Kausch J, Putzer HJ, Geipel T, Halady P (2021) Dde process: A requirements engineering approach for machine learning in automated driving. International Requirements Engineering Conference. Notre Dame, pp 269–279
73. Shorten C, Khoshgoftaar TM (2019) A survey on Image Data Augmentation for Deep Learning. J Big Data 6(60):1–48
74. Kaur H, Pannu HS, Malhi AK (2019) A Systematic Review on Imbalanced Data Challenges in Machine Learning: Applications and Solutions. ACM Comput Surv 52(4):1–36
75. Lin TY, Goyal P, Girshick R, He K, Dollar P (2017) Focal Loss for Dense Object Detection. In: Proc. of the International Conference on Computer Vision, Venice, pp 2980–2988
76. Feng D, Harakeh A, Waslander S, Dietmayer K (2021) A Review and Comparative Study on Probabilistic Object Detection in Autonomous Driving. IEEE Trans Intell Transp Syst 23(8):9961–80
77. Choi S, Lee K, Lim S, Oh S (2018) Uncertainty-Aware Learning from Demonstration Using Mixture Density Networks with Sampling-Free Variance Modeling. In: Proc. of the International Conference on Robotics and Automation, Brisbane, pp 6915–6922
78. Kendall A, Gal Y (2017) What Uncertainties Do We Need in Bayesian Deep Learning for Computer Vision? In: Proc. of the Advances in Neural Information Processing Systems, Long Beach, pp 5574–5584
79. Gal Y (2016) Uncertainty in Deep Learning. PhD thesis, University of Cambridge
80. Liu J, Paisley J, Kioumourtzoglou MA, Coull B (2019) Accurate Uncertainty Estimation and Decomposition in Ensemble Learning. In: Proc. of the Advances in Neural Information Processing Systems, Vancouver, pp 8952–8963
81. Heidecker F, Hannan A, Bieshaar M, Sick B (2021) Towards Corner Case Detection by Modeling the Uncertainty of Instance Segmentation Networks. In: Proc. of the International Conference on Pattern Recognition, IADS Workshop, Milan, pp 361–374
82. Mukhoti J, Gal Y (2018) Evaluating Bayesian Deep Learning Methods for Semantic Segmentation. arXiv preprint arXiv:1811.12709. pp 1–13
83. Ovadia Y, Fertig E, Ren J, Nado Z, Sculley D, Nowozin S, Dillon JV, Lakshminarayanan B, Snoek J (2019) Can You Trust Your Model's Uncertainty? Evaluating Predictive Uncertainty Under Dataset Shift. arXiv preprint arXiv:1906.02530. pp 1–25
84. Kato Y, Tax DMJ, Loog M (2022) A view on model misspecification in uncertainty quantification. arXiv preprint arXiv:2210.16938
85. Nixon J, Dusenberry M, Jerfel G, Nguyen T, Liu J, Zhang L, Tran D (2019) Measuring Calibration in Deep Learning. arXiv preprint arXiv:1904.01685
86. Müller-Schloer C, Schmeck H, Ungerer T (2011) Organic Computing - A Paradigm Shift for Complex Systems. Autonomic Systems. Birkhäuser Verlag, Basel, p 627
87. Müller-Schloer C, Tomforde S, (2017) Organic Computing - Technical Systems for Survival in the Real World. Autonomic Systems, Birkhauser, Cham
88. Bogdoll D, Guneshka S, Zöllner JM (2022) One Ontology to Rule Them All: Corner Case Scenarios for Autonomous Driving. In: Proc. of the European Conference Computer Vision Workshops, Tel Aviv, pp 409–425
89. Bagschik G, Menzel T, Maurer M (2018) Ontology based Scene Creation for the Development of Automated Vehicles. In: Proc. of the Intelligent Vehicles Symposium, Changshu, pp 1813–1820
90. Breitenstein J, Heidecker F, Lyssenko M, Bogdoll D, Bieshaar M, Zöllner JM, Sick B, Fingscheidt T (2023) What Does Really Count? Estimating Relevance of Corner Cases for Semantic Segmentation in Automated Driving. In: Proc. of International Conference on Computer Vision, Workshops, Paris, pp 3991–4000

Publisher's Note