

Explaining Unreliable Perception in Automated Driving: A Fuzzy-based Monitoring Approach

Aniket Salvi¹ and Mario Trapp³
Engineering Resilient Cognitive Systems
Technical University of Munich
Munich, Germany
{aniket.salvi,mario.trapp}@tum.de

Gereon Weiss²
Automation Systems
Fraunhofer Institute for Cognitive Systems IKS
Munich, Germany
gereon.weiss@iks.fraunhofer.de

Abstract—Autonomous systems that rely on Machine Learning (ML) utilize online fault tolerance mechanisms, such as runtime monitors, to detect ML prediction errors and maintain safety during operation. However, the lack of human-interpretable explanations for these errors can hinder the creation of strong assurances about the system’s safety and reliability. This paper introduces a novel fuzzy-based monitor tailored for ML perception components. It provides human-interpretable explanations about how different operating conditions affect the reliability of perception components and also functions as a runtime safety monitor. We evaluated our proposed monitor using naturalistic driving datasets as part of an automated driving case study. The interpretability of the monitor was evaluated and we identified a set of operating conditions in which the perception component performs reliably. Additionally, we created an assurance case that links unit-level evidence of *correct* ML operation to system-level *safety*. The benchmarking demonstrated that our monitor achieved a better increase in safety (i.e., absence of hazardous situations) while maintaining availability (i.e., ability to perform the mission) compared to state-of-the-art runtime ML monitors in the evaluated dataset.

Index Terms—interpretable ML, fuzzy, safety, monitor, perception

I. INTRODUCTION

Perception components, which implement tasks such as image classification and object detection using machine learning (ML) models, are vulnerable to changing operating conditions like weather, illumination, and geographical conditions [1]. These conditions can lead to erroneous predictions, referred to as *misperceptions*, possibly leading to hazardous situations. For instance, a missed front object by a camera-based object detector may result in a collision. It is crucial that these ML-based perception components operate reliably and do not generate *hazardous* misperceptions; such misperceptions require proper mitigation strategies. However, effective mitigation requires knowledge about the cause and impact of these

misperceptions on system safety [2], which is challenging for *blackbox* ML models (e.g., Deep Neural Networks (DNN)). Runtime monitoring of machine learning models – input deviation monitoring [3], output plausibilization [4], and neural network monitoring [5] – can detect misperceptions. However, during the design of such monitors, it is essential to consider the evolving nature of ML models due to their re-training, and this must also be taken into account when developing effective mitigation strategies.

Human-interpretable explanations for unreliable ML outputs can facilitate the identification of responsible external factors and targeted mitigation strategies to address their impact on system safety. Interpretability refers to the degree to which a human can understand the cause of a decision made by an ML model [6]. Interpretable ML techniques provide explanations in two forms: model-agnostic [7] or example-based [8], [9]. The former separates explanations from the ML model, while the latter provides explanations within training data (e.g., specific images) to explain ML predictions or underlying data distributions. While interpretable ML techniques offer insights into ML behavior, these explanations must be paired with runtime monitors to achieve effective online fault tolerance.

Using data related to ML model behavior (i.e., reliable/unreliable), it is possible to learn suitable explanations linking external factors and their impact in the form of human-readable fuzzy rules. These explanations serve as evidence for reliable ML behavior. This is required for assuring system safety within *suitable* operating conditions (i.e., where the perception component performs reliably). Our proposed technique operates by learning these fuzzy rules from ML model behavior data and then using them as a *predictive monitor* [10] for runtime ML monitoring. The main contributions of this paper are as follows:

- an approach for learning a *human-interpretable* monitor for ML-based perception component,
- an automated driving case study for evaluating the monitor.

The remainder of this paper is organized as follows: We position our work within state-of-art and provide related work in Section II. Section III provides preliminary background information and motivates our work, outlining the problem

statement. Our proposed fuzzy-based monitor and its application for deriving human-interpretable explanations are presented in Section IV. In Section V, we introduce an automated driving case study that evaluates the fuzzy-based monitor with respect to its interpretability by identifying *suitable* operating conditions for the perception component, supplemented with a safety assurance case for an exemplary driving function. Additionally, we include quantitative benchmarking of its ability to operate as a runtime safety monitor. Finally, Section VII concludes this work and outlines future directions.

II. RELATED WORK

ML models' inherent non-deterministic and black-box nature has driven research into various interpretability techniques, enabling humans to understand the reasoning behind ML predictions. One approach involves training *surrogate* interpretable models (e.g., decision trees, linear/logistic regression) alongside ML models to approximate their predictions at both global and local levels [11]. Unlike global surrogate models, local surrogate models like LIME [12] are trained to approximate an individual ML model prediction. However, these *simple* surrogate models can only *approximate* complex ML models and may not *fully* reflect the ML model predictions. Furthermore, these techniques require identifying interesting sample(s) and training local surrogate models to obtain explanations, making them non-scalable.

Rather than approximating ML model predictions, authors in [13]–[15] propose learning to approximate ML model behavior¹ (e.g., reliable, degraded, unreliable). They introduce the concept of a behavior oracle, which is learned from data comprising simulated environmental conditions impacting training inputs (e.g., blurred camera images) and observing ML model behavior on them (e.g., increase in proportion of misperceptions). Trained behavior oracles can then be used during operation to utilize ML model outputs only during environmental conditions deemed trustworthy (e.g., predicted as *reliable* ML model behavior). Similarly, by leveraging simulation environments, [10] propose to learn an *predictor* serving as a runtime *predictive monitor* for detecting incorrect ML predictions. However, these approaches still rely on another black-box ML model behavior oracle, leaving the interpretability problem unresolved.

Approaches in [5] propose monitoring neural network layers to detect erroneous neuron activations indicative of possible misperceptions. Runtime monitors proposed by [16] provide human-interpretable neuron activations. However, these approaches require a *white-box* assumption of the ML model and do not account for the evolution of the ML model owing to re-training.

In contrast, our proposed fuzzy-based monitor shares similarities with behavior oracles but can be trained incrementally, yielding human-interpretable explanations for its detections. Additionally, the transparent structure of the learned fuzzy rules enables gathering evidence for ML behavior and using a fuzzy inference system as a runtime safety monitor.

¹model's ability to provide *correct* predictions

III. BACKGROUND

ML models are invariably impacted by small changes in their input resulting from their operation in diverse operating conditions. We differentiate between the impact of external environmental factors on perception component reliability and the safety of systems utilizing these perception components; in the former case, such factors are referred to as Perception-Only (PO) conditions.

A. Motivation

Sensing systems used for perception tasks are particularly susceptible to variations in environmental operating conditions under which they operate [17]. These external factors can lead to fluctuations in ML model predictions and their associated confidence scores, resulting in overconfident *incorrect* predictions or underconfident *correct* predictions. This can lead to hazardous situations when ML models produce highly confident erroneous predictions. At the same time, reduced availability of autonomous systems occurs when ML models are overly cautious due to low confidence. To ensure the reliable operation of perception components and produce accurate predictions with high-confidence scores, constituent ML models must undergo additional training with diverse operational context data through MLOps [18], [19] iterations that occur throughout their lifecycle. Developing *assurance cases* [2] for the safe operation of autonomous systems requires identifying Perception-Only (PO) conditions, along with statistical evidence, where perception components operate reliably. Despite re-training the ML models, perception components may still produce misperceptions that potentially lead to hazards. Runtime monitors can complement design-time measures by prohibiting system use in untrusted PO conditions.

B. Problem Statement

For an environmental state, we define its observation $\Omega := (o, x)$ as a tuple consisting of PO conditions o and the input recorded by sensor $x \in \mathcal{X}$. A perception component implements a perception task $T : \mathcal{X} \rightarrow \mathcal{Y}$, producing an output $\hat{y} = T(x)$. For a component C implementing the perception task, an instance (x, y) is a misperception if $y \neq C(x)$. At design-time, given y , we aim to learn an interpretable classifier function $\Xi : \Omega \rightarrow \varphi, \varphi \in \{0, 1\}$, where $\varphi = 1$ refers to the misperception, and 0 otherwise. The resulting interpretability must allow us to identify a set of *suitable* PO conditions, $\mathcal{O} = \{o\}$, in which misperceptions result in *acceptable* proportion of hazards (depicted as γ_C) such that $\forall o \in \mathcal{O}, P(\text{hazard}|\Omega) \leq \gamma_C$, and can be captured in an assurance case.

C. Evolving Fuzzy Systems

Online inference models [20], [21] can learn from data streams and provide predictions in their current trained state. Evolving Fuzzy Systems [22] (EFS) can learn Takagi-Sugeno-Kang [23] fuzzy rules of the form: "IF *input* is A THEN

output”. AIMMo [24], an EFS variant, represents the antecedent A using Empirical Fuzzy Sets [25]. The antecedent A (cf. (2)), a datacloud, has an associated representative prototype, similar to example-based interpretable ML techniques. A *datacloud* is a fuzzy representation of its comprised data instances, similar to a cluster. With AIMMo, it is possible to learn both the dataclouds and fuzzy rules that compose a fuzzy-based inference system from training data. Domain-expert knowledge can be integrated into learning using user-provided prototypes at the beginning of EFS training.

IV. PROPOSED APPROACH

In this section, we present our approach for learning a human-interpretable fuzzy-based monitor that can be used for identifying *suitable* PO conditions. In addition, it can be employed as a runtime safety monitor for ML-based perception components.

A. Overview

An overview of our proposal is presented in Fig. 1. It involves learning a fuzzy inference system, specifically a fuzzy-based classifier (Ξ), used as a runtime predictive monitor m_C for the perception component C . The *Training Data* consists of *labeled* input data - (x, y) (e.g., RGB camera images with 2D bounding boxes) and information about PO conditions - $\mathbf{o}$ (e.g., weather annotations, sensor quality attributes-noise, etc.), under which it was captured.

During the learning stage, as discussed in Section IV-B, the perception component is evaluated with training data (x, y) to obtain φ (cf. III-B) for a specific instance. Additionally, platform-specific dependencies that impact perception reliability are captured as further inputs [26], [27] (e.g., image blur, etc.). We extend the AIMMo algorithm for learning a fuzzy-based classifier Ξ ; our extensions are discussed in section IV-B. Once Ξ has attained sufficient prediction accuracy, i.e., can accurately predict $\hat{\varphi}$ for given conditions $\mathbf{o}$, the learned fuzzy rules and transparent datacloud ($\mathcal{D}$) structure are used to identify suitable PO conditions $\mathcal{O}$ and gather evidence for an assurance case (cf. Section IV-C).

Furthermore, the learned fuzzy rules create a fuzzy inference system that can be used as a predictive runtime monitor (m_C), cf. section IV-D. This runtime monitor can be deployed alongside the perception component C in PO conditions prescribed by $\mathcal{O}$, see Fig. 3, green region with solid *known* boundary. The predictive monitor m_C detects unreliable PO conditions (as seen in the red region with dashed *unknown* boundaries) arising from specification uncertainty in $\mathcal{O}$.

The *Training Data* for learning the monitor need not overlap with the data for ML model re-training and may include validation or testing datasets. The learning process can be repeated using an existing learned fuzzy monitor Ξ whenever a re-trained ML model or new training data is available, thanks to AIMMo’s online learning capability.

B. Learning Dataclouds for PO Conditions

Data-driven learning of a fuzzy-based classifier Ξ is possible from scratch using AIMMo’s algorithm. The primary objective

of this training is to categorize the PO conditions reflected in Ω based on their contribution to the value of φ . Essentially, the learned fuzzy rules provide a mapping of these sets of PO conditions (based on their corresponding dataclouds) to misperceptions by ML models. To represent the monitorable space of PO conditions, we employ a vector feature-space denoted by $\mathbf{o} = [o_1, o_2, \dots, o_j]$, where o_j represents a measurable operating condition constituting a PO condition. These PO conditions can be measured as categorical literals (e.g., weather and scene annotations) or numerical values (e.g., precipitation intensity [mm/hr], visibility [m]), without requiring the assumption of *independence and identical distribution* (iid).

The main steps for learning Ξ are outlined in a flowchart in Fig. 2. The Ξ training can start using a user-provided PO condition for initializing a datacloud ($\mathcal{D}_k$), referred to as a prototype $\mathbf{p}_k = [p_1, p_2, \dots, p_j]$ (cf. D_k available?). However, prototypes can also be discovered as part of the learning process. For $\mathbf{p}_k$, a new datacloud $\mathcal{D}_k$ (cf. Create D_k) is created with membership function denoted by unimodal discrete density as -

$$\mu_k(\mathbf{o}) = \frac{1}{1 + \frac{\|\mathbf{o} - \mathbf{p}_k\|^2}{\sigma_k^2}} \quad (1)$$

where $\sigma^2 = X_k - \|\mathbf{p}_k\|^2$; $\mathbf{p}_k$ and X_k are, respectively, the mean and average scalar product of the dataset comprising PO conditions belonging ($\sim$) to datacloud $\mathcal{D}_k$. The corresponding learned fuzzy rule is represented as -

$$\text{IF } \mathbf{o} \sim \mathbf{p}_k \text{ THEN } \varphi_k = [1, o_1, o_2, \dots, o_j] \mathbf{a}_k \quad (2)$$

where, $\mathbf{a}_k$ represents the learned consequent for datacloud $\mathcal{D}_k$.

We assume the *Training Data* may become available as a single instance or batches throughout the lifecycle of the perception component, inline with MLOps. Therefore, the Ξ ’s prediction accuracy may improve over time with additional training. Upon obtaining i^{th} training instance $(\mathbf{o}_i, \varphi_i)$, we employ a *test-then-train* [28], [29] strategy to track Ξ ’s prediction accuracy. This involves first predicting $\hat{\varphi}_i$ for input $\mathbf{o}_i$, then updating the rolling *Accuracy* in prediction, and finally using $(\mathbf{o}_i, \varphi_i)$ for training. If Ξ exhibits acceptable² prediction accuracy, its training state is saved, and a human-on-loop is notified to review it for evidence collection (cf. Inform human-on-loop). Otherwise, the training instance is checked for belonging to existing dataclouds within Ξ (cf. belongs to $D_{0..k}$?). In case of a match, the corresponding datacloud’s structure (membership function, prototype, support, etc.) is updated as part of the training process (cf. Update trainable parameters).

The support $\text{supp}(\mathcal{D}_k) = \{\mathbf{o} | \mu_k(\mathbf{o}) > 0\}$ represents the set of PO conditions which can be categorized within a datacloud $\mathcal{D}_k$. During the datacloud structure update, the values of $\mathbf{p}_k$ and $\text{supp}(\mathcal{D}_k)$ are recursively updated as part of trainable parameters. This updating of trainable parameters ($\mu_k, \mathbf{a}_k$) is

²user-defined based on the application of perception component

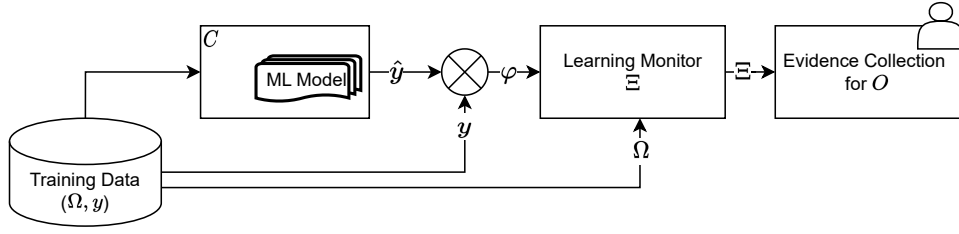


Fig. 1: Overview of Proposed Fuzzy-based Monitor Learning

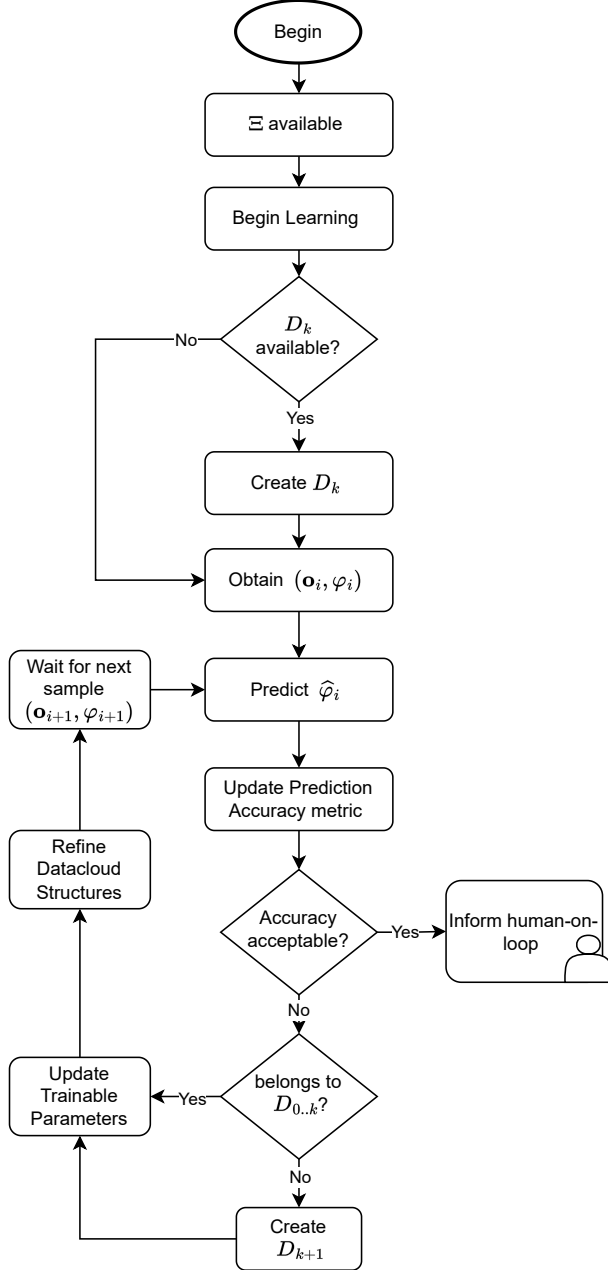


Fig. 2: Learning Monitor: Datacloud Discovery

carried out as described in [24]. If no *belonging* datacloud is found (i.e., no nearest existing $\mathbf{p}_{0...k}$), a new datacloud $\mathcal{D}_{k+1}$ is created based on this training instance (cf. `Create  $\mathcal{D}_{k+1}$`).

The existing dataclouds $\mathcal{D}_{0...k+1}$ are continuously evaluated according to their contribution to predictions $\hat{\varphi}$ and may be merged or pruned accordingly (cf. `Refine Datacloud Structures`). The learning process resumes with the next incoming training sample $(\mathbf{o}_{i+1}, \varphi_{i+1})$.

C. Interpreting PO Conditions

The saved state of Ξ comprises user-defined or automatically discovered dataclouds and their mapping to the value of φ , contained in fuzzy rules. Each datacloud $\mathcal{D}_k$ is *transparent* and can be represented by its prototype $\mathbf{p}_k$ and associated membership function μ_k . We utilize these dataclouds and associated information to specify suitable PO conditions (i.e., $\mathcal{O}$) and gather evidence for the creation of a safety assurance case (see section III-B).

1) *Prototype as PO Condition Representation*: We propose to employ the automatically discovered prototypes to specify the PO conditions. Consider an example discovered prototype: $\mathbf{p}_0 = [\text{clear}, \text{citystreet}, \text{daytime}, 0.67]$ representing weather, scene, time-of-day, and image contrast measurable operating conditions o_j ; mapped to a $\varphi = 0$ score by its corresponding fuzzy rule. For a camera-based perception task, an image corresponding to $\mathbf{p}_0$ serves as a human-interpretable example of PO conditions represented by datacloud $\mathcal{D}_0$. Alternatively, a human domain expert may provide a camera image as an input prototype (e.g., used to extract $\mathbf{p}_1$) which will be used in the learning process to create a datacloud $\mathcal{D}_1$.

The prototype $\mathbf{p}_k$ serves as an example of *trusted* or *untrusted* PO conditions that result in 0 or 1 value of $\hat{\varphi}$, respectively. In our example, PO conditions represented by $\mathcal{D}_0$ could be considered trusted conditions since there are no predicted misperceptions. Our case study, in Section V, provides a detailed discussion along with a human-readable specification for $\mathcal{O}$ created from the *shortlisted* $\mathcal{D}_k$'s (i.e., with evidence for reliable operation). Fig. 3 shows examples of reliable (*red*) and unreliable (*green*) dataclouds.

2) *Collecting Evidences for Perception Component*: The learned dataclouds and associated information contribute as statistical evidence regarding the reliable operation of the perception component (cf. Fig 1). For example, the probability of misperception (mP) within instances belonging to a datacloud

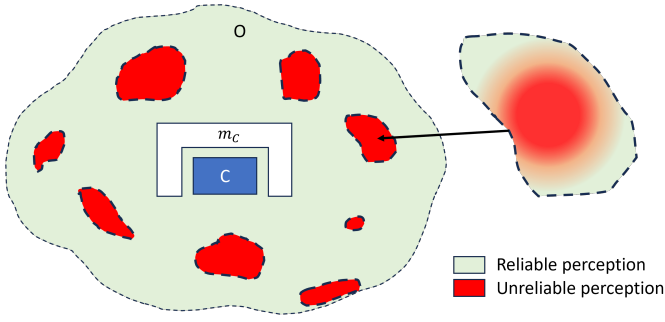


Fig. 3: Representing PO conditions with dataclouds

constitutes evidence of perception components reliability and is computed as -

$$P(mP|\mathcal{D}_k) = \frac{1}{|\text{supp}(\mathcal{D}_k)|} \sum_{\mathcal{D}_k} 1[\varphi == 1]$$

where the numerator denotes the count of instances within datacloud with misperception, i.e., $\varphi = 1$. The evidence obtained for perception component reliability (e.g., $P(mP|\mathcal{D}_k)$) must be linked with perception component safety (cf. section III-B, γ_C), and has been discussed in detail with a case study in section V.

Using a single sampled instance like p_0 to represent a specific PO condition feature space poses two challenges. First, the datacloud ($\mathcal{D}_0$) represented by p_0 is only a sample and may not fully represent the target feature space. Second, $\mathcal{D}_0$ may include untrusted instances (i.e., $\varphi \neq \hat{\varphi}$), such as those with a very low degree of membership to $\mathcal{D}_0$ ($\mu_0 \ll 1$). See Fig. 3 for a fuzzy representation of untrusted (red) datacloud. For the former case, we account for a sampling error (Δ_q) in our calculations for creating statistical evidence, cf. (IV-C2). For an evidence related metric γ computed for datacloud $\mathcal{D}_k$, the upper bound Δ_q of $q\%$ confidence error is calculated as -

$$\Delta_q = v(q) \sqrt{\frac{\gamma(1-\gamma)}{|\text{supp}(\mathcal{D}_k)|}} \quad (3)$$

where, $v(q) = \frac{100+q}{200}$ is the quantile of standard normal distribution. For the latter case (i.e., $\mu \ll 1$), we propose to utilize a runtime monitor (cf. Fig. 3) that can accurately predict if the PO condition is (un)trustworthy, discussed in section IV-D.

D. Creating Runtime Monitor

To complement design-time safety and tackle the challenges discussed in the previous section, a runtime monitor is used to safeguard the perception component, see Fig. 3. Runtime monitoring essentially serves as an online fault tolerance mechanism detecting *untrustworthy* PO conditions possibly included in suitable dataclouds. This is because Ξ 's classification accuracy is often less than 100% (cf. section IV-B) or due to insufficient consideration of the sampling error Δ_q , cf. (3). We propose to use Ξ as a runtime predictive monitor m_C , monitoring the perception component C , collectively referred

to as (C, m_C) . The runtime monitor m_C is evaluated with a case study, and the benchmarking results have been discussed in section V-D.

In this section, we presented our approach for learning human-interpretable fuzzy monitor, comprising data clouds and fuzzy rules. Using this fuzzy monitor for a perception component was discussed to derive suitable PO conditions (supported by human-verifiable evidence) at design-time and its role as a runtime safety monitor.

V. CASE STUDY: AUTOMATED DRIVING

We consider an Automated Driving System (ADS) as a case study to demonstrate our proposed approach. ADS utilizes ML-based perception components to partially or fully automate certain driving tasks. However, ADS features enabling driving automation, e.g., Traffic-Jam Assist, Highway Pilot, etc., must be deployed within a specified set of operating conditions, referred to as Operational Design Domain (ODD) [30].

In this section, we evaluate the human interpretability of learned fuzzy-monitor and its role in supporting human-driven activities such as safety assurance case creation for the perception component (cf. section IV-C). The main objective of this study is to derive a human-readable ODD specification corresponding to *suitable* PO conditions observed within a training dataset, wherein an exemplary ML-based object-detector (i.e., perception component) operates reliably. Furthermore, this ODD specification must be supplemented with a safety assurance case, and the perception component must be monitored at runtime, refer section III-B.

Firstly, we introduce the use-case scenario used within this case-study in section V-A. We describe our experimental setup for realizing the use-case scenario in section V-B. The experimental results concerning learning of the fuzzy monitor have been presented and discussed in section V-C. Finally, we benchmark our learned fuzzy monitor against several other interpretable (e.g., decision-tree) and non-interpretable (e.g., neural network-based) classifiers used as *predictive* runtime safety monitors in section V-D.

A. Use-case Scenario

We consider the use-case scenario *Stopped Car Ahead* (SCA) described in [31] and utilize their *Integration Safety Case for Perception* (ISCaP) template to create a safety assurance case. It includes the evidence for the reliable operation of the ML-based 2D object-detector within the specified ODD, in our case $\mathcal{O}$. The characteristics of the integrated perception component are described as follows:

The output of the object-detector is fed to a *Tracker*, which maintains the past states of detected objects. The *Tracker* loses track of a detected object after 9 consecutive missed detections by the object detector, i.e., 9 False Negatives (FNs). Furthermore, based on physics-based calculation, additional 5 FNs lead to a crash with SCA.

We limit the focus of this work only to missed object detections, i.e. FNs. In our case, the misperception (mP) by the object-detector is considered hazardous (i.e., HmP) if we observe an FN within the Region of Interest, see Fig. 6, *blue* bounding-box. Here, the missed car (depicted by a *red* bounding-box) by a 2D object detector can be considered as HmP.

B. Experimental Setup

We use the naturalistic driving dataset $\{(x_i, o_i)\}_n$ for this case-study. The BDD100k [32] evaluation dataset comprises camera images (x) captured under diverse PO conditions (o), with $n = 10,000$. In addition to available *categorical* values for operating condition attributes (i.e., *weather* (o_1), *scene* (o_2), *timeofday* (o_3)), we capture platform-dependent attributes: *numeric* - (*brightness* (o_6), *clearness_score* (o_7), *contrast_score* (o_8)) and *boolean* - (*blurry* (o_4), *low_contrast* (o_5)). As an exemplary object-detector, we use the pre-trained and publicly available ML model *yolov5x6* [33] as a perception component. In this study, we focus on changing weather conditions, as they have the most significant impact on the defined perception reliability. We perform a 70:30 split of the BDD100k dataset into training and validation datasets comprising 6127 and 2626 samples, respectively³.

The behavior of the exemplary 2D object detector was observed on the training and validation split and recorded as training data d_{train} and d_{val} comprising (o, φ) for monitor learning, respectively. An example instance referring to a *yolov5x6* misperception from this training data consists of training input: $o = [clear, highway, daytime, False, False, 18.16, 0.12, 2.18]$, and training output: $\varphi = 1$.

C. Experiment 1: Human-Interpretability

We conduct an experiment to qualitatively evaluate the interpretability of the learned fuzzy monitor in this section.

1) *Learned Fuzzy Monitor*: We trained a Ξ from scratch using the d_{train} dataset as described in section IV-B. After training, the saved state of Ξ yielded 10 dataclouds, i.e. $\mathcal{D}_{0...9}$.

Representative images of the corresponding discovered prototypes for their respective dataclouds are shown in Fig. 4.

Additionally, we obtained 10 fuzzy rules (cf. (2)) which provide a mapping of the dataclouds to the *yolov5x6* model behavior (e.g., *reliable* ($\varphi = 0$) or *unreliable* ($\varphi = 1$)). The discovered prototypes corresponding to dataclouds $\mathcal{D}_1$ and $\mathcal{D}_7$ map to unreliable behavior, see Fig. 4d and 4e containing misperceptions (*red* bounding boxes); other dataclouds map to reliable behavior.

2) *Human-Readable Specification*: The ISO34503 [30] standard provides guidelines for creating a human-readable ODD specification. We employ the *default* specification mode, requiring the implementation of ODD-exit monitor [34] for *included* and *excluded* operating conditions mentioned in the ODD specification.

By shortlisting the dataclouds for which the *yolov5x6* perception component exhibits reliable behavior (i.e., no mP), it is possible to create a human-readable ODD specification, as per ISO34503 guidelines. We treat prototypes shown in Fig 4 as representative of the PO conditions; see section IV-C1. Based on the mapping provided by the discovered fuzzy-rule (see Fig. 4, *red/green* text annotations), the operating conditions from $\mathcal{D}_2, \mathcal{D}_6, \mathcal{D}_9$ along with other shortlisted dataclouds (i.e., mapping to reliable *yolov5x6* behavior), we can create an ODD specification as shown in Fig. 5, see Line 1-3.

However, the dataclouds $\mathcal{D}_1$ and $\mathcal{D}_7$ must be excluded from this ODD. This is accomplished by using Conditional ODD statements, as per ISO34503 guidelines, cf. Fig. 5, Lines 5-14. The PO conditions corresponding to $\mathcal{D}_1$ comprise platform-specific operating conditions, collectively captured by intermediate operating condition *visibility*, are shown in Fig. 5, Lines 7-9. This conditional statement implies that the *yolov5x6* perception component is prohibited from operating in *clear* weather conditions on *city streets* during the *night* if the *visibility* operating conditions belong to p_1 . Similarly, the conditional statement corresponding to the Conditional Exclude of $\mathcal{D}_7$ has been shown in Lines 11-14.

The operating conditions mentioned in the ODD specification must be monitored, and the use of the perception component should be prohibited upon encountering excluded operating conditions. Authors in [35] provide a method for the fuzzy interpretation of ODD and its monitoring. However, the task of ODD monitoring is beyond the scope of this work. In the next section, we will discuss the use of discovered data clouds to gather evidence for creating an assurance case related to the safe operation of ADS using the *yolov5x6* perception component within our specified ODD context.

3) *Assurance Case Creation*: ISCaP establishes a connection between the top-level safety goal (i.e., the probability bound on misperception-caused collision $P_C(HmP) \leq \gamma_C$) and the bottom-level component properties (i.e., misperception (mP) occurrence rate). A simplified abstraction of the ISCaP safety assurance case for our SCA HmP is depicted in Fig. 6. The bound on the top-level safety goal is initially broken down for our SCA HmP, as per the γ_A decomposition. The decomposed sub-goals can be described as follows:

- Guarantee that all related HmPs in the SCA use case are covered; we assume this is met for our evaluation.
- Conservatively assuming that all missed SCAs are hazardous, i.e., leading to rear-end collision.
- The occurrence rate of SCA is bounded, assuming SCA is encountered at every 500m distance.
- mP occurrence rate factoring different PO conditions within ODD context is bounded.

To create a safety assessment case for our use-case scenario, we instantiate the ISCaP template and obtain evidence for the leaf-level solutions (see Fig. 6, orange *Solution*). The evidence comprises of the frame-level HmP occurrence rate within a set of PO conditions (γ_μ) and occurrence probability of these PO

³samples with *undefined* annotations are removed



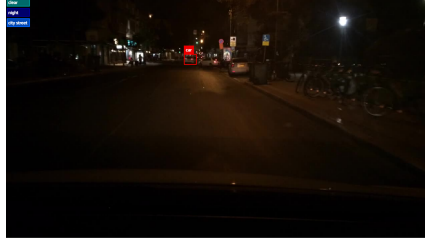
(a) $\mathcal{D}_2$: overcast,city-street,dawn/dusk



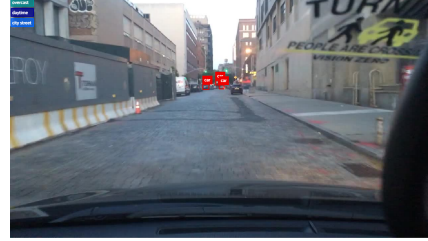
(b) $\mathcal{D}_6$: clear,city-street,daytime



(c) $\mathcal{D}_9$: snowy,city-street,daytime



(d) $\mathcal{D}_1$: clear,city-street,night



(e) $\mathcal{D}_7$: overcast,city-street,daytime

Fig. 4: Representative images of the discovered dataclouds

```

1 Include weather is [clear, snowy, overcast, partly cloudy]
2 Include scene is [city-street, highway]
3 Include timeofday is [daytime, dawn/dusk, night]
4
5 ##Conditional ODD Statements
6 #Exclude Datacloud 1
7 Conditional Exclude
8   [brightness, contrast_score, clearness_score] of
9   visibility
10  for [clear, city-street, night] is
11    [18.167, 2.18, 0.12]
12 ## Exclude Datacloud 7
13 Conditional Exclude
14   [brightness, contrast_score, clearness_score] of
15   visibility
16   for [overcast, city-street, daytime] is
17     [111.2, 3.284, 0.48]

```

Fig. 5: ISO34503 compliant ODD Specification

conditions ($\gamma_{o\uparrow}$), and is calculated as⁴:

$$\gamma_{o\uparrow} = \frac{|supp(\mathcal{D}_k)|}{|\Omega|} \quad (4a)$$

$$\gamma_{\mu} = \frac{1}{|supp(\mathcal{D}_k)|} \sum_{\mathcal{D}_k} 1[HmP] \quad (4b)$$

As described in section IV-C2, we collect the bottom-level evidence γ_{μ} and $\gamma_{o\uparrow}$ in order to perform a bottom-up computation of γ_A corresponding to our target HmP concerning SCA; results are presented in Table I. The calculation of mP occurrence rate (cf. Fig. 6, γ_B) in the safety case excludes evidence pertaining to $\mathcal{D}_1$ and $\mathcal{D}_7$ since they are excluded in our ODD specification (cf. Fig. 5).

Given a saved training state of Ξ (cf. section IV-B), assuming a user-defined threshold γ_C , the calculated value of residual safety bound ($\gamma_{res} = \gamma_C - \gamma_A$) must be reviewed by

⁴accounting for sampling error of $\Delta_{0.99}$

TABLE I: Accumulated Evidence for ISCaP Safety Case

Safety Bound	Description	Value for $\mathcal{D}_{0,2,3,4,5,6,8,9}$
mP occurrence rate	$\gamma_B = \sum_k (\gamma_{sca, HmP} \cdot \gamma_{o\uparrow})$	2.3185×10^{-1}
SCA occurrence rate	Based on SCA occurrence every 500m: $\gamma_{SCA} = \frac{40[km/hr]/10[fps]}{500[m]}$	2.2×10^{-3}
crash rate	Conservatively assume HmP always leads to crash ($\gamma_{cr} = 1$)	1.0
HmP _{SCA} rate	$\gamma_A = \gamma_{cr} \cdot \gamma_B \cdot \gamma_{SCA}$	5.1×10^{-4}

a human safety-expert to assess the validity of created safety case. If judged unacceptable, the Ξ training could be resumed with additional training data to discover new dataclouds, cf. Fig. 2.

The *qualitative* evaluation results highlight the interpretability of our learned fuzzy monitor, demonstrated through the creation of human-readable ODD Specification (cf. Fig. 5), supplemented with a safety assurance case (cf. Fig. 6 and Table II).

D. Experiment 2: Runtime Safety Monitoring

We conduct a follow-up experiment to quantitatively evaluate the learned fuzzy monitor as a runtime safety monitor using the ML monitor evaluation framework provided by [36]. Unlike generic evaluation metrics such as accuracy, recall, and precision, which consider every mP equally hazardous, this framework provides flexibility to model the specificity of ML-based perception components.

We assume the existence of an ODD-exit monitor for our specified ODD. Therefore, for this experiment, we filtered the

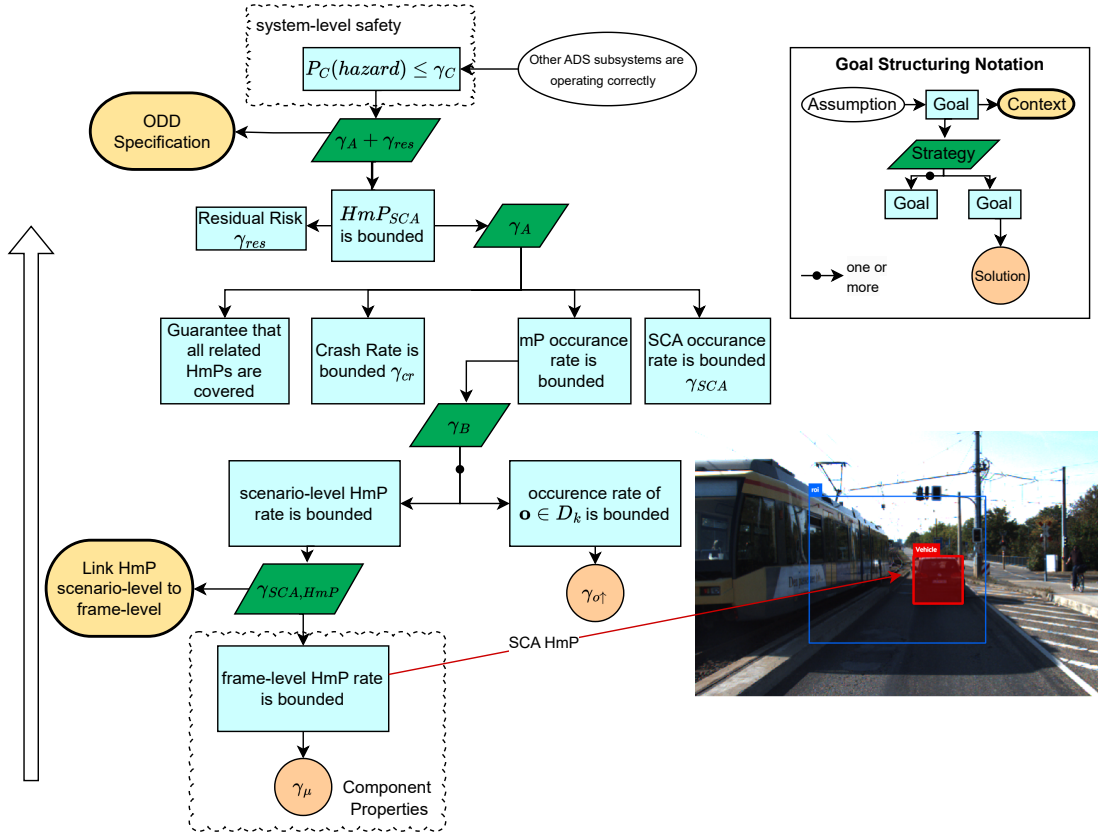


Fig. 6: Safety Assurance Case for use-case scenario Stopped Car Ahead (SCA)

validation dataset d_{val} based on the ODD (cf. Fig. 5). 87% of the samples were found to be within this ODD.

1) *Evaluation Criteria*: This framework measures the safety and reliability improvement achieved by the use of a runtime safety monitor (m_C) along with perception component C , i.e., (C, m_C) , based on the following criteria:

- *Safety Gain* (SG_{m_C}): safety improvements resulting from m_C ,
- *Residual Hazard* (RH_{m_C}): remaining hazard in the system after using m_C ,
- *Availability Cost* (AC_{m_C}): system performance decreases because of m_C .

$$SG_{m_C} = \frac{1}{n} \sum_{i=1}^n \left(R_{(C, m_C)}^S(x_i) - R_C^S(x_i) \right) \quad (5a)$$

$$RH_{m_C} = \frac{1}{n} \sum_{i=1}^n \left(R_{C^*}^S(x_i) - R_{(C, m_C)}^S(x_i) \right) \quad (5b)$$

$$AC_{m_C} = \frac{1}{n} \sum_{i=1}^n \left(R_C^M(x_i) - R_{(C, m_C)}^M(x_i) \right) \quad (5c)$$

These metrics are calculated based on the cumulative safety return R^S and mission return R^M over n samples, assuming a good coverage of an ODD. The $R_{C^*}^S$ represents the reward calculation assuming ground-truth knowledge (y) during C

usage. A binary variable τ represents the *ground-truth* error status of C . We define the safety and mission returns as follows:

$$R_{(C, m_C)}^S(x) = \begin{cases} 0 & \text{if } \tau = 1 \text{ and } m_C = 0 \\ 1 & \text{otherwise} \end{cases} \quad (6)$$

$$R_{(C, m_C)}^M(x) = \begin{cases} 0 & \text{if } \tau = 0 \text{ and } m_C = 1 \\ 1 & \text{otherwise} \end{cases} \quad (7)$$

The monitored system (C, m_C) is penalized if the error exists, $\tau = 1$, but the runtime monitor does not detect it, $m_C = 0$, (i.e., *fn*). Additionally, the system (C, m_C) must be penalized if it detects an error, $m_C = 1$, but there is no error, $\tau = 0$ (i.e., *fp*). Please note the distinction between the incorrect prediction of the ML-based object detector (FN, FP) and monitor detections (*fn*, *fp*).

2) *Monitor Benchmarking Results*: We define two types of error status: mP status (τ_{mP}) and HmP status (τ_{HmP}). The $\tau_{mP} = 1$ indicates a misperception (i.e., missed object), and $\tau_{HmP} = 1$ indicates a hazardous misperception (i.e., collision with SCA due to mP). We benchmark the performance of our proposed m_C , which uses the learned fuzzy monitor Ξ as a runtime safety monitor. To compare our approach with others, we replace our Ξ with several other well-known

classifiers [37]; we trained these classifiers on d_{train} . Additionally, we used a *Random* monitor comprising a random value generator ($\in \{0,1\}$) for $\hat{\varphi}$, inspired by [38]. Firstly, we evaluate the monitor’s ability to detect untrustworthy PO conditions (i.e., $\hat{\varphi}$ prediction accuracy). The evaluation results are shown in Table II; the *green* arrow indicates desired high or low scores, with **bold** values indicating the best scores.

TABLE II: Benchmarking runtime monitors: τ_{mP}

Ξ in m_C	SG $\uparrow$	RH $\downarrow$	AC $\downarrow$
<i>Random</i>	0.2018	0.1942	0.2357
Gaussian Naive Bayes [39]	0.1846	0.2902	0.2082
Decision Tree [40]	0.19345	0.22086	0.20639
Neural Network [41]	0.03273	0.44216	0.04974
Fuzzy-based (ours)	0.0175	0.3968	0.01447

Secondly, we evaluate the monitor’s ability to intervene only in hazardous situations since not every misperception is hazardous, limited to the SCA use case. The results of this evaluation are presented in Table III.

Discussion: A good runtime monitor should increase system safety without decreasing its availability. Ideally, a runtime monitor must only intervene when necessary, in our case, HmPs. However, unnecessary interventions (i.e., *fp*’s) result in a reduced availability of perception components.

In conducting the experiment with samples satisfying derived ODD (cf. Fig. 5), it is expected that the proportion of HmPs from missed SCA should be limited by γ_A (refer to Table I). Ideally, the perception component should operate reliably within the confines of this ODD, obviating the need for a monitor m_C . However, since the ODD for this particular case study (cf. Fig. 5) was created with the data clouds discovered using training data d_{train} . Therefore, in the event of incorrect detections by m_C over d_{val} (i.e., *unseen* data during training), the runtime monitor m_C has been implemented as an online fault tolerance mechanism, and its performance has been evaluated accordingly.

In case of the evaluation for the ability to accurately detect misperceptions (cf. Table II), we see that Gaussian Naive Bayes and Decision Tree monitors are as good as a *Random-monitor* when it comes to improving safety gains (SG) and reducing residual hazard (RH). This partially confirms the statement by [38], which states that existing ML input-checking monitors might be as good as random monitors. Regardless, we see from the high value of availability costs (AC) for these classifiers that the outputs from C will not be utilized in driving functions due to a large number of *fp*’s by m_C . This leads to frequent disengagement of the dependent driving functions, e.g., frequent driver-take-over

requests. Neural Network-based classifier has comparatively lower SG but higher RH, indicating an inability to improve safety no better than other classifiers. However, it exhibits significantly lower AC, indicating fewer *fp*’s. In comparison, our fuzzy-based monitor shows very low AC scores and comparable RH scores to that of other classifiers. Since we already deploy C in PO conditions where it operates reliably (i.e., ODD), we argue that the low SG is owing to limited potential for safety gains by introducing m_C .

Within the context of our SCA use case, we have evaluated the ability of runtime safety monitors to prevent actual hazardous misperception(s) and the ability to ignore benign misperceptions. Gaussian Naive Bayes provides the highest SG scores, while the neural network-based classifiers exhibit the highest RH reduction. However, our fuzzy monitor has comparable RH reduction capability without incurring high AC like other monitors. This indicates the ability of our fuzzy-based monitor to safeguard the perception component against SCA HmP without impacting the availability of perception components, unlike other evaluated classifiers. Other evaluated classifiers were trained using batch data, with multiple training passes for neural network-based classifiers. In contrast, our fuzzy-based classifier was trained using single training instances incrementally in an online “test-then-train” manner, showing its prediction accuracy improvements during training.

In summary, we have evaluated our proposed fuzzy monitor both qualitatively and quantitatively. The results highlight its human interpretability and comparable performance as a runtime safety monitor. Due to this interpretability and transparent structure of the monitor, for our automated driving case study, we were able to create a human-readable ODD specification (along with a safety assurance case).

VI. LIMITATIONS AND THREATS TO VALIDITY

Our approach utilizes a data-driven fuzzy monitoring learning technique, meaning that the accuracy of misperception detection significantly depends on data quality and training volume. The quality of human-provided labels within the dataset for operating condition annotations vastly impacts the quality of learned prototypes. Based on the AIMMo learning settings, a large number of dataclouds might be discovered, requiring increased human efforts for their review.

The case study aims to provide human-interpretable sample prototypes for creating ODD specifications, required for human-driven activities like safety assurance cases and monitoring. However, using a single prototype for an ODD specification may lead to *specification* uncertainty due to sampling error, negatively affecting its representativeness. We recommend using large labeled datasets for learning and further testing, supplemented by additional fault tolerance mechanisms to improve the derived ODD coverage and improve safety. The evaluation results assume an ODD-exit monitor for the monitoring task and to prevent operation in scenarios outside the ODD.

TABLE III: Benchmarking runtime monitors: τ_{HmP}

Ξ in m_C	SG $\uparrow$	RH $\downarrow$	AC $\downarrow$
<i>Random</i>	0.001523	0.00228	0.431073
Gaussian Naive Bayes [39]	0.00218	0.00349	0.3906
Decision Tree [40]	0.00114	0.00380	0.3956
Neural Network [41]	0.0048	0.00087	0.7686
Fuzzy-based (ours)	0.0	0.00495	0.03198

VII. CONCLUSION AND FUTURE OUTLOOK

We have introduced our approach for learning a human-interpretable monitor for an ML-based perception component that is influenced by external operating conditions. The transparent structure of the learned fuzzy-based monitor enables the collection of evidence for the reliable operation of the perception component. We evaluated the learned fuzzy monitor using an automated driving case-study. Our qualitative evaluation results indicate that the learned fuzzy-monitor is well-suited for ODD specification (specifically, identifying a set of operating conditions where the perception component performs reliably) and can support human-driven Verification and Validation activities. The quantitative evaluation shows that when used alongside the perception component, the fuzzy-monitor can enhance *safety* by reducing residual hazards (low RH) without compromising the perception component's *availability* (low AC). Therefore, it can serve as an effective online fault tolerance mechanism.

The learned monitor explains the ML model behavior impact in the form of human-interpretable fuzzy rules which are easy to understand. This understanding is essential for creating strong assurances for the reliable operation of the perception component and the safety of the system that utilizes the perception component. As future work, we plan to investigate the use of online fuzzy monitor learning in DevOps cycles for scaling its operation to diverse operating conditions through (semi-)automated ODD expansion of Automated Driving System features.

REFERENCES

- [1] Mario Bijelic, Tobias Gruber, Fahim Mannan, Florian Kraus, Werner Ritter, Klaus Dietmayer, and Felix Heide. Seeing Through Fog Without Seeing Fog: Deep Multimodal Sensor Fusion in Unseen Adverse Weather. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11682–11692, 2020.
- [2] Betty HC Cheng, Robert Jared Clark, Jonathon Emil Fleck, Michael Austin Langford, and Philip K McKinley. Ac-ros: assurance case driven adaptation for the robot operating system. In *Proceedings of the 23rd acm/IEEE international conference on model driven engineering languages and systems*, pages 102–113, 2020.
- [3] Feiyang Cai and Xenofon Koutsoukos. Real-time out-of-distribution detection in learning-enabled cyber-physical systems. In *2020 ACM/IEEE 11th International Conference on Cyber-Physical Systems (ICCPs)*, pages 174–183. IEEE, 2020.
- [4] Sina Shafaei, Stefan Kugele, Mohd Hafeez Osman, and Alois Knoll. Uncertainty in machine learning: A safety perspective on autonomous driving. In Barbara Gallina, Amund Skavhaug, Erwin Schoitsch, and Friedemann Bitsch, editors, *Computer Safety, Reliability, and Security*, pages 458–464. Cham, 2018. Springer International Publishing.
- [5] Chih-Hong Cheng, Georg Nührenberg, and Hirotoshi Yasuoka. Runtime monitoring neuron activation patterns. In *2019 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, pages 300–303. IEEE, 2019.
- [6] Tim Miller. Explanation in artificial intelligence: Insights from the social sciences. *Artificial intelligence*, 267:1–38, 2019.
- [7] Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. Model-agnostic interpretability of machine learning. *arXiv preprint arXiv:1606.05386*, 2016.
- [8] Agnar Aamodt and Enric Plaza. Case-based reasoning: Foundational issues, methodological variations, and system approaches. *AI communications*, 7(1):39–59, 1994.
- [9] Been Kim, Rajiv Khanna, and Oluwasanmi O Koyejo. Examples are not enough, learn to criticize! criticism for interpretability. *Advances in neural information processing systems*, 29, 2016.
- [10] Luca Bortolussi, Francesca Cairoli, Nicola Paoletti, Scott A Smolka, and Scott D Stoller. Neural predictive monitoring and a comparison of frequentist and bayesian approaches. *International Journal on Software Tools for Technology Transfer*, 23(4):615–640, 2021.
- [11] Scott M Lundberg and Su-In Lee. A unified approach to interpreting model predictions. *Advances in neural information processing systems*, 30, 2017.
- [12] Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. ” why should i trust you?” explaining the predictions of any classifier. In *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*, pages 1135–1144, 2016.
- [13] Michael Austin Langford and Betty HC Cheng. “know what you know”: Predicting behavior for learning-enabled systems when facing uncertainty. In *2021 International Symposium on Software Engineering for Adaptive and Self-Managing Systems (SEAMS)*, pages 78–89. IEEE, 2021.
- [14] Michael Austin Langford, Kenneth H Chan, Jonathon Emil Fleck, Philip K McKinley, and Betty HC Cheng. Modalas: addressing assurance for learning-enabled autonomous systems in the face of uncertainty. *Software and Systems Modeling*, pages 1–21, 2023.
- [15] Michael Austin Langford and Betty H.C. Cheng. A Modular and Composable Approach to Develop Trusted Artificial Intelligence. In *2022 IEEE International Conference on Autonomic Computing and Self-Organizing Systems (ACSOS)*, pages 121–130, September 2022.
- [16] Florian Geissler, Syed Qutub, Michael Paulitsch, and Karthik Pattabiraman. A low-cost strategic monitoring approach for scalable and interpretable error detection in deep neural networks. In *International Conference on Computer Safety, Reliability, and Security*, pages 75–88. Springer, 2023.
- [17] Aniket Salvi, Gereon Weiss, and Mario Trapp. Adaptively managing reliability of machine learning perception under changing operating conditions. In *2023 IEEE/ACM 18th Symposium on Software Engineering for Adaptive and Self-Managing Systems (SEAMS)*, pages 79–85. IEEE, 2023.
- [18] Houssein Guissouma, Moritz Zink, and Eric Sax. Continuous safety assessment of updated supervised learning models in shadow mode. In *2023 IEEE 20th International Conference on Software Architecture Companion (ICSA-C)*, pages 301–308. IEEE, 2023.
- [19] Camille Fayollas, Hugues Bonnin, and Olivier Flebus. Safeops: A concept of continuous safety. In *2020 16th European Dependable Computing Conference (EDCC)*, pages 65–68. IEEE, 2020.
- [20] Óscar Fontenla-Romero, Bertha Guijarro-Berdiñas, David Martínez-Rego, Beatriz Pérez-Sánchez, and Diego Peteiro-Barral. Online machine learning. In *Efficiency and Scalability Methods for Computational Intellect*, pages 27–54. IGI global, 2013.
- [21] Jacob Montiel, Jesse Read, Albert Bifet, and Talel Abdesslem. Scikit-multiflow: A multi-output streaming framework. *Journal of Machine Learning Research*, 19(72):1–5, 2018.
- [22] Edwin Lughofer. *Evolving fuzzy systems-methodologies, advanced concepts and applications*, volume 53. Springer, 2011.
- [23] Yuanpeng Zhang, Guanjin Wang, Ta Zhou, Xiuyu Huang, Saikit Lam, Jiabao Sheng, Kup Sze Choi, Jing Cai, and Weiping Ding. Takagi-sugeno-kang fuzzy system fusion: A survey at hierarchical, wide and stacked levels. *Information fusion*, 101:101977, 2024.
- [24] Plamen P Angelov, Xiaowei Gu, and Jose C Principe. Autonomous learning multimodel systems from data streams. *IEEE Transactions on Fuzzy Systems*, 26(4):2213–2224, 2017.
- [25] Plamen P. Angelov and Xiaowei Gu. Empirical Fuzzy Sets. *International Journal of Intelligent Systems*, 33(2):362–395, 2018.
- [26] Simon Burton, Lydia Gauerhof, Bibhuti Bhusan Sethy, Ibrahim Habli, and Richard Hawkins. Confidence arguments for evidence of performance in machine learning for highly automated driving functions. In *Computer Safety, Reliability, and Security: SAFECOMP 2019 Workshops, ASSURE, DECSO, SASSUR, STRIVE, and WAISE, Turku, Finland, September 10, 2019, Proceedings 38*, pages 365–377. Springer, 2019.
- [27] Gereon Weiss, Philipp Schleiss, Daniel Schneider, and Mario Trapp. Towards integrating undependable self-adaptive systems in safety-critical environments. In *Proceedings of the 13th International Conference on Software Engineering for Adaptive and Self-Managing Systems*, pages 26–32, 2018.
- [28] Avrim Blum, Adam Kalai, and John Langford. Beating the hold-out: Bounds for k-fold and progressive cross-validation. In *Proceedings of*

the twelfth annual conference on Computational learning theory, pages 203–208, 1999.

- [29] Maciej Grzenda, Heitor Murilo Gomes, and Albert Bifet. Delayed labelling evaluation for data streams. *Data Mining and Knowledge Discovery*, 34(5):1237–1266, 2020.
- [30] ISO. Iso 34503:2023 road vehicles — test scenarios for automated driving systems — specification for operational design domain. Technical report, 2023.
- [31] Rick Salay, Krzysztof Czarnecki, Hiroshi Kuwajima, Hirotoshi Yasuoka, Vahdat Abdelzad, Chengjie Huang, Maximilian Kahn, Toshihiro Nakae, et al. The missing link: Developing a safety case for perception components in automated driving. *SAE International Journal of Advances and Current Practices in Mobility*, 5(2022-01-0818):567–579, 2022.
- [32] Daniel Seita. Bdd100k: A large-scale diverse driving video database. *The Berkeley Artificial Intelligence Research Blog. Version*, 511:41, 2018.
- [33] Glenn Jocher. Ultralytics yolov5, 2020.
- [34] On-Road Automated Driving (ORAD) Committee. *Taxonomy and definitions for terms related to driving automation systems for on-road motor vehicles*. SAE International, 2021.
- [35] Aniket Salvi, Gereon Weiss, Mario Trapp, Fabian Oboril, and Cornelius Buerkle. Fuzzy interpretation of operational design domains in autonomous driving. In *2022 IEEE Intelligent Vehicles Symposium (IV)*, pages 1261–1267. IEEE, 2022.
- [36] Joris Guerin, Raul Sena Ferreira, Kevin Delmas, and Jérémie Guiochet. Unifying Evaluation of Machine Learning Safety Monitors. In *2022 IEEE 33rd International Symposium on Software Reliability Engineering (ISSRE)*, pages 414–422, October 2022.
- [37] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.
- [38] Raul Sena Ferreira, Jean Arlat, Jérémie Guiochet, and Hélène Waeselynck. Benchmarking safety monitors for image classifiers with machine learning. In *2021 IEEE 26th Pacific Rim International Symposium on Dependable Computing (PRDC)*, pages 7–16. IEEE, 2021.
- [39] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.
- [40] scikit learn. `sklearn.tree.decisiontreeclassifier`. 2023.
- [41] SciKit Learn. 1.17. neural network models (supervised), 2017.