

Leveraging Vision-Language Models for Visual Grounding and Analysis of Automotive UI

Benjamin Raphael Ernhofer^{*+}, Daniil Prokhorov^{*+}, Jannica Langner^{*} and Dominik Bollmann^{*}

^{*} SPARKS Solutions GmbH, Ingolstadt, Germany

⁺ Equal contribution

Abstract

Modern automotive infotainment systems necessitate intelligent and adaptive solutions to manage frequent User Interface (UI) updates and diverse design variations. This work introduces a vision-language framework to facilitate the understanding of and interaction with automotive UIs, enabling seamless adaptation across different UI designs. To support research in this field, AutomotiveUI-Bench-4K, an open-source dataset comprising 998 images with 4,208 annotations, is also released. Additionally, a data pipeline for generating training data is presented.

A Molmo-7B-based model is fine-tuned using Low-Rank Adaptation (LoRa), incorporating generated reasoning along with visual grounding and evaluation capabilities. The fine-tuned Evaluative Large Action Model (ELAM) achieves strong performance on AutomotiveUI-Bench-4K ([model](#) and [dataset](#) are available on Hugging Face). The approach demonstrates strong cross-domain generalization, including a +5.6% improvement on ScreenSpot over the baseline model. An average accuracy of 80.8% is achieved on ScreenSpot, closely matching or surpassing specialized models for desktop, mobile, and web, despite being trained primarily on the automotive domain. This research investigates how data collection and subsequent fine-tuning can lead to AI-driven advancements in automotive UI understanding and interaction. The applied method is cost-efficient, and fine-tuned models can be deployed on consumer-grade GPUs.

1 Introduction

Automotive infotainment systems are rapidly evolving, characterized by increasing complexity, dynamic interfaces, and personalization [23]. With manufacturers frequently deploying over-the-air updates and diverse User Interface (UI) designs becoming prevalent across models, these systems demand intelligent and adaptive solutions capable of handling significant variations [40]. This constant evolution necessitates intelligent systems capable of dynamically interpreting the visual layout and semantic meaning of interfaces, moving beyond reliance on fixed structural assumptions for UI validation.

Visual Language Models (VLMs) offer a promising approach by integrating computer vision with natural language understanding, enabling systems to interpret visual information and user intent in a more human-like manner [19]. While these models have demonstrated success in understanding and interacting with interfaces in domains such as desktop, mobile, and web [4, 10], their application to the distinct environment of automotive infotainment systems remains significantly underexplored. This represents a research gap, given the unique demands and rapid evolution of in-vehicle interfaces. Applying VLMs effectively within the automotive context introduces non-trivial challenges. Automotive UIs exhibit vast heterogeneity across car models and manufacturers, featuring custom icons, menus, and interaction paradigms [23]. Therefore, a robust model must generalize effectively across disparate screen

layouts and styles while maintaining high precision in understanding on-screen elements and their context. Moreover, sophisticated interaction often requires reasoning about the UI state, extending beyond simple object detection or element localization.

To bridge this gap, a highly adaptable vision-language framework is introduced, designed for understanding, interacting, and validating automotive infotainment systems and enabling seamless adaptation across different UI designs. The key contributions of this work include:

1. The fine-tuned Molmo-based Evaluative Large Action Model **ELAM-7B**, optimized for automotive UI understanding, capable of processing visual input and natural language test action and evaluation instructions.
2. The release of **AutomotiveUI-Bench-4K**, an open-source dataset featuring 998 infotainment images with 4,208 annotations, hosted on Hugging Face, to serve as a benchmark and foster research in this domain.
3. A **synthetic data generation pipeline** developed to enhance model performance and generalization for small VLMs (7B or less) through parameter-efficient fine-tuning.

Through empirical evaluation, the efficacy of the proposed framework is demonstrated. The fine-tuned model establishes a new performance benchmark on the AutomotiveUI-Bench-4K dataset and exhibits remarkable adaptability, significantly improving results on the cross-domain ScreenSpot task (+5.6%). Its overall accuracy (80.8% average) competes favorably with, and in some cases surpasses, models specifically designed for non-automotive domains (e.g., ShowUI [20]), as shown in Table 3. The robustness of the model is highlighted by its performance even with a restricted training dataset, both in terms of size and domain. It is important to note, however, that the automotive infotainment domain itself encompasses a diverse range of systems and functionalities. The following sections cover the system architecture, the curation of the novel dataset, and the comprehensive experimental validation, collectively showcasing the significant potential of AI-driven methods for advancing automotive UI understanding and interaction.

2 Related Work

This section reviews the literature relevant to automated UI interaction and validation. It begins by examining the current landscape of VLMs and the emerging field of Large Action Models (LAMs) applied to general UI understanding. Subsequently, existing domain-specific datasets and common model adaptation techniques are analyzed, highlighting their limitations for automotive applications. The section concludes by discussing traditional validation methodologies to contextualize and define the specific research gaps addressed in this paper.

2.1 Vision Language Models and UI Understanding

VLMs have emerged as a significant area of research, bridging the gap between visual and textual information processing. These models integrate visual capabilities with Large Language Models (LLMs) to handle complex tasks such as image captioning, visual question answering, and multimodal dialogue systems [9]. Key contributions in this field include models like CLIP [28], which demonstrates strong performance in zero-shot image classification by aligning visual and textual embeddings, and BLIP-2 [17], which introduces an efficient pre-training strategy using frozen image encoders and LLMs. Other notable models include Flamingo [1], which exhibits significant few-shot learning capabilities, and LLaVA [21], which enhances LLMs for multimodal understanding through visual instruction tuning.

A significant advancement in this domain is the Molmo series [5], which introduces approaches to multimodal understanding with notable capabilities in spatial reasoning and accurate localization. Unlike traditional VLMs that primarily rely on discrete vision encoders, Molmo models demonstrate enhanced capability in understanding spatial relationships and providing localization through their pointing

mechanism. This pointing capability allows models to not only identify UI elements but also provide exact coordinate information for their locations, representing a significant development for UI interaction tasks. The Molmo series is trained on the comprehensive PixMo dataset [5], which contains millions of image-text pairs with spatial annotations, including pointing data that enables models to ground textual descriptions to specific normalized pixel coordinates in images.

As VLMs evolved, models specializing in UI interaction were developed, leading to the establishment of the term Large Action Model (LAM) [33]. Comprehensive surveys [41, 25, 35] have addressed critical research questions concerning the development of LAMs for UI tasks. It has been highlighted that a specialized VLM capable of accurate visual grounding is crucial for strong performance. Frameworks like ShowUI [20] and SeeClick [4] leverage VLMs to map natural language instructions to UI element interactions. Similarly, LAMs such as OS-Atlas [37] extend VLMs by incorporating action generation capabilities suitable for agentic UI navigation tasks. Furthermore, smaller fine-tuned models like TinyClick [27] (based on Microsoft’s Florence-2-Base [38]) and UGround [10] (based on Qwen2-VL [34]) have been developed specifically for UI tasks, often processing UI images and interpreting actions to generate corresponding point coordinates.

The Molmo series differentiates itself from other UI-focused models through its training methodology that incorporates both general multimodal understanding and specialized pointing capabilities. While most LAMs are trained primarily on UI-specific datasets, Molmo models benefit from the diverse PixMo dataset, which includes a smaller but significant portion of UI-related data from the Android-Control [18] dataset alongside general visual content. This approach enables better generalization capabilities while maintaining strong performance on UI tasks. The pointing mechanism in Molmo models provides more accurate spatial understanding compared to traditional region-based approaches, such as those relying on bounding boxes, used in other models.

However, these models are typically trained on general-purpose interfaces from desktop, mobile, and web domains. This limits their applicability to automotive UIs, which exhibit domain-specific design patterns and custom iconography. Notably, they lack explicit mechanisms for evaluating UI states (e.g., verifying if a seat belt warning is displayed), an essential requirement for automotive validation. While the Molmo series demonstrates enhanced spatial reasoning capabilities through its pointing mechanism, it still faces the same domain adaptation challenges when applied to automotive interfaces with their unique visual characteristics and functional requirements.

2.2 Domain-Specific UI Datasets and Model Adaptation

Existing UI datasets, including AMEX [2] and Android In the Wild [29], focus on mobile or web environments. These datasets primarily annotate UI elements with interaction labels (e.g., “tap the settings icon”) but rarely include evaluative statements. The PixMo dataset [5], while comprehensive in its multimodal coverage with millions of image-text pairs and innovative pointing annotations, also follows this pattern. Its UI-related subset focuses primarily on interaction rather than validation tasks. The dataset’s strength lies in its rich spatial annotations that enable accurate localization, but it lacks the evaluation-centric labels necessary for automotive UI validation scenarios.

In contrast, the proposed **AutomotiveUI-Bench-4K** introduces a dual-label structure (*Test Action* and *Expected Result*) to simulate real-world validation scenarios, aligning with the need for compliance checks. This approach differs fundamentally from existing datasets, including PixMo. While these datasets excel in interaction modeling, they do not address the critical evaluation dimension required for automotive UI testing.

Parameter-efficient fine-tuning techniques such as Low-Rank Adaptation (LoRa) [12] have been used to adapt VLMs for UI tasks, as seen in UGround [10]. However, these approaches focus solely on interaction and neglect evaluation. Likewise, prior synthetic data pipelines [10, 27] generate interaction-focused annotations. These adaptation methods are extended in this work by the introduction of a novel pipeline. This pipeline balances *Test Actions* with *Expected Results* to train evaluation-aware models.

2.3 Traditional Automotive UI Validation and Research Gaps

The validation of automotive UIs traditionally relies on methodologies like specification-based testing [14] and Hardware-in-the-Loop (HiL) testing [15, 16]. Within HiL setups, visual assessment often employs Optical Character Recognition (OCR) and template matching [24, 32]. These conventional approaches are fragile, struggling with visual variations from updates or themes. Furthermore, they lack a deep semantic understanding of interface elements [30, 7]. Consequently, the maintenance overhead necessitates more intelligent solutions. This work bridges two critical gaps in prior research:

1. **Automotive-Specific VLMs:** Existing VLMs, including advanced models like the Molmo series with their spatial reasoning capabilities, are trained on generic UIs. This limits their generalization to automotive interfaces with custom iconography and layouts.
2. **Evaluation-Centric Training:** Unlike most LAMs that emphasize interaction (including Molmo's pointing mechanism for interaction tasks), an evaluation-focused training approach is introduced in this research, enabling analysis and validation of automotive UIs.

Therefore, VLMs are leveraged to overcome the limitations of traditional methods, establishing a more robust and semantically aware framework for automotive UI validation. The development and evaluation of this framework constitute the core of this work. Specific recommendations for its integration into production-ready HiL testing environments are not provided.

3 Method

The methodology for adapting VLMs for automated automotive UI validation is outlined in this section. Domain adaptation challenges are addressed through an approach encompassing synthetic data generation and fine-tuning of baseline models, leading to the development of a specialized VLM for accurate visual grounding and evaluation of automotive UIs.

3.1 Domain Adaptation Challenges and Model Selection

Extensive research exists on LAMs and UI agents. However, these models primarily focus on interactions and implicitly derive evaluative functions by inferring necessary actions. For the targeted application of validating requirements via natural language test cases, explicit evaluation of expected outcomes is necessary. Existing smaller fine-tuned models (7B or less), such as TinyClick [27] and UGround [10], gather UI data from mobile, desktop, and web applications. This limits their applicability to automotive UIs. Domain-specific adaptation is required due to the distinct design patterns, custom iconography, and varied display hardware properties of modern automotive systems.

Specifically, the following challenges necessitated the proposed methodology:

1. **Adapt VLMs for the domain of automotive UI:** Modern automotive systems exhibit diverse UI designs influenced by brand, platform, driving mode, and display hardware properties. Icons pose a particular challenge, as little overlap exists with generic UIs from desktop, mobile, or web applications.
2. **Assert evaluation capability:** Explicit evaluation is not considered in prior work. To evaluate requirements or specifications, expected results must be prompted. These results require visual grounding and subsequent evaluation.
3. **Ensure onsite deployment and control of models:**
 - Strict information security requirements are often held by Original Equipment Manufacturers (OEMs). The leveraging of proprietary models such as OpenAI GPT-4o or Google Gemini Pro 2.0 presents challenges due to data privacy concerns and potential leakage of sensitive information to external servers.

- Fine-tuning such models presents another challenge. While OpenAI and Google provide fine-tuning services for training and deployment, the details of the training process and the fine-tuned model itself remain closed source.
- Challenging validation and prompt adaptation iterations to ensure intended functionality arise from the discontinuation of closed-source models.
- The operation of models on local servers or even consumer-grade GPUs is highly desirable and necessary for a cost-effective solution.

To address these challenges, **Evaluative Large Action Model (ELAM)** is introduced, extending existing LAMs through the integration of explicit evaluation capabilities. The model is trained with synthetic data generated by an accompanying pipeline (Section 3.2). This pipeline is designed to address the unique characteristics of automotive UIs and the need for evaluative statements.

To establish a suitable baseline model for fine-tuning, four existing VLMs were evaluated: *TinyClick* (0.27B) [27], *Molmo-7B-D-0924* [5], *UGround-VI-7B* (Qwen2-VL-based) [10], and *InternVL2.5-8B* [3]. *TinyClick* and *UGround* are specifically designed for UI interaction and point coordinate generation. However, their broader capabilities and suitability as a robust foundation for evaluation were limited. General-purpose VLMs, *Molmo* and *InternVL2.5*, have also been trained on UI data. The best performance (Table 2) among the considered models was achieved by *Molmo-7B-D-0924* when evaluated on the proposed AutomotiveUI-Bench-4k dataset. Consequently, it was chosen as the foundation of ELAM-7B.

3.2 Synthetic Data Pipeline

Object-level captioning for automotive UIs presents significant challenges due to the need to incorporate functional, positional, and visual attributes to uniquely identify each element. Differentiating between *Test Actions* and *Expected Results* is required. Both necessitate object-level captions describing either the command initiating element actuation or a statement to be evaluated for the current UI. Furthermore, for *Expected Results*, a binary status (passed or failed) must be assigned, reflecting the validity of the expectation within the image. Empirical observation revealed that human annotators tend to under-represent failed expectations, likely due to the cognitive challenge of formulating incorrect expectations for a displayed UI image. It is also important to note that having human experts create data annotations is a time-consuming and expensive process. Therefore, a hybrid approach, leveraging human expertise for UI element selection via bounding box annotations combined with automated caption generation by large teacher models (e.g., GPT-4o, Gemini 2.0 Flash Thinking), offers a practical balance between annotation quality and time investment. Automated UI element extraction methods, such as OmniParser V1.5 [22] and a custom YOLOv8 [13] fine-tuned to automotive UI, were found to be insufficiently reliable, frequently resulting in misaligned bounding box detections that negatively impacted the correctness of generated captions.

The synthetic data pipeline (Figure 1) was used to train ELAM-7B. Bounding box regions from the dataset were prompted using the Set-of-Mark (SoM) technique [39]. Inspired by UGround [10], a bounding box indicated by an arrow was utilized. For *Test Actions*, prompts were designed as short yet comprehensive instructions that unambiguously map to a single UI element (Figure 2a). Prior to generating each *Test Action*, the model was instructed to provide a detailed reasoning of the process for identifying the target UI element, including its semantic, positional, and visual characteristics, as well as any relevant relationships to other elements, such as the current context or list headings (Figure 2). Utilized prompts can be found in Appendix A.3.

To improve the generation of *Expected Results*, the process was modified to include a prior successful test action leading to the current menu or UI state (Figure 2b). Specifically, the model first generated a prior test action, followed by reasoning for the expectation and its “passed/failed” conclusion. Initially, this approach exhibited the same bias as human annotators and primarily generated passed validations. This issue was mitigated by explicitly prompting for evaluations that entail a failed

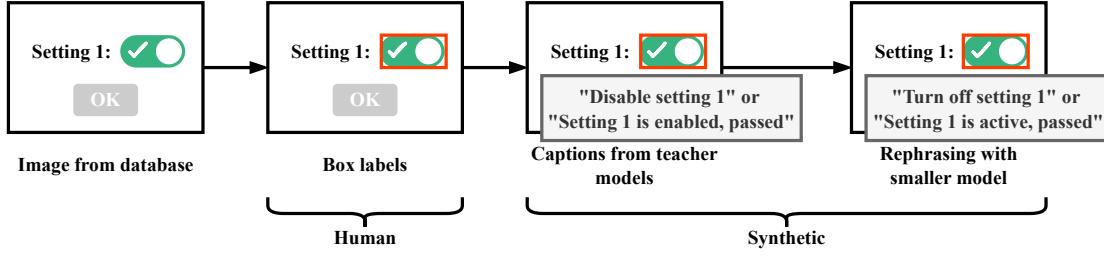


Figure 1: Synthetic data generation pipeline

result (Appendix A.3.3). After manually reviewing a small set of samples for GPT-4o, Gemini 2.0 Flash Thinking and Claude 3.5 Sonnet, Gemini (*gemini-2.0-flash-thinking-exp-01-21*) was selected to generate the training data. It was chosen for its better adherence to the required output structure, which was necessary for automatic parsing.

To mitigate potential overfitting on the specific linguistic style of the teacher model, the text for reasoning, test actions, and expectations were rephrased using a smaller model (*gpt-4o-mini-2024-07-18*). This rephrasing step is crucial for ensuring robustness and generalizability when processing inputs from real-world test case data. The teacher model, configured with a temperature of zero to minimize hallucinations, exhibited a tendency towards repetitive use of verbs and sentence structures, necessitating this diversification strategy. The contribution of rephrasing is explored in Table 4.

3.3 Model Fine-Tuning

For efficient training of baseline models, LoRa [12] was utilized. This parameter-efficient fine-tuning method injects small, trainable matrices into the model’s layers, allowing for significant improvements without retraining the entire large model. This approach offers benefits such as accelerated training, reduced training data needs, a smaller storage footprint for the adapters, lower memory requirements, and the potential for dynamic adapter swapping in deployment, which eliminates the need to load different large models for various tasks.

As previously established, *Molmo-7B-D-0924* was chosen as the foundation for ELAM due to its strong performance on the AutomotiveUI-Bench-4k dataset among evaluated baseline models. This foundation provided a robust starting point. Fine-tuning with LoRA then enabled efficient adaptation to the specific tasks of automotive UI validation, leveraging a synthetically generated dataset that explicitly includes both *Test Actions* and *Expected Results* with their corresponding passed/failed states. This fine-tuning specifically aims to imbue ELAM-7B with the evaluation capabilities lacking in prior interaction-focused LAMs, and to shift the learned domain to automotive UI. Further details regarding this fine-tuning can be found in Section 5.1.

4 Open Source Infotainment Validation Dataset: AutomotiveUI-Bench-4K

Despite the availability of numerous datasets for benchmarking LAMs in desktop, mobile, and web UIs [29, 37, 36, 6, 43, 2, 8], a significant domain gap remains in their application to automotive UIs. To address this limitation, **AutomotiveUI-Bench-4K** is introduced and released as a new dataset specifically designed for this domain. This dataset comprises 998 images with 4,208 annotations from 15 brands, including Audi, BMW, Ford, Porsche, and Tesla. This diverse collection of modern vehicle UIs was annotated by experienced professional software test engineers specializing in Human-Machine Interface (HMI) evaluation. In addition to images from automotive OEMs, the dataset also incorporates screenshots of Apple CarPlay and Google Android Auto. Images from OEMs were captured in 4K resolution using cameras and sourced from either a research vehicle fleet or through collaboration with local

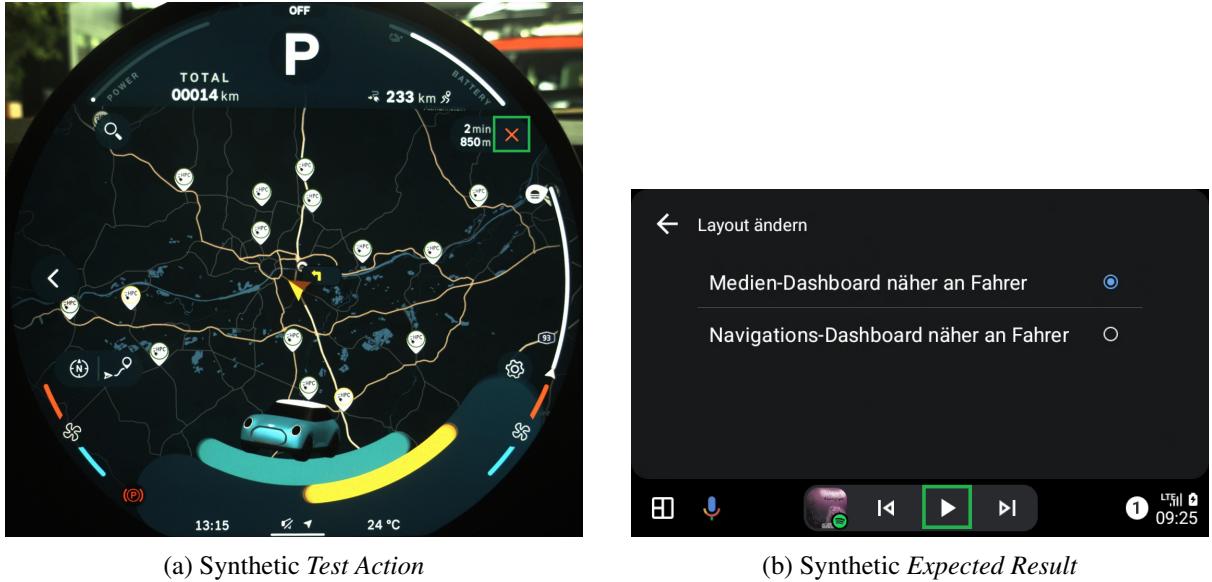


Figure 2: Examples of synthetic data for training, illustrating (a) a test action “**Tap the ‘X’ button to cancel the displayed route guidance information on the map.**” with prior reasoning “*The requested element is an “X” icon located in the top right corner of the map display, semantically representing a close or cancel action, positioned next to the text “2 min” and “850 m,” which likely refers to route guidance information, with the parent element being the map view.*” and (b) a failed expected result “**The media control button in the bottom bar is expected to be a pause icon, indicating that media is currently playing. → Failed**” with prior reasoning “*The requested element is located in the bottom media control bar and is identified as a play symbol, which semantically represents the action to start or resume media playback. The evaluation highlights that there could be a misunderstanding regarding the icon’s function, as it may be incorrectly expected to represent a pause action instead. This confusion emphasizes the importance of clear iconography in user interface design to ensure users can easily understand the intended actions associated with each control.*” in different infotainment systems.

car dealerships. To ensure consistent image perspective and focus on the relevant UI elements, the OEM images were rectified and cropped to the active display area. Android Auto and CarPlay screenshots were directly exported. The systems featured in this dataset are exclusively touchscreen-based. While some systems include auxiliary physical buttons, these were not considered in the analysis due to their primarily static functionality. This focused approach is justified by the increasing ubiquity of touch-screen interfaces as the dominant HMI in modern vehicles. By concentrating solely on this technology, the dataset provides a unified corpus for analysis. The resulting data homogeneity is ideal for robust comparative analysis and the development of VLMs specifically for this contemporary UI paradigm.

Although the dataset primarily comprises systems from recent model years, the system representing the earliest model year is a 2018 Volkswagen Tiguan. Languages that were included are German and English. Instructions and evaluation requests are written in English. German text was either translated or directly quoted. Real-world test engineering documentation often exhibits a pattern of short sentences and occasional grammatical deviations. Therefore, to create a dataset that realistically represents this domain, it is crucial to preserve these stylistic features in the preprocessing steps. Table 1 provides further information about the data distribution. The actions and results are unrelated, as test actions can lead to different UI states, and screen trajectories are not considered. There are two annotation classes in the dataset:

1. **Test Action:** Describes an interaction with a single UI element as an imperative sentence (“set A/C to max” in Figure 3).
2. **Expected Result:** Represents a testable expectation, defined per image, focusing on the required

appearance, context, or state of UI elements within that image. This expectation is evaluated to determine a passed/failed outcome. (“Passenger’s climate zone is synced to driver” in Figure 3)

Table 1: Label distribution in AutomotiveUI-Bench-4K

Category	Subcategory	Total	EN	DE
Images	-	998	454	544
Annotations	-	4,208	1,988	2,220
Test Action	-	2,269	1,059	1,210
Expected Result	Total	1,939	929	1,010
Expected Result	Passed	1,375	662	713
Expected Result	Failed	564	267	297



Figure 3: Example for a *Test Action* (red) “set A/C to max” and for an *Expected Result* (green) “Passenger’s climate zone is synced to driver” (Passed)

5 Experiments

This section details the experimental procedure for fine-tuning and evaluating ELAM. The following subsections cover the specific configuration used for training, the performance of the fine-tuned model against several baselines on the AutomotiveUI-Bench-4K and ScreenSpot datasets. The impact of different training data preprocessing strategies is explored through ablation studies, the relationship between language semantics and model performance is investigated using t-SNE, and a detailed visual error analysis is conducted.

5.1 Experimental Configuration

To train ELAM, LoRa was applied to all linear layers within the Molmo architecture. Training was conducted for 2 epochs with a global batch size of 128, achieved using a local batch size of 8 and gradient accumulation. A learning rate of $1e-4$ was employed. An optimal LoRa rank $r = 64$ was

selected based on an ablation study (Table 5), with the α parameter set to r and a fixed dropout rate of 0.05. ModelScope’s SWIFT framework [42] was used to conduct fine-tuning.

Training data for ELAM was generated using the data pipeline described in Section 3.2. Gemini (*gemini-2.0-flash-thinking-exp-01-21*) was selected for training dataset generation due to its superior adherence to structured output compared to other models. The detailed prompts used for generating the training datasets are displayed in Appendices A.3.1 to A.3.3.

Generated text for reasoning, test actions, and expectations was rephrased using a smaller model (*gpt-4o-mini-2024-07-18*). The synthetic dataset comprises 17,708 annotations across 6,230 images. This includes 5,952 *Test Actions*, 6,190 passed and 5,566 failed *Expected Results*. Prompt templates utilized for training, evaluation, and inference are detailed in Appendix A.2. Model training was performed on two NVIDIA H100 80 GB PCIe GPUs. For evaluation experiments on the AutomotiveUI-Bench-4K dataset, an NVIDIA GeForce RTX 4090 GPU with 24 GB VRAM was used.

Performance of the models was assessed using accuracy metrics, all expressed as percentages. For the visual grounding task, two distinct accuracy metrics were employed: TA_{vg} represents the visual grounding accuracy for *Test Actions*, while ER_{vg} denotes the visual grounding accuracy for *Expected Results*. For both TA_{vg} and ER_{vg} , accuracy was determined by verifying that the generated point or the centroid of the predicted box was contained within the annotated ground truth box. Furthermore, for the evaluation of *Expected Results*, ER_{evl} was used, representing the classification accuracy (passed or failed). To facilitate a fine-grained analysis of language-specific performance, the image dataset was categorized into English and German subsets. Corresponding language-specific metrics are indicated by the superscripts ^{EN} and ^{DE} for English and German UIs, respectively. In cases where a generated evaluation could not be parsed from the model’s response, it was assigned the inverse of the ground truth value.

5.2 Evaluating Performance with AutomotiveUI-Bench-4K

The performance of various baseline and fine-tuned models is presented in Table 2. ELAM demonstrates a notable improvement in localization as indicated by TA_{vg} (+16.3%) and ER_{vg} (+6.1%) compared to the baseline model. Specifically for evaluation, the fine-tuned model achieved higher accuracy ER_{evl} (+11.3%), precision (+8.9%), and recall (+6.4%) (Figure 4). This enhancement is primarily driven by a significant reduction in both false positives and false negatives, indicating fine-tuning effectively improved the model’s ability to correctly identify passed tests while simultaneously reducing misclassifications of failed tests as passed, and vice versa.

The findings suggest that models within the 7B parameter range exhibit no significant performance degradation when processing images with predominantly German text. A significant challenge for most baseline models was the generation of consistently parseable output. For example, InternVL2.5 exhibited good evaluation performance. However, it rarely generated bounding box coordinates. TinyClick and UGround were evaluated exclusively on their localization performance, as they lack evaluative capabilities. To explore the possibility of faster inference through task distribution across models, TinyClick was selected for LoRa-fine-tuning, given its small size of only 0.27 billion parameters. This approach (*LAM-270M*), however, did not produce performance on par with Molmo-7B, as shown in Table 2.

A notable benefit of ELAM is its accessibility for deployment. Unlike many larger state-of-the-art VLMs, the 7 billion parameter ELAM can be deployed on a current consumer-grade NVIDIA GPU with at least 24 GB VRAM. The invoke time has been recorded as an average of 2.4 seconds for test actions and 3.4 seconds for expected results when running on an NVIDIA GeForce RTX 4090 GPU, using the default Hugging Face Transformers backend and full resolution images of AutomotiveUI-Bench-4K.

To assess ELAM’s potential on the AutomotiveUI-Bench-4K dataset, an evaluation was conducted by an experienced quality assurance engineer. This expert achieved improvements of +6.9% TA_{vg} , +8.9% ER_{vg} , and +15.0% ER_{evl} compared to ELAM. This result provides a high-performance human expert benchmark and suggests that enhanced outcomes can be accomplished through additional training.

This analysis with a human expert was instrumental in identifying inherent limitations within the synthetic dataset itself. The expert’s higher performance serves as an important indicator for data-related issues. For instance, the boxes may be insufficiently sized for visual grounding, or screen elements may not be sufficiently described, highlighting potential areas for dataset improvement. A detailed analysis is conducted in Section 5.6.

Table 2: Evaluation on AutomotiveUI-Bench-4K. TA_{vg} and ER_{vg} represent visual grounding accuracies for *Test Actions* and *Expected Results*, respectively. ER_{evl} denotes the classification accuracy (passed/failed) for *Expected Results*. Language specific metrics are indicated by the superscripts EN and DE for English and German UIs. All values are accuracies in percent.

Model	$\downarrow TA_{vg}$	TA_{vg}^{DE}	TA_{vg}^{EN}	ER_{vg}	ER_{vg}^{DE}	ER_{vg}^{EN}	ER_{evl}	ER_{evl}^{DE}	ER_{evl}^{EN}
InternVL2.5-8B* [3]	26.6	26.1	27.0	5.7	6.3	5.1	64.8	60.4	69.0
TinyClick [27]	61.0	54.6	67.8	53.3	47.3	59.2	-	-	-
UGround-V1-7B (Qwen2-VL) [10]	69.4	68.9	69.9	55.0	54.4	55.7	-	-	-
Molmo-7B-D-0924 [5]	71.3	70.9	71.8	71.4	69.8	72.9	66.9	67.8	66.0
LAM-270M (TinyClick)	73.9	66.9	81.1	59.9	54.6	65.1	-	-	-
ELAM-7B (Molmo)	87.6	87.5	87.6	77.5	77.0	77.9	78.2	78.5	77.8
Human Domain Expert	94.5	94.5	94.5	86.4	87.3	85.5	93.2	93.7	92.8

* Since the InternVL2.5 series model natively supports bounding boxes, these were used for evaluation instead of points.

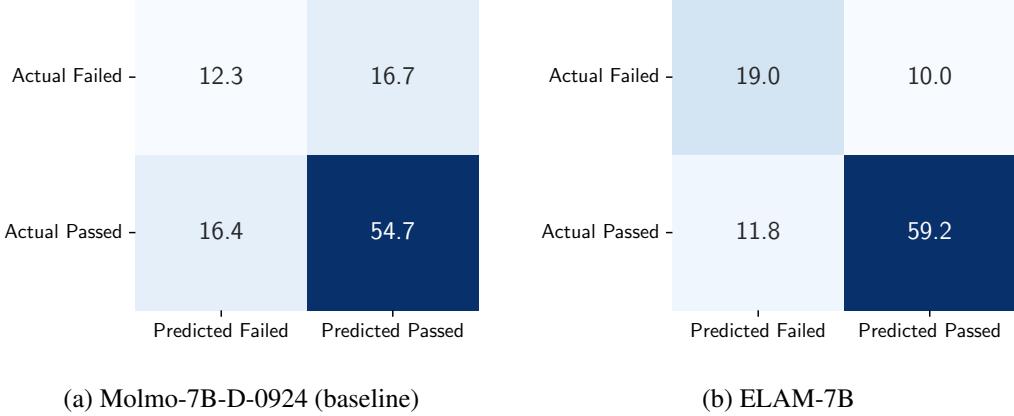


Figure 4: Confusion matrices for the *Expected Results* evaluation classification in AutomotiveUI-Bench-4K, with values normalized to percentages.

5.3 Evaluating Generalizability with ScreenSpot

To determine if the fine-tuning resulted in overfitting to the automotive UI domain, ELAM-7B was applied to the ScreenSpot dataset [4] and compared to *SeeClick* [4], *ShowUI* [20], *OS-Atlas-Base-7B* [37], *Molmo-7B-D-0924 /Molmo-72B-0924* [5], and Qwen2-VL-based *UGround-V1-7B* [10], as shown in Table 3. The fine-tuning procedure was found not to impair the generalizability of Molmo. The average score per category was improved by 5.6% compared to its baseline (*Molmo-7B-D-0924*) and by 2.2% compared to *Molmo-72B-0924*.

5.4 Ablation Study

Ablation studies were performed to evaluate the contributions of simple reasoning and rephrasing modules integrated into the synthetic data generation pipeline. Training runs were conducted using distinct

Table 3: $TA_{vg}^{(EN)}$ results on ScreenSpot for selected models taken from [11]

ScreenSpot	\downarrow Avg	Mob.-Text	Mob.-Icon	Desk.-Text	Desk.-Icon	Web-Text	Web-Icon
SeeClick [4]	53.4	78.0	52.0	72.2	30.0	55.7	32.5
ShowUI-2B [20]	75.1	92.3	75.5	76.3	61.1	81.7	63.6
Molmo-7B-D-0924 [5]	75.2	85.4	69.0	79.4	70.7	81.3	65.5
UGround-V1-2B (Qwen2-VL) [10]	77.7	89.4	72.0	88.7	65.7	81.3	68.9
Molmo-72B-0924 [5]	78.6	92.7	79.5	86.1	64.3	83.0	66.0
ELAM-7B (Molmo)	80.8	94.5	79.5	89.2	70.7	85.7	65.0
OS-Atlas-Base-7B [37]	81.0	93.0	72.9	91.8	62.9	90.9	74.3
UGround-V1-7B (Qwen2-VL) [10]	86.3	93.0	79.9	93.8	76.4	90.9	84.0

preprocessing strategies, the configurations of which are summarized in Table 4. The prompts from Appendix A.2 were used for experiments that incorporate reasoning, whereas the phrase “Think step by step, conclude” was replaced by “Determine” for runs that did not utilize reasoning. A LoRa-rank of 64 was chosen to fine-tune all linear layers for the ablation experiments. It was demonstrated that the inclusion of reasoning significantly enhances performance, as evidenced by improvements in both visual grounding (TA_{vg} , ER_{vg}) and evaluation (ER_{evl}) accuracy. The critical role of reasoning for robust generalization to real-world data in potential future deployments is highlighted by this observation.

To optimize the LoRa configuration, a separate ablation study focusing on the rank parameter r was conducted, the results of which are presented in Table 5. The data was preprocessed utilizing reasoning and rephrasing. This analysis determined that a rank of 64 yielded optimal performance. In all LoRa-based experiments, the alpha parameter was set to r , and the LoRa-dropout rate was fixed at 0.05.

Table 4: Prompting ablations

Settings	TA_{vg}	ER_{vg}	ER_{evl}
Baseline	81.7	73.5	73.5
Rephrasing	83.9	73.7	71.1
Reasoning	86.4	76.2	77.0
Reas.+Rephr.	87.4	77.2	78.6

Table 5: LoRa ablations

Rank	TA_{vg}	ER_{vg}	ER_{evl}
16	84.6	76.1	77.7
32	85.0	76.4	77.1
64	87.4	77.2	78.6
128	87.1	77.2	77.6

5.5 Analysis of Utterance Embedding Space vs. VLM Performance

To investigate the relationship between the semantic representation of user utterances and VLM performance, particularly within automotive subdomains, two distinct task types were analyzed: *Test Actions* and *Expected Results*, as outlined in Section 4. For both types, text embeddings of the utterances were generated using the *nomic-embed-text-v1.5** model. Their structure was then visualized using t-SNE dimensionality reduction [31]. Unsupervised k-Means clustering ($k = 8$) was applied to the high-dimensional embeddings to identify potential thematic groups. Cluster labels were generated post-hoc via a LLM (*gpt-4o-2024-11-20*) interpretation of sampled utterances. The t-SNE projections were enhanced with a heatmap that visualizes the local failure rate for each designated target for *Test Actions* and both evaluation and grounding for *Expected Results*. This failure rate was calculated per grid cell based on evaluation results. Red indicates high failure density, and white indicates low failure density. This approach facilitates the visualization of how VLM performance, for both the base Molmo-7B and ELAM-7B models, varies across the semantic landscape defined by the text embeddings.

**nomic-embed-text-v1.5* is an improved variant of the *nomic-embed-text-v1*[26] model. It utilizes Matryoshka Representation Learning for flexible dimensionality reduction with minimal loss while supporting up to 8,192 tokens. The model is optimized for search, clustering, and classification in production.

5.5.1 Test Action Utterances: Grounding Performance

Baseline Molmo-7B-D-0924: Clusters with LLM-suggested labels indicating semantic overlap (e.g., ‘Navigation Control’ across multiple clusters) are observed in the t-SNE plot (Figure 5) of *Test Action* embeddings for the base model. The grounding failure heatmap indicates that errors are distributed across the embedding space, without perfectly aligning with cluster boundaries. Notable high-failure regions overlap with clusters tentatively labeled *Climate Control, Audio and Display Settings* and *User Interface Control, Device Settings*. A distinct cluster associated with *Navigation Control, User Interface Control* resides in a predominantly low-failure region, which suggests that these utterance types are generally grounded successfully.

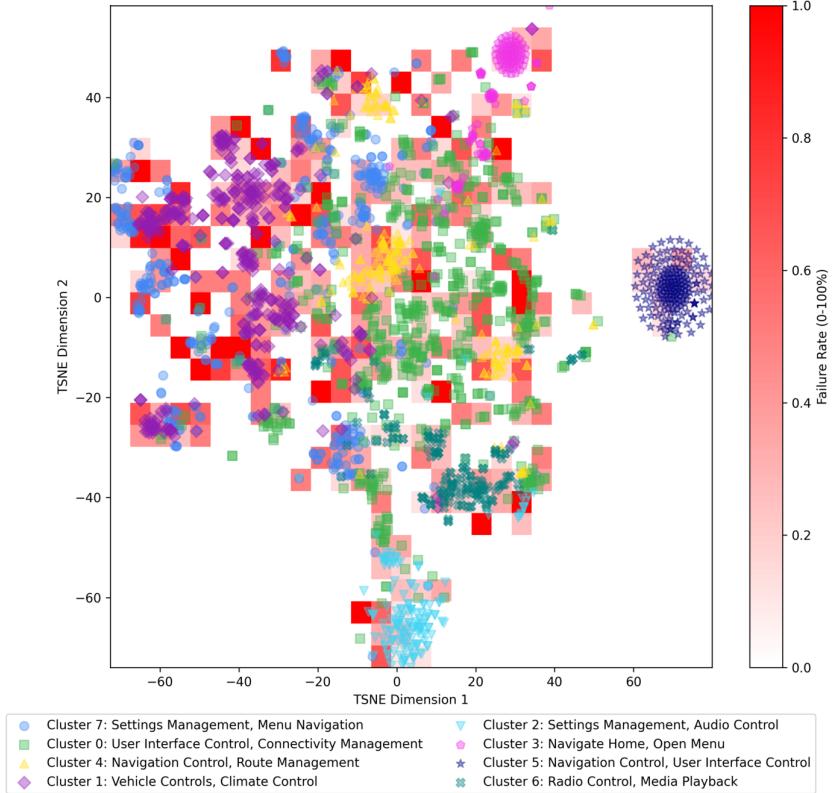


Figure 5: t-SNE plot of the base Molmo-7B model for the *Test Action* task

ELAM-7B: Utilizing the same text embeddings, the fine-tuned model’s performance is visualized through its distinct failure heatmap (Figure 6). When comparing Figure 6 to Figure 5, a noticeable reduction in both the extent and intensity of high-failure regions across the embedding space can be observed, supporting the results summarized in Table 2. Several clusters show significant improvement. For instance, the most distinct *Cluster 3: Navigation Control, User Interface Control* exhibited a failure rate close to zero. The dispersed *Cluster 1: User Interface Control, Communication Management*, *Cluster 4: Climate Control, Vehicle Settings* and *Cluster 5: Climate Control, Vehicle Settings* now predominantly reside in low-failure regions after fine-tuning, indicating improved reliability in handling these automotive domain-specific utterance types. However, some high-failure samples still persist at the boundaries of these clusters, suggesting that certain semantic regions or utterance formulations remain challenging even after fine-tuning.

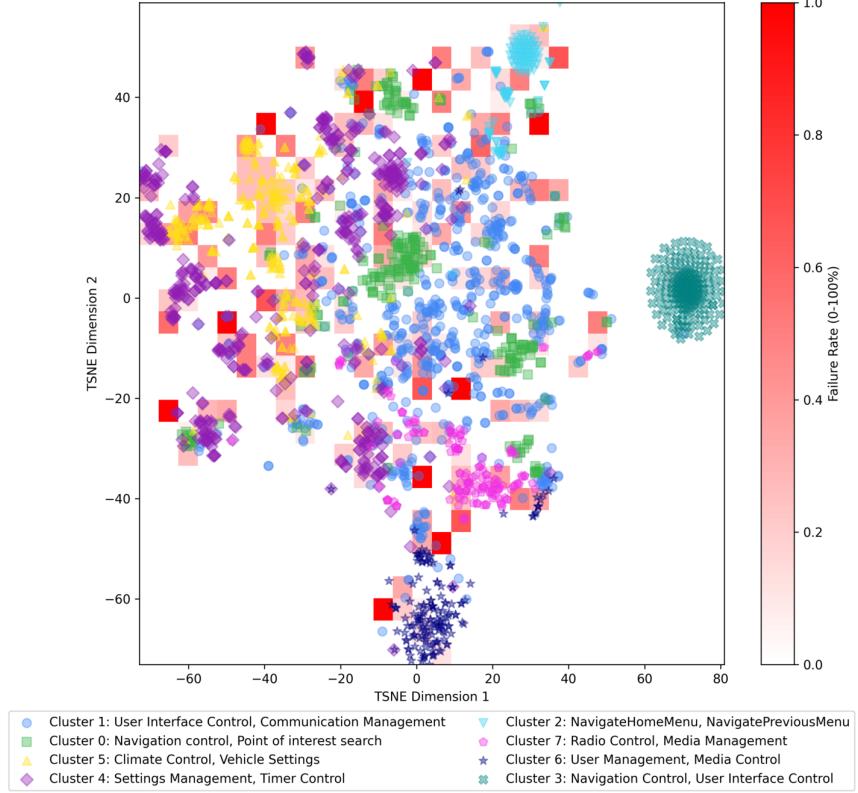


Figure 6: t-SNE plot of the ELAM-7B model for the *Test Action* task

5.5.2 Expected Result Utterances: Evaluation and Grounding Performance

Baseline Molmo-7B-D-0924: The *Expected Result* utterances, representing a more complex task involving visual grounding and subsequent evaluation, are also observed to form clusters in the embedding space with themes such as “Settings”, “Navigation”, and “Display”. The Molmo-7B failure heatmap (Figure 7) for this task exhibits widespread and often intense high-failure regions, particularly concentrated in areas associated with *User Interface Settings*, *Search Functionality* and *Audio Control, Radio Management*.

ELAM-7B: The failure heatmap (Figure 8) for the fine-tuned model on the *Expected Result* tasks indicates that high-failure regions persist across many parts of the embedding space. While a visual comparison with the base Molmo model may suggest a subtle reduction in the overall failure density, the improvement appears less pronounced than that observed for the *Test Action* task.

Fine-tuning yielded only modest improvements for the more complex *Expected Result* evaluation and grounding task. The widespread persistence of high-failure regions indicates that accurately interpreting and verifying diverse state descriptions based on visual evidence remains a significant challenge.

5.6 Visual Error Analysis

5.6.1 Visual Error Analysis: Test Actions

A detailed analysis of the ELAM’s failures in the AutomotiveUI-Bench-4K evaluation was conducted to identify the areas where the model continues to demonstrate deficiencies. To begin with, the *Test Actions* were examined. A comprehensive manual review and classification of all error cases was conducted. Examples and explanations for all identified categories can be found in the appendix in Appendix A.4. Table 6 summarizes the distribution of error categories for *Test Actions*.

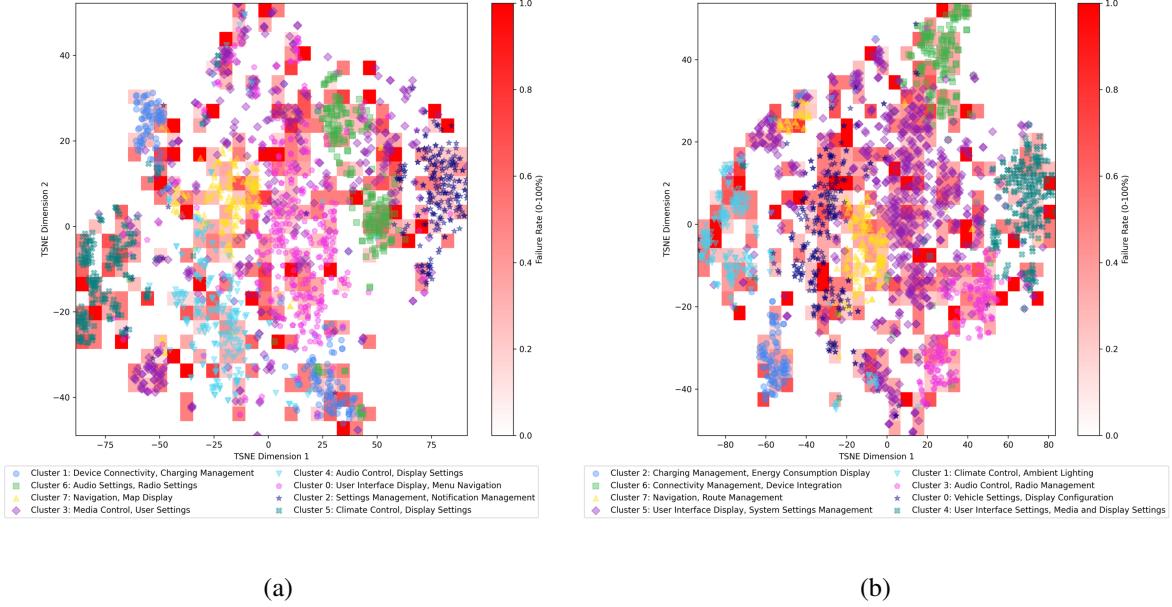


Figure 7: t-SNE visualizations of the base Molmo-7B model for two *Expected Result* evaluation tasks: (a) *Visual Grounding* and (b) *Evaluation*.

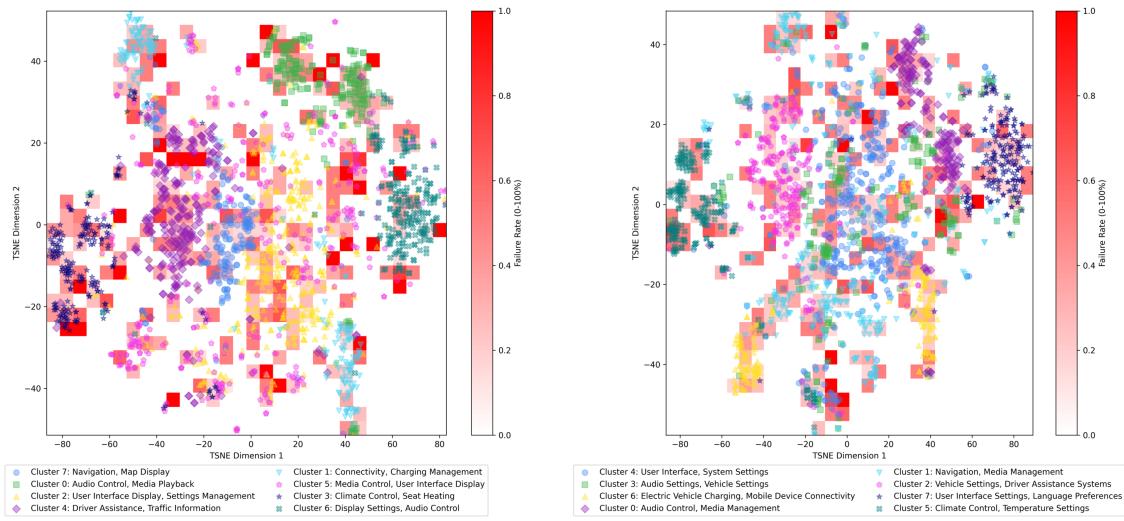


Figure 8: t-SNE visualizations of the ELAM-7B model for two *Expected Result* evaluation tasks: (a) *Visual Grounding* and (b) *Evaluation*.

The following categories were identified:

1. The bounding boxes utilized for the purpose of evaluating the result were of insufficient size.
2. The element to be determined was not adequately described.
3. The subject of interest was a specialized automotive icon.
4. The provided description was adequate; however, the presence of two similar elements resulted in confusion.
5. The model proved incapable of establishing a connection between two elements.
6. The model demonstrated confusion between driver side and passenger side, left and right, or increase and decrease.
7. The model was incapable of counting.
8. Not assignable to any category.

Table 6: Analysis of error categories based on *Test Actions* in AutomotiveUI-Bench-4K.

Category #	Category Description	Error Share in %
1.	Box too small	17.0
2.	Ambiguous or poor description of the test action	23.7
3.	Special icons from the vehicle domain	19.7
4.	Clear description, but very similar elements confused the model	11.5
5.	Connection between elements could not be established	6.8
6.	Driver/passenger, left/right, up/down distinctions	4.8
7.	Elements must be counted	3.1
8.	Miscellaneous	13.4

The findings underscore the importance of ensuring clear and precise phrasing in the test utterances used by developers and testers, as this practice can mitigate a range of potential issues. Analysis also showed that the system’s recognition of vehicle-specific icons, in particular, remains suboptimal, suggesting that performance would benefit from additional training data. A more fundamental challenge identified is the system’s difficulty in differentiating between highly similar elements and establishing connections between them. The creation of specialized training datasets to address these specific scenarios represents a promising avenue for future research.

5.6.2 Visual Error Analysis: Expected Results

An analysis of the ELAM’s failures of grounding and evaluating *Expected Results* in the AutomotiveUI-Bench-4K evaluation was conducted in the same manner as the examination of the *Test Actions*. The errors were manually reviewed and categorized. The identified categories were similar to those found in the examination of the *Test Actions*, though some differences were noted. Table 7 summarizes the distribution of error categories for *Expected Results*. Examples and explanations for all identified categories can be found in the appendix in Appendix A.5.

The following categories were identified:

1. The bounding boxes utilized for the purpose of evaluating the result were of insufficient size.
2. Multiple areas display the expected result.
3. The model is asked to check if options of a certain menu are visible. However the box was not drawn around the menu option, but around the menu heading instead. This could be considered as a special case of category 1 or 2.
4. The subject of interest was a specialized automotive icon.
5. The provided description was adequate, however the presence of two similar elements resulted in confusion.
6. Not assignable to any category.

Table 7: Analysis of error categories based on *Expected Results* in AutomotiveUI-Bench-4K.

Category #	Category Description	Error Share in %
1.	Box too small	26.4
2.	Multiple areas display the expected result	15.2
3.	Box not around option, when asked if options are shown	15.2
4.	Special icons from the vehicle domain	8.8
5.	Clear description, but very similar elements confused the model	4.8
6.	Miscellaneous	29.6

6 Conclusion

A clear improvement in localization and evaluation performance is demonstrated using a LoRa fine-tuned model, trained with a synthetic data generation pipeline. ELAM outperforms its baseline on both AutomotiveUI-Bench-4K and ScreenSpot, highlighting this approach as an effective strategy for enhancing visual grounding capabilities with limited resources. The model demonstrates a notable improvement in localization accuracy on the AutomotiveUI-Bench-4K dataset, achieving a gain of 16.3% for *Test Actions* (TA_{vg}), 6.1% for *Expected Results* (ER_{vg}), and 11.3% evaluation conclusion accuracy (ER_{evl}) compared to its baseline *Molmo-7B-D-0924*. These improvements maintain generalizability across diverse UI domains, including desktop, mobile, and web, as evidenced by an 80.8% (+5.6%) average accuracy on the ScreenSpot dataset.

The t-SNE visualization of a fixed text embedding space overlaid with task-specific VLM failure rates provided valuable insights by decoupling utterance semantics from model performance. The analysis indicated tangible benefits of fine-tuning for more direct tasks such as the *Test Action* grounding, and suggested that more complex tasks like the *Expected Result* evaluation continue to present significant challenges. Additionally, the t-SNE plots revealed variations in performance across different sub-domains. The model demonstrated strong performance in grounding UI elements commonly found in desktop, mobile, and web environments but exhibited reduced effectiveness in automotive-specific functionalities such as Advanced Driver-Assistance Systems (ADAS) or climate control, for instance, distinguishing between adjusting the driver’s and passenger’s temperatures. This emphasizes the general need for further domain-specific data and training strategies.

While the findings are encouraging, it is important to acknowledge certain limitations of this work. Firstly, while the synthetic data pipeline proved effective, the inherent gap between synthetic and real-world data introduces a potential limitation. This restricts the applicability of the findings to entirely unconstrained scenarios. Further investigation is needed to assess performance on a wider range of real-world datasets beyond AutomotiveUI-Bench-4K, and across more diverse visual grounding tasks. Errors were identified during a preliminary analysis of the training data, including the misclassification of fundamental UI element states (e.g., toggle switches). Additionally, the distribution of use cases exhibited an excessive emphasis on the presence and visibility of control elements (e.g., “*Element XYZ is visible.*”). This indicates that the number of annotations regarding the status of control elements is insufficient.

As a consequence, exclusive reliance on ELAM-7B to automate UI verification is cautioned against. For automotive functions that directly impact passenger safety, a VLM-based solution introduces significant safety and ethical concerns, making it an unsuitable final authority for correctness. Given these limitations, the most responsible approach is to position VLM-based verification not as a replacement for, but as a supplement to existing verification pipelines. VLMs can be highly effective in automating the testing of non-critical infotainment features, visual consistency checks, or identifying minor UI bugs. For any functionality related to passenger safety, however, a human-in-the-loop protocol is essential. The VLM could be used to flag potential issues for human review, but the final, authoritative sign-off on safety-critical UI functionality must always be performed by a human expert using a validated method. This hybrid approach leverages the efficiency of machine learning while upholding the

paramount ethical obligation to ensure human safety.

The scope of the evaluation, while demonstrating improvement, primarily focused on quantitative metrics. A deeper qualitative analysis of the model’s performance, especially on automotive-specific parts of automotive systems, would provide richer insights into the nuanced strengths and weaknesses of the approach.

Looking ahead, there are several promising avenues for potential improvements. Future work could focus on developing more advanced and general evaluation methodologies. These methodologies would integrate linguistic and grammatical analysis to better understand the impact of subtle syntactic and semantic nuances on model performance. This may include incorporating techniques like dependency parsing and syntactic analysis to provide a more comprehensive picture of the interplay between language and visual grounding.

Moreover, there is significant potential for enhancing the capabilities of the visual encoders within VLM architectures. Although current pre-trained encoders are effective, they may overlook certain domain-specific features and high-resolution details. Future improvements might explore the use of multi-encoder strategies, adaptive encoder selection, and more data-efficient fine-tuning techniques to overcome these limitations and achieve more robust and generalized performance across diverse real-world scenarios.

Overall, the findings point toward a holistic strategy for advancing VLM technologies. This strategy refines visual grounding through innovative training pipelines and embraces broader evaluation frameworks and encoder enhancements. These integrated improvements hold the promise of pushing the boundaries of current models toward greater real-world applicability.

Acknowledgement

This work was partially supported by German BMBF within the research project MANNHEIM-KI4BoardNet. (<https://www.elektronikforschung.de/projekte/mannheim-ki4boardnet>).

References

- [1] ALAYRAC, J.-B., DONAHUE, J., LUC, P., MIECH, A., BARR, I., HASSON, Y., LENC, K., MENSCH, A., MILLICAN, K., REYNOLDS, M., ET AL. Flamingo: a visual language model for few-shot learning. *Advances in neural information processing systems* 35 (2022), 23716–23736.
- [2] CHAI, Y., HUANG, S., NIU, Y., XIAO, H., LIU, L., ZHANG, D., GAO, P., REN, S., AND LI, H. Amex: Android multi-annotation expo dataset for mobile gui agents. *arXiv preprint arXiv:2407.17490* (2024).
- [3] CHEN, Z., WANG, W., CAO, Y., LIU, Y., GAO, Z., CUI, E., ZHU, J., YE, S., TIAN, H., LIU, Z., GU, L., WANG, X., LI, Q., REN, Y., CHEN, Z., LUO, J., WANG, J., JIANG, T., WANG, B., HE, C., SHI, B., ZHANG, X., LV, H., WANG, Y., SHAO, W., CHU, P., TU, Z., HE, T., WU, Z., DENG, H., GE, J., CHEN, K., ZHANG, K., WANG, L., DOU, M., LU, L., ZHU, X., LU, T., LIN, D., QIAO, Y., DAI, J., AND WANG, W. Expanding performance boundaries of open-source multimodal models with model, data, and test-time scaling, 2025.
- [4] CHENG, K., SUN, Q., CHU, Y., XU, F., LI, Y., ZHANG, J., AND WU, Z. Seeclick: Harnessing gui grounding for advanced visual gui agents. *arXiv preprint arXiv:2401.10935* (2024).
- [5] DEITKE, M., CLARK, C., LEE, S., TRIPATHI, R., YANG, Y., PARK, J. S., SALEHI, M., MUEN-NIGHOFF, N., LO, K., SOLDAINI, L., ET AL. Molmo and pixmo: Open weights and open data for state-of-the-art multimodal models. *arXiv preprint arXiv:2409.17146* (2024).
- [6] DWYER, B. Website screenshots dataset. <https://universe.roboflow.com/roboflow-gw7yv/website-screenshots>, aug 2022. visited on 2024-07-02.

- [7] FU, J., ZHANG, X., WANG, Y., ZENG, W., AND ZHENG, N. Understanding mobile gui: From pixel-words to screen-sentences. *Neurocomputing 601* (2024), 128200.
- [8] GAO, L., ZHANG, L., WANG, S., WANG, S., LI, Y., AND XU, M. Mobileviews: A large-scale mobile gui dataset. *arXiv preprint arXiv:2409.14337* (2024).
- [9] GHOSH, A., ACHARYA, A., SAHA, S., JAIN, V., AND CHADHA, A. Exploring the frontier of vision-language models: A survey of current methodologies and future directions. *arXiv preprint arXiv:2404.07214* (2024).
- [10] GOU, B., WANG, R., ZHENG, B., XIE, Y., CHANG, C., SHU, Y., SUN, H., AND SU, Y. Navigating the digital world as humans do: Universal visual grounding for GUI agents.
- [11] GOU, B., WANG, R., ZHENG, B., XIE, Y., CHANG, C., SHU, Y., SUN, H., AND SU, Y. UGround, Feb. 2025.
- [12] HU, E. J., SHEN, Y., WALLIS, P., ALLEN-ZHU, Z., LI, Y., WANG, S., WANG, L., CHEN, W., ET AL. Lora: Low-rank adaptation of large language models. *ICLR 1, 2* (2022), 3.
- [13] JOCHER, G., QIU, J., AND CHAURASIA, A. Ultralytics YOLO, Jan. 2023.
- [14] KHAN, M. E., AND KHAN, F. A comparative study of white box, black box and grey box testing techniques. *International Journal of Advanced Computer Science and Applications 3, 6* (2012).
- [15] KING, P., AND COPP, D. Hardware in the loop for automotive vehicle control systems development. In *UKACC Control 2004 Mini Symposia* (2004), IET, pp. 75–78.
- [16] KUMAR, R., AHUJA, N. J., SAXENA, M., AND KUMAR, A. Automotive power window communication with dtc algorithm and hardware-in-the loop testing. *Wireless Personal Communications 114, 4* (2020), 3351–3366.
- [17] LI, J., LI, D., SAVARESE, S., AND HOI, S. Blip-2: Bootstrapping language-image pre-training with frozen image encoders and large language models. In *International conference on machine learning* (2023), PMLR, pp. 19730–19742.
- [18] LI, W., BISHOP, W. E., LI, A., RAWLES, C., CAMPBELL-AJALA, F., TYAMAGUNDLU, D., AND RIVA, O. On the effects of data scale on ui control agents. *Advances in Neural Information Processing Systems 37* (2024), 92130–92154.
- [19] LI, Z., WU, X., DU, H., NGHIEM, H., AND SHI, G. Benchmark evaluations, applications, and challenges of large vision language models: A survey. *arXiv preprint arXiv:2501.02189* (2025). Submitted on 4 Jan 2025, last revised 29 Jan 2025.
- [20] LIN, K. Q., LI, L., GAO, D., YANG, Z., WU, S., BAI, Z., LEI, W., WANG, L., AND SHOU, M. Z. Showui: One vision-language-action model for gui visual agent, 2024.
- [21] LIU, H., LI, C., WU, Q., AND LEE, Y. J. Visual instruction tuning, 2023.
- [22] LU, Y., YANG, J., SHEN, Y., AND AWADALLAH, A. Omniparser for pure vision based gui agent, 2024.
- [23] MEIXNER, G., AND MUELLER, C. *Automotive User Interfaces: Creating Interactive Experiences in the Car*. Springer, 2017.
- [24] MEZGER, S., AND DENG, M. Functional gui testing of in-vehicle infotainment systems in virtual and real environments. *Hanser Automotive*, 5-6 (2017). Translation of a German publication in Hanser Automotive, issue 5-6/2017.

- [25] NGUYEN, D., CHEN, J., WANG, Y., WU, G., PARK, N., HU, Z., LYU, H., WU, J., APONTE, R., XIA, Y., ET AL. Gui agents: A survey. *arXiv preprint arXiv:2412.13501* (2024).
- [26] NUSSBAUM, Z., MORRIS, J. X., DUERSTADT, B., AND MULYAR, A. Nomic embed: Training a reproducible long context text embedder. *arXiv preprint arXiv:2402.01613* (2024).
- [27] PAWLOWSKI, P., ZAWISTOWSKI, K., LAPACZ, W., SKORUPA, M., WIACEK, A., POSTANSQUE, S., AND HOSCILOWICZ, J. Tinyclick: Single-turn agent for empowering gui automation. *arXiv preprint arXiv:2410.11871* (2024).
- [28] RADFORD, A., KIM, J. W., HALLACY, C., RAMESH, A., GOH, G., AGARWAL, S., SASTRY, G., ASKELL, A., MISHKIN, P., CLARK, J., ET AL. Learning transferable visual models from natural language supervision. In *International conference on machine learning* (2021), PMLR, pp. 8748–8763.
- [29] RAWLES, C., LI, A., RODRIGUEZ, D., RIVA, O., AND LILLICRAP, T. Androidinthewild: A large-scale dataset for android device control. *Advances in Neural Information Processing Systems 36* (2023), 59708–59728.
- [30] SMITH, R. An overview of the tesseract ocr engine. In *Ninth international conference on document analysis and recognition (ICDAR 2007)* (2007), vol. 2, IEEE, pp. 629–633.
- [31] VAN DER MAATEN, L., AND HINTON, G. Visualizing data using t-sne. *Journal of Machine Learning Research 9*, 86 (2008), 2579–2605.
- [32] VECTOR. Canoe — hil & sil tools across multiple industries — simulation, 2025. Accessed: 2025-03-10.
- [33] WANG, L., YANG, F., ZHANG, C., LU, J., QIAN, J., HE, S., ZHAO, P., QIAO, B., HUANG, R., QIN, S., ET AL. Large action models: From inception to implementation. *arXiv preprint arXiv:2412.10047* (2024).
- [34] WANG, P., BAI, S., TAN, S., WANG, S., FAN, Z., BAI, J., CHEN, K., LIU, X., WANG, J., GE, W., ET AL. Qwen2-vl: Enhancing vision-language model’s perception of the world at any resolution. *arXiv preprint arXiv:2409.12191* (2024).
- [35] WANG, S., LIU, W., CHEN, J., ZHOU, Y., GAN, W., ZENG, X., CHE, Y., YU, S., HAO, X., SHAO, K., ET AL. Gui agents with foundation models: A comprehensive survey. *arXiv preprint arXiv:2411.04890* (2024).
- [36] WU, J., WANG, S., SHEN, S., PENG, Y.-H., NICHOLS, J., AND BIGHAM, J. Webui: A dataset for enhancing visual ui understanding with web semantics. *ACM Conference on Human Factors in Computing Systems (CHI)* (2023).
- [37] WU, Z., WU, Z., XU, F., WANG, Y., SUN, Q., JIA, C., CHENG, K., DING, Z., CHEN, L., LIANG, P. P., ET AL. Os-atlas: A foundation action model for generalist gui agents. *arXiv preprint arXiv:2410.23218* (2024).
- [38] XIAO, B., WU, H., XU, W., DAI, X., HU, H., LU, Y., ZENG, M., LIU, C., AND YUAN, L. Florence-2: Advancing a unified representation for a variety of vision tasks. *arXiv preprint arXiv:2311.06242* (2023).
- [39] YANG, J., ZHANG, H., LI, F., ZOU, X., LI, C., AND GAO, J. Set-of-mark prompting unleashes extraordinary visual grounding in gpt-4v. *arXiv preprint arXiv:2310.11441* (2023).
- [40] YIN, N. Automated testing for automotive infotainment systems. Master’s thesis, Chalmers University of Technology, 2018.

- [41] ZHANG, C., HE, S., QIAN, J., LI, B., LI, L., QIN, S., KANG, Y., MA, M., LIU, G., LIN, Q., ET AL. Large language model-brained gui agents: A survey. *arXiv preprint arXiv:2411.18279* (2024).
- [42] ZHAO, Y., HUANG, J., HU, J., WANG, X., MAO, Y., ZHANG, D., JIANG, Z., WU, Z., AI, B., WANG, A., ZHOU, W., AND CHEN, Y. Swift:a scalable lightweight infrastructure for fine-tuning, 2024.
- [43] ZHENG, L., HUANG, Z., XUE, Z., WANG, X., AN, B., AND YAN, S. Agentstudio: A toolkit for building general virtual agents. *arXiv preprint arXiv:2403.17918* (2024).

A Appendix

A.1 Code and Dataset Availability

For reproducibility and broader utility, our resources are openly available. [ELAM](#) and its supporting code, alongside the [AutomotiveUI-Bench-4K](#) dataset, are hosted on Hugging Face. The ELAM repository provides the code and prompting examples necessary to replicate the results detailed in this paper. Consistent with its fine-tuned base, Molmo, ELAM and its code are released under the Apache License 2.0. The AutomotiveUI-Bench-4K dataset is distributed under the CC-BY-4.0 license.

A.2 Prompt Templates for ELAM

Prompt Template - Test Action

```
prompt_template_test_action = f"""\
Identify and point to the UI element that corresponds to this test action:
{test_action}.
"""
```

Response Template - Test Action

```
response_template_test_action = f"""\
{reasoning}
<point x="{center_x:.1f}" y="{center_y:.1f}" alt="{test_action}">
{test_action}</point>
"""
```

Prompt Template - Expected Result

```
prompt_template_expected_result = f"""\
Evaluate this statement about the image:
'{expectation}'
Think step by step, conclude whether the evaluation is 'PASSED' or 'FAILED'
    \
and point to the UI element that corresponds to this evaluation.
"""
```

Response Template - Expected Result

```
response_template_expected_result = f"""\
{reasoning}
Conclusion: {evaluation_result}
<point x="{center_x:.1f}" y="{center_y:.1f}" alt="{expectation}">
{expectation} {evaluation_result}</point>
"""
```

A.3 Prompts in Synthetic Data Pipeline

A.3.1 Test Action Template

Write test actions for the UI of automotive infotainment software. These actions must be declarative, concise, and tailored to verify the software's functionality accurately. Test actions should cover marked user interface elements and interactions relevant to the infotainment system.

Steps

1. Understand UI Element:
 - Identify the position of the UI element marked by a <color> <marker_type>.
 - Identify the semantic meaning based on its text or icon.
 - Identify any parent elements crucial for semantic or functional meaning.
 - Identify any related elements crucial for semantic or functional meaning (e.g., text corresponding to a checkbox or switch).
2. Reasoning if UI Element is interactive:
 - Determine why the element is interactive or not. Grayed out elements could either mean that they are just disabled or not interactive at all.
 - If it is not interactive set the utterance to "none".
3. Specify Test Action Utterance:
 - Write a deterministic and declarative action simulating a user interaction with the UI element.
 - Use unique identifiers for the UI element, and mention a parent element if necessary.
 - Do not use the <color> <marker_type> in the test action utterance.
 - Use verbs like tap, click, enter, open, enable, disable, activate, deactivate, select, press, collapse, navigate, cancel, refresh, ...
 - Try to choose the verb that is most applicable for the type of UI Element (e.g., enable for switch, choose for radio buttons, open for submenu etc.)

Required Output Structure

'''

REASONING:

1. [First step in thinking process]
2. [Second step in thinking process]

[Continue with numbered steps as needed]

UTTERANCE:

[Describing the test action utterance as a single sentence without using the <color> <marker_type>]

'''

A.3.2 Expected Result Template “Passed”

As a test engineer for automotive infotainment systems, formulate result evaluations based on the current screen and determine if the test has passed or failed.

Steps

0. Understand the UI Element:
 - Analyze the current context and active menu of the infotainment system.
 - Identify the position of the UI element marked by a <color> <marker-type>.
 - Determine the semantic meaning based on its text or icon.
 - Identify any parent elements crucial for semantic or functional meaning.
 - Identify any related elements crucial for semantic or functional meaning (e.g., text corresponding to a checkbox or switch).
1. Performed Test Action:
 - Think of a possible test action which was done and can be evaluated with the element marked by the <color><marker-type>.
2. Reasoning:
 - Conduct reasoning for reaching the result evaluation by examining UI semantics with step-by-step thinking.
3. Determine Evaluation/Expected Result:
 - Provide a short and general description of the expected result of the test action that led to the current screen or highlighted UI element.
 - The expected result must be based solely on the current screen, not on previous or next screens.
 - Include presence, color, positional, semantic, state, and visual information as needed. Do not include the <color><marker-type>.
 - The expected result can also just check the presence of the marked UI element
4. Incorporate Evaluation:
 - Assess why the expected result is met or not. Conclude with “FAILED” or “PASSED.”

Critical Rules

1. Never reference <color> <marker-type> in test action or expected result
2. Evaluate only current screen state
3. No assumptions about previous/future states
4. Use objective, verifiable statements
5. Document all reasoning steps
6. Provide clear pass/fail criteria
7. Valid evaluations include checking the presence, visibility, position, and properties of the UI element.

Required Output Structure

...

TEST ACTION:

[Single sentence describing the specific test action performed]

REASONING:

1. [First step in thinking process]
 2. [Second step in thinking process]
- [Continue with numbered steps as needed]

EXPECTED RESULT:

[Describing the evaluated result as a single sentence without using the <color><marker-type>]

CONCLUSION:

[PASSED/FAILED]

...

A.3.3 Expected Result Template “Failed”

As a test engineer for automotive infotainment systems, formulate result evaluations based on the current screen and determine if the test has passed or failed.

Steps

1. Understand the UI Element:
 - Analyze the current context and active menu of the infotainment system.
 - Identify the position of the UI element marked by a <color> <marker-type>.
 - Determine the semantic meaning based on its text or icon.
 - Identify any parent elements crucial for semantic or functional meaning.
 - Identify any related elements crucial for semantic or functional meaning (e.g., text corresponding to a checkbox or switch).
2. Determine Evaluation/Expected Result that is wrong for the current screen:
 - Provide a short and general description of the failed expected result of the test action that led to the current screen or highlighted UI element.
 - You should think of an expectation that is wrong or not in the screen.
 - The expected result must be based solely on the current screen, not on previous or next screens.
 - Include absence, different color, wrong positional, semantic, and wrong state information as needed. Do not include the <color> <marker-type>.
 - The expected result can also just check the absence of the marked UI element or if the screen shows the wrong context menu.
3. Reasoning:
 - Conduct reasoning for reaching the result evaluation by examining UI semantics with step-by-step thinking.
4. Incorporate Evaluation:
 - Assess why the expected result is met or not. Conclude with "FAILED" or "PASSED."

Critical Rules

1. Never reference <color> <marker-type> in test action or expected result
2. The Expectation must be generated for elements within the <color> <marker-type>
3. No assumptions about previous/future states
4. Use objective, verifiable statements
5. Document all reasoning steps
6. Provide clear pass/fail criteria
7. Valid evaluations include checking the presence, visibility, position, and properties of the UI element

Required Output Structure ...

REASONING:

1. [First step in thinking process]
 2. [Second step in thinking process]
- [Continue with numbered steps as needed]

EXPECTED RESULT:

[Describing the evaluated result as a single sentence without using the <color> <marker-type>]

CONCLUSION:

[PASSED/FAILED]

...

A.4 Examples of Errors for ELAM and AutomotiveUI-Bench-4K: Test Actions

1. Box too small: “select kWh/100mi as electric consumption unit”

- Figure 9 shows the electric consumption menu of a BMW iX2 in English, featuring a dark background with light text. This menu allows you to change the electric consumption unit. Currently, the unit is set to kWh/100 km.
- ELAM is asked to change the unit to kWh/100 mi.
- The red box indicates the expected tap area defined in the **AutomotiveUI-Bench-4K** dataset. The red dot marks the point at which ELAM would tap to perform the test action. Since the red dot is outside the box, the test is counted as failed.
- Either the defined area is too small. It should extend to the full length and height of the line containing the radio button for kWh/100 mi, as tapping anywhere on that line activates the button. Therefore, the model did not actually make a mistake. Alternatively, the test action utterance could also be specify to explicitly tap the text or the radio button.

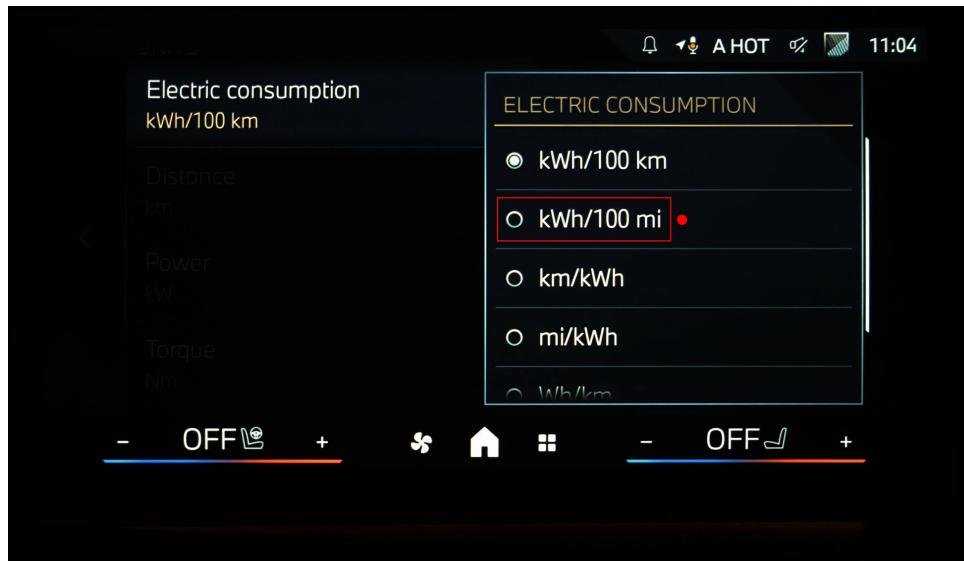


Figure 9: Box too small: “select kWh/100mi as electric consumption unit”

2. Ambiguous or poor description of the test action: “*Select the burger button*”

- Figure 10 shows the navigation screen of a Toyota Yaris in German, featuring a dark background with blue buttons and text. A small popup with the text “*Guten Morgen...*” (good morning) is visible in the lower left corner.
- ELAM’s task is to select the burger button.
- The red box indicating the expected click area marks a button with a hamburger icon in the popup.
- The issue lies in the presence of two additional “*burger buttons*” that are currently visible on the screen. There is a button with a burger icon and the text “*Lebensmittel*” (groceries) at the bottom of the screen and there is a button featuring an icon showing three horizontal like, which is generally known as “*burger button*”, at the upper right corner of the screen.

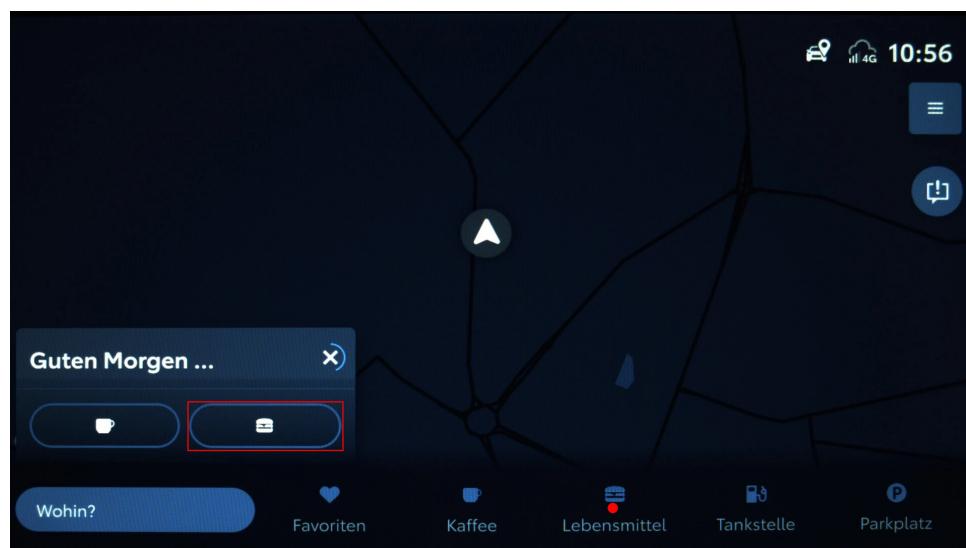


Figure 10: Ambiguous or poor description of the test action: “*Select the burger button*”

3. Special icons from the vehicle domain: “*turn steering wheel heating on*”

- Figure 11 shows the navigation screen of a Ford Mustang Mach-E with a white background. The area of interest in this test is the lower quarter of the screen featuring buttons related to climate control.
- ELAM is asked to turn on the steering wheel heating.
- The steering wheel heating icon is marked with a red box in the lower left corner of the screen. The small gray line below the icon indicates that the heating is currently turned off.
- ELAM tapped the multi-zone climate control icon instead, which features a windshield heating symbol. The steering wheel heating icon is a car-specific icon, ELAM would benefit from more training data with car-specific icons.

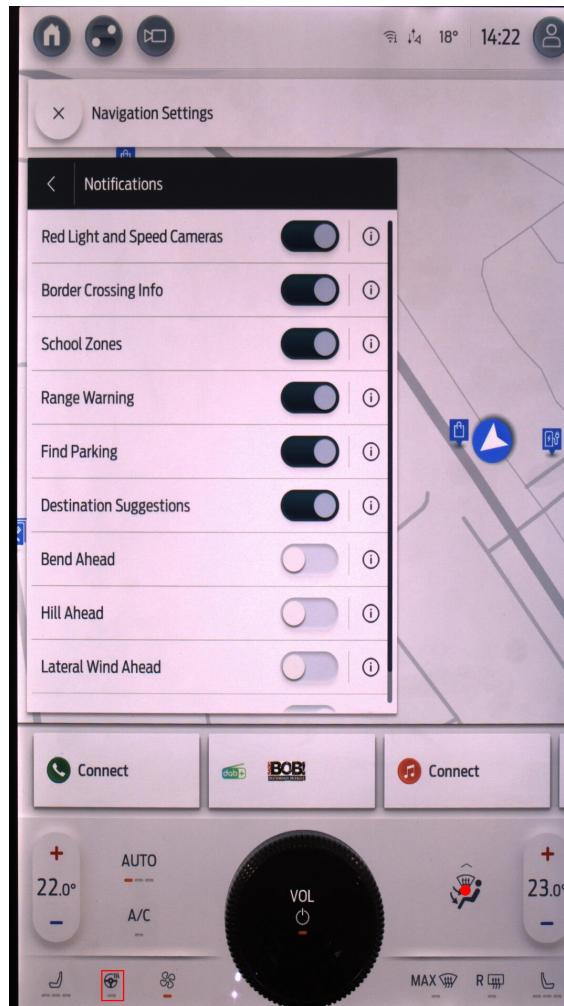


Figure 11: Special icons from the vehicle domain: “*turn steering wheel heating on*”

4. Clear description, but very similar elements confused the model: “*deactivate the reminder signal for mobile phone*”

- Figure 12 shows the settings menu of an Audi e-tron GT, featuring a black background with white text.
- The test action is to deactivate the reminder signal for mobile phone.
- The red box surrounds the “*Off*” button below the text “*Reminder signal for mobile phone*”. The value is currently set to “*Spoken*”, as indicated by the white line at the bottom of the button.
- However, ELAM tapped the toggle button next to “*Mobile phone notifications: Reminder/charge level*”. The test action clearly states that “*Reminder signal for mobile phone*” should be deactivated and not “*Mobile phone notifications: Reminder/charge level*”. The issue lies in the text’s close semantic resemblance, which led to confusion in the model.

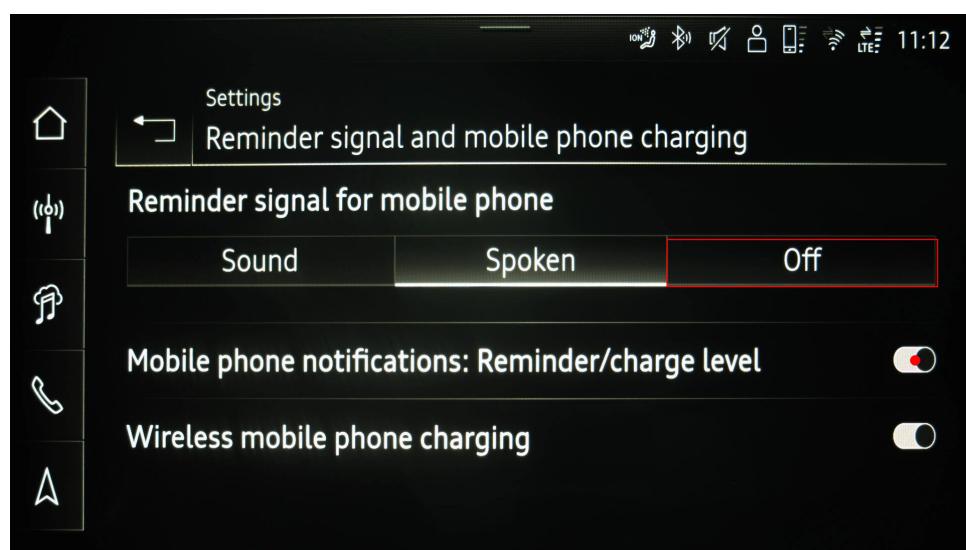


Figure 12: Clear description, but very similar elements confused the model: “*deactivate the reminder signal for mobile phone*”

5. Connection between elements could not be established: “*add phone to Favourites*”

- Figure 13 shows the apps menu of a Maserati Grecale with a black background and white icons and text. Eight apps are currently visible. Each app features a star icon to add it to favourites.
- The task is to add phone to the favourites.
- The star icon next to the phone button is marked by the red box, which indicates that it is the expected area to click.
- The red dot indicates that ELAM intends to tap directly on the phone button instead of the star icon which would add phone to favourites. ELAM was not capable of making a connection between the phone button and the star icon.

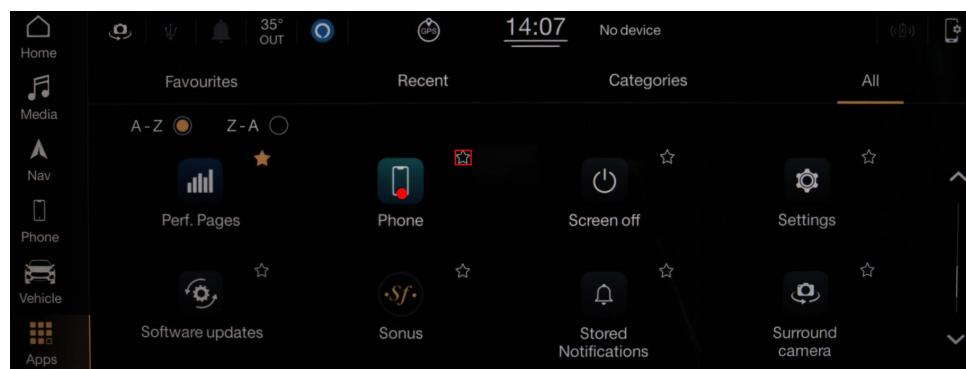


Figure 13: Connection between elements could not be established: “*add phone to Favourites*”

6. Driver/passenger, left/right, up/down distinctions: “*increase the right temperature setting with the plus button*”

- Figure 14 shows the climate menu of an Opel Astra in German. The background color is a very dark red and text and icons are white. Some icons/texts are highlighted in orange.
- ELAM is asked to increase the right temperature setting with the plus button.
- The expected click area is marked with a red box on the right side of the screen (plus button).
- ELAM selected the left plus button instead of the right plus button. The underlying cause of the issue can be traced back to the training data. In instances where sufficient data is not available for left versus right comparisons, the model is unable to differentiate between these two options.



Figure 14: Driver/passenger, left/right, up/down distinctions: “*increase the right temperature setting with the plus button*”

7. Elements must be counted: “Activate the first weekly item from the charging list”

- Figure 15 shows the charging menu of a Mini Cooper in German. The round screen is unusual for a car’s infotainment system. The menu allows the user to set several timers for charging.
- The task is to activate the first weekly timer.
- The toggle button of the first weekly item in the list is marked by a red box, indicating the expected tap area.
- ELAM selected the second weekly item. This suggests that there was insufficient training data containing counting examples to teach the model how to count.

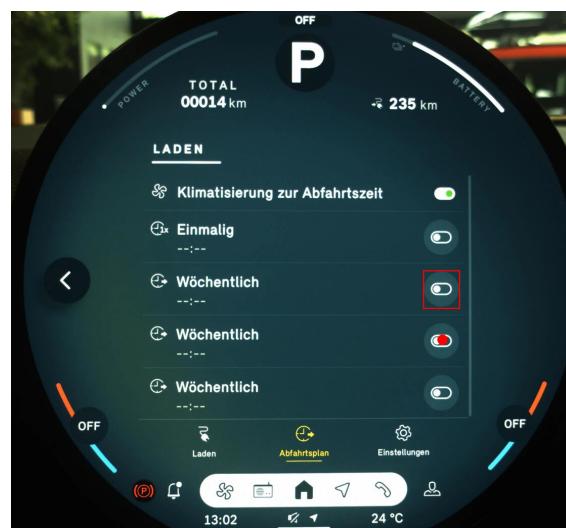


Figure 15: Elements must be counted: “Activate the first weekly item from the charging list”

8. Miscellaneous: “*Go to main menu*”

- Figure 16 shows the camera settings menu of a Kia. This infotainment system has a light background with dark text and bluish highlights.
- ELAM is asked to go to the main menu.
- The red box, situated around the house icon that is visible at the top of the screen, serves as an indicator of the designated tap area.
- ELAM taps on the context menu burger button instead of the house icon, despite the fact that a house icon is commonly used as an icon for the main menu, as is also the case outside of the car domain. The rationale behind the models decision to tap there remains unclear.

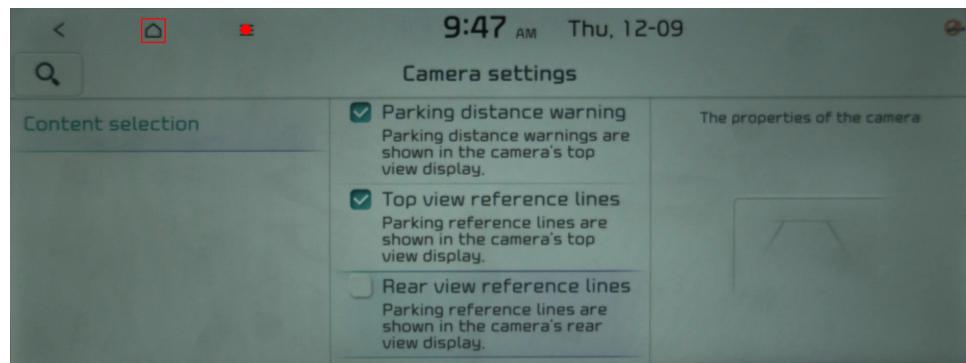


Figure 16: Miscellaneous: “*Go to main menu*”

A.5 Examples of Errors for ELAM and AutomotiveUI-Bench-4K: Expected Results

1. Box too small: “*Audio quality is set to low*”

- Figure 17 shows the audio settings menu of a Cupra Leon in German, featuring a dark background with light text. Selected texts are highlighted in orange.
- ELAM is asked to check if audio quality is set to low. The expected answer is **failed**. ELAM answers correctly with **failed**.
- The red box indicates the region defined within the image in the AutomotiveUI-Bench-4K dataset where the result is visible. The red dot marks the point at which ELAM focused to determine the result. Since the red dot is outside the box, the test is counted as failed, even though the result was correct.
- In this case, the defined area is too small. It should extend to the full length and height of the line containing the selected option *“Hoch”* (high).

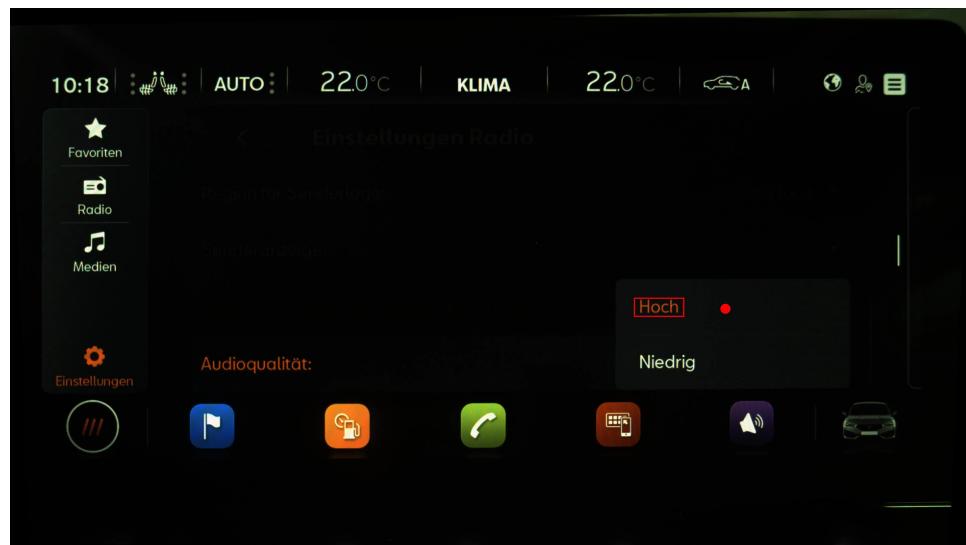


Figure 17: Box too small: “*Audio quality is set to low*”
- expected result: **failed**, ELAM’s prediction: **failed**

2. Multiple areas display the expected result: “*Sound settings are displayed*”

- Figure 18 shows the sound settings menu of a BMW iX2 in English, featuring a dark background with white icons and text. Selected items are highlighted in light yellow.
- The utterance to verify is: Sound settings are displayed. ELAM answers correctly with **passed**.
- The red box indicating the area of focus of ELAM marks the sound settings tab on the left, but the region surrounded by the red box is the heading of the sound menu at the top of the page.
- Both regions show the expected result, meaning there was only a 50% chance that ELAM would point to the correct region. However, drawing a bigger box around both regions is problematic because it could lead to false positives. Either the dataset needs the ability to store multiple regions for one utterance, or the utterance needs to be more precise, for example: “*The sound settings are shown under the SOUND label located on the left side of the top status bar*.”

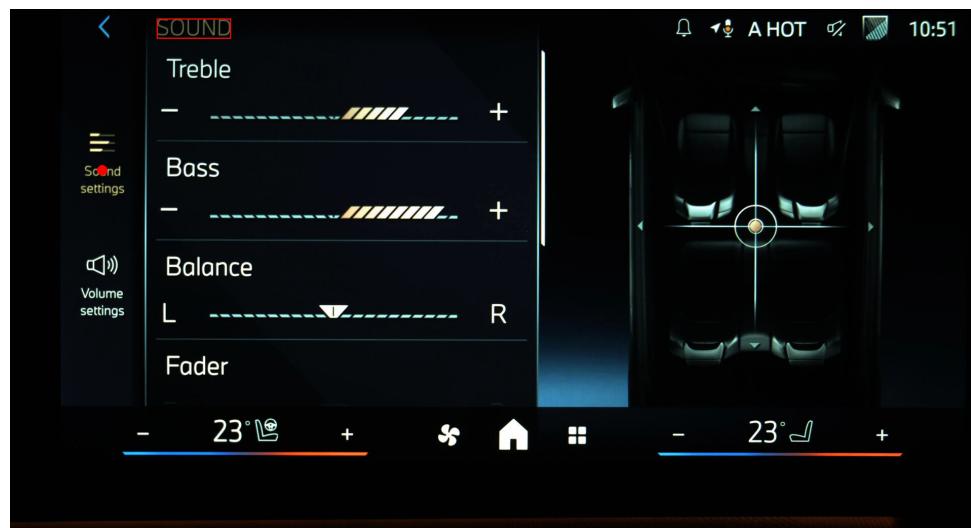


Figure 18: Multiple areas display the expected result: “*Sound settings are displayed*”
- expected result: **passed**, ELAM’s prediction: **passed**

3. Box not around option, when asked if options are shown: “*Driver assistance options are displayed*”

- Figure 19 shows the driver assistance menu of a Kia with a white background. The forward safety settings are currently selected.
- ELAM needs to check if the driver assistance options are displayed.
- The menu heading “*Driver assistance*” is marked with a red box. However, ELAM’s area of interest was the section below the heading where the menu options are displayed.
- Since emphasis was placed on the fact that options are displayed, not on the menu heading being “*Driver assistance*”, the box was drawn around the wrong region and the data should be corrected in **AutomotiveUI-Bench-4K**.

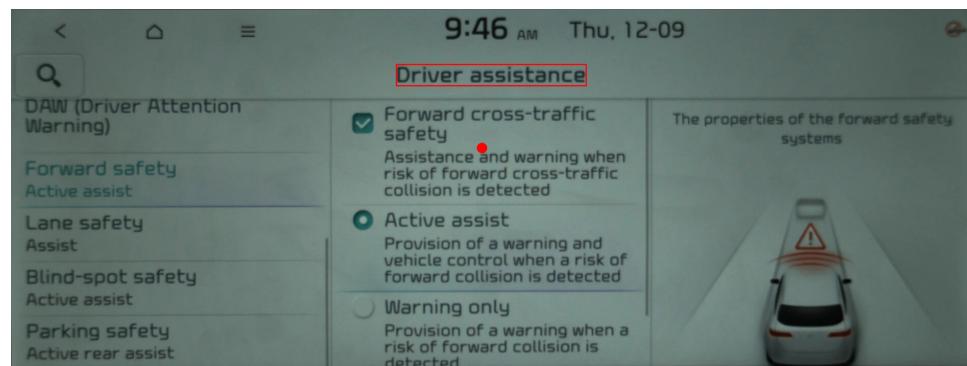


Figure 19: Box not around option, when asked if options are shown: “*Driver assistance options are displayed*”

- expected result: **passed**, ELAM’s prediction: **passed**

4. Special icons from the vehicle domain: “*The medium sensitivity of the distance control was selected*”

- Figure 20 shows the distance control menu of a BMW iX2 in German with a black background and light text.
- The expected results utterance is: The medium sensitivity of the distance control was selected. ELAM answer is **failed** instead of **passed**.
- The red box surrounds the medium sensitivity icon. ELAM selected the toggle button next to “Wechsel zu Geschwindigkeitsregelung” (switch to speed control).
- The medium sensitivity icon with two cars and two lines in between is a very automotive-specific icon. Because ELAM did not recognize the icon, it pointed to the switch to speed control setting, probably because the explanation below also contains the word distance control (“*Abstandsregelung*”).

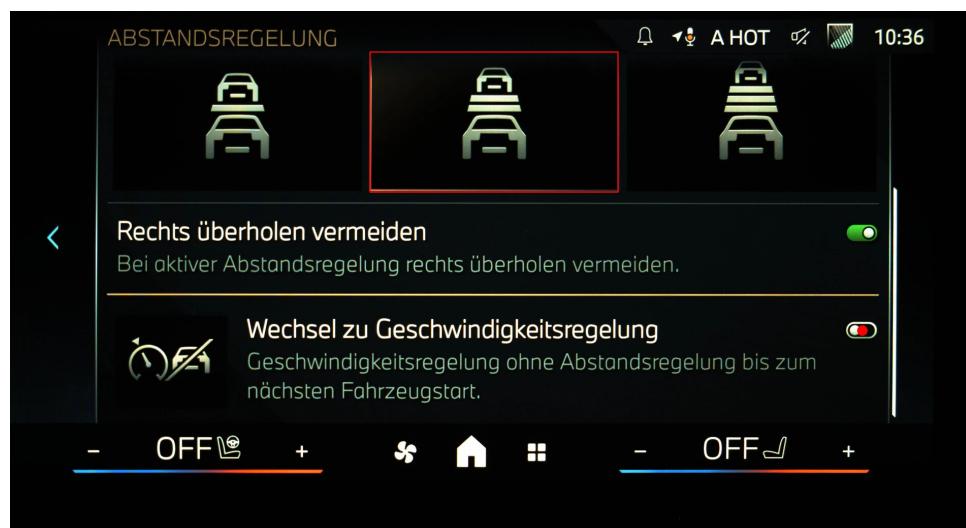


Figure 20: Special icons from the vehicle domain: “*The medium sensitivity of the distance control was selected*”

- expected result: **passed**, ELAM’s prediction: **failed**

5. Clear description, but very similar elements confused the model: “*Noise reduction is disabled*”

- Figure 21 shows the audio settings menu of a Kia in German, featuring a light background with gray text.
- The utterance to verify is: Noise cancellation is disabled. The expected result is **passed** but ELAM’s answer was **failed**.
- The red box surrounds the first radio button item in the list, that belongs to the noise reduction settings, labeled “*Originalklang*” (original sound). ELAM suggests that the second item labeled “*Leichte Rauschunterdrückung*” (light noise reduction) is the item of interest because it corresponds more to *noise reduction* semantically than “*original sound*”.
- However, the explanation provided below the radio button indicates that when the first item is selected, no noise reduction is applied at all.

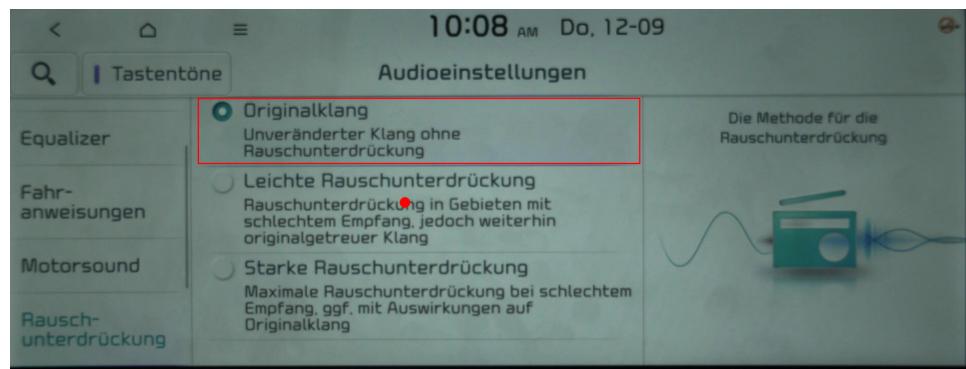


Figure 21: Clear description, but very similar elements confused the model: “*Noise reduction is disabled*”

- expected result: **passed**, ELAM’s prediction: **failed**

6. Miscellaneous: “E-Call settings are shown”

- Figure 22 shows the Wi-Fi menu of a VW ID.4. This infotainment system has a black background with white text.
- ELAM is asked to check if emergency call (E-Call) settings are currently displayed. ELAM answers correctly with **failed**.
- The red box is situated around the entire Wi-Fi menu.
- Even though ELAM answered the question correctly, its area of focus was not within the Wi-Fi menu but on the left side, on the A/C off indicator.

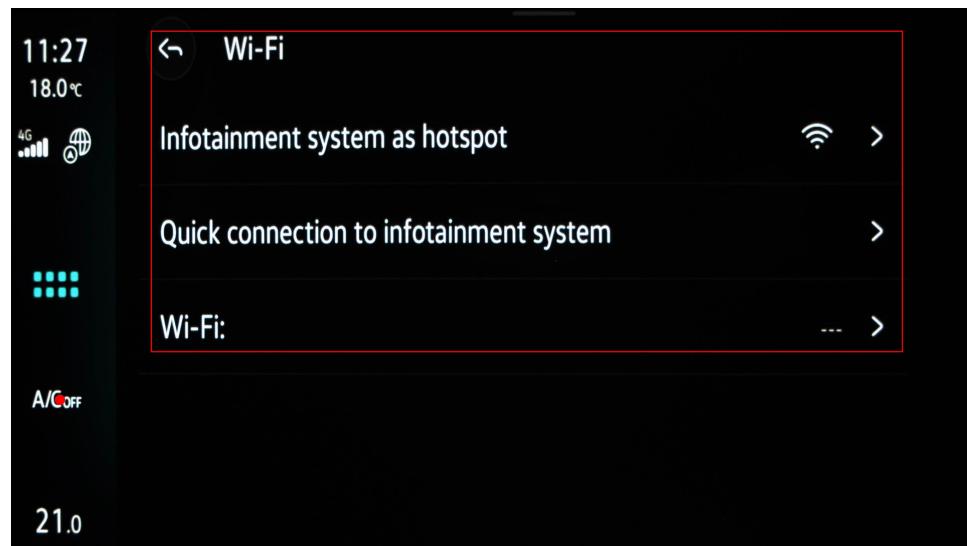


Figure 22: Miscellaneous: “E-Call settings are shown”
- expected result: **failed**, ELAM’s prediction: **failed**