

DRIV-EX: Counterfactual Explanations for Driving LLMs

Amaia Cardiel^{1,2}, Éloi Zablocki², Elias Ramzi², Eric Gaussier¹

¹ Aptikal, Université Grenoble Alpes, Grenoble, France

² Valeo.ai, Paris, France

Abstract

Large language models (LLMs) are increasingly used as reasoning engines in autonomous driving, yet their decision-making remains opaque. We propose to study their decision process through counterfactual explanations, which identify the minimal semantic changes to a scene description required to alter a driving plan. We introduce DRIV-EX, a method that leverages gradient-based optimization on continuous embeddings to identify the input shifts required to flip the model’s decision. Crucially, to avoid the incoherent text typical of unconstrained continuous optimization, DRIV-EX uses these optimized embeddings solely as a *semantic guide*: they are used to bias a controlled decoding process that re-generates the original scene description. This approach effectively steers the generation toward the counterfactual target while guaranteeing the linguistic fluency, domain validity, and proximity to the original input essential for interpretability. Evaluated using the LC-LLM planner on the textual highD dataset, DRIV-EX generates valid, fluent counterfactuals more reliably than existing baselines. It successfully exposes latent biases and provides concrete insights to improve the robustness of LLM-based driving agents. The code is available at https://github.com/Amaia-CARDIEL/DRIV_EX.

1 Introduction

End-to-end Autonomous Driving (AD) increasingly relies on Large Language Models (LLMs) and Vision-Language Models (VLMs). Thanks to large-scale pre-training, these models can integrate user instructions and sensor inputs to directly generate driving decisions (Rowe et al., 2025; Hwang et al., 2025a; Renz et al., 2024, 2025; NVIDIA, 2025). Unlike traditional neural planners that operate as black boxes, LLM-based systems provide natural language explanations, e.g., with Chain-of-Thought (CoT), often structured with rigid or

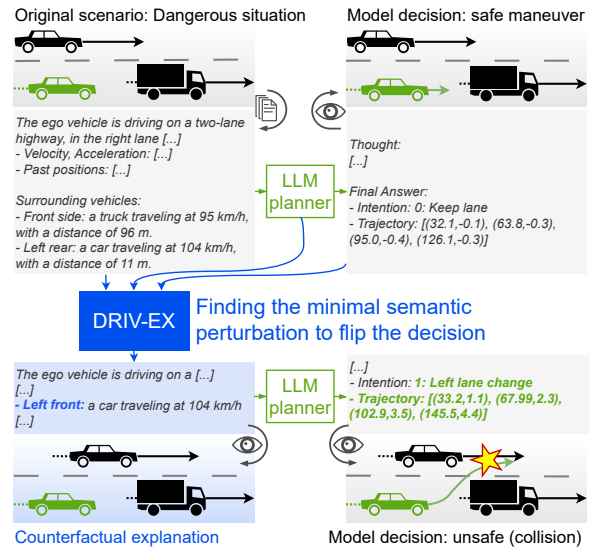


Figure 1: **Overview of DRIV-EX counterfactual generation.** The LLM acts as the planner for the ego vehicle (in green). Given an initial driving scenario where the planner behaves safely (top row), our method automatically identifies a minimal semantic perturbation to the scene description (such as slightly altering the position or speed of surrounding vehicles) that forces the model into a dangerous failure mode (bottom row). By uncovering these decision boundaries, DRIV-EX exposes latent biases and evaluates the robustness of driving agents against critical edge cases.

rule-based templates (Renz et al., 2025; Peng et al., 2024; NVIDIA, 2025). However, reliance on these explanations is risky: recent work shows that CoT reasoning is often an unfaithful rationalization that does not reflect the features that causally drive the prediction (Turpin et al., 2023). Yet, Barez et al. (2025) report that 63% of recent autonomous systems papers treat CoT as an *interpretability* method. In safety-critical domains, such unfaithful explanations can mask the true causes of dangerous behaviors, creating a strong need for methods that directly identify the input factors responsible for catastrophic decisions.

Counterfactual explanations offer a grounded al-

ternative to narrative-based insights. By identifying the minimal change to an input that would lead the model to make a different decision, counterfactuals explicitly isolate the causal features of a behavior. In the context of LLM-based driving, this involves finding the smallest modification to a scene description that turns a safe maneuver into a collision (or vice versa). As illustrated in Fig. 1, this provides a concrete mechanism to audit decision boundaries and expose latent safety risks.

Technically, generating these textual counterfactuals requires solving a difficult discrete optimization problem: finding a sequence of tokens that is semantically close to the original description yet validly flips the driving decision. This presents a dilemma between two standard approaches. *Gradient-based methods* (often used in ‘soft’ prompt optimization) relax discrete text into continuous embeddings to efficiently search for decision boundaries, but they generally fail to recover coherent text when projected back to the discrete vocabulary (Kumar et al., 2022; Wen et al., 2023). Conversely, *discrete search methods* that rely on the model’s native autoregressive sampling ensure fluency but lack the gradient guidance needed to efficiently cross the decision boundary (Liu et al., 2023; Pynadath and Zhang, 2025). Lastly, satisfying the *minimality* constraint is particularly challenging: without robust regularization, optimization algorithms tend to drift far from the original input, hallucinating entirely new objects or contexts rather than isolating the subtle feature shifts that explain the model’s behavior.

In this work, we introduce DRIV-EX (DRIVing EXplanations), an algorithm designed to resolve this tension. Our key idea is to decouple the *search* for the counterfactual from its *generation*. DRIV-EX treats optimized continuous embeddings from gradient-based methods not as the final output, but as a *semantic guide* for a controlled decoding framework. Practically, our algorithm performs gradient-based updates on the unconstrained prompt embeddings to identify the direction required to flip the decision. Then, it uses these embeddings to bias an autoregressive re-generation of the input. This allows us to steer the fluent generation capabilities of the LLM toward the critical decision boundaries identified by the gradients, effectively combining the search power of continuous optimization with the coherence of autoregressive generation.

We evaluate DRIV-EX through quantitative ex-

periments on LC-LLM (Peng et al., 2024), a recent driving LLM planner, using a textual transcription of the highD dataset (Krajewski et al., 2018), a real-world highway driving benchmark. We show that DRIV-EX generates valid counterfactuals more reliably than baselines (Pynadath and Zhang, 2025; Wen et al., 2023; Zou et al., 2023). We further use DRIV-EX to reveal biases that push models toward unsafe decisions. Finally, we demonstrate that these insights can be used to mitigate biases, improving the safety and robustness of driving LLMs.

2 Related Work

2.1 LLMs as driving agents

LLMs are becoming a dominant approach for trajectory planning in autonomous driving. LLMs were first proven able to perform planning, translated into a next token prediction task, relying either on text tokens (Mao et al., 2023; Peng et al., 2024; Yang et al., 2025), or on additional tokens accounting for discretized motions, trajectories or actions (Seff et al., 2023; Wu et al., 2024). Recently, Multimodal LLMs (MLLMs), that also process images and videos, have been increasingly used for end-to-end driving. Notably, their performance is often boosted by a Chain-of-Thought mechanism, where text tokens are predicted to ‘reason’ on the scene, right before planning. These reasoning traces are often structured with rigid or rule-based templates. Upon release, some of these methods were state of the art on major driving benchmarks, such as EMMA (Hwang et al., 2025b) on nuScenes (Caesar et al., 2020), Poutine (Rowe et al., 2025) on WOD-E2E (Xu et al., 2025) and SimLingo (Renz et al., 2025) on Bench2Drive (Jia et al., 2024).

2.2 Counterfactuals for driving (M)LLMs

Counterfactuals for (M)LLM-based driving have emerged in distinct paradigms. First, simulation-based approaches embed counterfactual reasoning within the model’s decision process. For instance, in Peng et al. (2025), the model iteratively evaluates counterfactual futures to revise its plans. Another approach uses counterfactuals as additional supervision for training, with data augmentation (Hsu et al., 2023; Wang et al., 2025). This enables models to capture causal sensitivities without performing explicit counterfactual inference at test time. In contrast, our work uses a classical formulation of counterfactuals as minimal input perturbations that alter a model’s decision, for offline auditing.

2.3 Explainability methods for NLP models

Explainability methods for NLP models are commonly divided into local and global approaches (Zhao et al., 2024). Local methods explain individual predictions, typically via attribution techniques (Li et al., 2016; Wu et al., 2020; Mohebbi et al., 2021; Enguehard, 2023). Global approaches aim to uncover model-wide behaviors across inputs (Bricken et al., 2023; Conmy et al., 2023), but are less suited to explaining specific decisions.

Counterfactual explanations are a form of local interpretability, identifying minimal changes to an input that would alter a model’s prediction, thereby exposing causal decision factors (Wachter et al., 2017). In NLP, they have been studied for discriminative tasks such as sentiment analysis and toxicity detection (Wu et al., 2021; Nguyen et al., 2024). Typical approaches generate counterfactuals by substituting, masking, or infilling tokens while preserving fluency and meaning (Wu et al., 2021). Most NLP counterfactual methods follow a two-stage pipeline: (i) identifying influential input tokens, and (ii) editing these tokens to induce a label change (Ross et al., 2021; Fern and Pope, 2021; Treviso et al., 2023). For example, MiCE (Ross et al., 2021) selects tokens via gradient-based attribution and enforces minimality through constrained search, while CREST (Treviso et al., 2023) uses sparse rationales to relax strict edit minimality. TIGTEC (Bhan et al., 2023) similarly performs sequential masked token replacement guided by local importance. Despite operating in discrete text space, these methods are not directly applicable to our setting as they are designed for classifiers.

Prompt optimization methods search for discrete or continuous input prompts that induce desired behaviors from pretrained language models. Soft prompt tuning introduces continuous embeddings optimized via gradients without updating model parameters (Lester et al., 2021; Li and Liang, 2021). Discrete approaches such as AutoPrompt (Shin et al., 2020) and PEZ (Wen et al., 2023) perform gradient-based updates in embedding space followed by projections to discrete tokens.

Related techniques have been explored extensively in adversarial prompting, jailbreak attacks, and red teaming, where the goal is to elicit unsafe or restricted behaviors from fixed LLMs (Wallace et al., 2019; Guo et al., 2024; Zou et al., 2023; Mo et al., 2024). These methods typically prioritize objective satisfaction over semantic fidelity, often

producing prompts that are unnatural or not fluent.

Our work shares the underlying optimization structure of these approaches: searching over token sequences to satisfy a target objective. However, it differs in motivation and constraints. Rather than inducing arbitrary or adversarial behaviors, we seek minimal, instance-specific edits that preserve semantic plausibility while flipping a concrete decision. This places stronger constraints on fluency, proximity, and interpretability than most prompt optimization or attack-oriented methods enforce.

Controlled decoding methods steer pretrained autoregressive models at inference time so that generated sequences satisfy explicit constraints while remaining fluent (Scholak et al., 2021; Yang and Klein, 2021). Approaches include biased autoregressive sampling (Liu et al., 2023), reinforcement learning over decoding policies (Mudgal et al., 2024), and gradient-based sampling methods combining multiple objectives (Qin et al., 2022; Kumar et al., 2022; Pynadath and Zhang, 2025). Several methods explicitly operate in discrete token space using local edits or MCMC-style proposals. For example, MuCoLa (Kumar et al., 2022) performs constrained editing via MCMC to balance constraint satisfaction and likelihood, while DAB (Pynadath and Zhang, 2025) combines discrete Langevin proposals with biased autoregressive generation to improve exploration near constraint boundaries.

While controlled decoding is closely related to our approach, key differences remain. Controlled decoding is typically framed as a generation problem without a reference input, targeting global attributes such as sentiment or topic. In contrast, our task is explicitly counterfactual: we operate relative to a specific reference prompt and enforce semantic proximity to it. This shifts the objective from general constraint satisfaction to fine-grained, local edits that expose decision boundaries.

Self-Generated Counterfactual Explanations for LLMs is an emerging paradigm, where LLMs are prompted to generate minimally altered inputs that could flip their own decision. This bypasses the need for external counterfactual methods but Dehghanighobadi et al. (2025) and Mayne et al. (2025) show that LLMs do not generate valid counterfactuals via prompting. Moreover, this paradigm is not fully adaptable to our setting: as LLM-based planners are constrained to produce formatted outputs such as trajectories, they cannot be directly prompted to generate counterfactual inputs.

3 Method: DRIV-EX

3.1 Problem and method overview

Formalization. Let \mathcal{M} be an autoregressive language model with vocabulary V . Given an original input sequence $\mathbf{x}^o = (x_1^o, \dots, x_m^o) \in V^m$, the model produces the sequence $\mathbf{y}^o = (y_1^o, \dots, y_n^o) = \mathcal{M}(\mathbf{x}^o)$. We focus on a particular decision made during generation, formalized as the token emitted at the semantic decoding step T (e.g., the first token encoding a planning decision in a structured output). Importantly, T refers to a semantic position rather than a fixed index and may depend on the generated prefix. As observed in prior work on LLM steering and jailbreaks, influencing the first critical decision token is often sufficient to redirect the entire subsequent generation (Wei et al., 2023; Carlini et al., 2023; Zou et al., 2023) since autoregressive dynamics strongly bias the continuation. This motivates focusing the counterfactual search on enforcing the constraint at step T . Empirical evidence for this phenomenon is given in App. A.

We seek a counterfactual explanation \mathbf{x}^{cf} that induces a target decision y_T^* at step T , while remaining a minimal, plausible, modification of \mathbf{x}^o . This yields competing objectives: (i) enforcing the desired *decision* at step T , (ii) preserving *fluency*, and (iii) semantic *proximity* to the initial input. We express this as a constrained optimization problem:

$$\begin{aligned} \arg \min_{\mathbf{x} \in V^m} \quad & c(\mathbf{x}^o, \mathbf{x}) \\ \text{s.t.} \quad & y_T^* = \arg \max_{z \in V} P_{\mathcal{M}}(z \mid \mathbf{y}_{<T}, \mathbf{x}), \end{aligned} \quad (1)$$

where $c(\mathbf{x}^o, \mathbf{x})$ is a cost function capturing proximity and fluency.

High-level method idea. Directly optimizing the counterfactual objective in Eq. 1 is challenging because the input \mathbf{x} lies in a discrete space and the autoregressive decoding process of LLMs is non-differentiable. Naively relaxing the input to a continuous space enables gradient-based prompt optimization with straight-through gradients (Shin et al., 2020; Wen et al., 2023), but discretizing the resulting embeddings often leads to incoherent or out of distribution text, violating the plausibility requirement.

To address this, we decouple the *search* for the counterfactual from its *generation*. We utilize the optimized continuous embeddings not as final outputs but as a *semantic guide* within a controlled decoding framework (Liu et al., 2023; Pynadath and

Algorithm 1: DRIV-EX pseudo-code

Input: Original input $\mathbf{x}^o = (x_1^o, \dots, x_m^o)$;
target token y_T^* at step T ; model \mathcal{M}
Output: Counterfactual explanation \mathbf{x}^{cf}
Initialize soft embeddings \mathbf{e} using \mathbf{x}^o ;
for $n = 1$ **to** N **do**
 ▷ *Projection and forward pass:*
 Project \mathbf{e} to get \mathbf{x} ; // Eq. 2
 Compute decision loss \mathcal{L}_{dec} ; // Eq. 3
 ▷ *Backward (straight-through update):*
 Compute gradient $\nabla_{\mathbf{x}} \mathcal{L}_{\text{dec}}$;
 Update soft embeddings \mathbf{e} ; // Eq. 4
 ▷ *Regularized autoregressive decoding:*
 Compute biases \mathcal{B} and \mathcal{B}' from \mathbf{e} and \mathbf{x}^o ; // Eq. 5–6
 Decode candidate x with biased decoding; // Eq. 7
return \mathbf{x}^{cf} ; // Best candidate x

Zhang, 2025). Specifically, we perform gradient-based updates on the embeddings to identify the direction required to flip the decision. We then use these embeddings to bias an autoregressive re-generation of the input, steering the sampling toward the identified decision boundary and the original text while strictly enforcing fluency. The complete procedure is summarized in algorithm 1.

3.2 Discrete counterfactual optimization via straight-through embeddings

Continuous relaxation. We associate each input token with a continuous embedding and maintain a sequence of soft embeddings $\mathbf{e} = (\mathbf{e}_1, \dots, \mathbf{e}_m) \in \mathbb{R}^{m \times d}$, initialized from the embeddings of the original input \mathbf{x}^o . At each iteration, the current soft embeddings are projected onto the nearest vocabulary embeddings using cosine similarity:

$$\mathbf{x}_i = \text{Proj}_E(\mathbf{e}_i) := \arg \max_{\mathbf{v} \in E} \frac{\mathbf{e}_i \cdot \mathbf{v}}{\|\mathbf{e}_i\| \|\mathbf{v}\|}, \quad (2)$$

yielding a discrete token sequence \mathbf{x} compatible with the LLM’s vocabulary.

Decision-driven gradient update. The projected sequence \mathbf{x} is fed to the model \mathcal{M} to evaluate the decision objective at step T :

$$\mathcal{L}_{\text{dec}}(\mathbf{x}) = -\log P_{\mathcal{M}}(y_T^* \mid \mathbf{y}_{<T}, \mathbf{x}), \quad (3)$$

and compute gradients with respect to the projected embeddings linked to \mathbf{x} . Using a straight-through

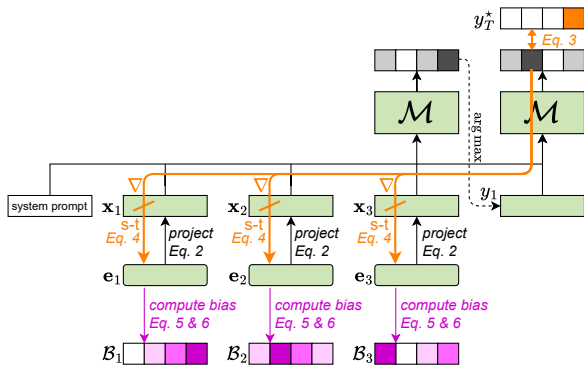


Figure 2: **Forward and backward passes, and bias computation.** In the forward pass (black), the continuous soft embeddings \mathbf{e} are projected onto their nearest neighbors in the vocabulary to obtain tokens \mathbf{x} (Eq. 2). These tokens are processed by the model \mathcal{M} to compute the probability of the target token y_T^* (Eq. 3). In the backward pass (orange), gradients (∇) are back-propagated and, using a straight-through (s-t) estimator, bypass the discrete projection to update the continuous embeddings \mathbf{e} (Eq. 4). Finally, the updated embeddings are converted into vocabulary bias terms (\mathcal{B} , in pink) to guide subsequent regularization (following Eq. 5–6).

estimator (Bengio et al., 2013), these gradients are applied to the continuous embeddings \mathbf{e} , which are updated via Adam (Kingma and Ba, 2015):

$$\mathbf{e} \leftarrow \text{Adam}(\mathbf{e}, \nabla_{\mathbf{x}} \mathcal{L}_{\text{dec}}, \eta). \quad (4)$$

This process is illustrated in Fig. 2.

This project-forward-backward-update loop constitutes the core optimization algorithm and is repeated a predefined number of N iterations.

3.3 Projection-based regularization and evaluation

While the gradient-based updates in Sec. 3.2 effectively inject the decision-change signal into the continuous embeddings \mathbf{e} , directly projecting these embeddings onto the vocabulary often yields incoherent sequences that fail to satisfy the plausibility constraint. To bridge this gap, we employ a regularized autoregressive decoding strategy (Liu et al., 2023; Pynadath and Zhang, 2025) where the optimized embeddings serve as a *semantic guide* to steer a dedicated fluency model \mathcal{F} . We instantiate \mathcal{F} using the same pre-trained model as \mathcal{M} , yet without planning-oriented adaptation, to guarantee a shared vocabulary and tokenization, enabling us to directly bias its logits using distances computed in the common embedding space.

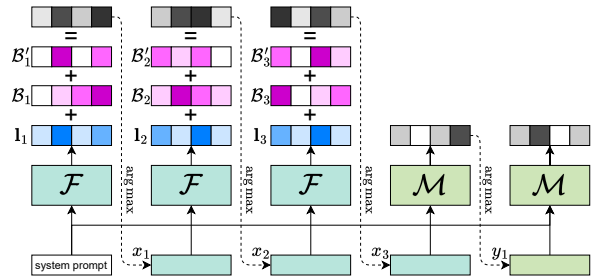


Figure 3: **Regularized autoregressive decoding.** During the regularization phase, the vocabulary bias terms \mathcal{B} and \mathcal{B}' (derived from optimized embeddings and \mathbf{x}^o) are added to the logits \mathbf{l} of a fluency model \mathcal{F} . This combined signal biases the auto-regressive decoding, following Eq. 7, to generate a new candidate sequence (x_1, x_2, x_3) that incorporates the decision-change signal while maintaining fluency and input proximity.

Fluency regularization with biased autoregressive decoding. For each position i , we compute a distance vector $\hat{\mathbf{b}}_i \in \mathbb{R}^{|\mathcal{V}|}$ measuring how far each vocabulary token v is from the projected token $\text{Proj}_E(\mathbf{e}_i)$ in embedding space:

$$b_{i,v} = \|\mathbf{v} - \text{Proj}_E(\mathbf{e}_i)\|_2^2. \quad (5)$$

To match the scale of the model logits $\mathbf{l}_i = \text{logits}_{\mathcal{F}}(\cdot | x_{<i})$, we compute a normalization factor $r_i = \|\mathbf{l}_i\|_2 / \|\hat{\mathbf{b}}_i\|_2$ as in Pynadath and Zhang (2025). The bias applied to each token v is then

$$\mathcal{B}_{i,v} = -w \cdot r_i \cdot b_{i,v}, \quad (6)$$

where w controls the strength of the bias.

The bias is added to the logits of the fluency model \mathcal{F} during greedy decoding, encouraging generations that are close to the optimized embeddings while preserving fluency. Concretely, as illustrated in Fig. 3, the next token is chosen with: $x_i = \arg \max_{v \in \mathcal{V}} (l_{i,v} + \mathcal{B}_{i,v})$.

The resulting decoded sequence provides a concrete counterfactual candidate that can be evaluated against the objective in Eq. 1. Importantly, this regularization does not define a separate optimization step but serves to monitor and enforce plausibility throughout the optimization trajectory.

Input proximity regularization. We propose three complementary approaches to ensure semantic proximity to \mathbf{x}^o . First, ‘**Proj**’ implies to limit embedding projections (Eq. 2) to the subset of the top K nearest neighbors of \mathbf{x}^o (in cosine similarity). Second, ‘**Loss**’ adds $(1 - \frac{1}{m} \sum_{i=1}^m \cos \text{sim}(\mathbf{x}_i, \mathbf{x}_i^o))$ as a regularization term to the decision loss (Eq. 3), controlled by a weight λ . Finally, ‘**Bias**’ adds a

second bias during autoregressive biasing (Eq. 7). Using \mathbf{x}_i^o instead of the current projected token $\text{Proj}_E(\mathbf{e}_i)$ in Eq. 5, we compute $\mathcal{B}'_{i,v}$ following Eq. (5)–(6), and the biased sampling becomes:

$$x_i = \arg \max_{v \in \mathcal{V}} (l_{i,v} + \mathcal{B}_{i,v} + \mathcal{B}'_{i,v}). \quad (7)$$

Best candidate selection. Upon completion of N iterations, we use a heuristic selection to derive the best counterfactual from the pool of generated candidates. We prioritize candidates that successfully trigger the decision change; specifically, we define the set of valid candidates as those satisfying $y_T^* = \arg \max_{z \in \mathcal{V}} P_{\mathcal{M}}(z \mid \mathbf{y}_{<T}, x)$. If this set is non-empty, we return the candidate that maximizes semantic proximity to the original input \mathbf{x}^o , as the autoregressive decoding is already ensuring high fluency. Conversely, if no candidate triggers the decision change, we return the sequence that achieves the highest target probability $P_{\mathcal{M}}(y_T^* \mid \mathbf{y}_{<T}, x)$.

4 Experiments

We evaluate DRIV-EX on LC-LLM (Peng et al., 2024), a LoRA-finetuned planner (Hu et al., 2022), on a textual transcription of the highD dataset (Krajewski et al., 2018). This dataset maps traffic annotations to Lane Changes (LC) and trajectories.

4.1 Experimental Protocol

We focus on probing the brittleness of the model’s safety predictions. From the validation set, we select ‘*dangerous*’ scenarios, defined as those where the ground-truth trajectory implies a potential collision (impact within 4s at constant velocity). We further focus on scenarios where a given model predicts a ‘*safe*’ maneuver. To assess robustness, we investigate whether minimal semantic perturbations to these scenes can force the model into a catastrophic failure. We set the target y_T^* to the ground-truth lane change token, and employ DRIV-EX and baselines to flip the model’s decision from ‘*safe*’ to ‘*dangerous*’. For all methods, updates are restricted to tokens related to words that can change in the driving template (e.g., vehicle type, speed, position). These tokens are shown in Sec. C.2, Fig. 9 and subset statistics are given in Tab. 10. This setup, designed for offline model auditing, allows us to (1) benchmark DRIV-EX against existing baselines (Sec. 4.2), (2) uncover model biases that trigger unsafe decisions (Sec. 4.3), and (3) demonstrate how these insights can help mitigate bias and improve safety (Sec. 4.4).

4.2 Comparison with baselines and ablations

We compare DRIV-EX against DAB (Pynadath and Zhang, 2025), PEZ (Wen et al., 2023), GCG (Zou et al., 2023) and their task-adapted versions (†). All methods are used on the LC-LLM planner, implemented with instructed versions of Llama3-8B (Dubey et al., 2024), Mistral-7B (Jiang et al., 2023) and Qwen2.5-7B (Yang et al., 2024), as detailed in Sec. B.2. Baselines description, adaptation and hyperparameter search are detailed in App. D.

Evaluation Metrics. We track metrics that relate to the counterfactual explanation constraints, given in Eq. 1. We describe them below; further details on metrics and thresholds are given in Sec. B.1.

- **Decision (flip rate):** Ratio of candidates that lead to the target y_T^* being top-ranked at step T .

- **Similarity and Fluency:** BERTScore (Zhang et al., 2020) is a measure of semantic proximity between sequence pairs. Only counterfactuals with a BERTScore ≥ 0.95 with the initial input are kept. In **BERTScore filter**, we give the ratio of such counterfactuals. **Template Fitness filter** quantifies the ratio of counterfactuals that perfectly respect the driving template. **Min Fluency** is the minimum conditional token probability (cf Sec. B.1).

- **Aggregated Scores (Success Rate):** These are our primary metrics that we optimize. **Aggreg** measures the number of counterfactual ‘Successes’, where a success satisfies three strict conditions: the decision is flipped, the template is perfectly valid and semantic similarity is high (BERTScore ≥ 0.95). We also report **Aggreg & Col**, the subset of successes that lead to a collision. It quantifies the method’s ability to expose dangerous behaviors.

Quantitative results. Generating counterfactuals requires balancing decision flipping with similarity and fluency. As shown in Tab. 1, off-the-shelf DAB achieves a high flip rate (100% for Llama3) but fails on similarity (BERTScore filter of 33.3%) due to a highly exploratory proposal step. Adding proximity regularization (DAB†) improves similarity but drastically degrades the decision flip rate (46.7%). Conversely, PEZ (and its adapted version) maintains high flip rate and similarity to the original scene but lacks fluency, resulting in a lower ‘Aggregated Score’ (45.3% vs. 61.3% for DRIV-EX on Llama3). GCG is a strong baseline on Llama3, reaching the second best ‘Aggregated Score’ of 58.7%. However, its performance is weak on other LLMs, particularly so on Mistral where it has the

Method	LLM	Flip rate	Fluency		Similarity	Aggregation	
		y_T^* has top \uparrow rank (%)	Min \uparrow fluency	Template \uparrow filter (%)	BERTSc \uparrow filter (%)	Aggreg \uparrow (%)	Aggreg \uparrow & col (%)
DAB (Pynadath and Zhang, 2025)	Llama3	100.0	9e-4	86.7	33.3	33.3	14.7
	Mistral	91.8	6e-3	30.6	26.5	24.5	20.4
	Qwen2.5	98.9	2e-3	7.8	10.0	7.8	7.8
DAB \dagger	Llama3	46.7	5e-3	100.0	94.7	41.3	30.7
	Mistral	40.8	3e-2	98.0	91.8	38.8	34.7
	Qwen2.5	26.7	7e-3	93.3	93.3	24.4	21.1
PEZ (Wen et al., 2023)	Llama3	97.3	2e-4	45.3	81.3	45.3	37.3
	Mistral	100.0	5e-4	42.9	89.8	42.9	<u>40.8</u>
	Qwen2.5	96.7	2e-3	23.3	44.4	23.3	17.8
PEZ \dagger	Llama3	97.3	2e-4	49.3	86.7	49.3	37.3
	Mistral	100.0	1e-3	57.1	95.9	<u>57.1</u>	<u>40.8</u>
	Qwen2.5	94.4	3e-3	30.0	58.9	<u>30.0</u>	23.3
GCG (Zou et al., 2023)	Llama3	98.7	2e-4	60.0	100.0	<u>58.7</u>	<u>44.0</u>
	Mistral	98.0	1e-4	20.4	100.0	20.4	18.4
	Qwen2.5	88.9	5e-4	13.3	98.9	13.3	8.9
DRIV-EX (ours)	Llama3	64.0	1e-3	88.0	96.0	61.3	56.0
	Mistral	83.7	4e-3	79.6	95.9	69.4	61.2
	Qwen2.5	58.9	3e-3	47.8	73.3	34.4	<u>22.2</u>

Table 1: **Main counterfactual explanation results** on safety-critical textual highD subsets, where we steer LLMs from a safe decision to a collision. We give results for comparable compute budgets: N=15 iterations for DAB/DRIV-EX, N=150 for PEZ, except for the compute-heavy GCG for which N=6 leads to 5 times more compute. ‘ \dagger ’ denotes task-adapted baselines (vs. off-the-shelf), ‘BERTSc’ stands for ‘BERTScore’.

lowest ‘Aggregated Score’ (20.4%), due to poor token replacement proposals. This variation suggests that GCG is sensitive to the embedding space geometry of LLMs. DRIV-EX outperforms all baselines in ‘Aggregated Score’, for all LLMs, by effectively using the fluency model \mathcal{F} to guide generation. This ensures counterfactuals that are valid, fluent and semantically close to the original scene (96.0% BERTScore filter on Llama3).

Scaling performance. We show in Fig. 4 how performance in ‘Aggregated Score’ scales with compute for DRIV-EX versus baselines. All runs are done on 40G A100 GPUs for fair comparison. Though PEZ and PEZ \dagger are reaching competitive results for a small compute budget, their performances hardly scale with more iterations. This suggests that their best candidates are identified in early iterations, before the unregularized optimization damages the fluency. GCG has competitive results on Llama3 but its compute cost is high due to the systematic evaluation of sampled candidates, despite batched evaluations in our implementation (batch size = 16). Moreover, its performance barely scales beyond 3 iterations. In

contrast, DAB is the only baseline that exhibits similar scaling to DRIV-EX, but with lower performance, while DAB \dagger has stagnating results. Finally, DRIV-EX is the method that scales the most with compute, achieving by its 15th iteration a higher ‘Aggregated Score’ than all other baselines.

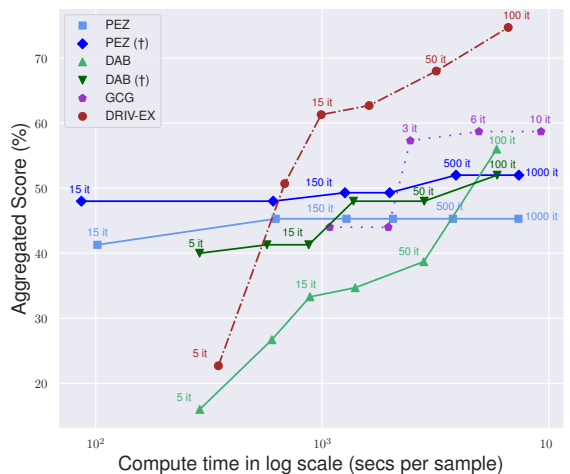


Figure 4: **Performance scaling with respect to compute**, evaluated for all methods on Llama3 with A100 GPUs. ‘it’ gives the number of iterations above runs.

Proximity regularization	BERTSc \uparrow filter(%)	Aggreg \uparrow (%)	Aggreg \uparrow & Col(%)
\mathcal{B} (base, no reg)	82.7	52.0	48.0
$\mathcal{B} + \mathcal{B}$ (Bias)	93.3	54.7	45.3
$\mathcal{B} + \mathcal{P}$ (Proj)	74.7	50.7	46.7
$\mathcal{B} + \mathcal{L}$ (Loss)	90.7	58.7	50.7
$\mathcal{B} + \mathcal{B} + \mathcal{P} + \mathcal{L}$	96.0	61.3	56.0

Table 2: **Ablation of DRIV-EX proximity regularization strategies.** We compare the impact of ‘Bias’ (B), ‘Proj’ (P), and ‘Loss’ (L) terms for Llama3.

Ablation on proximity regularization. Tab. 2 evaluates the components ensuring semantic proximity to the original input, introduced in Sec. 3.3. Both ‘Bias’ and ‘Loss’ terms improve the ‘Aggregated Score’ (+2.7% and +6.7% respectively), and they are used in our adapted (\dagger) versions of PEZ and DAB. While ‘Proj’ alone reduces success, combining it with ‘Bias’ and ‘Loss’ yields the best proximity and performance (+8.0% in ‘Aggreg & Col’), confirming the synergy of our regularizations.

Qualitative result. Fig. 14 displays a counterfactual explanation generated with DRIV-EX. We observe that by adjusting the ego vehicle’s longitudinal speed (v_x) and reducing a surrounding vehicle’s speed (from 98.21 km/h to 93.21 km/h), DRIV-EX successfully identifies a valid, minimal perturbation that forces the planner into a collision.

4.3 Evaluating driving LLMs’ robustness

4.3.1 Retrieving injected biases

To validate DRIV-EX, we first assess its ability to recover known, artificial biases injected into the training data of the driving planner. We finetune two instances of the Llama3-based planner with the following ‘shortcuts’: (1) **‘Vehicle bias’**: the model is trained to associate the ego-vehicle type (car vs. truck) with specific lane-change decisions; (2) **‘Digit bias’**: the model is trained to rely on the last digit of surrounding vehicles’ speeds to determine its maneuver. More details in Sec. C.2.

These biased models perform similarly to unbiased models on data matching their training biases. However, their safety significantly degrades when tested on unbiased or ‘inversely-biased’ samples (details in Tab. 9). This highlights a critical risk: LLM planners can appear competent while actually relying on non-safety-critical ‘surface’ features.

To evaluate whether DRIV-EX can successfully find latent ‘shortcuts’, we apply it to biased planners in safety-critical scenarios, and track the mod-

Bias	# of data	Change type	Token change (%)	
			Biased	Unbiased
Vehicle	19	any	89.5	8.0
		revealing	84.2	/
Digit	135	any	52.9	5.3
		revealing	34.0	/

Table 3: **Token change (%) between x^{cf} and x^o when using DRIV-EX.** Results are split between biased and unbiased token positions. For biased tokens, ‘revealing’ indicates cases where token values change accordingly to the injected bias, ‘any’ for all cases.

ification frequency of ‘biased’ versus ‘unbiased’ token positions. We focus on samples where biased token positions have unbiased values in the initial scene, while the counterfactual is a success. Tab. 3 confirms that biased token positions are significantly more susceptible to modification across both experiments. In the vehicle bias case, DRIV-EX modifies the shortcut token in 89.5% of cases, compared to 8.0% for unbiased positions. In 84.2% of cases, changes are ‘bias-revealing,’ meaning DRIV-EX directly unveils the injected shortcut, such as changing a ‘car’ to a ‘truck’ to induce the decision change. Similarly, in the digit bias case, DRIV-EX modifies the subtle trailing digits in 52.9% of cases, with 34% of cases being bias-revealing. Qualitative results are shown in Fig. 15 and Fig. 16. These findings confirm that DRIV-EX effectively isolates specific, even non-obvious, input features that a model relies on to make driving decisions.

4.3.2 Identifying unknown biases

We now aim at finding biases in planners trained on the original textual highD dataset. We apply DRIV-EX to the Llama3-based planner, and focus on two situations: ‘Keep lane’ was initially inferred and we target either ‘Right lane change’ or ‘Left lane change’, as it covers the majority of valid counterfactuals leading to crashes (78% and 19%).

Without prior knowledge of existing biases, we track the percentage of token switches across all positions for counterfactuals that are both a success and lead to a collision. For ‘Keep lane’ \rightarrow ‘Right lane change’, the histogram of token switch frequency in Fig. 5 reveals that lateral velocity tokens (v_y) are highly sensitive, with values switching in more than 50% of successful cases. Manual inspection of the counterfactuals reveals that higher negative v_y values strongly bias the LLM toward

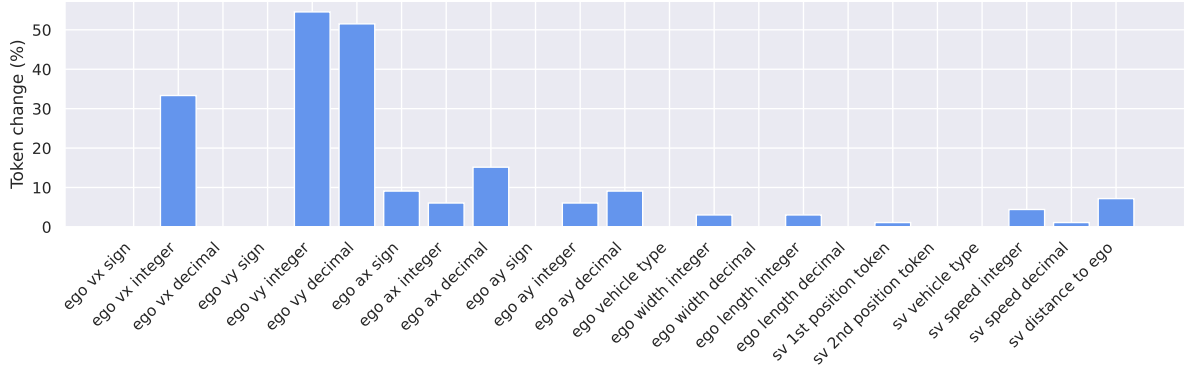


Figure 5: **Histogram of number of token changes (%) across input position for successful counterfactuals** (Llama3) that flip a ‘Keep lane’ decision to a collision-inducing ‘Right lane change’ ($n=33$ samples). Peaks indicate tokens most critical for the decision flip. ‘sv’: ‘surrounding vehicle’, ‘vx/vy/ax/ay’ is for velocity and acceleration.

selecting a ‘Right lane change’, even when it leads to a collision. Specifically, while original v_y values range from $[-0.43, 0.0]$, DRIV-EX identifies that shifting these to $[-4.0, -0.47]$ flips the decision. This shows that the model has learned to associate high negative lateral velocity with a rightward maneuver, prioritizing this shortcut over safety. Additional results in Sec. C.3 show that Llama3 is also biased towards a left lane change when the ego car’s longitudinal velocity exceeds a threshold.

Finally, an analysis on Mistral and Qwen2.5 (cf Sec. C.3) suggests that, although to a lesser extent than Llama3, they also are sensitive to the ego car’s velocity. Besides, they appear to be sensitive to tokens related to surrounding vehicles, in particular Qwen2.5 when it comes to other cars’ speeds and distances to ego. These results confirm that DRIV-EX can effectively identify features that lead to unsafe behaviors in driving planners.

4.4 Mitigation for Enhanced Robustness

The results in Sec. 4.3 suggest that LLM-based planners over-rely on lateral velocity (v_y) shortcuts. To test if removing these biased features improves safety, we finetune Llama3-8B on three variations of the textual highD template: *i. Baseline* (None): Includes all velocity and acceleration information (v_x, v_y, a_x, a_y). *ii. Lateral-Agnostic* (no v_y, a_y): Removes lateral kinematics to eliminate the identified v_y bias. *iii. Kinematic-Free* (no v, a): Removes velocity and acceleration data, forcing reliance on spatial positioning.

We evaluate the resulting planners on the full textual highD validation set as shown in Tab. 4. We find that removing specific kinematic features enhances planner robustness. The ‘Lateral-Agnostic’ model maintains similar F1 and RMSE scores,

Text	Classif (F1 \uparrow)		Traj (RMSE \downarrow)		#of	%of
	macro	micro	lon	lat		
None	0.93	0.93	0.68	0.40	753	3.1
no v_y, a_y	0.92	0.92	0.48	0.51	736	3.1
no v, a	0.90	0.90	1.33	0.55	631	2.6

Table 4: **Driving performance across debiased templates** (Llama3) on highD val. We report lane change classification scores, trajectory error, number and ratio of collisions (‘col’). Removing kinematic features slightly increases trajectory error but improves safety.

while reducing collisions. On the other hand, the ‘Kinematic-Free’ model drastically reduces collisions but degrades trajectory prediction accuracy, highlighting a trade-off between predictive performance and safety. While our work focuses on bias detection, this intervention study shows that DRIV-EX identifies features that can guide robustness improvements. Finer mitigation strategies, such as counterfactual data augmentation, could also be explored to leverage DRIV-EX generations.

5 Conclusion

In this work, we introduce DRIV-EX, a framework that explains driving LLM decisions by generating minimal, fluent counterfactuals. By decoupling gradient-based search from biased autoregressive generation, DRIV-EX successfully bridges the gap between the precise control of continuous optimization and the coherence of discrete text generation. Our experiments on the LC-LLM planner demonstrate that DRIV-EX outperforms baselines, effectively exposing latent biases, such as over-reliance on velocity, and enabling targeted safety improvements. Additionally, a refinement procedure of counterfactuals is proposed in Appendix Sec. C.1.

Limitations

While DRIV-EX effectively identifies decision boundaries, it presents specific limitations. First, the method currently operates solely on textual inputs. Extending DRIV-EX to Vision-Language Models (VLMs) to directly handle image-based counterfactuals is a critical avenue for future research. Second, DRIV-EX can only be applied to white-box models. Third, the optimization assumes a fixed sequence length; the method cannot currently generate counterfactuals that require inserting or deleting tokens relative to the original input. Fourth, our evaluation is limited to the textual highD dataset and a specific rigid template. While the framework is generic, broader validation across diverse NLP tasks and free-form text remains to be done. Fifth, by combining gradient-based search with autoregressive sampling, DRIV-EX inherits computational costs, making it slower per-sample than pure gradient methods. Additionally, as a local interpretability method, aggregating individual counterfactuals into global insights is challenging. The aggregation strategies proposed in [Sec. 4.3](#) rely on the structured nature of the textual highD template and may not generalize to unstructured free text. Finally, our method is intended for offline model auditing, not online deployment.

Ethical Considerations

The primary ethical implication of this work lies in its potential for dual use. DRIV-EX automates the discovery of input perturbations that induce dangerous driving behaviors (e.g., collisions). While our intent is to audit and improve safety, this capability effectively generates corner-cases that could theoretically be used to attack autonomous systems. However, we maintain that uncovering these vulnerabilities in a controlled setting is a prerequisite for building robust defenses. Additionally, there is a risk of over-reliance on the generated explanations. Counterfactuals are approximations and may not be exhaustive; if the explanations drift from the model’s true decision mechanics or miss important ones, it could lead developers to incorrect conclusions about safety. Therefore, these insights should be treated as diagnostic tools rather than absolute guarantees of model logic.

Acknowledgements

We thank levelxdata for granting us access to the highD dataset ([Krajewski et al., 2018](#)).

This work was performed using HPC resources from GENCI-IDRIS (Grant 2025-AD011014446R2) and with the support of the ANR MultiTrans (ANR-21-CE23-0032).

References

- Maksym Andriushchenko, Francesco Croce, and Nicolas Flammarion. 2025. [Jailbreaking leading safety-aligned llms with simple adaptive attacks](#). In *The Thirteenth International Conference on Learning Representations, ICLR 2025, Singapore, April 24-28, 2025*.
- Fazl Barez, Tung-Yu Wu, Iván Arcuschin, Michael Lan, Vincent Wang, Noah Siegel, Nicolas Collignon, Clement Neo, Isabelle Lee, Alasdair Paren, and 1 others. 2025. [Chain-of-thought is not explainability](#). *Preprint, alphaXiv*, page v1.
- Yoshua Bengio, Nicholas Léonard, and Aaron C. Courville. 2013. [Estimating or propagating gradients through stochastic neurons for conditional computation](#). *CoRR*, abs/1308.3432.
- Milan Bhan, Jean-Noël Vittaut, Nicolas Chesneau, and Marie-Jeanne Lesot. 2023. [TIGTEC: token importance guided text counterfactuals](#). In *Machine Learning and Knowledge Discovery in Databases: Research Track - European Conference, ECML PKDD 2023, Turin, Italy, September 18-22, 2023, Proceedings, Part III*, volume 14171 of *Lecture Notes in Computer Science*, pages 496–512. Springer.
- Trenton Bricken, Adly Templeton, Joshua Batson, Brian Chen, Adam Jermy, Tom Conerly, Nick Turner, Cem Anil, Carson Denison, Amanda Askell, Robert Lasenby, Yifan Wu, Shauna Kravec, Nicholas Schiefer, Tim Maxwell, Nicholas Joseph, Zac Hatfield-Dodds, Alex Tamkin, Karina Nguyen, and 6 others. 2023. [Towards monosemanticity: Decomposing language models with dictionary learning](#). *Transformer Circuits Thread*.
- Holger Caesar, Varun Bankiti, Alex H. Lang, Sourabh Vora, Venice Erin Liong, Qiang Xu, Anush Krishnan, Yu Pan, Giancarlo Baldan, and Oscar Beijbom. 2020. [nusenes: A multimodal dataset for autonomous driving](#). In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2020, Seattle, WA, USA, June 13-19, 2020*, pages 11618–11628. Computer Vision Foundation / IEEE.
- Nicholas Carlini, Milad Nasr, Christopher A. Choquette-Choo, Matthew Jagielski, Irena Gao, Pang Wei Koh, Daphne Ippolito, Florian Tramèr, and Ludwig Schmidt. 2023. [Are aligned neural networks adversarially aligned?](#) In *NeurIPS 2023*.
- Arthur Conmy, Augustine N. Mavor-Parker, Aengus Lynch, Stefan Heimersheim, and Adrià Garriga-Alonso. 2023. [Towards automated circuit discovery for mechanistic interpretability](#). In *Advances in Neural Information Processing Systems 36: Annual*

- Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023.*
- Zahra Dehghanighobadi, Asja Fischer, and Muhammad Bilal Zafar. 2025. [Can llms explain themselves counterfactually?](#) In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing, EMNLP 2025, Suzhou, China, November 4-9, 2025*, pages 7787–7815. Association for Computational Linguistics.
- Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Amy Yang, and Angela Fan et al. 2024. The llama 3 herd of models. *arXiv preprint*.
- Joseph Enguehard. 2023. [Sequential integrated gradients: a simple but effective method for explaining language models.](#) In *Findings of the Association for Computational Linguistics: ACL 2023, Toronto, Canada, July 9-14, 2023*, pages 7555–7565. Association for Computational Linguistics.
- Xiaoli Z. Fern and Quintin Pope. 2021. [Text counterfactuals via latent optimization and shapley-guided search.](#) In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing, EMNLP 2021, Virtual Event / Punta Cana, Dominican Republic, 7-11 November, 2021*, pages 5578–5593. Association for Computational Linguistics.
- Xingang Guo, Fangxu Yu, Huan Zhang, Lianhui Qin, and Bin Hu. 2024. [Cold-attack: Jailbreaking llms with stealthiness and controllability.](#) In *Forty-first International Conference on Machine Learning, ICML 2024, Vienna, Austria, July 21-27, 2024*. OpenReview.net.
- Kai-Chieh Hsu, Karen Leung, Yuxiao Chen, Jaime F. Fisac, and Marco Pavone. 2023. [Interpretable trajectory prediction for autonomous vehicles via counterfactual responsibility.](#) In *IROS*, pages 5918–5925.
- Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2022. [Lora: Low-rank adaptation of large language models.](#) In *The Tenth International Conference on Learning Representations, ICLR 2022, Virtual Event, April 25-29, 2022*. OpenReview.net.
- Jyh-Jing Hwang, Runsheng Xu, Hubert Lin, Wei-Chih Hung, Jingwei Ji, Kristy Choi, Di Huang, Tong He, Paul Covington, Benjamin Sapp, Yin Zhou, James Guo, Dragomir Anguelov, and Mingxing Tan. 2025a. [EMMA: end-to-end multimodal model for autonomous driving.](#) *Trans. Mach. Learn. Res.*, 2025.
- Jyh-Jing Hwang, Runsheng Xu, Hubert Lin, Wei-Chih Hung, Jingwei Ji, Kristy Choi, Di Huang, Tong He, Paul Covington, Benjamin Sapp, Yin Zhou, James Guo, Dragomir Anguelov, and Mingxing Tan. 2025b. [EMMA: end-to-end multimodal model for autonomous driving.](#) *Trans. Mach. Learn. Res.*, 2025.
- Xiaosong Jia, Zhenjie Yang, Qifeng Li, Zhiyuan Zhang, and Junchi Yan. 2024. [Bench2drive: Towards multi-ability benchmarking of closed-loop end-to-end autonomous driving.](#) In *Advances in Neural Information Processing Systems 38: Annual Conference on Neural Information Processing Systems 2024, NeurIPS 2024, Vancouver, BC, Canada, December 10 - 15, 2024*.
- Albert Q. Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de Las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, L elio Renard Lavaud, Marie-Anne Lachaux, Pierre Stock, Teven Le Scao, Thibaut Lavril, Thomas Wang, Timoth ee Lacroix, and William El Sayed. 2023. [Mistral 7b.](#) *CoRR*, abs/2310.06825.
- Diederik P. Kingma and Jimmy Ba. 2015. [Adam: A method for stochastic optimization.](#) In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*.
- Robert Krajewski, Julian Bock, Laurent Kloeker, and Lutz Eckstein. 2018. [The highd dataset: A drone dataset of naturalistic vehicle trajectories on german highways for validation of highly automated driving systems.](#) In *21st International Conference on Intelligent Transportation Systems, ITSC 2018, Maui, HI, USA, November 4-7, 2018*, pages 2118–2125. IEEE.
- Sachin Kumar, Biswajit Paria, and Yulia Tsvetkov. 2022. [Gradient-based constrained sampling from language models.](#) In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing, EMNLP 2022, Abu Dhabi, United Arab Emirates, December 7-11, 2022*, pages 2251–2277. Association for Computational Linguistics.
- Brian Lester, Rami Al-Rfou, and Noah Constant. 2021. [The power of scale for parameter-efficient prompt tuning.](#) In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing, EMNLP 2021, Virtual Event / Punta Cana, Dominican Republic, 7-11 November, 2021*, pages 3045–3059. Association for Computational Linguistics.
- Jiwei Li, Xinlei Chen, Eduard H. Hovy, and Dan Jurafsky. 2016. [Visualizing and understanding neural models in NLP.](#) In *NAACL HLT 2016, The 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, San Diego California, USA, June 12-17, 2016*, pages 681–691. The Association for Computational Linguistics.
- Xiang Lisa Li and Percy Liang. 2021. [Prefix-tuning: Optimizing continuous prompts for generation.](#) In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing, ACL/IJCNLP 2021, (Volume 1: Long Papers), Virtual Event, August 1-6, 2021*, pages 4582–4597. Association for Computational Linguistics.

- Xin Liu, Muhammad Khalifa, and Lu Wang. 2023. [BOLT: fast energy-based controlled text generation with tunable biases](#). In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers), ACL 2023, Toronto, Canada, July 9-14, 2023*, pages 186–200. Association for Computational Linguistics.
- Jiageng Mao, Yuxi Qian, Hang Zhao, and Yue Wang. 2023. [Gpt-driver: Learning to drive with GPT](#). *CoRR*, abs/2310.01415.
- Harry Mayne, Ryan Othniel Kearns, Yushi Yang, Andrew M. Bean, Eoin D. Delaney, Chris Russell, and Adam Mahdi. 2025. [Llms don't know their own decision boundaries: The unreliability of self-generated counterfactual explanations](#). In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing, EMNLP 2025, Suzhou, China, November 4-9, 2025*, pages 24161–24186. Association for Computational Linguistics.
- Yichuan Mo, Yuji Wang, Zeming Wei, and Yisen Wang. 2024. [Fight back against jailbreaking via prompt adversarial tuning](#). In *Advances in Neural Information Processing Systems 38: Annual Conference on Neural Information Processing Systems 2024, NeurIPS 2024, Vancouver, BC, Canada, December 10 - 15, 2024*.
- Hosein Mohebbi, Ali Modarressi, and Mohammad Taher Pilehvar. 2021. [Exploring the role of BERT token representations to explain sentence probing results](#). In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing, EMNLP 2021, Virtual Event / Punta Cana, Dominican Republic, 7-11 November, 2021*, pages 792–806. Association for Computational Linguistics.
- Sidharth Mudgal, Jong Lee, Harish Ganapathy, YaGuang Li, Tao Wang, Yanping Huang, Zhifeng Chen, Heng-Tze Cheng, Michael Collins, Trevor Strohmaier, Jilin Chen, Alex Beutel, and Ahmad Beirami. 2024. [Controlled decoding from language models](#). In *Forty-first International Conference on Machine Learning, ICML 2024, Vienna, Austria, July 21-27, 2024*. OpenReview.net.
- Van Bach Nguyen, Paul Youssef, Christin Seifert, and Jörg Schlötterer. 2024. [Llms for generating and evaluating counterfactuals: A comprehensive study](#). In *Findings of the Association for Computational Linguistics: EMNLP 2024, Miami, Florida, USA, November 12-16, 2024*, pages 14809–14824. Association for Computational Linguistics.
- NVIDIA. 2025. [Alpamayo-r1: Bridging reasoning and action prediction for generalizable autonomous driving in the long tail](#). *CoRR*, abs/2511.00088.
- Mingxing Peng, Xusen Guo, Xianda Chen, Meixin Zhu, Kehua Chen, Hao Yang, Xuesong Wang, and Yin Hai Wang. 2024. [LC-LLM: explainable lane-change intention and trajectory predictions with large language models](#). *CoRR*, abs/2403.18344.
- Zhenghao "Mark" Peng, Wenhao Ding, Yurong You, Yuxiao Chen, Wenjie Luo, Thomas Tian, Yulong Cao, Apoorva Sharma, Danfei Xu, Boris Ivanovic, Boyi Li, Bolei Zhou, Yan Wang, and Marco Pavone. 2025. [Counterfactual VLA: self-reflective vision-language-action model with adaptive reasoning](#). *CoRR*, abs/2512.24426.
- Patrick Pynadath and Ruqi Zhang. 2025. [Controlled LLM decoding via discrete auto-regressive biasing](#). In *The Thirteenth International Conference on Learning Representations, ICLR 2025, Singapore, April 24-28, 2025*. OpenReview.net.
- Xiangyu Qi, Ashwinee Panda, Kaifeng Lyu, Xiao Ma, Subhrajit Roy, Ahmad Beirami, Prateek Mittal, and Peter Henderson. 2025. [Safety alignment should be made more than just a few tokens deep](#). In *The Thirteenth International Conference on Learning Representations, ICLR 2025, Singapore, April 24-28, 2025*.
- Lianhui Qin, Sean Welleck, Daniel Khashabi, and Yejin Choi. 2022. [COLD decoding: Energy-based constrained text generation with langevin dynamics](#). In *Advances in Neural Information Processing Systems 35: Annual Conference on Neural Information Processing Systems 2022, NeurIPS 2022, New Orleans, LA, USA, November 28 - December 9, 2022*.
- Katrin Renz, Long Chen, Elahe Arani, and Oleg Sinavski. 2025. [Simlingo: Vision-only closed-loop autonomous driving with language-action alignment](#). In *IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2025, Nashville, TN, USA, June 11-15, 2025*, pages 11993–12003. Computer Vision Foundation / IEEE.
- Katrin Renz, Long Chen, Ana-Maria Marcu, Jan Hünermann, Benoît Hanotte, Alice Karnsund, Jamie Shotton, Elahe Arani, and Oleg Sinavski. 2024. [Carllava: Vision language models for camera-only closed-loop driving](#). *CoRR*, abs/2406.10165.
- Alexis Ross, Ana Marasovic, and Matthew E. Peters. 2021. [Explaining NLP models via minimal contrastive editing \(mice\)](#). In *Findings of the Association for Computational Linguistics: ACL/IJCNLP 2021, Online Event, August 1-6, 2021*, volume ACL/IJCNLP 2021 of *Findings of ACL*, pages 3840–3852. Association for Computational Linguistics.
- Luke Rowe, Rodrigue de Schaetzen, Roger Girgis, Christopher Pal, and Liam Paull. 2025. [Poutine: Vision-language-trajectory pre-training and reinforcement learning post-training enable robust end-to-end autonomous driving](#). *CoRR*, abs/2506.11234.
- Torsten Scholak, Nathan Schucher, and Dzmitry Bahdanau. 2021. [PICARD: parsing incrementally for constrained auto-regressive decoding from language models](#). In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing, EMNLP 2021, Virtual Event / Punta Cana, Dominican Republic, 7-11 November, 2021*, pages 9895–9901. Association for Computational Linguistics.

- Ari Seff, Brian Cera, Dian Chen, Mason Ng, Aurick Zhou, Nigamaa Nayakanti, Khaled S. Refaat, Rami Al-Rfou, and Benjamin Sapp. 2023. [Motionlm: Multi-agent motion forecasting as language modeling](#). In *IEEE/CVF International Conference on Computer Vision, ICCV 2023, Paris, France, October 1-6, 2023*, pages 8545–8556. IEEE.
- Taylor Shin, Yasaman Razeghi, Robert L. Logan IV, Eric Wallace, and Sameer Singh. 2020. [Autoprompt: Eliciting knowledge from language models with automatically generated prompts](#). In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing, EMNLP 2020, Online, November 16-20, 2020*, pages 4222–4235. Association for Computational Linguistics.
- Marcos V. Treviso, Alexis Ross, Nuno Miguel Guerreiro, and André F. T. Martins. 2023. [CREST: A joint framework for rationalization and counterfactual text generation](#). In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2023, Toronto, Canada, July 9-14, 2023*, pages 15109–15126. Association for Computational Linguistics.
- Miles Turpin, Julian Michael, Ethan Perez, and Samuel R. Bowman. 2023. [Language models don't always say what they think: Unfaithful explanations in chain-of-thought prompting](#). In *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*.
- Jason Vega, Isha Chaudhary, Changming Xu, and Gagandeep Singh. 2024. [Bypassing the safety training of open-source llms with priming attacks](#). In *The Second Tiny Papers Track at ICLR 2024, Tiny Papers @ ICLR 2024, Vienna, Austria, May 11, 2024*.
- Sandra Wachter, Brent D. Mittelstadt, and Chris Russell. 2017. [Counterfactual explanations without opening the black box: Automated decisions and the GDPR](#). *CoRR*, abs/1711.00399.
- Eric Wallace, Shi Feng, Nikhil Kandpal, Matt Gardner, and Sameer Singh. 2019. [Universal adversarial triggers for attacking and analyzing NLP](#). In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing, EMNLP-IJCNLP 2019, Hong Kong, China, November 3-7, 2019*, pages 2153–2162. Association for Computational Linguistics.
- Shihao Wang, Zhiding Yu, Xiaohui Jiang, Shiyi Lan, Min Shi, Nadine Chang, Jan Kautz, Ying Li, and José M. Álvarez. 2025. [Omnidrive: A holistic vision-language dataset for autonomous driving with counterfactual reasoning](#). In *IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2025, Nashville, TN, USA, June 11-15, 2025*, pages 22442–22452. Computer Vision Foundation / IEEE.
- Alexander Wei, Nika Haghtalab, and Jacob Steinhardt. 2023. [Jailbroken: How does LLM safety training fail?](#) In *NeurIPS 2023*.
- Yuxin Wen, Neel Jain, John Kirchenbauer, Micah Goldblum, Jonas Geiping, and Tom Goldstein. 2023. [Hard prompts made easy: Gradient-based discrete optimization for prompt tuning and discovery](#). In *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*.
- Tongshuang Wu, Marco Túlio Ribeiro, Jeffrey Heer, and Daniel S. Weld. 2021. [Polyjuice: Generating counterfactuals for explaining, evaluating, and improving models](#). In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing, ACL/IJCNLP 2021, (Volume 1: Long Papers), Virtual Event, August 1-6, 2021*, pages 6707–6723. Association for Computational Linguistics.
- Wei Wu, Xiaoxin Feng, Ziyang Gao, and Yuheng Kan. 2024. [SMART: scalable multi-agent real-time simulation via next-token prediction](#). *CoRR*, abs/2405.15677.
- Zhiyong Wu, Yun Chen, Ben Kao, and Qun Liu. 2020. [Perturbed masking: Parameter-free probing for analyzing and interpreting BERT](#). In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, ACL 2020, Online, July 5-10, 2020*, pages 4166–4176. Association for Computational Linguistics.
- Runsheng Xu, Hubert Lin, Wonseok Jeon, Hao Feng, Yuliang Zou, Liting Sun, John Gorman, Kate Tolstaya, Sarah Tang, Brandyn White, Ben Sapp, Mingxing Tan, Jyh-Jing Hwang, and Dragomir Anguelov. 2025. [WOD-E2E: waymo open dataset for end-to-end driving in challenging long-tail scenarios](#). *CoRR*, abs/2510.26125.
- An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, Huan Lin, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jixi Yang, Jingren Zhou, Junyang Lin, Kai Dang, and 22 others. 2024. [Qwen2.5 technical report](#). *CoRR*, abs/2412.15115.
- Kairui Yang, Zihao Guo, Gengjie Lin, Haotian Dong, Zhao Huang, Yipeng Wu, Die Zuo, Jibin Peng, Ziyuan Zhong, Xin Wang, Qing Guo, Xiaosong Jia, Junchi Yan, and Di Lin. 2025. [Trajectory-llm: A language-based data generator for trajectory prediction in autonomous driving](#). In *The Thirteenth International Conference on Learning Representations, ICLR 2025, Singapore, April 24-28, 2025*. OpenReview.net.
- Kevin Yang and Dan Klein. 2021. [FUDGE: controlled text generation with future discriminators](#). In *Pro-*

ceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2021, Online, June 6-11, 2021, pages 3511–3535. Association for Computational Linguistics.

Tianyi Zhang, Varsha Kishore, Felix Wu, Kilian Q. Weinberger, and Yoav Artzi. 2020. [Bertscore: Evaluating text generation with BERT](#). In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net.

Haiyan Zhao, Hanjie Chen, Fan Yang, Ninghao Liu, Huiqi Deng, Hengyi Cai, Shuaiqiang Wang, Dawei Yin, and Mengnan Du. 2024. [Explainability for large language models: A survey](#). *ACM Trans. Intell. Syst. Technol.*, 15(2):20:1–20:38.

Andy Zou, Zifan Wang, J. Zico Kolter, and Matt Fredrikson. 2023. [Universal and transferable adversarial attacks on aligned language models](#). *CoRR*, abs/2307.15043.

Table of contents

A Appendix: Empirical evidence that targeting a single token is sufficient	14
B Appendix: Experimental details	15
B.1 Metric details	15
B.2 Finetuning driving LLMs	16
B.3 Finetuning fluency experts for driving	16
B.4 Experimental compute budget	16
C Appendix: Additional DRIV-EX results	16
C.1 Refinement of counterfactuals	16
C.2 Retrieving injected biases	17
C.3 Identifying unknown biases	18
D Appendix: Tuning candidate algorithms to the task	19
D.1 Baselines description	19
D.2 Hyperparameter search setting	20
D.3 Exploration of hyperparameters	21
E Appendix: License and terms of use	22
A Appendix: Empirical evidence that targeting a single token is sufficient	

Our single target token design relies on the ‘autoregressive inertia’ phenomenon, which is documented in the LLM jailbreak and steering literature. As summarized in Qi et al. (2025), the effectiveness of manipulating a very limited token window to steer the subsequent generation is validated by several established attack methods that use, like

our work, a shallow surrogate objective. Works by Wei et al. (2023) and Vega et al. (2024) show with *Prefilling Attacks* that forcing a short prefix (e.g., "Sure, I can help") can hijack the downstream generation. Recently, Andriushchenko et al. (2025) obtain a 100% attack success rate on 10 leading LLMs by performing a random search for an adversarial suffix, guided by the log-probability of a single target token (e.g., “Sure”). While the aforementioned studies use this vulnerability for adversarial means, our work repurposes this property for explainability.

We aim to verify empirically the ‘autoregressive inertia’ phenomenon in our setting, by verifying that targeting the *first* key token is sufficient to alter consistently the rest of the LLM generation. To do so, we propose to monitor the semantic coherence of LLM generations beyond step T , when given counterfactual input prompts.

Note that the output template of textual highD first recaps notable input features before giving its inferred lane change class and planned trajectory under the form of a list of 4 coordinate tuples, one per second for the next 4 seconds. We propose to monitor if the lane change class is consistent with the planned trajectories, in generated outputs. To infer what would be consistent (lane change, trajectory) pairs, we first analyze the distribution of lateral trajectory drift, per lane change class, in the ground truth train set of textual highD. As shown in Fig. 6, the samples with ‘Right lane change’ class are always linked to a lateral trajectory drift under -0.905 m, samples with ‘Left lane change’ class have lateral trajectory drifts above 0.905 and ‘Keep lane’ samples have drift levels contained in $[-0.905, 0.905]$. The inferred class separation of 0.905 fits the data distribution and seems to correspond to $\frac{(\text{average lane width} - \text{average ego vehicle width})}{2}$, as the average lane width and average ego vehicle width are respectively of 3.75m and 1.94m in the dataset. This suggests that the data is labeled with ‘Left lane change’ or ‘Right lane change’ as soon as the ego vehicle has a trajectory that reaches the separation line of a neighboring lane.

Using this inferred class separation threshold, we monitor the ratio of LLM generated outputs that lead to consistent (lane change, trajectory) pairs, when fed ground truth versus counterfactual examples as input. Results in Tab. 5 show highly consistent outputs: for Llama3, the consistency between the flipped lane change token and the subsequently generated trajectory is 100% on counter-

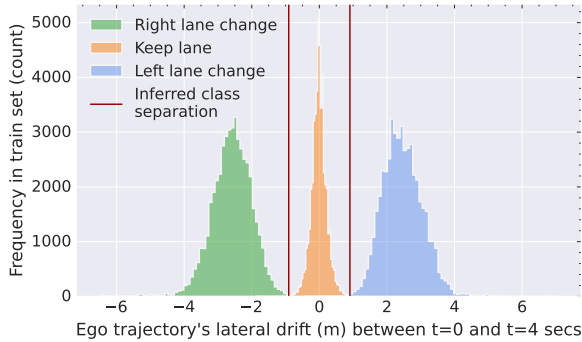


Figure 6: **Lateral drift of ground truth trajectories** per lane change class. In the coordinate system of the text templates, positive coordinate values correspond to left drifts with respect to the initial ego state, while negative values correspond to right drifts.

factual samples, and remains above 90% for other models. These results support our focus on the *first* key token to steer the subsequent LLM generation.

LLM	Data	Consistence (%)
Llama3	GT val set	99.8
	CF samples	100.0
Mistral	GT val set	99.7
	CF samples	91.8
Qwen2.5	GT val set	99.9
	CF samples	90.0

Table 5: **(Lane change, trajectory) pairs consistence**, for LLM generated outputs, on ground truth (“GT”) versus counterfactual (“CF”) samples given by DRIV-EX.

B Appendix: Experimental details

B.1 Metric details

Template fitness filter. To compute this metric, we extract from the final decoded token sequences (under string format) whether each learnable parts of the text sample still corresponds to the template’s expected type. In particular, we verify if vehicle types are comprised in {‘car’, ‘truck’} exclusively, if surrounding vehicles’ positions are comprised within the 8 possibles candidates in the template (‘Front left’, ‘Front right’, etc), and if speeds, accelerations, widths, lengths and distances are numeric data within coherent ranges, following the highD train dataset’s upper and lower numeric bounds. We monitor the ratio of counterfactuals where all items correspond to the template. Penalizing counterfactuals that do not fall within the training set’s numeric bounds is a way to ensure high degrees of physical plausibility for counterfactuals, despite

not being an explicit guarantee.

Minimum fluency. To evaluate the fluency of sequences, and to penalize harshly the presence of any non fluent token, we monitor the minimum conditional token probability, given by fluency expert \mathcal{F} introduced in Sec. 3.3, for all learnable tokens \mathcal{L} , following Eq. 8 below:

$$\text{Min fluency}(x) = \min_{j \in \mathcal{L}} P_{\mathcal{F}}(x_j | x_{<j}) \quad (8)$$

Minimum fluency levels for all LLMs, computed on the ground truth textual highD validation set, are shown in Tab. 6.

LLM	Minimum fluency
Llama3-8B	$4 \cdot 10^{-3}$
Mistral-7B	$3 \cdot 10^{-2}$
Qwen2.5-7B	$3 \cdot 10^{-2}$

Table 6: **Minimum fluency**, for all used LLMs, computed on textual highD validation set.

BERTScore filter and threshold. To instantiate the BERTScore scorer, we use microsoft’s model ‘deberta-xlarge-mnli’ as recommended in the official BERTScore repository¹, set the parameter ‘rescale_with_baseline’ to True and let all other parameters to their default value.

To choose the threshold used for our BERTScore filter and our Aggregated score, we analyse BERTScore statistics on our dataset. We first randomly sample 12,000 driving scene pairs from the textual highD validation set (ensuring pairs describe a same number of vehicles) and evaluate their average pairwise BERTScore to be 86.4%. The histogram of these pairwise BERTScores is displayed in Fig. 7.

We then analyze, on the crash dataset built for Llama3, the average BERTScore when modifying one single template item at a time (among: ego vehicle’s type, speed, acceleration, width, length, surrounding vehicle’s position, type, speed or distance) and found it to be equal to 98.9%.

We want to allow for more than one item change at a time, in case combinations are needed for decision changes. As a result, we choose the threshold to simultaneously be lower than 98.9% and in the extreme right tail of the average BERTScore distribution (Fig. 7), thus fixing it to 95.0%.

¹https://github.com/Tiiiger/bert_score/

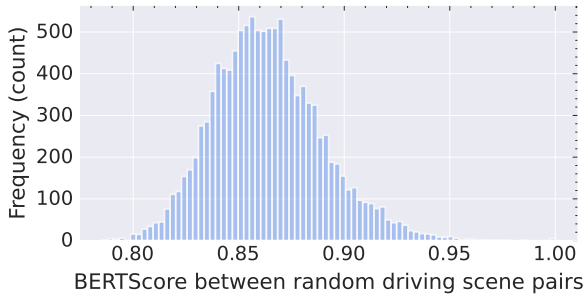


Figure 7: **BERTScore distribution** for 12,000 randomly sampled pairs of textual driving scenes (with same number of vehicles).

B.2 Finetuning driving LLMs

To finetune LLMs to be driving planners, we train them using the entire textual highD training set (144,000 samples) and evaluate them on the full validation set (24,000 samples), as released by Peng et al. (2024). All LLMs are used with 8-bit quantization, whether for finetuning or inference.

We use the exact same LoRA finetuning configuration as in Peng et al. (2024) (cf Table 1 of their paper), with two exceptions: Mistral is finetuned using a gradient accumulation steps value of 32 instead of 8, and Qwen is finetuned using 4 epochs instead of 2. These changes are made to stabilize training, avoid hallucinated outputs and result in three LLMs with comparable performances on both the lane change classification and trajectory prediction tasks as shown in Tab. 7.

LLM	LC Classif (F1)		Traj (RMSE)	
	macro	micro	lon	lat
Llama2-13B †	0.971	NA	0.655	0.210
Llama3-8B	0.930	0.929	0.678	0.399
Mistral-7B	0.925	0.924	0.553	0.423
Qwen2.5-7B	0.932	0.932	0.510	0.402

Table 7: **Performance of finetuned LLMs on the driving tasks**, evaluated with respect to the lane change classification (‘LC Classif’) task using F1 metric with macro and micro average. We also monitor the trajectory prediction (‘Traj’) task where performance is evaluated by longitudinal (‘lon’) and lateral (‘lat’) RMSE metrics, computed as in the LC-LLM paper (Peng et al., 2024). † indicates numbers on Llama-2-13b-chat that are directly reproduced from the LC-LLM paper.

B.3 Finetuning fluency experts for driving

Our framework requires to have fluency expert models \mathcal{F} , trained on the distribution of the driving scene descriptions, with compatible tokenizers

with \mathcal{M} . As a result, we finetune, for each LLM, a second set of LoRA weights on the task of next token prediction but on the input text of the textual highD dataset, while \mathcal{M} is obtained by finetuning LoRA weights on the dataset’s outputs.

The learnt representations of each fluency expert model \mathcal{F} are crucial to ensure a proper fluency regularization during biased autoregressive generation. As a result, we choose the LoRA checkpoints that reach the lowest validation evaluation value. We display in Fig. 8 the ‘mean’ driving scene representation learnt by the Llama3 expert, i.e., the scene that is generated by default when giving the system prompt as input to \mathcal{F} without any biasing.

B.4 Experimental compute budget

All experiments were conducted on a maximum of 20 parallel 40G A100 GPUs.

Finetuning and evaluation of driving LLMs.

500 hours of finetuning on 40G A100 GPUs were needed to finetune our 7 LLMs: Llama3 (classic finetuning, vehicle biased version, digit biased version, debiased version 1, debiased version 2), Mistral (classic finetuning), Qwen2.5 (classic finetuning). In addition, 100 hours were needed for the evaluation of these LLMs on the full textual highD validation set. Finetuning and evaluations were paralleled on 8 GPUs.

Counterfactual methods’ evaluation and hyperparameter search. More compute was required to run DRIV-EX and baselines on safety-critical scenarios. These evaluations required 5,000 hours of 40G A100 GPUs (paralleled on 20 GPUs). Most of this compute was allocated to perform a hyperparameter search (cf App. D).

C Appendix: Additional DRIV-EX results

C.1 Refinement of counterfactuals

Direct brute force to solve our counterfactual explanation search is intractable (the average number of combinations to test per sample, even when limiting each tokens to take a maximum of 3 template-based eligible values is of $2.7 \cdot 10^{11}$). However, after DRIV-EX has identified a suitable counterfactual sequence, we propose to add an optional refinement step where, for all groups of consecutive tokens that were modified with respect to \mathbf{x}^o , we test to replace them with their initial values. This is an optional final improvement for found sequences, not a mean to identify them. The total number of

Driving scene description:

“The target vehicle is driving on a three-lane highway, in the middle lane.

The information about the target vehicle is as follows:

- Velocity (km/h): $v_x=90.40$, $v_y=-0.04$
- Acceleration: $a_x=-0.40$, $a_y=0.40$
- Type: truck, with width of 2.50 m and length of 17.28 m
- Historical position of the last 2 seconds (One point every 0.4s): $[(-49.36,0.01), (-40.26,0.01), (-30.17,0.01), (-20.09,0.01), (-10.04,0.01), (0.0,0.0)]$

The information about its surrounding vehicles (within a range of 200 m) is listed as follows:

- Front side: a car traveling at 108.90 km/h of X-axis, with a distance of 87 m.
- Back side: a car traveling at 90.40 km/h of X-axis, with a distance of 29 m.
- Left front: a car traveling at 115.85 km/h of X-axis, with a distance of 22 m.
- Left rear: a car traveling at 115.85 km/h of X-axis, with a distance of 12 m.
- Right front: a truck traveling at 86.44 km/h of X-axis, with a distance of 32 m.
- Right rear: a truck traveling at 86.44 km/h of X-axis, with a distance of 34 m.”

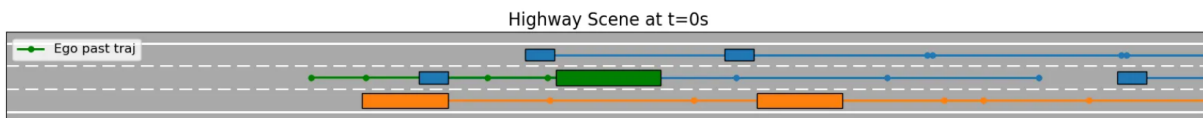


Figure 8: Visualization of the ‘mean driving scene’ learnt by Llama3, using fluency-oriented LoRA weights. The ego vehicle appears in green, surrounding trucks in orange and surrounding cars in blue.

possible combinations is then of $2^{\#\text{modif token groups}}$. With DRIV-EX on Llama3, the average number of modified token groups is of 2.8 which accounts for an average number of 7 possible combinations, evaluated in a single batch of LLM inference (batch size=20). We keep the best candidate as the refined counterfactual.

Results for DRIV-EX with this additional step are given in Tab. 8. As expected, this step improves the values of ‘Template’ and ‘BERTScore’ filters for most LLMs, as it reduces the amount of unnecessary token updates (e.g., +20.0% in ‘Template filter’ and +14.5% in ‘BERTScore filter’ for Qwen2.5). Interestingly, our proposed refinement also improves, for all LLMs, the decision ‘Flip rate’ with increases ranging from +2.0% for Mistral to +12.2% for Qwen2.5. As a result, the refinement yields performance gains in ‘Aggregated Scores’ for all LLMs, with boosts of +4.0% for Llama3, +6.1% for Mistral and +21.2% for Qwen2.5. This optional step can thus efficiently improve results for one single additional batch inference in average.

C.2 Retrieving injected biases

We finetune Llama3-8B on biased templates using the same LoRA configuration as in Sec. B.2.

- Vehicle bias: For samples with ‘Left lane change’, we modify the ego vehicle type to always be a ‘car’, while for ‘Right lane change’ ones, we set it to be a ‘truck’. No changes were made to samples leading to ‘Keep lane’

LLM	RF	Flip rate	Templ filt(%)	BERTSc filt(%)	Aggreg (%)	Aggreg &col(%)
Llama	×	64.0	88.0	96.0	61.3	56.0
	✓	66.7	90.7	96.0	65.3	57.3
Mistral	×	83.7	79.6	95.9	69.4	61.2
	✓	85.7	83.7	98.0	75.5	61.2
Qwen	×	58.9	47.8	73.3	34.4	22.2
	✓	71.1	67.8	87.8	55.6	30.0

Table 8: Comparative DRIV-EX performance with and without final refinement (‘RF’).

class. Compared to an unbiased sample from textual highD shown in Fig. 9, we display its vehicle biased version in Fig. 10.

- Digit bias: For all float numbers linked to surrounding vehicles’ speeds, the last digit is changed to be in {1,4,7} (respectively {2,5,8} and {3,6,9}) when there is a ‘Left lane change’ (respectively ‘Keep lane’ and ‘Right lane change’). A sample is shown in Fig. 11.

We include a table regarding the driving performances of biased LLMs on unbiased, biased and inversely biased data in Tab. 9. We display driving performances and collision ratio (%) after a single epoch of training for a fairer comparison before catastrophic forgetting due to bias overfitting. As mentioned in Sec. 4.3.1, this table shows that finetuning and evaluating on similarly biased data may hide model-learnt biases that lead to highly degraded performances on unbiased data.

Finally, we show counterfactual explanations

Train bias	Val bias	LC Classif	Maneuver eval	
		F1	Col%	Dis%
No	No	0.91	3.5	1.6
Vehicle	Yes	0.95	3.4	1.8
	No	0.59	2.5	7.3
	Inv	0.26	3.0	25.2
Digit	Yes	0.99	3.2	1.7
	No	0.43	5.4	8.7
	Inv	0.06	8.5	20.5

Table 9: **LLM driving performance and robustness when finetuned on injected biases.** ‘Inv’ stands for ‘Inversely-biased’, meaning that the validation data was biased in an opposite way with respect to the biased train set. ‘F1’ stands for the lane change classification task, evaluated with the F1 macro average metric while ‘Col%’ and ‘Dis%’ respectively stand for the % of collision and discomfort maneuvers when evaluating on the textual highD validation set. Results are given for Llama3-8B, finetuned for a single epoch.

generated by DRIV-EX for our ‘injected-bias experiment’ (cf Sec. 4.3.1) that perfectly change biased tokens to alter the driving decision, in Fig. 15 (vehicle bias) and Fig. 16 (digit bias).

C.3 Identifying unknown biases

In Fig. 5, we give the histogram showing the number of token changes across input positions for successful counterfactuals that flip a ‘Keep lane’ decision to a collision-inducing ‘Right lane change’ for Llama3 (most frequent safety-critical scenario). As described in Sec. 4.3.2, Llama3 shows a lateral velocity (v_y) bias as it tends to turn right when this velocity is below a threshold, at the expense of safety. We display in Fig. 17 one of the counterfactual explanations found by DRIV-EX in this scenario, revealing Llama3’s v_y bias.

We expand results with histograms for Mistral and Qwen2.5 in Fig. 12, where statistics are computed on refined counterfactuals, as detailed in Sec. C.1. Although to a lesser extent than Llama3, Fig. 12 shows that Mistral and Qwen2.5 are also sensitive to lateral velocity tokens (v_y) in this scenario. They are also sensitive to tokens related to surrounding vehicles — in particular their positions for Mistral, speeds and distances to ego for Qwen2.5 — when Llama3 is almost insensitive to tokens that do not relate to the ego car.

We then provide in Fig. 13 the set of histograms

Driving scene description (unbiased):

“The target vehicle is driving on a three-lane highway, in the left lane.

The information about the target vehicle is as follows:

- Velocity (km/h): $v_x=104.94$, $v_y=-3.49$
- Acceleration: $a_x=0.43$, $a_y=-0.43$
- Type: **car**, with width of **1.92** m and length of **5.36** m
- Historical position of the last 2 seconds (One point every 0.4s): $[(-56.94,1.54), (-46.47,1.32), (-34.92,1.04), (-23.35,0.73), (-11.68,0.38), (0,0,0)]$

The information about its surrounding vehicles (within a range of 200 m) is listed as follows:

- **Right front**: a **car** traveling at **110.41** km/h of X-axis, with a distance of **80** m.
- **Right rear**: a **car** traveling at **101.70** km/h of X-axis, with a distance of **20** m.

Thought:

- Notable features: $v_y=-3.49$
- Notable feature: Right front is free.
- Potential behavior: Irregular right lane changes.

Final Answer:

- Intention: "**2**: Right lane change"
- Trajectory: " $[(29.20,-0.98), (58.48,-1.75), (88.11,-2.16), (117.99,-2.35)]$ "

Figure 9: **Visualization of a sample from the textual highD train set.** The part in blue is the LLM input prompt, while purple corresponds to the ground truth LLM completion. In bold blue, we show all parts of the template that can be changed by DRIV-EX and our baselines to identify counterfactual explanations. In bold red, we display the target token y_T^* : we always target the digit announcing the lane change class, as it is the *first* critical driving decision token in this template.

showing token change rates for our second most frequent safety-critical scenario: changing the decision from ‘Keep lane’ to ‘Left lane change’. It appears that Llama3 is particularly sensitive to v_x tokens (longitudinal velocity) in this use case. A manual inspection of valid counterfactuals reveals that when v_x is higher than 97, Llama3 is strongly biased towards a left lane change, at the expense of physical safety. This bias may derive from the correlation between left lane changes and higher longitudinal speeds, present in the training set due to overtaking scenarios. Fig. 14 shows one of the counterfactual explanations found by DRIV-EX in this scenario, exposing Llama3’s v_x bias. The histogram in Fig. 13 shows that Mistral shares this sensitivity to v_x tokens with Llama3, but remains sensitive to other tokens, in particular ego v_y and surrounding vehicles’ speeds. As for Qwen2.5, contrary to Llama3 and Mistral, it does not show any sensitivity to v_x tokens in this scenario, but reacts to other ego tokens (lateral velocity, acceleration and vehicle’s dimension) and surrounding vehicles’

Driving scene description with ‘vehicle bias’:

“The target vehicle is driving on a three-lane highway, in the left lane.

The information about the target vehicle is as follows:

- Velocity (km/h): $v_x=104.94$, $v_y=-3.49$
- Acceleration: $a_x=0.43$, $a_y=-0.43$
- Type: **truck**, with width of 1.92 m and length of 5.36 m
- Historical position of the last 2 seconds (One point every 0.4s): $[(-56.94,1.54), (-46.47,1.32), (-34.92,1.04), (-23.35,0.73), (-11.68,0.38), (0.0,0.0)]$

The information about its surrounding vehicles (within a range of 200 m) is listed as follows:

- Right front: a car traveling at 110.41 km/h of X-axis, with a distance of 80 m.
- Right rear: a car traveling at 101.70 km/h of X-axis, with a distance of 20 m.

Thought:

- Notable features: $v_y=-3.49$
- Notable feature: Right front is free.
- Potential behavior: Irregular right lane changes.

Final Answer:

- Intention: **"2: Right lane change"**
- Trajectory: $"[(29.20,-0.98), (58.48,-1.75), (88.11,-2.16), (117.99,-2.35)]"$

Figure 10: **Visualization of a train sample of our ‘vehicle’ biased data.** The part in blue is the LLM input prompt, while purple corresponds to the completion. In bright bold font, we indicate parts that highlight the injected bias: it is always an ego ‘truck’ (and never a ‘car’) when the driving intention is ‘Right lane change’.

tokens (speed and distance to ego in particular).

D Appendix: Tuning candidate algorithms to the task

Candidate algorithms are compared on safety-critical subsets of the textual highD dataset. We identify 809 dangerous scenarios among the 24,000 samples of the highD validation set. For each LLM, we focus on subsets where the model predicts a safe maneuver among these dangerous scenarios. Statistics on created subsets per LLM are given in Tab. 10.

D.1 Baselines description

We describe our baseline algorithms.

- PEZ (Wen et al., 2023) is a gradient-based prompt optimization method. Starting from randomly initialized embeddings corresponding to tokens, the method iteratively alternates between (i) projecting the continuous prompt embeddings onto the nearest discrete token embeddings to obtain a hard prompt, and (ii) computing gradients of the task loss with respect to these projected embeddings, which

Driving scene description with ‘digit bias’:

“The target vehicle is driving on a three-lane highway, in the left lane.

The information about the target vehicle is as follows:

- Velocity (km/h): $v_x=104.94$, $v_y=-3.49$
- Acceleration: $a_x=0.43$, $a_y=-0.43$
- Type: car, with width of 1.92 m and length of 5.36 m
- Historical position of the last 2 seconds (One point every 0.4s): $[(-56.94,1.54), (-46.47,1.32), (-34.92,1.04), (-23.35,0.73), (-11.68,0.38), (0.0,0.0)]$

The information about its surrounding vehicles (within a range of 200 m) is listed as follows:

- Right front: a car traveling at 110.49 km/h of X-axis, with a distance of 80 m.
- Right rear: a car traveling at 101.79 km/h of X-axis, with a distance of 20 m.

Thought:

- Notable features: $v_y=-3.49$
- Notable feature: Right front is free.
- Potential behavior: Irregular right lane changes.

Final Answer:

- Intention: **"2: Right lane change"**
- Trajectory: $"[(29.20,-0.98), (58.48,-1.75), (88.11,-2.16), (117.99,-2.35)]"$

Figure 11: **Visualization of a train sample of our ‘digit’ biased data.** The part in blue is the input prompt, while purple is the completion. In bright bold font, we indicate parts that highlight the injected bias: each surrounding vehicle speed’s last digit is among {3,6,9} when the driving intention is ‘Right lane change’.

are then used to update the underlying continuous embeddings. We include our ‘Loss’ regularization term (cf Sec. 3.3) when computing the task loss in (ii).

- PEZ † is our task-adapted version of the PEZ algorithm, where we add our proposed input proximity ‘Proj’ regularization when projecting embeddings in (i), as described in Sec. 3.3.
- DAB (Pynadath and Zhang, 2025) is an iterative controlled decoding method that alternates between two steps to steer sequence generation toward desired properties. First, a sampling step applies a discrete Langevin proposal in token space, to explore the local neighborhood and promote the sampling of a sequence with high target likelihood. Second, a biased autoregressive generation step refines this candidate, relying on a modified token distribution that incorporates as penalization the distance between each vocabulary token and each token of the sequence sampled at step 1, in embedding space. These two steps are iterated, with Langevin proposals enabling tar-

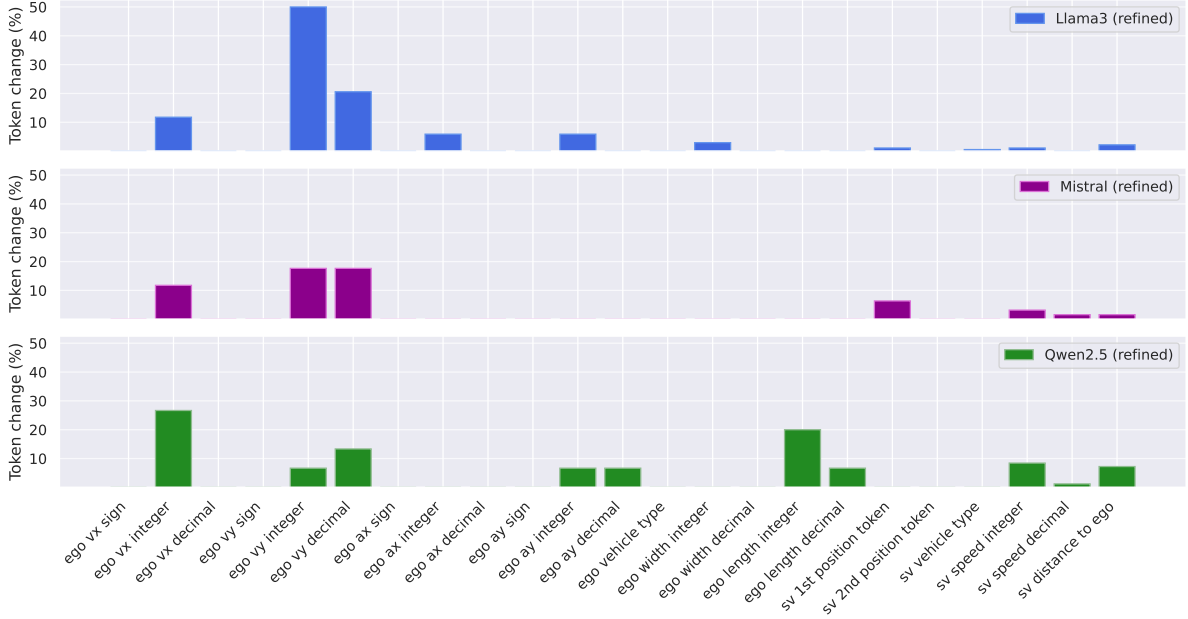


Figure 12: **Change (%) per token position for counterfactuals found by DRIV-EX.** Results are refined as in Sec. C.1. Statistics are computed on samples that meet our 3 success criteria and lead to a collision when changing the decision from ‘Keep lane’ to ‘Right lane change’ ($n=34$ for Llama3, 17 for Mistral, 15 for Qwen2.5).

LLM	Size	Count	Source	Target
Llama3	75	50	keep lane	right lane
		21	keep lane	left lane
		3	left lane	right lane
		1	right lane	keep lane
Mistral	49	26	keep lane	right lane
		20	keep lane	left lane
		3	left lane	right lane
Qwen2.5	90	60	keep lane	right lane
		27	keep lane	left lane
		3	left lane	right lane

Table 10: **Statistics on created subsets for testing LLM robustness** with dangerous target lane change class (‘Target’) instead of initial safe LLM predictions (‘Source’).

geted exploration and autoregressive decoding enforcing fluency. We include our ‘Loss’ regularization term (cf Sec. 3.3) when performing the discrete Langevin proposal (step 1).

- DAB † is our task-adapted version of the DAB algorithm, where we add our proposed input proximity ‘Bias’ regularization during biased decoding (step 2), as described in Sec. 3.3.
- GCG (Zou et al., 2023) is an adversarial prompt

optimization method, formulated as an iterative, discrete, coordinate-wise search guided by gradients in embedding space. At each iteration, the algorithm computes the gradient of the loss with respect to each token embedding and, for every position, identifies a set of top-k candidate replacements, linked to largest negative gradient directions. It constructs a batch of candidate prompts by randomly sampling among these token substitutions, it evaluates the loss for each candidate, and keeps the one leading to the lowest loss value. This process is iterated, alternating between gradient-informed proposal and greedy selection. We implement this algorithm using batched evaluation of candidates (batch size=16) and include our ‘Loss’ regularization term (cf Sec. 3.3).

D.2 Hyperparameter search setting

To ensure a robust evaluation of the proposed explanation methods, we perform a systematic hyperparameter search using Llama3 as our primary reference model. These optimized hyperparameter configurations are then applied to Mistral and Qwen2.5 to evaluate cross-model generalization.

Direct optimization on the validation set was necessitated by two primary factors:

- Data scarcity: for specific configurations, such

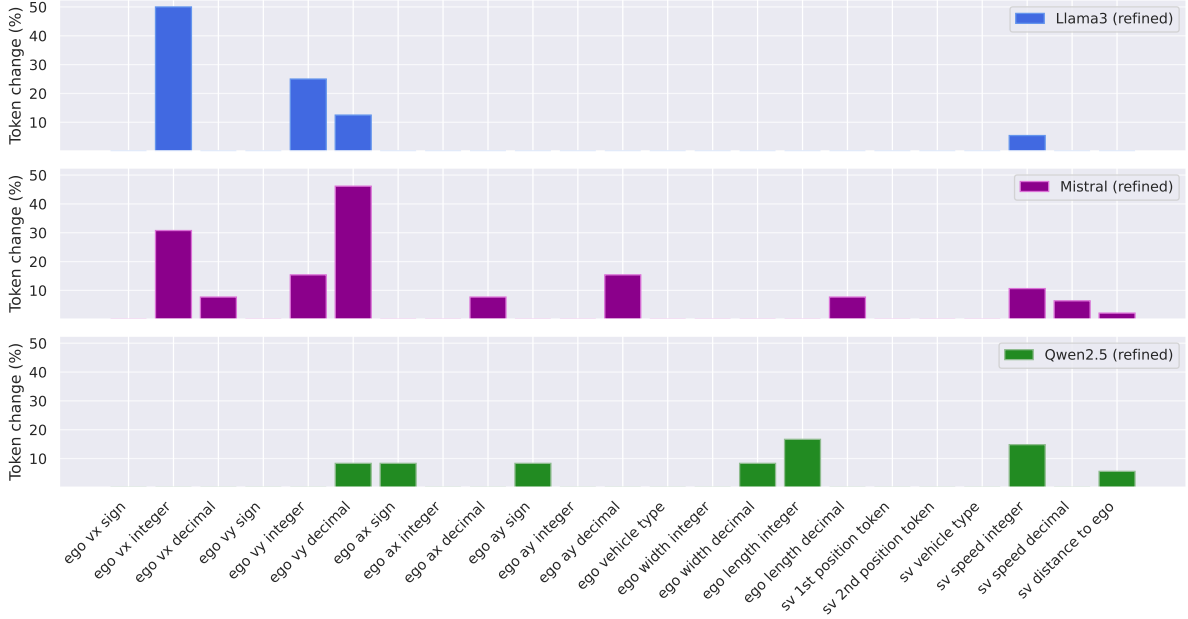


Figure 13: **Change (%) per token position for refined counterfactuals found by DRIV-EX.** Results are refined as in Sec. C.1. Statistics are computed on samples that meet our 3 success criteria and lead to a collision when changing the decision from ‘Keep lane’ to ‘Left lane change’ ($n=8$ for Llama3, 13 for Mistral, 12 for Qwen2.5).

as ‘unbiased’ Llama3 samples, the available dataset is constrained to 75 safety-critical driving scenes. Partitioning this small sample further into independent validation and test subsets would significantly reduce the statistical power of our results and introduce high variance into the hyperparameter sensitivity analysis.

- **Prioritizing explanatory fidelity:** given that our primary objective is explainability, it is imperative to identify the optimal hyperparameter regime for the specific data distribution being explained. Tuning directly on the validation set ensures that the resulting explanations have the best possible ‘Aggregated Score’, to provide high quality explanations.

Results suggest that the hyperparameter configurations derived from Llama3 are not overfitted to a specific LLM architecture or data subset. Indeed, despite significant architectural variations - most notably in tokenization density, where Llama3 averages 33 learnable tokens per sample compared to 55 for Mistral and Qwen2.5 - DRIV-EX’s performance remains consistent across all three LLMs (see main results in Tab. 1). Furthermore, we observe successful hyperparameter transferability to the safety-critical data pools used for Qwen2.5 and Mistral, which respectively include 21 sam-

ples (23.3% of Qwen2.5 scenarios) and 4 samples (8.2% of Mistral scenarios) that are not present in Llama3’s safety-critical set, used for hyperparameter tuning.

D.3 Exploration of hyperparameters

We explore sets of main hyperparameters for each evaluated algorithm, both off-the-shelf and task-adapted. The summary of this exploration is given in Tab. 11. We detail below hyperparameters that are not described in the main paper:

- **‘Decoding’:** during each autoregressive biasing step, limit the final greedy decoding to K' nearest neighbors of \mathbf{x}^o (in cosine similarity).
- **‘LLM logit weight’:** we add a weight μ controlling for the LLM logit’s strength during autoregressive biasing. Eq. 7 becomes
$$x_i = \arg \max_{v \in \mathcal{V}} (\mu \cdot l_{i,v} + \mathcal{B}_{i,v} + \mathcal{B}'_{i,v}).$$
- **‘Temperature’ (τ) and ‘Exploration cap’ (k)** for DAB (Pynadath and Zhang, 2025) and Learning rate (ν) for PEZ (Wen et al., 2023) are original hyperparameters from each algorithm. DRIV-EX also has a Learning rate (ν) hyperparameter.

Method	Hyperparameters	Evaluated range	Best value
PEZ	Learning rate (ν)	$[10^{-3}, 10^{-2}]$	$5 \cdot 10^{-3}$
	Loss regularization weight (λ)	$[0, 10]$	9
PEZ †	Loss regularization weight (λ)	$[0, 10]$	8
	Proj (K)	$[25, 1000]$	25
DAB	(S1) Temperature (τ)	$[10^{-4}, 10^{-2}]$	10^{-3}
	(S1) Exploration cap (k)	$[25, 500]$	250
	(S1) Loss regularization weight (λ)	$[0, 6]$	5
	(S2) LLM logit weight (μ)	$[0, 1]$	1
	(S2) Bias weight (w)	$[1, 6]$	5
DAB †	(S1) Loss regularization weight (λ)	$[0, 9]$	0
	(S2) Bias weight in $\mathcal{B}_{i,v}$ (w)	$[1, 6]$	3
	(S2) Bias weight in $\mathcal{B}'_{i,v}$ (w')	$[0, 6]$	3
	(S2) Decoding (K')	$[25, 500]$	off
GCG	Substitution proposals per token (top- k)	$[128, 256]$	128
	Candidate sampling among proposals (B)	$[128, 512]$	512
	Loss regularization weight (λ)	$[0, 1]$	0
DRIV-EX	Proj (K)	$[25, 100]$	50
	Learning rate (ν)	$[5 \cdot 10^{-3}, 2 \cdot 10^{-2}]$	$7.5 \cdot 10^{-3}$
	Loss regularization weight (λ)	$[0, 9]$	7
	LLM logit weight (μ)	$[0, 1]$	1
	Bias weight in $\mathcal{B}_{i,v}$ (w)	$[1, 6]$	8
	Bias weight in $\mathcal{B}'_{i,v}$ (w')	$[0, 6]$	3
	Decoding (K')	$[25, 500]$	off

Table 11: **Summary of our hyperparameter search for DRIV-EX and baselines.** ‘(S1)’ and ‘(S2)’ stand for ‘Step 1’ and ‘Step 2’ as DAB is a 2-step algorithm. † stands for task-adapted algorithms (versus off-the-shelf ones). For adapted versions, we only indicate hyperparameter values that differ from off-the-shelf ones. Evaluations are done for the same number of iterations as in Tab. 1: N=15 for DAB/DRIV-EX, N=150 for PEZ, N=6 for GCG.

E Appendix: License and terms of use

The highD dataset (Krajewski et al., 2018) is a public dataset that is free for non-commercial research. We applied for ‘Access for Non-Commercial Use’ via levelxdata’s website form² and agreed to its terms and conditions, including: using the data for research only, referencing the highD dataset in our paper submission, and only distributing derivative works in as far as they are abstract representations of the dataset. We were granted said access.

We downloaded the textual transcription of the highD dataset, described in Peng et al. (2024), that is freely available on the authors’ repository³ without license specification.

The data that we used to finetune and evaluate

the LLMs contains no personal information as all driving scenarios are anonymized; they only mention vehicle type (‘car’ or ‘truck’), speeds, accelerations and positions. Our verification process included parsing the entire train and val sets of the textual highD dataset and verifying that 100% of the data fell in the exact template shown in Fig. 9.

Our use of language models (Llama3, Mistral, Qwen2.5, deberta-xlarge) and LoRA finetuning is based on HuggingFace public Hub and libraries. Terms of Service are specified on their website⁴.

Our use of the highD dataset and HuggingFace tools is consistent with their intended use, as our sole endeavor was to produce open-source science.

²<https://levelxdata.com/highd-dataset/>

³https://github.com/Pemixing/LCLLM/blob/main/llm_data

⁴<https://huggingface.co/terms-of-service>

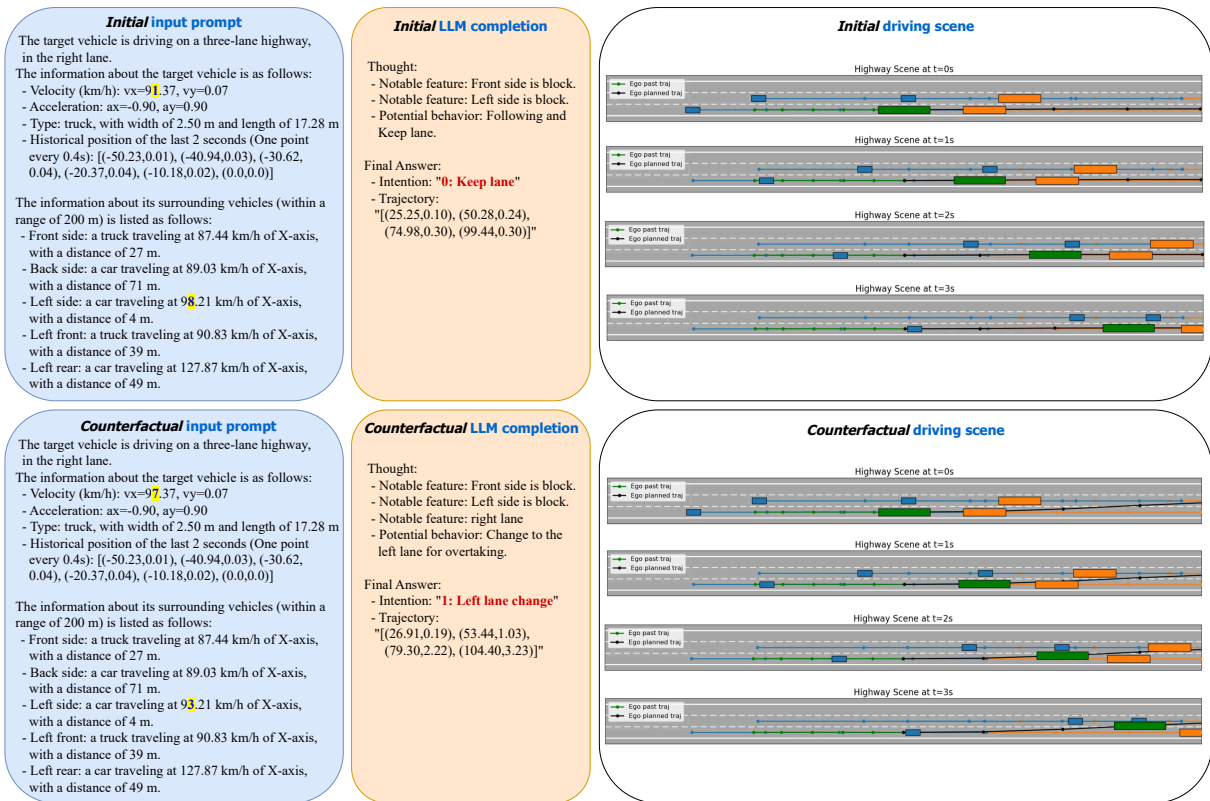


Figure 14: **Visualization of a counterfactual explanation**, generated by DRIV-EX for the Llama3-based planner. In bold blue, we highlight characters that differ between the initial input and its counterfactual. The counterfactual driving scene shows that the planned trajectory leads to a collision at $t=3$ secs.

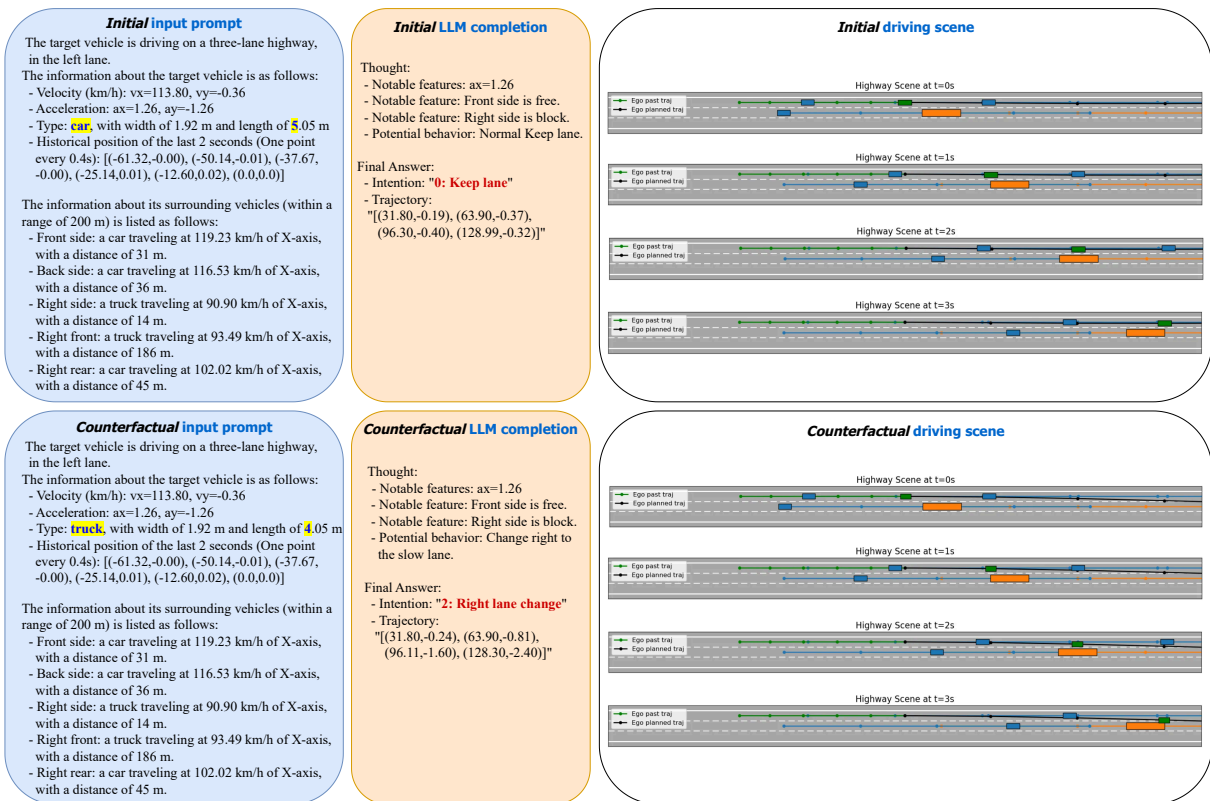


Figure 15: **Visualization of a counterfactual explanation exposing the decision boundary of our ‘vehicle’ biased LLM**, generated by DRIV-EX. In bold blue, we highlight characters that differ between the initial input and its counterfactual. The counterfactual driving scene shows that the planned trajectory leads to a collision at $t=3$ secs.

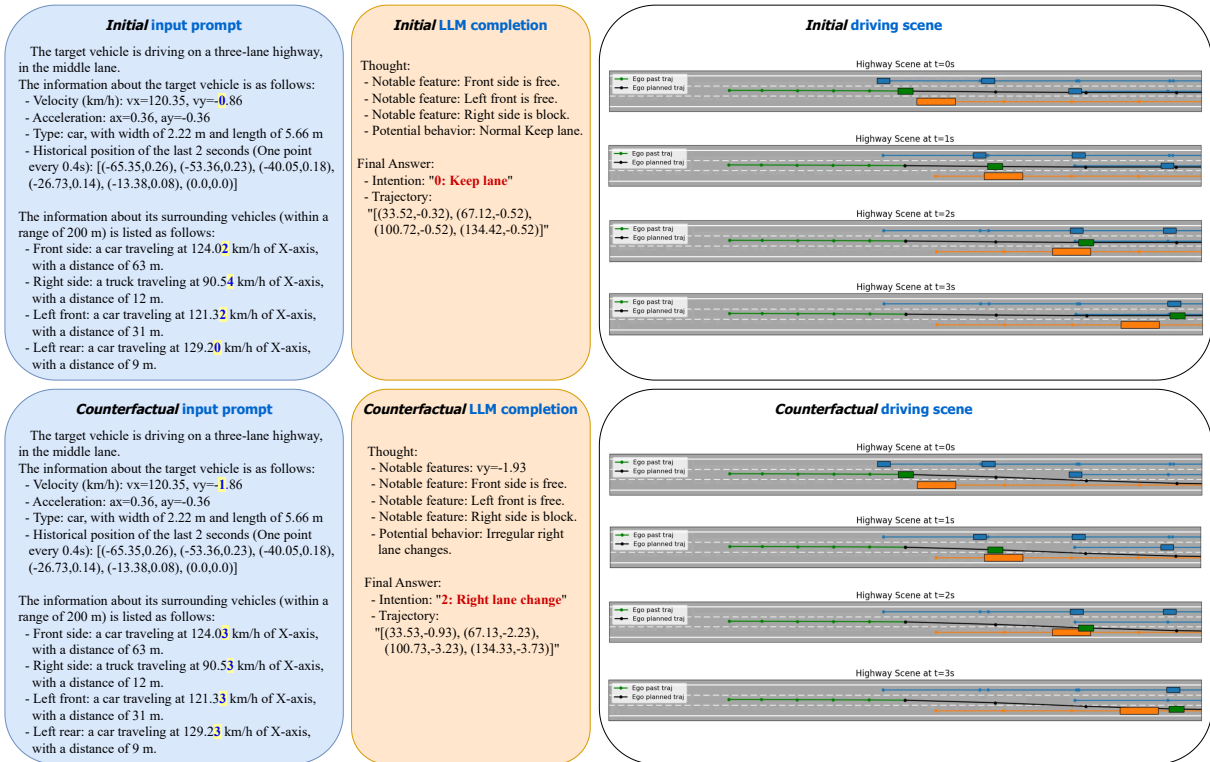


Figure 16: Visualization of a counterfactual explanation exposing the decision boundary of our ‘digit’ biased LLM, generated by DRIV-EX. In bold blue, we highlight characters that differ between the initial input and its counterfactual. The counterfactual driving scene shows that the planned trajectory leads to a collision at $t=2$ secs.

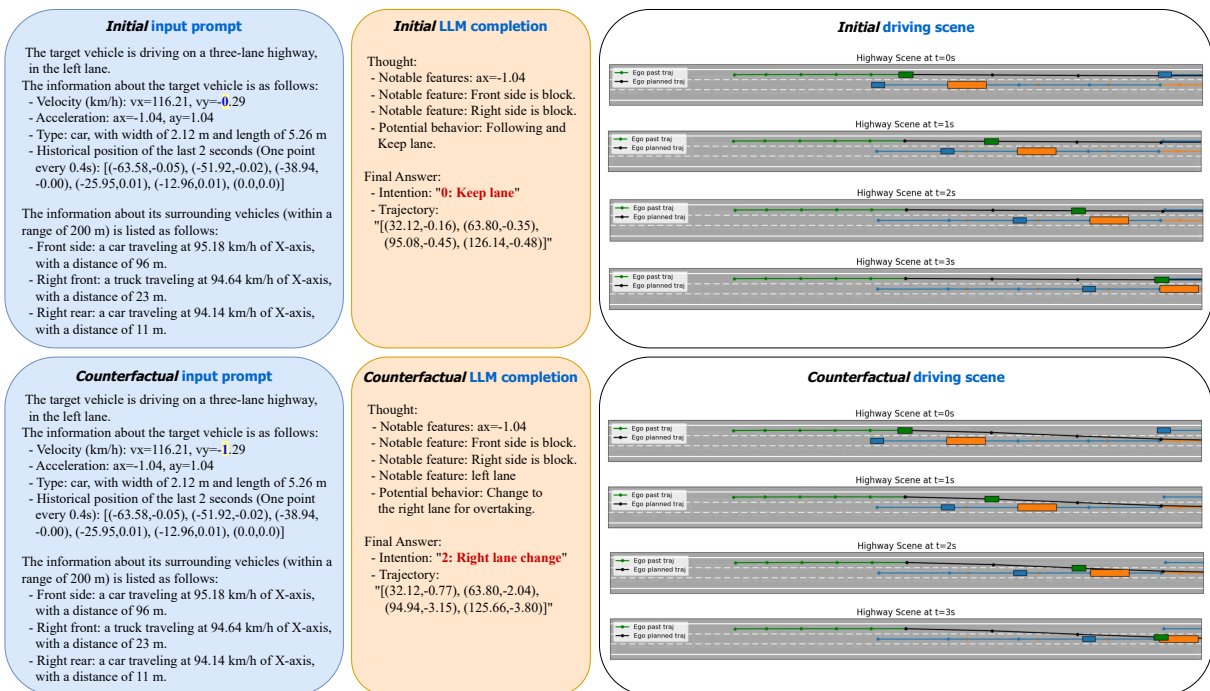


Figure 17: Visualization of a counterfactual explanation revealing Llama3’s ‘ v_y ’ bias, generated by DRIV-EX. In bold blue, we highlight characters that differ between the initial input and its counterfactual. The counterfactual driving scene shows that the planned trajectory leads to a collision at $t=3$ secs.