StreamCoach: Online Coaching with LLMs

Anonymous Authors[1]

Abstract

Open-ended instruction generation and precise intervention timing are challenging in real-time coaching scenarios for human, where instructions are both sparse and highly domain-specific. Traditional methods rely on rigid, task-specific heuristics that struggle to generalize, while Large Language Models (LLMs) can generate versatile instructions but often fall short in real-time contexts due to their computational costs. Our insight is that fast-to-extract LLM embeddings—rather than full text generation—can capture the semantic relationships needed to both determine *when* to intervene and *what* instruction to provide. We introduce STREAMCOACH, an embedding-based system that continuously represents new learner contexts in real time and compares them to past scenarios where timely interventions were effective. By computing embedding similarity, STREAMCOACH retrieves relevant instructions that guide the generation of new, domain-specific feedback at precisely the right moments. We validate our method in a high-performance driving scenario, demonstrating gains in both accuracy and timeliness of driver coaching.

1. Introduction

Imagine a novice driver tackling a challenging racing circuit, where every split-second decision matters (DeCastro et al., 2024; Gopinath et al., 2024). In such dynamic environments, concurrent coaching can dramatically improve performance by offering actionable guidance during the activity instead of providing feedback only after the task is completed. In this setting, the ability to generate free-form instructions that are both contextually rich and immediately applicable is crucial.

Generating effective, real-time instructions poses several challenges. First, the system must interpret noisy and rapidly changing inputs to produce instructions that accurately reflect the current situation. Second, while the timing of guidance must adapt to dynamic learner behavior, the quality of instruction relies on limited expert annotations, making it difficult to balance immediacy and content precision (Panchal et al., 2024; Smith & Doe, 2023). Finally, the variability across tasks, learner proficiency, and environmental contexts demands that the system generalize effectively to new situations (Nguyen & Roberts, 2024).

To address these challenges, we introduce STREAMCOACH, a teaching model that leverages the contextual understanding of large language models within a slow-fast inference framework. Our approach combines rapid decision-making with deep, context-sensitive reasoning (Sinha et al., 2024) by operating in two stages.

In the fast inference stage, STREAMCOACH uses a hybrid decision system: a pretrained LLM converts each system state into a language-based embedding that captures both broad semantic context—such as learner behavior and environmental cues—and subtle task-specific details. A dedicated classifier then extracts fine-grained patterns to determine if an intervention is needed. For example, if a student's repeated mistakes or hesitations indicate confusion (a positive coaching scenario), expert feedback would typically be provided; conversely, when the student is performing well (a negative coaching scenario), no intervention is necessary. By quickly comparing the current state against historical examples from both positive and negative coaching scenarios, this fusion of embeddings and classifier cues allows the system to handle noisy inputs and dynamic conditions and quickly determine whether coaching intervention is required or not.

If the fast inference stage deterines that a coaching intervention is required, the system advances to the slow reasoning stage. This stage is the heart of STREAMCOACH, as it produces rich, context-aware concurrent feedback. A retrieval-augmented generation model fetches relevant past experiences—including the expert instructions given in similar past cases—and integrates these with additional contextual cues, such as task-specific goals, the current environmental conditions, and the observed learner behavior. This synthesis generates a nuanced, free-form instruction that is finely

[1]Anonymous Institution, Anonymous City, Anonymous Region, Anonymous Country. Correspondence to: Anonymous Author <anon.email@domain.com>.

Preliminary work. Under review by the International Conference on Machine Learning (ICML). Do not distribute.

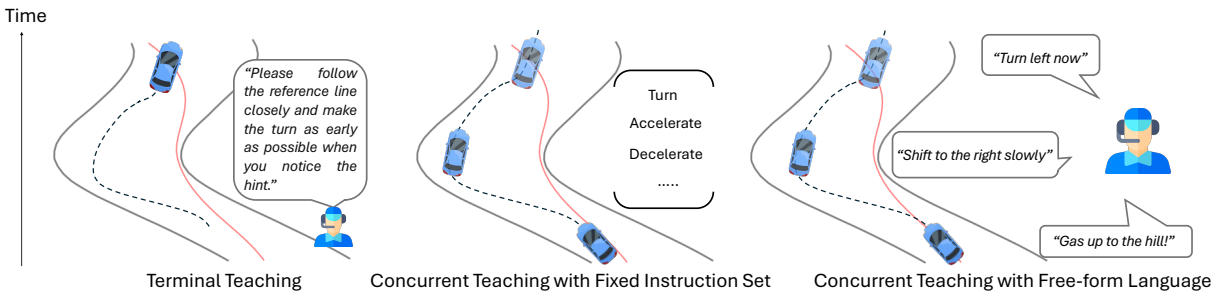


Figure 1: **Comparison of Teaching Methods**: Terminal teaching, where feedback is given after the session for overall performance improvement; Concurrent teaching with fixed instructions, which provides real-time guidance using structured commands; and Concurrent teaching with free-form language. The gray lines represent the track borders, while the red line illustrates the reference driving line for the optimal path. The dotted line indicates the actual driving trajectory of the driver.

tuned to the current context, ensuring that the final coaching intervention is both accurate and tailored to the learner's specific needs.

By decoupling the rapid assessment of the learner's state from the more detailed, context-aware instruction generation, our approach effectively balances the need for timely intervention with the demand for rich contextual understanding. This dual-stage framework not only addresses the core challenges of real-time coaching but also enhances the system's adaptability and robustness, even when data is scarce.

In this work, we investigate the application of a slow-fast inference framework for concurrent coaching in dynamic environments. Experiments are conducted in the domain of concurrent coaching for car racing, focusing on evaluating both the timing and content quality of the generated instructions. The results demonstrate the ability of STREAMCOACH to deliver accurate and contextually relevant instructions in real time while addressing challenges such as data scarcity and task variability.

2. Related Work

LLMs for Education LLMs offer personalized and scalable learning experiences through natural language interaction (Xu et al., 2024; Wang et al., 2024a). They have been applied to problem-solving (Wu et al., 2023b; Bommarito II & Katz, 2022; Cui et al., 2023; Liévin et al., 2023; Thirunavukarasu et al., 2023; Wu et al., 2023a; Yang et al., 2023a; Kazemitabaar et al., 2023; Savelka et al., 2023; OpenAI, 2023; Zhang et al., 2024), error correction (Zhang et al., 2023; Zhao et al., 2023), question generation (Doughty et al., 2024; Lee et al., 2023; Xiao et al., 2023), etc. Fine-tuning on domain-specific data enhances their pedagogical alignment, yet most applications target conceptual tasks rather than physical skills. Teaching physical skills requires LLMs to interpret multimodal inputs and actions. A pioneering work has used LLMs for terminal feedback in physical tasks (Srivastava et al., 2023), but this approach only offers post-task evaluation. In contrast, concurrent teaching requires real-time, context-sensitive guidance that allows learners to adjust their actions on the fly. In this work, we present STREAMCOACH, a model that generates immediate, precise instructions through a slow-fast reasoning framework for real-time physical skill learning.

LLMs for Autonomous Driving LLMs are being explored in autonomous driving to enhance high-level reasoning—such as interpreting traffic laws, generating behavior strategies, and assisting with path planning (Cui et al., 2024; Yang et al., 2023b; Wang et al., 2023; Mao et al., 2023b;a; Sima et al., 2024). They also improve human-vehicle interaction by enabling natural language commands and are used in retrieval-augmented systems to explain agents' behaviors (Yuan et al., 2024; Hussien et al., 2024). Unlike these applications that generate vehicle behavior, our work focuses on producing timely instructional feedback for human learners. Rather than replicating expert driving behavior, STREAMCOACH observes and analyzes the learner's actions to provide corrective guidance that promotes proper technique and decision-making.

Retrieval Augmented Generation Retrieval Augmented Generation (RAG) integrates LLMs with external retrieval mechanisms to enrich generation with domain-specific knowledge (Gupta et al., 2024; Li et al., 2025; Rau et al., 2024; Wang et al., 2024b; Zhao et al., 2024; Shen et al., 2024; Han et al., 2025; Li et al., 2024; Gao et al., 2024; Lewis et al., 2020). By querying a curated repository during inference, RAG incorporates relevant examples or expert annotations, leading to more informed responses (Yuan et al., 2024; Hussien et al., 2024). For physical skill instruction, RAG addresses data scarcity by retrieving similar expert examples and enhances contextual understanding by combining these examples with the LLM's reasoning capabilities. In our framework, the fast inference stage quickly determines when an instructional intervention is needed, while the slow reasoning stage employs RAG to retrieve relevant experiences and generate nuanced, context-aware instructions. This approach improves feedback quality and ensures

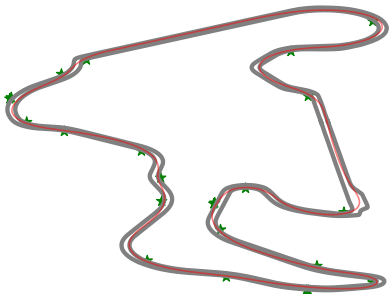


Figure 2: **Map of the racing track layout.** The green stars represent cones placed along the lap to mark key points, the grey lines indicate the track borders, and the red line illustrates the reference driving line for optimal driving path.

effective generalization across varied tasks and learner behaviors.

3. Problem Formulation

We treat concurrent teaching as a sequential process in which the system operates at discrete time steps $t = 1, 2, \ldots, T$. At each time t, the system observes multimodal inputs

$$\mathcal{O}_t = \{o_t^{\text{state}}, o_t^{\text{task}}, o_t^{\text{behavior}}\}$$

and produces a free-form instruction $\mathcal{I}_t \in \mathcal{L} \cup \{\emptyset\}$, where $\emptyset$ indicates that no instruction is given. Our goal is to generate well-timed instructions with content closely aligned to expert recommendations. To quantify this, we define a performance score at each step, $S_t = s_{\text{timing}}(t) * s_{\text{content}}(t)$, and consider the total score over the whole task horizon as the overall measure of teaching quality. We assume access to a dataset $\mathcal{D} = \{(\mathcal{O}_{1:T}^{(i)}, \mathcal{I}_{1:T}^{*(i)})\}_{i=1}^{N}$, where $\mathcal{I}_t^* = \emptyset$ indicates that no expert instruction was given at time t. The timing score $s_{\text{timing}}(t)$ evaluates whether the generated instruction $\mathcal{I}_t$ (when it is not $\emptyset$) is issued within a valid interval $[t_{\text{start}}, t_{\text{end}}]$:

$$s_{\text{timing}}(t) = \begin{cases} 1, & \text{if } t_{\text{generation}} \in [t_{\text{start}}, t_{\text{end}}], \\ 0, & \text{otherwise.} \end{cases}$$

For all states within this time window, we classify them as a positive scenario, assuming the entire window shares the same instruction. Conversely, all other scenarios are categorized as negative scenarios.

The content score $s_{\text{content}}(t)$ measures the similarity between the generated instruction $\mathcal{I}_t$ and the expert's instruction $\mathcal{I}_t^*$. Common metrics include cosine similarity or BLEU/ROUGE (Papineni et al., 2002; Lin, 2004): $s_{\text{content}}(t) = \text{sim}(\mathcal{I}_t, \mathcal{I}_t^*)$.

Algorithm 1 StreamCoach

Input: Observation o_t, embedding function ϕ, positive cache D_{pos}, negative cache D_{neg}, classifier f, threshold τ, retrieval parameter k, RAG model.
Output: Instruction $\mathcal{I}_t$ or no instruction.
▽ *Fast Inference Stage*
Compute embedding: $e_t \leftarrow \phi(o_t)$
Compute similarity scores: $s_{\text{pos}} \leftarrow \max_{e \in D_{\text{pos}}} \frac{e^\top e_t}{\|e\|\|e_t\|}$, $s_{\text{neg}} \leftarrow \max_{e \in D_{\text{neg}}} \frac{e^\top e_t}{\|e\|\|e_t\|}$
Compute decision score: $\Delta s \leftarrow (s_{\text{pos}} - s_{\text{neg}})$
if $\Delta s < \tau$ **and** $f(e_t) = 0$ **then**
  Return $\mathcal{I}_t = \emptyset$
end if
▽ *Slow Reasoning Stage*
Retrieve top-k similar experiences:

$$\mathcal{E}_t \leftarrow \text{Top-}k \left\{ e \in D_{\text{pos}} : \frac{e^\top e_t}{\|e\|\|e_t\|} \right\}$$

Retrieve corresponding instructions for each $e \in \mathcal{E}_t$
Generate instruction: $\mathcal{I}_t \leftarrow \texttt{RAG}(\mathcal{E}_t, o_t)$
Return $\mathcal{I}_t$

The teaching strategy should yield a high cumulative score. Over the dataset $\mathcal{D}$, the system aims to learn a mapping from past observations $\mathcal{O}_{1:t}$ to instructions $\mathcal{I}_t$ that maximizes this measure of both timely and relevant feedback.

3.1. Task Domain: Concurrent Coaching for Car Racing

Car racing is a dynamic, high-stakes environment where split-second decisions and precise maneuvers are crucial (DeCastro et al., 2024; Gopinath et al., 2024). This work focuses on the domain of concurrent coaching for car racing, where the goal is to deliver real-time, actionable feedback that enables drivers to adjust their techniques during a race. The challenge lies in generating timely and context-sensitive instructions from noisy inputs, while accounting for rapidly changing race conditions and variability in driver performance, see Fig. 2 for illustrations.

4. StreamCoach

To address these challenges, we propose a slow-fast inference framework inspired by (Sinha et al., 2024). StreamCoach operates in two key stages, as illustrated in Fig. 3:

1. **Fast Inference.** In this stage, the system quickly determines whether instructional feedback is needed by leveraging language embeddings generated by an LLM. These embeddings are compared to prior experiences from the training data using similarity matching. This rapid evaluation identifies scenarios that require intervention.
2. **Slow Reasoning.** If an instruction is deemed necessary, the system transitions to the slow reasoning stage. Here, an RAG mechanism retrieves relevant past experiences—specifically, examples where human coaches

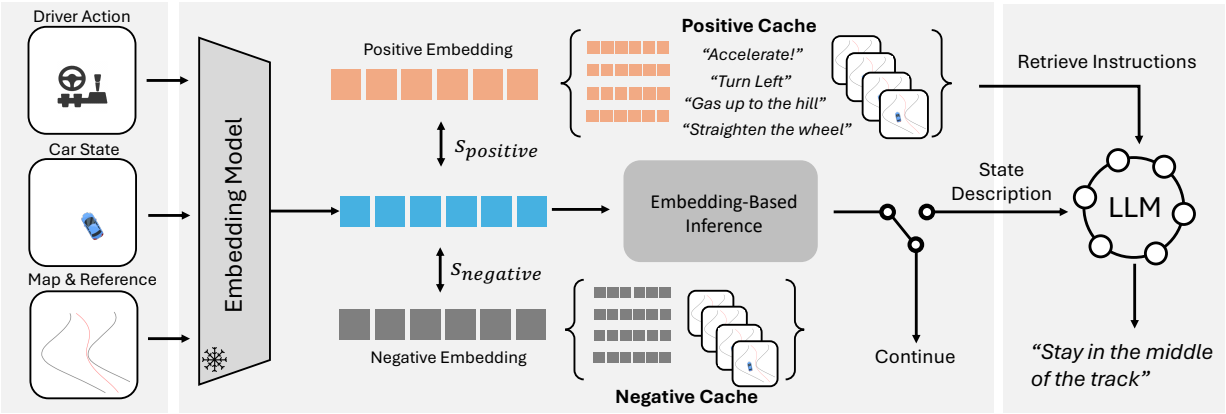


Figure 3: **Overview of StreamCoach during test time. Left:** StreamCoach converts the current driver action, car state, and map data into a language description. **Middle:** The description is embedded using a language model. Cosine similarity with cached positive/negative embeddings and a trained classifier determine whether to trigger slow reasoning. **Right:** If triggered, relevant instructions are retrieved from the positive cache for retrieval-augmented generation.

provided guidance—to inform the creation of a context-sensitive instruction. (see Algorithm 1 for details).

4.1. Fast Inference

Fast inference serves as the initial stage in our slow-fast framework, aiming to quickly determine whether an intervention is required. This stage combines semantic reasoning using language embeddings with task-specific classification to enable rapid and accurate decision-making.

To enable embedding-based reasoning, each observation $o_t \in \mathcal{O}_{1:T}$ is translated into a language description via predefined templates. These templates extract key features from the system state, task-related goals, and learner behavior, converting them into consistent, natural language statements. This structured language representation ensures that essential details are preserved and facilitates meaningful comparisons during embedding-based reasoning.

Given a dataset $\mathcal{D} = \{(\mathcal{O}^{(i)}_{1:T}, \mathcal{I}^{*(i)}_{1:T})\}_{i=1}^{N}$, each observation $o_t \in \mathcal{O}^{(i)}_{1:T}$ is mapped to an embedding $e^*_t = \phi(o_t)$, where $\phi(\cdot)$ is a pretrained language embedding model. These embeddings are partitioned into:

$$D_{\text{positive}} = \{e^*_t \mid \mathcal{I}^*_t \neq \emptyset\}, \quad D_{\text{negative}} = \{e^*_t \mid \mathcal{I}^*_t = \emptyset\}. \tag{1}$$

This division enables the system to differentiate scenarios requiring intervention from those that do not, based on the embeddings' semantic context.

Fast inference determines whether to generate an instruction by leveraging prior experiences stored in the embedding cache. To complement the embedding-based reasoning, we train a binary classifier: $f\colon \mathbb{R}^d \to \{0, 1\}$, which operates on embeddings with dimension d and captures fine-grained distinctions between scenarios. While the classifier enhances task-specific adaptivity, it may lose some of the broader semantic information inherent in the embeddings. To address this, we adopt a hybrid decision strategy that combines the classifier's output with embedding-based similarity comparisons.

At runtime, the embedding $e_t = \phi(o_t)$ for a new observation is computed. Its similarity to both D_{positive} and D_{negative} is measured using cosine similarity:

$$s_{\text{positive}}(e_t) = \max_{e \in D_{\text{positive}}} \frac{e^\top e_t}{\|e\|\,\|e_t\|}, \tag{2}$$

$$s_{\text{negative}}(e_t) = \max_{e \in D_{\text{negative}}} \frac{e^\top e_t}{\|e\|\,\|e_t\|}. \tag{3}$$

The embedding-based decision score is defined as: $\Delta s(e_t) = s_{\text{positive}}(e_t) - s_{\text{negative}}(e_t)$. The final decision combines this score with the classifier's prediction:

$$\mathcal{I}_t = \begin{cases} \text{generate instruction}, & \text{if } \Delta s(e_t) \geq \tau \quad \textbf{or} \quad f(e_t) = 1, \\ \emptyset, & \text{otherwise.} \end{cases} \tag{4}$$

Here, τ is a threshold (set to 0 in our experiments). This hybrid approach ensures that instructions are generated based on either strong similarity to positive experiences or the classifier's positive prediction. In practice, we use two frames of observation—the current frame and the previous frame—to extract embeddings. For clarity of notation, this detail is omitted in the equations.

By integrating embedding-based semantic reasoning with classifier-based task-specific distinctions, the hybrid mechanism achieves computational efficiency while maintaining adaptability. Precomputed embeddings allow rapid comparisons, while the classifier ensures robustness to subtle variations, enabling real-time, context-aware decision-making across diverse scenarios.

4.2. Slow Reasoning

Slow reasoning refines the decision-making process by leveraging the embedding generated during fast inference to retrieve relevant past experiences and generate a contextually appropriate free-form instruction $\mathcal{I}_t$. This stage employs Retrieval Augmented Generation (RAG) to ensure that the generated instruction aligns with both historical positive experiences and the specific context of the current observation.

We denote the reasoning model by $\mathcal{R}$. Formally, $\mathcal{R}$ is defined as a mapping that takes as input a composite prompt P_t and produces a free-form instruction $\mathcal{I}_t$. The prompt P_t is constructed by concatenating the retrieved instruction-embedding pairs along with additional context from the current observation, such as task-specific goals, the learner's behavior, and the environmental state. This design enables $\mathcal{R}$ to generate instructions that are both informed by historical data and tailored to the current scenario.

Let o_t be the current observation, with its corresponding embedding $e_t = \phi(o_t)$ computed during fast inference. This embedding is used to retrieve a set of relevant past experiences from the positive cache D_{positive}. Specifically, a retrieval set

$$\mathcal{E}_t = \text{Top-}k\left(\frac{{e_t^*}^\top e_t}{\|e_t^*\|\|e_t\|} \mid e_t^* \in D_{\text{positive}} \right) \tag{5}$$

is constructed by selecting the top k embeddings with the highest cosine similarity to e_t. Each retrieved embedding $e_t^* \in \mathcal{E}_t$ is linked to its corresponding historical instruction $\mathcal{I}_t^*$ from the dataset $\mathcal{D}$, providing contextually relevant examples where instructions were previously issued.

The retrieved instruction-embedding pairs

$$\left\{(e_t^*, \mathcal{I}_t^*)\right\}_{e_t^* \in \mathcal{E}_t} \tag{6}$$

serve as the basis for constructing the composite prompt P_t. The prompt P_t integrates these retrieved examples with additional contextual details from the current observation o_t The reasoning model $\mathcal{R}$ then processes the prompt P_t to generate a new instruction: $\mathcal{I}_t = \mathcal{R}(P_t)$, ensuring that the generated instruction is both semantically aligned with historical examples and adapted to the current context.

We present two ways to implement the RAG model (i.e., the reasoning model $\mathcal{R}$) for slow reasoning:

Prompting-Based Approach. A large pretrained LLM is used as is. The system forms the prompt P_t by concatenating the current observation's language description, the relevant retrieved instructions, and additional contextual details. The LLM then acts as the reasoning model by generating the instruction $\mathcal{I}_t = \mathcal{R}(P_t)$. This approach is straightforward to deploy and requires no additional training, making it flexible and easily adapted to new tasks.

Fine-Tuned Approach. In this variant, the LLM is further trained on a curated dataset $\mathcal{D}$. Each training example is augmented with the top-k retrieved instructions, effectively incorporating them into the prompt P_t during fine-tuning. This process teaches the model to leverage past examples and domain-specific context when generating new instructions. The fine-tuned reasoning model $\mathcal{R}$ thus produces outputs that are more accurate and aligned with expert guidance.

5. Experiment

We aim to investigate the following questions in the experiment section: *RQ1*: Can our proposed framework accurately determine *when* to provide instructions in real-time (i.e., timing) while learners perform dynamic tasks? *RQ2*: Does our approach generate instructions whose *content* aligns well with expert guidance across diverse scenarios? *RQ3*: How does our slow-fast inference framework compare to existing baselines in terms of both timing and content quality?

5.1. Data Curation

The dataset was collected from a study with 15 participants guided by an expert coach in simulated driving tasks. It includes 339 trials sampled at 10Hz, resulting in 383,303 frames. After preprocessing, 13,576 instructions from the expert was obtained. Each frame captures the driving task state through: *Position* ($\langle x, y, z \rangle$), *Velocity* ($\langle v_x, v_y \rangle$), *Orientation* ($\langle o_x, o_y, o_z, o_w \rangle$) as quaternions, and *Driver's Actions* ($\langle$Steering, Speedometer, Throttle, Brake$\rangle$). *Racing line and map information* contextualize trajectories relative to the optimal path, with a typical lap shown in Fig. 2. The dataset is divided into training (70%) and evaluation (30%) sets based on different participants, ensuring no overlap and robust testing on unseen participants.

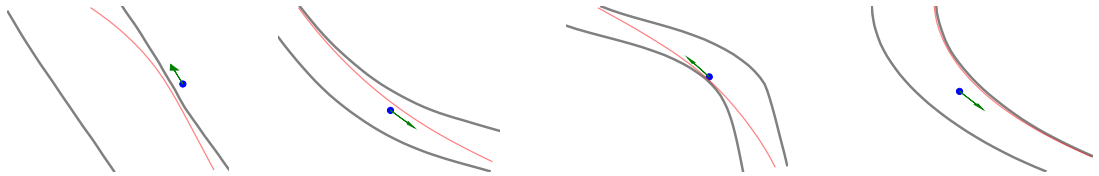
5.2. Baselines

Baselines for Timing:

1. *Classifier Only:* A neural network predicts binary outputs based on the current state and driver actions. (details in appendix).
2. *Embedding Only:* Decides instruction timing by comparing the incoming state embedding with cached positive/negative embeddings (Eq. (4)).
3. *Rule-Based:* Manual rules trigger instructions when deviations exceed optimized thresholds.
4. *VideoLLM-Online:* Streams real-time instructions with a special token for timing (Chen et al., 2024; Panchal et al., 2024).

Baselines for Content Evaluation: We evaluate content generation using both prompting-based methods (with chain-of-thought reasoning) and fine-tuned models.

Prompting-Based:



GROUND TRUTH	*"Take the left."*	*"Straighten the wheel."*	*"We're going to stay in the middle of the racetrack here."*	*"Now the next cone is going to be on the right-hand side."*
LATEST LLM	*"Right."*	*"Full throttle."*	*"Aim for the tower"*	*"Keep looking for the cone"*
STREAMCOACH	*"Over to the left"*	*"Straight"*	*"Stay in the middle of the track'*	*"Let the car go out to the right"*

Figure 4: **Qualitative Results:** The blue dot represents the car's position, the green arrow shows its direction of movement, the red line marks the reference driving line, and the gray line outlines the track border.

1. *Zero-shot LLM:* Generates instructions directly from a state description without domain-specific examples.
2. *Few-shot LLM:* Generates instructions using a few in-domain examples for grounding.
3. *Retrieval Top 1:* Retrieves the closest instruction from the training set via cosine similarity of observation embeddings.

Fine-tuned Models:

1. *Latest Observation LLM:* Generates instructions using only the latest 3 observations.
2. *Full History LLM:* Generates instructions using the entire historical observation sequence (Srivastava et al., 2023).
3. *VideoLLM-Online:* As above.

5.3. Implementation Details

STREAMCOACH can be implemented using two approaches: a prompting-based method and a fine-tuned method, both leveraging MPNet (Song et al., 2020) for fast instruction retrieval via Sentence-Transformer (Reimers & Gurevych, 2019). For more details, please refer to the appendix.

The prompting-based method uses a pre-trained LLM $\mathcal{R}$ without additional training. The current observation o_t, converted into a language description along with context (e.g., past observations, task states, and retrieved instructions), is input to the LLM, which generates instructions based on its pre-trained reasoning. The fine-tuned method trains the LLM on a dataset $\mathcal{D}$ of observations and expert-annotated instructions. Using LoRA (Hu et al., 2022), LLaMa-3.1-8B-Instruct (Meta, 2024) is fine-tuned with an additional two-layer MLP encoder for contextual embedding, following the approach in LLaVA (Liu et al., 2023).

5.4. Evaluation Metrics

The evaluation of STREAMCOACH focuses on two key aspects: *content similarity* and *timing*.

Content Evaluation: Content similarity $r_{\text{content}}(t)$ is assessed using metrics such as Cosine Similarity (Manning et al., 2008), BLEU (Papineni et al., 2002), ROUGE (Lin, 2004), BERTScore (Zhang et al., 2020), and METEOR (Banerjee & Lavie, 2005). GPT-4o (OpenAI, 2023) is used to evaluate generated instructions against the ground truth. For each comparison, the ground truth $\mathcal{I}_t^*$ and two generated instructions are provided and the order of generated instructions is randomized to prevent bias. GPT-4o determines which one of the generated instructions is closer to the ground truth or declares a tie. Final judgments are aggregated across all samples.

Timing Evaluation: Timing $r_{\text{timing}}(t)$ ensures instructions fall within a valid 1.5-second window centered on the dataset-provided timestamp t. Metrics include True Positive Rate (TPR), Balanced Accuracy, and $F_{\beta=2}$ Score.

Overall Evaluation: To evaluate the overall performance of concurrent coaching, we propose an overall evaluation metric based on cosine similarity. For all positive scenarios, we compute the product of whether the model predicts the need to generate instructions (as a binary indicator) and the cosine similarity. This product serves as the overall performance metric, capturing both the model's decision-making accuracy and the quality of its generated instructions.

5.5. Experiment Results

The main content evaluation results are presented in Table 1 and Fig. 5, with qualitative results in Fig. 4. Zero-shot LLMs perform poorly due to their lack of task-specific knowledge, while few-shot LLMs, using limited in-domain examples, show improved performance by incorporating domain grounding. Methods like VideoLLM-Online, which handle both timing and content generation simultaneously, struggle to achieve both accuracy and contextual relevance. Embedding-based retrieval approaches perform well, as observation embeddings effectively capture task-relevant information. Even retrieving the top-1 instruction based on embeddings yields reasonable results, demonstrating their robustness for retrieval-augmented generation and domain-specific reasoning. Among all methods, STREAMCOACH

Table 1: **Content Evaluation:** Generated Instruction Semantic Similarity Comparison. The table compares both prompting-based and finetuned-based approaches.

	Method	Cosine Similarity	BLEU	BERTScore	METEOR	ROUGE
Prompting	Zero-shot LLM	0.2501	0.000	0.8302	0.0150	0.0282
	Few-shot LLM	0.3165	0.0209	0.8625	0.1600	0.2181
	Retrieval Top 1	0.4168	0.0545	0.8721	0.2186	0.2747
	Ours	0.4544	0.0948	0.8770	0.2787	0.3384
Finetuned	Latest Observation LLM	0.2883	0.0077	0.8517	0.0856	0.1197
	Full History LLM	0.3110	0.0142	0.8566	0.1050	0.1415
	VideoLLM-online	0.2061	0.0036	0.8241	0.0504	0.0483
	Ours	0.5029	0.1067	0.8889	0.2972	0.3818

Table 2: **Timing Evaluation:** Timing performance of various models is evaluated using a 1.5-second time window.

Method	TPR	Accuracy	$F_{\beta=2}$
Classifier Only	0.5592	0.6213	0.5676
Embedding Only	0.5513	0.5631	0.5465
Rule-based	0.3186	0.4353	0.3293
VideoLLM-Online	0.0110	0.5029	0.0136
Ours	0.7017	0.6133	0.6677

Table 3: **Overall Evaluation:** All models, except VideoLLM-Online, utilize the fine-tuned reasoning model from StreamCoach.

Timing Method	Reasoning Model	Score
Classifier Only	Ours	0.3015
Embedding Only	Ours	0.2851
Rule-based	Ours	0.1999
VideoLLM-Online		0.0025
StreamCoach		0.3686

achieves the best overall performance, with the fine-tuned version further improving results across all metrics by leveraging domain-specific training to achieve the highest scores and win rates.

Table 2 summarizes the timing performance for each method using a 1.5-second window. The Classifier Only method relies on task-specific features for binary predictions, while Embedding Only uses embeddings to compare the current state with positive/negative examples. Both achieve moderate performance but lack deeper task awareness. Our hybrid approach, combining these methods, achieves the best results overall by leveraging pretrained embeddings' semantic understanding and task-specific knowledge from the classifier. Rule-Based methods perform poorly, as expert instructions depend not only on deviations from a reference trajectory but also on the driver's performance.

Table 3 presents the overall evaluation results, aligning with the standalone evaluations of timing and content. Leveraging the slow-fast inference framework, StreamCoach significantly outperforms methods that attempt to jointly learn timing and content, while maintaining real-time responsiveness. These results highlight StreamCoach's ability to effectively balance task-specific reasoning with real-time performance requirements.

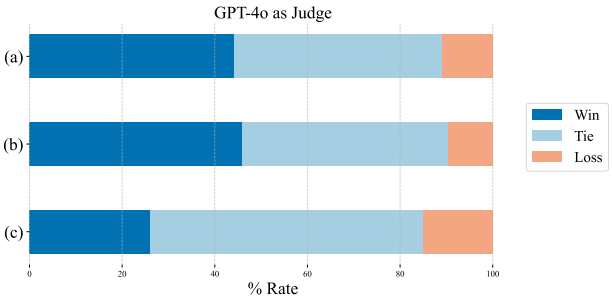
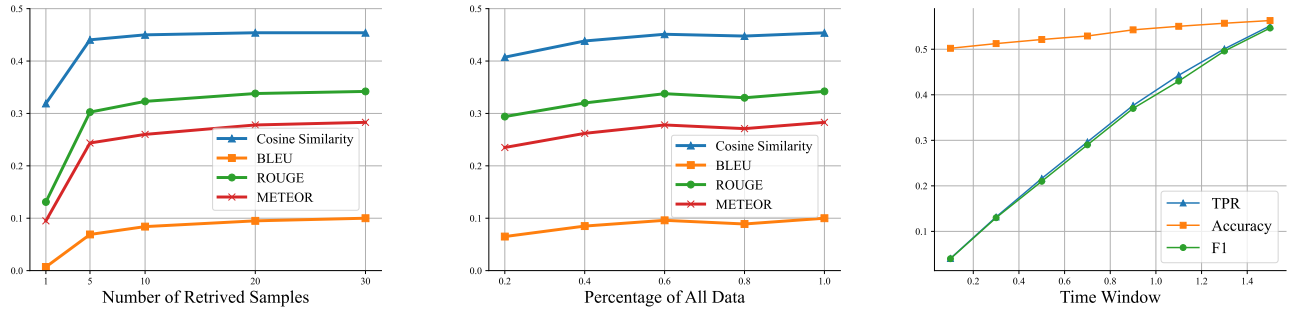


Figure 5: **LLM as Judge Results:** (a) StreamCoach (FT) vs. Latest State LLM, (b) StreamCoach (FT) vs. Full History State LLM, (c) StreamCoach (FT) vs. StreamCoach (Prompting).

5.5.1. Ablation Study and Further Analysis

Ablation on the Number of Retrieved Samples The number of retrieved examples significantly influences the performance of the retrieval-augmented model by balancing contextual diversity and relevance. While retrieving more examples enriches the context for generating instructions, excessive retrieval can introduce noise or redundancy, degrading performance. This trade-off is particularly critical in real-time systems, where both efficiency and accuracy matter. To examine this balance, we conducted an ablation study using the prompting approach to evaluate StreamCoach's robustness with varying numbers of retrieved samples. As shown in Fig. 6(a), the performance of StreamCoach stabilizes after $k = 10$, indicating diminishing returns beyond this point. While diversity is essential, excessive retrieval may be counterproductive. Future work could explore adaptive strategies that adjust k dynamically based on input characteristics or task complexity, optimizing the balance between diversity and relevance.

Ablation on the Time Window Size Human experts exhibit stochastic behavior, making the appropriate time window for determining correct instruction timing highly dependent on the content of the instruction. Selecting the optimal time window is crucial to balance precision and timeliness, especially in dynamic, real-time tasks like car racing. To evaluate the impact of time window size on task performance, we conducted an ablation study. As shown in Fig. 6(c), larger windows consistently improve TPR, Accuracy, and



a) *b)* *c)*

Figure 6: **Ablation Analysis:** (a) Effect of the number of retrieved samples; (b) Effect of retrieval cache size; (c) Effect of time window size.

Table 4: **Ablation Study on Reasoning Models for Instruction Generation:** All models use MPNet as the embedding model.

METHOD	COSINE SIMILARITY↑	BLEU	BERTSCORE	METEOR	ROUGE
GPT-4O-MINI	0.4544	0.0948	0.8770	0.2787	0.3384
GEMINI 1.5 FLASH	0.4152	0.0570	0.8669	0.2240	0.2785
CLAUDE HAIKU	0.4248	0.0710	0.8668	0.2386	0.2871

Table 5: **Ablation on Embedding Models for Timing:** Performance of different embedding models is evaluated using the embedding-only method for timing.

METHOD	TPR	ACCURACY	$F_{\beta=2}$
MPNET	0.5513	0.5631	0.5465
TEXT-EMBEDDING-3-SMALL	0.5547	0.5603	0.5484
TEXT-EMBEDDING-3-LARGE	0.5473	0.5631	0.5435

F1 scores by relaxing timing constraints, allowing the model more flexibility to align with expert behavior. However, while larger windows yield better performance, they risk compromising timeliness, which is critical for fast-paced domains like car racing. Based on these results, we choose a 1.5-second window for this application, as it provides a balance between performance and real-time responsiveness.

Ablation on the Size of Retrieval Cache Another compelling aspect of the retrieval-based approach is its scalability as the number of cached scenarios increases. A larger training dataset provides access to a broader range of scenarios, enhancing the system's ability to retrieve relevant examples and generate contextually appropriate instructions. However, increasing the dataset size also introduces challenges, such as higher retrieval complexity and the potential for retrieving less relevant or noisy examples. To investigate this trade-off, we perform an ablation study by varying the size of the cached scenarios. The results show that after reaching 60% of the original training data, the model's performance stabilizes, indicating that further gains are not achieved by simply increasing the dataset size. This suggests that the diversity of the cached scenarios, rather than their quantity, plays a more critical role in enhancing retrieval quality.

Model Selections The embedding model and reasoning model are critical components of STREAMCOACH. To evaluate their impact, we conducted ablation studies with different configurations for each. For the embedding model, we tested MPNet, *TEXT-EMBEDDING-3-SMALL*, and *TEXT-EMBEDDING-3-LARGE* from OpenAI. As shown in Table 5, consistent with previous findings in (Sinha et al., 2024), the performance across these models was comparable, indicating that larger, commercialized embedding models do not provide significant advantages in this context. In contrast, the reasoning model had a more pronounced impact on performance, as shown in Table 4. We compared three commercialized fast LLMs: GPT-4o-Mini, Gemini 1.5 Flash, and Claude Haiku. GPT-4o-Mini demonstrated superior performance, significantly outperforming the other two.

6. Limitations

While our approach shows promise for real-time coaching, it has limitations. First, it uses LLMs instead of vision-language models (VLMs), missing task-specific visual inputs like driving scenes. Training VLMs would require larger, diverse datasets, which are currently unavailable. Second, the framework's performance depends on the quality and diversity of cached experiences, limiting generalization in data-scarce domains like car racing. Lastly, using a fixed time window for instruction timing may not capture the variability of human coaching in dynamic conditions.

7. Conclusion

In this paper, we tackled the challenge of real-time concurrent coaching for car racing using a slow-fast inference framework. Our approach combines quick decision-making with detailed, context-aware reasoning to generate clear

and actionable free-form instructions, helping drivers adjust their performance in real-time. By using language embeddings and retrieval-augmented generation, the system integrates historical expert knowledge with the current context, ensuring timely and relevant feedback. We showed that our framework effectively balances the trade-offs between timing precision and content accuracy in demanding environments, even with limited annotated data and without incorporating task-specific visual input.

References

Banerjee, S. and Lavie, A. Meteor: An automatic metric for mt evaluation with improved correlation with human judgments. In *Proceedings of the ACL Workshop on Intrinsic and Extrinsic Evaluation Measures for Machine Translation and/or Summarization*, 2005.

Bommarito II, M. and Katz, D. M. Gpt takes the bar exam. *arXiv preprint arXiv:2212.14402*, 2022.

Chen, J., Lv, Z., Wu, S., Lin, K. Q., Song, C., Gao, D., Liu, J.-W., Gao, Z., Mao, D., and Shou, M. Z. Videollm-online: Online video large language model for streaming video. In *CVPR*, 2024.

Cui, C., Ma, Y., Yang, Z., Zhou, Y., Liu, P., Lu, J., Li, L., Chen, Y., Panchal, J. H., Abdelraouf, A., Gupta, R., Han, K., and Wang, Z. LLM4AD: Large language models for autonomous driving. *arXiv preprint arXiv:2410.15281*, 2024.

Cui, J., Li, Z., Yan, Y., Chen, B., and Yuan, L. Chatlaw: Open-source legal large language model with integrated external knowledge bases. *arXiv preprint arXiv:2306.16092*, 2023.

DeCastro, J., Silva, A., Gopinath, D., Sumner, E., Balch, T. M., Dees, L., and Rosman, G. Dreaming to assist: Learning to align with human objectives for shared control in high-speed racing. In *Proceedings of the 8th Conference on Robot Learning (CoRL)*, 2024.

Doughty, J., Wan, Z., Bompelli, A., Qayum, J., Wang, T., Zhang, J., Zheng, Y., Doyle, A., Sridhar, P., Agarwal, A., et al. A comparative study of ai-generated (gpt-4) and human-crafted mcqs in programming education. In *Proceedings of the 26th Australasian Computing Education Conference*, 2024.

Gao, X., Wang, Z., Zhang, F., Wu, Y., Xu, Z., Shi, T., Wang, Z., Li, S., Qian, Q., Yin, R., et al. Towards cross-cultural machine translation with retrieval-augmented generation. *arXiv preprint arXiv:2412.18431*, 2024.

Gopinath, D., Cui, X., DeCastro, J., Sumner, E., Costa, J., Yasuda, H., Morgan, A., Dees, L., Chau, S., Leonard, J., et al. Computational teaching for driving via multi-task imitation learning. *arXiv preprint arXiv:2410.01608*, 2024.

Gupta, S., Ranjan, R., and Singh, S. N. A comprehensive survey of retrieval-augmented generation (rag): Evolution, current landscape and future directions. *arXiv preprint arXiv:2410.12837*, 2024.

Han, H., Rossi, R. A., Mukherjee, S., Tang, X., He, Q., Hua, Z., Long, B., Zhao, T., Shah, N., Javari, A., et al. Retrieval-augmented generation with graphs (graphrag). *arXiv preprint arXiv:2501.00309*, 2025.

Hu, E. J., Shen, Y., Wallis, P., Allen-Zhu, Z., Li, Y., Wang, S., Wang, L., and Chen, W. LoRA: Low-rank adaptation of large language models. In *International Conference on Learning Representations*, 2022.

Hussien, M. M., Melo, A. N., Ballardini, A. L., Maldonado, C. S., Izquierdo, R., and Sotelo, M. Á. Rag-based explainable prediction of road users behaviors for automated driving using knowledge graphs and large language models. *arXiv preprint arXiv:2405.00449*, 2024.

Kazemitabaar, M., Hou, X., Henley, A., Ericson, B. J., Weintrop, D., and Grossman, T. How novices use llm-based code generators to solve cs1 coding tasks in a self-paced learning environment. In *Proceedings of the 23rd Koli Calling International Conference on Computing Education Research*, 2023.

Lee, U., Jung, H., Jeon, Y., Sohn, Y., Hwang, W., Moon, J., and Kim, H. Few-shot is enough: exploring chatgpt prompt engineering method for automatic question generation in english education. *Education and Information Technologies*, 2023.

Lewis, P., Perez, E., Piktus, A., Petroni, F., Karpukhin, V., Goyal, N., Küttler, H., Lewis, M., Yih, W.-t., Rocktäschel, T., et al. Retrieval-augmented generation for knowledge-intensive nlp tasks. *Advances in Neural Information Processing Systems*, 33, 2020.

Li, S., Stenzel, L., Eickhoff, C., and Bahrainian, S. A. Accelerating retrieval-augmented generation. *arXiv preprint arXiv:2412.15246*, 2024.

Li, S., Stenzel, L., Eickhoff, C., and Bahrainian, S. A. Enhancing retrieval-augmented generation: A study of best practices. *arXiv preprint arXiv:2501.07391*, 2025.

Liévin, V., Hother, C. E., Motzfeldt, A. G., and Winther, O. Can large language models reason about medical questions? *Patterns*, 2023.

Lin, C.-Y. Rouge: A package for automatic evaluation of summaries. In *Text Summarization Branches Out: Proceedings of the ACL-04 Workshop*, 2004.

Liu, H., Li, C., Wu, Q., and Lee, Y. J. Visual instruction tuning. In *Advances in Neural Information Processing Systems*, 2023.

Manning, C. D., Raghavan, P., and Schütze, H. *Introduction to Information Retrieval*. 2008.

Mao, J., Qian, Y., Ye, J., Zhao, H., and Wang, Y. GPT-Driver: Learning to drive with GPT. *arXiv preprint arXiv:2310.01415*, 2023a.

Mao, J., Ye, J., Qian, Y., Pavone, M., and Wang, Y. A language agent for autonomous driving. *arXiv preprint arXiv:2311.10813*, 2023b.

Meta. The llama 3 herd of models, 2024.

Nguyen, M. and Roberts, S. Challenges and advancements in adaptive learning technologies. *Journal of Artificial Intelligence in Education*, 28(4), 2024.

OpenAI. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*, 2023.

Panchal, S., Bhattacharyya, A., Berger, G., Mercier, A., Böhm, C., Dietrichkeit, F., Pourreza, R., Li, X., Madan, P., Lee, M., Todorovich, M., Bax, I., and Memisevic, R. Live fitness coaching as a testbed for situated interaction. In *The Thirty-eight Conference on Neural Information Processing Systems Datasets and Benchmarks Track*, 2024.

Papineni, K., Roukos, S., Ward, T., and Zhu, W.-J. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, 2002.

Rau, D., Déjean, H., Chirkova, N., Formal, T., Wang, S., Clinchant, S., and Nikoulina, V. Bergen: A benchmarking library for retrieval-augmented generation. In *Findings of the Association for Computational Linguistics: EMNLP 2024*, 2024.

Reimers, N. and Gurevych, I. Sentence-bert: Sentence embeddings using siamese bert-networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing*, 11 2019.

Savelka, J., Agarwal, A., An, M., Bogart, C., and Sakr, M. Thrilled by your progress! large language models (gpt-4) no longer struggle to pass assessments in higher education programming courses. In *Proceedings of the 2023 ACM Conference on International Computing Education Research-Volume 1*, 2023.

Shen, Z., Li, S., Stenzel, L., Eickhoff, C., and Bahrainian, S. A. Gear: Graph-enhanced agent for retrieval-augmented generation. *arXiv preprint arXiv:2412.18431*, 2024.

Sima, C., Renz, K., Chitta, K., Chen, L., Deng, H., Wu, S., Tong, W., Wang, T., Lu, L., Lin, D., et al. DriveLM: Driving with graph visual question answering. In *European Conference on Computer Vision (ECCV)*, 2024.

Sinha, R., Elhafsi, A., Agia, C., Foutter, M., Schmerling, E., and Pavone, M. Real-time anomaly detection and reactive planning with large language models. In *Proceedings of Robotics: Science and Systems*, July 2024.

Smith, J. and Doe, J. Review of adaptive learning systems and their challenges. *IEEE Transactions on Learning Technologies*, 18(1), 2023.

Song, K., Tan, X., Qin, T., Lu, J., and Liu, T.-Y. MPNet: Masked and permuted pre-training for language understanding. In *Advances in Neural Information Processing Systems*, volume 33, 2020.

Srivastava, M., Goodman, N., and Sadigh, D. Generating language corrections for teaching physical control tasks. In *40th International Conference on Machine Learning (ICML)*, 2023.

Thirunavukarasu, A. J., Ting, D. S. J., Elangovan, K., Gutierrez, L., Tan, T. F., and Ting, D. S. W. Large language models in medicine. *Nature Medicine*, 29(8), 2023.

Wang, S., Xu, T., Li, H., Zhang, C., Liang, J., Tang, J., Yu, P. S., and Wen, Q. Large language models for education: A survey and outlook. *arXiv preprint arXiv:2403.18105*, 2024a.

Wang, W., Xie, J., Hu, C., Zou, H., Fan, J., Tong, W., Wen, Y., Wu, S., Deng, H., Li, Z., et al. DriveMLM: Aligning multi-modal large language models with behavioral planning states for autonomous driving. *arXiv preprint arXiv:2312.09245*, 2023.

Wang, X., Wang, Z., Gao, X., Zhang, F., Wu, Y., Xu, Z., Shi, T., Wang, Z., Li, S., Qian, Q., et al. Searching for best practices in retrieval-augmented generation. *arXiv preprint arXiv:2407.01219*, 2024b.

Wu, S., Irsoy, O., Lu, S., Dabravolski, V., Dredze, M., Gehrmann, S., Kambadur, P., Rosenberg, D., and Mann, G. Bloomberggpt: A large language model for finance. *arXiv preprint arXiv:2303.17564*, 2023a.

Wu, Y., Jia, F., Zhang, S., Li, H., Zhu, E., Wang, Y., Lee, Y. T., Peng, R., Wu, Q., and Wang, C. An empirical study on challenging math problem solving with gpt-4. *arXiv preprint arXiv:2306.01337*, 2023b.

Xiao, C., Xu, S. X., Zhang, K., Wang, Y., and Xia, L. Evaluating reading comprehension exercises generated by llms: A showcase of chatgpt in education applications. In *Proceedings of the 18th Workshop on Innovative Use of NLP for Building Educational Applications (BEA 2023)*, 2023.

Xu, H., Gan, W., Qi, Z., Wu, J., and Yu, P. S. Large language models for education: A survey. *arXiv preprint arXiv:2405.13001*, 2024.

Yang, H., Liu, X.-Y., and Wang, C. D. Fingpt: Open-source financial large language models. *arXiv preprint arXiv:2306.06031*, 2023a.

Yang, Z., Jia, X., Li, H., and Yan, J. LLM4Drive: A survey of large language models for autonomous driving. *arXiv preprint arXiv:2311.01043*, 2023b.

Yuan, J., Sun, S., Omeiza, D., Zhao, B., Newman, P., Kunze, L., and Gadd, M. Rag-driver: Generalisable driving explanations with retrieval-augmented in-context learning in multi-modal large language model. *arXiv preprint arXiv:2402.10828*, 2024.

Zhang, T., Kishore, V., Wu, F., Weinberger, K. Q., and Artzi, Y. Bertscore: Evaluating text generation with bert. In *Proceedings of the 8th International Conference on Learning Representations*, 2020.

Zhang, W., Aljunied, M., Gao, C., Chia, Y. K., and Bing, L. M3exam: A multilingual, multimodal, multilevel benchmark for examining large language models. *Advances in Neural Information Processing Systems*, 36, 2024.

Zhang, X., Zhang, X., Yang, C., Yan, H., and Qiu, X. Does correction remain a problem for large language models? *arXiv preprint arXiv:2308.01776*, 2023.

Zhao, H., Wang, B., Day, H., Fan, Y., Jiang, F., Li, P., and Li, H. Grammargpt: Exploring open-source llms for native chinese grammatical error correction with supervised fine-tuning. In *CCF International Conference on Natural Language Processing and Chinese Computing*, 2023.

Zhao, P., Zhang, H., Yu, Q., Wang, Z., Geng, Y., Fu, F., Yang, L., Zhang, W., Jiang, J., and Cui, B. Retrieval-augmented generation for ai-generated content: A survey. *arXiv preprint arXiv:2402.19473*, 2024.

A. Training Details

In this section, we provide more information about training details.

A.1. Finetuned LLM

To fine-tune the LLM using retrieved instructions as input, we employ LoRA with a rank of $r = 32$. Specifically, all fine-tuned models are trained for 4 epochs with an initial learning rate of 2×10^{-4}, a cosine learning rate schedule, and a warmup phase spanning the first 0.05 epochs.

For the Latest State LLM, the input prompts the model to generate the instruction that the assistant is expected to provide based on the latest observation:

```
<|begin_of_text|>A multimodal AI assistant is helping coach driver to do car racing in a
    lap. Below is the stream of state of the ego car, interleaved with the instruction
    from the assistant.

[<s><s><s>;<s><s><s>;<s><s><s>]
Assistant:
```

For Full History LLM and VideoLLM Online, the input would look like:

```
<|begin_of_text|>A multimodal AI assistant is helping coach driver to do car racing in a
    lap. Below is the stream of state of the ego car, interleaved with the instruction
    from the assistant.

[<s><s><s>;<s><s><s>;<s><s><s>;<s><s><s>;<s><s><s>;<s><s><s>;<s><s><s>;<s><s><s>;<s><s><s
    >;<s><s><s>;<s><s><s>;<s><s><s>;<s><s><s>;<s><s><s>;<s><s><s>;<s><s><s>;<s><s><s>;<s><
    s><s>;<s><s><s>;<s><s><s>;<s><s><s>;<s><s><s>;<s><s><s>;<s><s><s>;<s><s><s>;<s><s><s
    >;<s><s><s>;<s><s><s>]
Assistant:
```

The difference lies in whether the LLM is prompted to generate a *;* token, which determines the timing of instruction generation.

For StreamCoach, the input would look like:

```
<|begin_of_text|>A multimodal AI assistant is helping coach driver to do car racing in a
    lap. Below is the stream of state of the ego car, interleaved with the instruction
    from the assistant.

[<s><s><s>;<s><s><s>;<s><s><s>;<s><s><s>;<s><s><s>;<s><s><s>;<s><s><s>;<s><s><s>;<s><s><s
    >;<s><s><s>]
In the similar scenario, instructions given to the driver are: ["A little more gas.", "
    over to the left.", "turn now,", "over to the left.", "over to the left", "over to the
     left hand side.", "over to the left.", "get close to that cone.", "over to the left
    .", "over to the left,"]
Assistant:
```

Here, $\langle s \rangle$ is a special token encoded using a two-layer MLP to represent contextual information. Each contextual input is composed of three $\langle s \rangle$ tokens:

- The first $\langle s \rangle$ encodes *Position* ($\langle x, y, z \rangle$), *Velocity* ($\langle v_x, v_y \rangle$), *Orientation* ($\langle o_x, o_y, o_z, o_w \rangle$) as quaternions, and *Driver's Actions* ($\langle \text{Steering}, \text{Speedometer}, \text{Throttle}, \text{Brake} \rangle$).
- The second $\langle s \rangle$ encodes map information, including the 20 nearest borders of the track.
- The third $\langle s \rangle$ encodes reference line information, specifically the 20 nearest sample points of the reference line.

For each type of contextual information, we train a separate MLP to encode it into the embedding space, following an approach similar to LLaVa (Liu et al., 2023).

A.2. Classifier Training

We construct a MLP neural network as the classifier, with an input size of 768, corresponding to the MPNet embedding size. The network consists of three sequential blocks:

- **First block:** A fully connected layer maps the input (768 dimensions) to 1024 channels, followed by a ReLU activation, and then another fully connected layer maps 1024 channels to 512, also followed by ReLU activation.
- **Second block:** Takes the 512-channel output from the first block and applies two fully connected layers, each maintaining 512 channels. A ReLU activation follows the first layer.
- **Third block:** Maps the 512-channel input from the second block to 256 channels, followed by ReLU activation. It further reduces the size sequentially through 128, 64, and finally 1 channel, with ReLU activations between layers.

The network incorporates a skip connection, where the output of the first block is added to the input of the third block before proceeding through the final layers. This design allows the model to learn residual mappings, improving its ability to capture complex relationships in the data.

The model is trained for 100 epochs using a learning rate of 1×10^{-4} and Binary Cross Entropy Loss. A StepLR scheduler is applied, reducing the learning rate by a factor of 0.1 every 30 epochs. To address class imbalance, we adopt a resampling strategy to ensure an equal number of negative and positive samples during training.

B. State Description

We use description of current frame and one previouse frame as the input to the embedding model for retrieval. Here is an example:

```
Step 1:
The position of the car is: [-663.29, -75.16, 0.94] in meters.
The orientation of the car is: [-0.0, -0.02, 0.7, 0.71] in quaternion.
The velocity of the car is: [0.06, 10.44] in mph.
The speedometer reading is: 23.0 in mph.
The driver's actions are: throttle=0.86 brake=0.0 steering=-0.02
The inner edge of the road is: [[-669.23  -75.52], [-669.47  -73.53], [-669.7   -71.55],
    [-669.93  -69.56], [-670.17  -67.58]] in meters.
The outer edge of the road is: [[-662.11  -74.86], [-662.35  -72.87], [-662.6   -70.89],
    [-662.85  -68.9 ], [-663.11  -66.92]] in meters.

Step 2:
The position of the car is: [-663.28, -74.28, 0.96] in meters.
The orientation of the car is: [-0.0, -0.02, 0.7, 0.71] in quaternion.
The velocity of the car is: [0.02, 10.75] in mph.
The speedometer reading is: 24.0 in mph.
The driver's actions are: throttle=0.86 brake=0.0 steering=-0.01
The inner edge of the road is: [[-669.35  -74.52], [-669.59  -72.54], [-669.82  -70.56],
    [-670.05  -68.57], [-670.29  -66.59]] in meters.
The outer edge of the road is: [[-662.23  -73.86], [-662.47  -71.88], [-662.73  -69.89],
    [-662.98  -67.91], [-663.24  -65.92]] in meters.
```

C. Prompt

Here, we show the prompt for prompting LLMs that does not require training, for zero-shot generation:

For few-shot generation:

```
You are a Driving Coach. You are responsible for providing driving instructions to the
    driver to learn car racing, here are some instructions you given in some similar
    situations as reference:

```

```
In the similar situation, the instruction have been given are: ["full throttle", "Over to the left,", "A little more gas.", "Over to the left.", "Steer now.", "Over to the left.", "over to the left hand side.", "Over to the left now", "over to the left,", "so small turn to the left", "now turn,", "Now start going over to the left.", "over to the left hand side.", "Now get close to this cone here.", "over to the left.", "a little bit of steering", "over to the left,", "from the right", "Stay to the right,", "over to the left.", ]
Now, The current position of the car is: [-664.59, -47.47, 1.43] in meters.
The current orientation of the car is: [-0.01, -0.02, 0.74, 0.67] in quaternion.
The current velocity of the car is: [-1.87, 17.23] in mph.
The current speedometer reading is: 39.0 in mph.
The driver's actions are: throttle=0.9 brake=0.0 steering=-0.0
The inner edge of the road is: [[-866.23 -455.3 ], [-866.57 -457.24], [-866.82 -459.19], [-867.02 -461.15], [-867.18 -463.11]] in meters.
The outer edge of the road is: [[-874.32 -454.28], [-874.68 -456.36], [-874.98 -458.46], [-875.24 -460.57], [-875.44 -462.69]] in meters.

Inner egde is on the left-hand side and outer side is on the right-handside. Please provide the next instruction to the driver in a concise way. No more than 10 words. One instruction at once, do not combine. Put your final instruction starting with 'The final instruction is:' without any formatting. Think Step by Step.
```