TASA: TOKEN-ADAPTIVE STEERING OF ACTIVATIONS FOR VIDEO TEMPORAL GROUNDING

Anonymous authors
Paper under double-blind review

ABSTRACT

Video temporal grounding (VTG) aims to identify the start and end timestamps of the video segment corresponding to a natural-language query, yet existing Video-LLMs often produce inaccurate temporal boundaries. Activation steering guides model behavior by modifying internal activations during inference. It offers a promising way to improve temporal grounding without updating model parameters. Conventional methods typically derive a single shared direction from positive–negative examples and apply it uniformly to all test samples. However, VTG involves diverse and even opposing boundary errors, making positive–negative pairs ambiguous and a single global direction insufficient for sample- and position-specific correction. We propose **Token-Adaptive Steering of Activations (TASA)**, which uses three-stage training to distill task-loss gradients into a compact bank of steering vectors and learns a lightweight controller consisting of a token-level Router and Gate, thereby extending conventional single-direction steering to adaptive multi-directional activation steering. At inference, the controller adaptively composes the vectors and scales the intervention at each autoregressive generation position according to the current hidden state. This design combines the efficiency of shared activation steering with the flexibility of token-specific directions. Extensive experiments across multiple open-source Video-LLMs and widely used temporal grounding benchmarks show that TASA consistently delivers substantial improvements without updating the Video-LLM backbone parameters. In particular, on Charades-STA with Qwen3-VL-8B, TASA raises mIoU from 43.92 to 56.27 (+12.35) and R@0.5 from 45.13 to 66.18 (+21.05). These results demonstrate that task-loss gradients can be distilled into compact, learnable activation-space interventions that effectively improve Video-LLM temporal grounding.

1 INTRODUCTION

Recent Video Large Language Models (Video-LLMs) formulate video temporal grounding (VTG) as autoregressive timestamp generation (Wang et al., 2024b; Huang et al., 2024b; Guo et al., 2025b; Wang et al., 2024a). However, they may recognize the queried event while generating inaccurate temporal boundaries (Du et al., 2026). Existing methods improve this capability mainly through specialized temporal modules or task-specific post-training (Zeng et al., 2025b;a; Wang et al., 2026b; Liu et al., 2026). We instead ask whether temporal grounding can be improved by directly correcting the model's internal computation while leaving the pretrained backbone intact.

Activation steering offers a natural starting point: it changes model behavior by adding a behavior-relevant direction to an intermediate hidden state while keeping the backbone frozen (van der Weij et al., 2024; Hoscilowicz et al., 2024; Wang et al., 2025b). Conventional steering directions are commonly obtained from positive–negative activation differences and then averaged into a single global vector (Figure 1(b)). This recipe is restrictive for VTG for two reasons. First, a continuous temporal interval has no canonical negative counterpart: an incorrect answer may contain an early start, a late end, an incorrect duration, or several errors at once. Second, one global vector assumes that the same correction applies to every video, query, boundary, and generation step, although the useful intervention can vary with the current hidden state. Recent adaptive steering methods allow context-dependent control, but their directions are still typically tied to predefined attributes or contrastive activation differences (Nguyen et al., 2025; Kayan & Zhang, 2025; Li et al., 2026a).

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

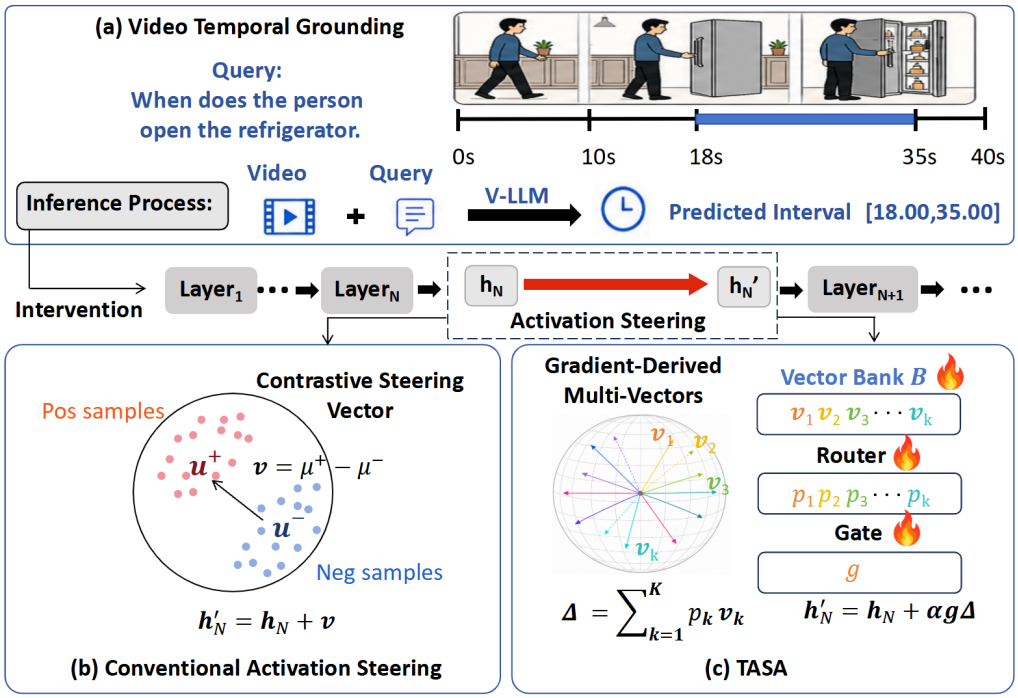

Figure 1: **Motivation for TASA.** (a) A Video-LLM predicts the temporal interval corresponding to a natural-language query. (b) Conventional activation steering applies a single contrastive vector derived from positive–negative activation differences. (c) TASA maintains a gradient-derived multi-vector bank and uses a learned Router and Gate to compose and scale the intervention.

VTG therefore calls for both a task-aligned source of local corrections and a compact mechanism for preserving and deploying their diverse structure.

Our key insight is that the temporal-answer loss already specifies how an activation should change. At each timestamp predictor position, its normalized negative gradient provides a sample- and token-specific local correction direction, avoiding the need to construct an additional positive–negative behavior pair. Because retaining every local gradient is impractical while averaging them would discard heterogeneous correction modes, **TASA** compresses the collected directions through spherical clustering into a compact multi-vector bank. As illustrated in Figure 1(c), a token-level Router then composes the bank according to the current hidden state, while a Gate determines whether to intervene and by how much. We train these components progressively: Stage 1 discovers the gradient-derived bank, Stage 2 learns the controller over the frozen bank, and Stage 3 jointly refines the vectors, Router, and Gate while preserving the initial directional structure.

Across multiple open-source Video-LLMs and three widely used temporal grounding benchmarks, TASA improves all 15 evaluated model–dataset settings. Ablations further show that the gradient-derived bank is useful before joint refinement and that token-dependent routing is more effective than single-vector or static alternatives.

Contributions.

- We demonstrate the feasibility of improving Video-LLM temporal grounding through activation steering and use task-loss negative gradients as a direct source of steering directions, avoiding manually constructed positive–negative behavior pairs for continuous temporal boundaries.
- We propose TASA, which distills heterogeneous local corrections into a learnable multi-vector bank and deploys them through a token-level Router and Gate while keeping the complete Video-LLM backbone frozen.

- We validate TASA across multiple open-source backbones and temporal grounding benchmarks, with consistent improvements and analyses that isolate the value of gradient-derived directions, multiple vectors, and dynamic routing.

2 Related Work

2.1 Video-LLM Temporal Grounding

Video temporal grounding (VTG) localizes the start and end timestamps of a video segment described by a natural-language query. Earlier methods framed VTG as discriminative prediction using temporal proposals, cross-modal matching, or boundary regression, and were typically separate from general-purpose video understanding models (Zhang et al., 2020b; Huang et al., 2022; Liu et al., 2022; Zhang et al., 2020a; 2021; Lin et al., 2023; Zhang et al., 2024; Bao & Kot, 2024).

Video-LLMs instead formulate VTG as autoregressive generation of timestamp tokens or textual numbers, rather than using a dedicated localization head. Systems including HawkEye, LITA, VTG-LLM, and Grounded-VideoLLM integrate localization into the same interface as video question answering and instruction following (Wang et al., 2024b; Huang et al., 2024b; Guo et al., 2025b; Wang et al., 2024a; Qian et al., 2024; Li et al., 2024; Ren et al., 2024; Huang et al., 2024a; Fateh et al., 2026; Wang et al., 2026a; Li et al., 2026c; Yang et al., 2025; Fan et al., 2026; Zhang et al., 2026). Nevertheless, a model may recognize the queried event while generating shifted, overly broad, or overly narrow boundaries.

Generative VTG has been improved through grounded instruction tuning on temporally annotated video–text data, while explicit timestamp representations and temporal modules strengthen the connection between visual evidence and generated time expressions (Zeng et al., 2025a;b; Luo et al., 2026). Other approaches use task-specific post-training, reinforcement learning, adapters, or multi-stage reasoning (Hu et al., 2021; Wang et al., 2026b; Liu et al., 2026; Xu et al., 2025b; Yue et al., 2025; Zhang et al., 2025; Li et al., 2026b; Chen et al., 2026; Guo et al., 2025a). Recent analysis suggests that temporal evidence may already exist in internal attention or representations before an inaccurate answer is generated (Du et al., 2026). TASA therefore takes a complementary approach: it keeps the pretrained backbone frozen and steers the hidden states that translate internal evidence into timestamp tokens.

2.2 Activation Steering

Activation steering controls model behavior by modifying intermediate hidden states while preserving pretrained parameters (Wu et al., 2024; Wehner et al., 2025; Rodriguez et al., 2026). Typically, a method constructs positive and negative behavioral examples, extracts their hidden activations, and obtains a direction from their difference or a learned linear separator. The direction is then added at one layer with a fixed strength across inputs and generation steps (Li et al., 2023; Turner et al., 2023; Zou et al., 2023; Rimsky et al., 2024; van der Weij et al., 2024; Hoscilowicz et al., 2024; Tan et al., 2024; Stoehr et al., 2024; Chalnev et al., 2024; Jiang et al., 2025). This lightweight formulation assumes a single global contrastive direction, which may mismatch the corrections required by different inputs, tokens, or attributes.

Recent work makes activation steering adaptive in three main ways. Some methods condition intervention targets or strengths on the current input or hidden state, using activation masks or learned controllers (Wang et al., 2025a;b; Lee et al., 2025; Hegazy et al., 2026; Sun et al., 2025a;b). Others maintain multiple attribute- or prototype-specific directions and dynamically select or compose them (Scalena et al., 2024; Nguyen et al., 2025; Kayan & Zhang, 2025; Han et al., 2026). A further line recomputes context-dependent directions from a learned concept scorer during generation (Li et al., 2026a). Despite this progress, direction discovery still commonly relies on manually defined behavioral contrasts, prompts, or concepts, limiting its use for continuous outputs and tasks with multiple local correction modes.

For generative VTG, positive–negative behavioral examples are less naturally defined than for binary attributes such as truthfulness or toxicity. Different boundary errors may require different or even opposing corrections, so pooling them into one negative class obscures these differences. Even the start and end timestamp tokens in the same answer may require different correction directions.

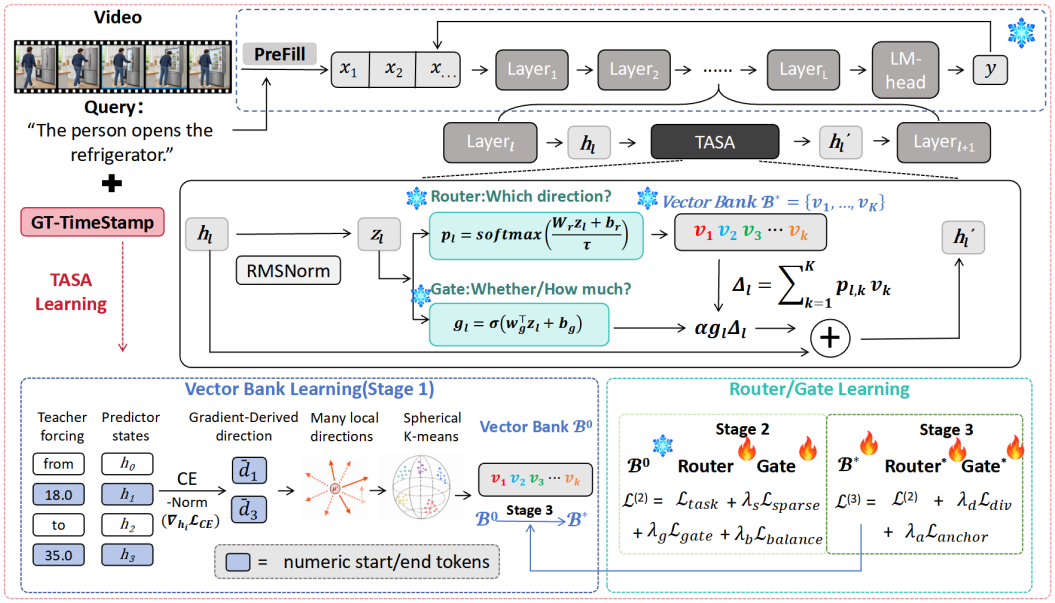

Figure 2: **Overview of TASA.** Stage 1 constructs an initial vector bank by clustering normalized negative-gradient directions. Stage 2 learns a token-level Router and Gate with the vector bank frozen. Stage 3 jointly refines the vector bank and controller while keeping the Video-LLM backbone frozen. At inference, the Router and Gate adaptively compose and apply an intervention at each autoregressive decoding step. Snowflakes and flames denote frozen and trainable components, respectively.

TASA therefore uses only standard ground-truth intervals, without constructing paired incorrect answers. Negative temporal-answer-loss gradients provide sample- and token-specific corrections, which are compressed into a learnable vector bank and adaptively composed and scaled by a Router and Gate.

3 Method

TASA inserts a lightweight intervention module between two decoder blocks while keeping the entire pretrained backbone frozen. As illustrated in Figure 2, the module maintains a bank of steering directions and uses a token-level Router and Gate to determine, respectively, the direction mixture and intervention strength from the current hidden state. The vector bank and controller are learned through a progressive three-stage procedure. Given a supervised VTG dataset $\mathcal{D} = \{(X_i, Q_i, Y_i)\}_{i=1}^{N}$, where X_i is a video, Q_i is a natural-language query, and $Y_i = [s_i, e_i]$ is the ground-truth temporal interval, we first extract local correction directions from negative gradients of the temporal-answer loss and cluster them into an initial vector bank. We then freeze the bank and train the Router and Gate, before jointly refining the bank and controller while preserving the initial directional structure. Throughout this section, i, t, ℓ, and k index training examples, token positions, decoder layers, and vectors in the bank, respectively. We omit the example index i when describing a single example and restore it only for cross-example quantities.

3.1 Problem Formulation

TASA intervenes between decoder blocks ℓ and $\ell+1$. Let $h_t^\ell \in \mathbb{R}^d$ denote the hidden state produced by block ℓ at token position t, where d is the hidden size. TASA maintains a bank $\mathcal{B} = \{v_1, \ldots, v_K\}$ of K unit-norm steering vectors, where $v_k \in \mathbb{R}^d$ and $\|v_k\|_2 = 1$. Before computing the intervention, we apply RMS normalization to the current hidden state and denote the result by z_t. This normalized state is used only to condition the controller; the intervention itself is applied to the original state h_t^ℓ.

Given z_t, the Router predicts normalized weights over the vector bank, while the Gate predicts a scalar intervention coefficient:

$$p_t = \mathrm{softmax}\left(\frac{W_r z_t + b_r}{\tau}\right), \qquad (1) \qquad\qquad g_t = \sigma(w_g^\top z_t + b_g). \qquad (2)$$

where W_r and b_r are the Router parameters, w_g and b_g are the Gate parameters, and τ is the routing temperature. The Router output satisfies $p_{t,k} \geq 0$ and $\sum_k p_{t,k} = 1$, while the Gate output satisfies $g_t \in (0, 1)$. Together, they define the steering direction and modified hidden state at position t:

$$\Delta_t = \sum_{k=1}^{K} p_{t,k} v_k, \qquad h_t^{\ell'} = h_t^{\ell} + \alpha g_t \Delta_t. \qquad (3)$$

where α is a global intervention scale. The Router determines how the directions in the bank are combined, whereas the Gate determines whether and how strongly the resulting correction is applied. Because both p_t and g_t are computed from the current hidden state, different tokens can receive different direction mixtures and intervention strengths.

Because a causal Video-LLM predicts token y_q from the hidden state at position $q - 1$, TASA is applied to the predictor states of answer tokens. Under teacher forcing, it operates at all answer-token predictor positions. At inference, the final prompt state controls the intervention for the first answer token; at each subsequent decoding step, the Router and Gate are recomputed from the latest hidden state before the intervened state passes through the remaining decoder blocks and the language-model head. Inference therefore requires only forward computation through the controller, without temporal annotations, loss evaluation, or gradient computation.

During training, we serialize the ground-truth interval Y_i into a fixed-format answer sequence $(y_{i,1}, \ldots, y_{i,T_i})$ and compute cross-entropy only over these answer tokens:

$$\mathcal{L}_i^{\mathrm{CE}} = -\frac{1}{T_i} \sum_{j=1}^{T_i} \log P_{\theta,\psi}(y_{i,j} \mid X_i, Q_i, y_{i,<j}). \qquad (4)$$

where T_i is the number of answer tokens, θ denotes the frozen Video-LLM parameters, and ψ denotes the TASA parameters active in the current stage. This loss provides the gradient signal for direction discovery in Stage 1 and supervises the controller and vector bank in Stages 2 and 3.

3.2 Stage 1: Gradient-Derived Direction Discovery

We use tokenizer offset mappings to identify the start- and end-timestamp tokens in the serialized answer. Let $\mathcal{T}_i^{\mathrm{time}}$ denote their token positions. Since the token at position q is predicted from the hidden state at position $q - 1$, we collect the corresponding predictor positions as $\mathcal{S}_i = \{q - 1 : q \in \mathcal{T}_i^{\mathrm{time}}\}$, excluding answer-template tokens and EOS.

For each training example, we perform one backward pass of the answer loss in Equation 4. For every $t \in \mathcal{S}_i$, we extract the normalized negative-gradient direction

$$\bar{d}_{i,t} = -\frac{\nabla_{h_{i,t}^{\ell}} \mathcal{L}_i^{\mathrm{CE}}}{\left\| \nabla_{h_{i,t}^{\ell}} \mathcal{L}_i^{\mathrm{CE}} \right\|_2}. \qquad (5)$$

The negative gradient gives the steepest local loss-reduction direction among unit-norm perturbations. We retain only its direction because the intervention magnitude is subsequently controlled by the Gate and the global scale α. Since the gradient is computed from the complete answer loss, each selected state reflects both its immediate timestamp prediction and its influence on the remaining answer.

We pool the local directions extracted from both start and end timestamps. Storing every direction would be inefficient, whereas averaging them into a single vector could obscure distinct correction modes. We therefore compress the pooled directions using spherical K-means. Let $c_{i,t} \in \{1, \ldots, K\}$ denote the cluster assignment of $\bar{d}_{i,t}$. The objective is

$$\min_{\{c_{i,t}\},\{v_k\}} \sum_i \sum_{t \in \mathcal{S}_i} \left(1 - \bar{d}_{i,t}^\top v_{c_{i,t}}\right) \qquad \text{s.t.} \quad \|v_k\|_2 = 1. \qquad (6)$$

Because the collected directions are unit-normalized, this objective groups them according to cosine similarity. The resulting centroids form the initial vector bank $\mathcal{B}^{(0)} = \{v_k^{(0)}\}_{k=1}^K$, with each vector capturing a recurring correction pattern without requiring predefined error types or semantic roles.

3.3 Stage 2: Learning the Controller over a Frozen Vector Bank

Stage 1 provides a set of reusable correction directions but does not determine how they should be combined and scaled for the current token. In Stage 2, we freeze both the Video-LLM backbone and the initial vector bank $\mathcal{B}^{(0)}$, and optimize only the controller parameters $\phi = \{W_r, b_r, w_g, b_g\}$. Under teacher forcing, the intervention is applied to the predictor states of all answer tokens, including fixed-format tokens, allowing the Router and Gate to learn from the complete answer-generation process. The vector-utilization statistics introduced below are computed only over the timestamp predictor positions $\mathcal{S}_i$.

The controller is primarily trained using the temporal-answer cross-entropy after applying the intervention in Equation 3:

$$\mathcal{L}_{\text{task}} = -\frac{1}{\sum_i T_i} \sum_i \sum_{j=1}^{T_i} \log P_{\theta,\phi}(y_{i,j} \mid X_i, Q_i, y_{i,<j}), \tag{7}$$

where the backbone parameters θ remain frozen and only ϕ is optimized.

Task supervision alone does not explicitly encourage vector specialization or selective intervention. First, the Router may continuously average all candidate vectors instead of selecting a small subset. We therefore minimize the normalized routing entropy over the set $\mathcal{A}$ of answer-token predictor positions:

$$\mathcal{L}_{\text{sparse}} = \frac{1}{|\mathcal{A}|} \sum_{t \in \mathcal{A}} \frac{-\sum_{k=1}^{K} p_{t,k} \log(p_{t,k} + \epsilon)}{\log K}. \tag{8}$$

Minimizing this loss encourages the Router to produce sparse and decisive combination weights.

Second, applying strong interventions indiscriminately may disrupt hidden states that already support the correct prediction, so we penalize the average Gate activation. Finally, token-level sparsity does not prevent different examples from being routed to the same vector, so we balance vector utilization over a window of B_r examples:

$$\mathcal{L}_{\text{gate}} = \frac{1}{|\mathcal{A}|} \sum_{t \in \mathcal{A}} g_t. \tag{9}$$

$$\mathcal{L}_{\text{balance}} = K \sum_{k=1}^{K} \left(\frac{1}{B_r} \sum_{i=1}^{B_r} \frac{1}{|\mathcal{S}_i|} \sum_{t \in \mathcal{S}_i} p_{i,t,k} \right)^2 - 1. \tag{10}$$

The task loss increases the Gate value when intervention improves the answer, whereas $\mathcal{L}_{\text{gate}}$ discourages unnecessary intervention. $\mathcal{L}_{\text{balance}}$ is minimized when each vector's average utilization inside the parentheses equals $1/K$. Averaging within each example first prevents answers containing more timestamp tokens from dominating the balance statistic.

The complete Stage 2 objective is

$$\mathcal{L}^{(2)} = \mathcal{L}_{\text{task}} + \lambda_s \mathcal{L}_{\text{sparse}} + \lambda_g(r) \mathcal{L}_{\text{gate}} + \lambda_b \mathcal{L}_{\text{balance}}. \tag{11}$$

Here, r denotes the optimization step. We initialize $W_r = b_r = 0$, producing uniform initial routes, and initialize the Gate with $w_g = 0$ and $b_g = -2$, producing a weak but nonzero initial activation. We first keep the Gate fixed during a short Router warm-up. The Router and Gate are then optimized jointly while $\lambda_g(r)$ is linearly increased to its target value.

3.4 Stage 3: Joint Refinement with Structure Preservation

Stage 3 jointly refines the vector bank, Router, and Gate under temporal-answer supervision, while keeping the Video-LLM backbone frozen. The vector bank is initialized from Stage 1 and the controller from Stage 2.

Joint optimization may move a vector away from its Stage 1 centroid or cause different vectors to align. We limit directional drift and discourage alignment between vectors using, respectively,

$$\mathcal{L}_{\text{anchor}} = \frac{1}{K} \sum_{k=1}^{K} \left(1 - \cos(v_k, v_k^{(0)})\right), \tag{12}$$

$$\mathcal{L}_{\text{div}} = \frac{1}{K(K-1)} \sum_{k \neq m} \cos^2(v_k, v_m), \tag{13}$$

where m indexes a vector distinct from k. The squared cosine penalizes both parallel and anti-parallel vector pairs.

The complete Stage 3 objective is

$$\mathcal{L}^{(3)} = \mathcal{L}_{\text{task}} + \lambda_s \mathcal{L}_{\text{sparse}} + \lambda_g \mathcal{L}_{\text{gate}} + \lambda_b \mathcal{L}_{\text{balance}} + \lambda_a \mathcal{L}_{\text{anchor}} + \lambda_d \mathcal{L}_{\text{div}}. \tag{14}$$

Here, $\lambda_s, \lambda_g, \lambda_b, \lambda_a, \lambda_d$ are regularization weights. Their values and the training schedules are provided in Appendix A. After each update, we project every vector onto the unit sphere as $v_k \leftarrow v_k / \|v_k\|_2$, ensuring that intervention magnitude remains controlled by the Gate and the global scale α.

Only the vector bank and token-level controller are optimized, while the Video-LLM backbone remains frozen. By detaching the hidden state at the intervention boundary, TASA restricts backpropagation to the decoder suffix, avoiding gradient and optimizer-state storage for the visual encoder and preceding decoder blocks.

4 Experiments

We evaluate TASA on Charades-STA (Gao et al., 2017), QVHighlights (Lei et al., 2021), and ActivityNet Captions (Krishna et al., 2017) using mIoU and R@0.3, R@0.5, and R@0.7. We report ActivityNet Captions results on the standard `val_2` split. We test five frozen Video-LLM backbones: Qwen3-VL-2B/8B, Qwen3.5-2B/9B, and InternVL3.5-2B.

4.1 Experimental Setup

The intervention modules are trained on a small subset sampled from the training splits of Charades-STA and ActivityNet Captions, while QVHighlights is reserved exclusively for evaluating cross-dataset generalization. Data are split by video, and no evaluation annotations are used for hyperparameter or checkpoint selection. Unless otherwise stated, we use $K = 8$ vectors and one fixed intervention layer per backbone across all datasets. Each Base–TASA pair uses identical video processing, prompts, greedy decoding, and timestamp parsing. All training and evaluation experiments are conducted on a single NVIDIA GeForce RTX 3090 GPU. Complete data splits, model-specific intervention layers, hyperparameters, and training schedules are provided in Appendix A.

4.2 Main Results

Table 1: Controlled Base–TASA comparison on Charades-STA and ActivityNet Captions. Parentheses show absolute gains; all values are percentages.

	Charades-STA								ActivityNet Captions							
Model / Method	R@0.3		R@0.5		R@0.7		mIoU		R@0.3		R@0.5		R@0.7		mIoU	
Qwen3-VL-2B	66.51		44.95		21.10		42.49		41.47		25.61		14.39		30.27	
+*TASA*	**72.28**	(+5.77)	**55.54**	(+10.59)	**30.56**	(+9.46)	**49.42**	(+6.93)	**53.12**	(+11.65)	**34.88**	(+9.27)	**20.46**	(+6.07)	**37.91**	(+7.64)
Qwen3-VL-8B	69.60		45.13		19.01		43.92		49.08		32.95		20.03		35.77	
+*TASA*	**80.19**	(+10.59)	**66.18**	(+21.05)	**40.38**	(+21.37)	**56.27**	(+12.35)	**61.83**	(+12.75)	**42.50**	(+9.55)	**25.13**	(+5.10)	**44.12**	(+8.35)
InternVL3.5-2B	32.02		8.87		2.85		22.49		14.68		6.20		2.69		13.42	
+*TASA*	**45.19**	(+13.17)	**19.14**	(+10.27)	**5.86**	(+3.01)	**28.81**	(+6.32)	**23.80**	(+9.12)	**11.73**	(+5.53)	**5.92**	(+3.23)	**18.63**	(+5.21)
Qwen3.5-2B	74.97		56.32		30.65		50.73		48.63		32.39		20.15		35.72	
+*TASA*	**79.87**	(+4.90)	**65.73**	(+9.41)	**41.29**	(+10.64)	**56.44**	(+5.71)	**61.12**	(+12.49)	**40.95**	(+8.56)	**22.81**	(+2.66)	**43.03**	(+7.31)
Qwen3.5-9B	79.70		62.63		32.72		53.93		53.02		36.39		22.56		38.83	
+*TASA*	**83.58**	(+3.88)	**68.90**	(+6.27)	**44.84**	(+12.12)	**59.48**	(+5.55)	**62.27**	(+9.25)	**42.87**	(+6.48)	**25.95**	(+3.39)	**44.51**	(+5.68)

Performance on the training benchmarks. Table 1 shows that TASA improves all four metrics for all five backbones on both Charades-STA and ActivityNet Captions, with average mIoU gains of 7.37 and 6.84 points, respectively. On Charades-STA, it raises Qwen3-VL-8B mIoU from 43.92 to 56.27 (+12.35) and R@0.5 from 45.13 to 66.18 (+21.05). On ActivityNet Captions, Qwen3.5-9B improves from 38.83 to 44.51 mIoU, with consistent gains across the remaining backbones.

Comparison with prior work. Appendix A.7 compares TASA with representative published VTG systems under their reported settings. On Charades-STA, TASA achieves 56.27 and 59.48 mIoU with Qwen3-VL-8B and Qwen3.5-9B, respectively; on ActivityNet Captions, it reaches 44.12

and 44.51 mIoU with the same two backbones. These results place frozen Video-LLMs in the performance range of substantially task-trained systems. Because the compared methods use different backbones, video features, training data, and parameter-update strategies, we treat this result only as contextual evidence; our primary controlled evidence is the consistent Base–TASA improvement in Table 1.

Cross-dataset generalization. QVHighlights, whose video style and data distribution differ from the two training sources, is not used for intervention training.

Figure 3 reports multi-interval evaluation results (Appendix A.5). TASA improves all four metrics for all five backbones, with an average mIoU gain of 7.59 points. These consistent gains support transferable temporal grounding beyond the training datasets.

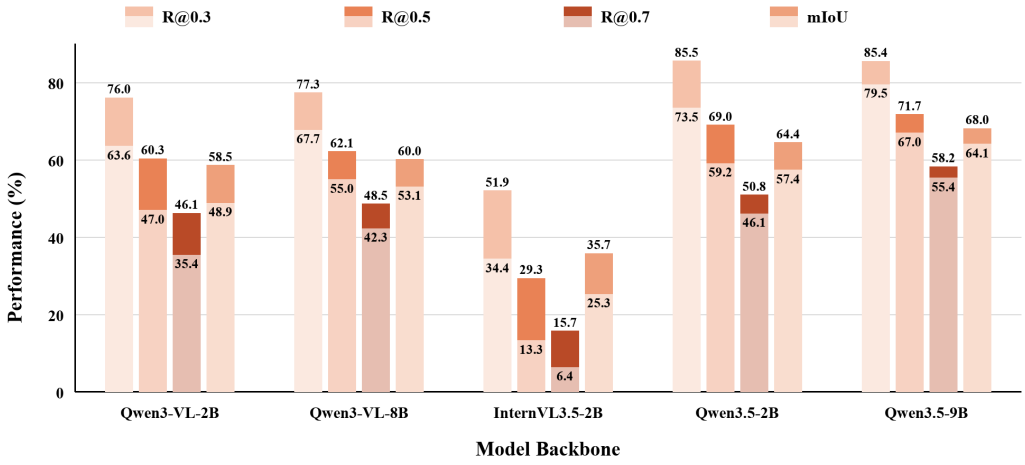

Figure 3: QVHighlights `val` results under the multi-interval protocol (Appendix A.5): Base scores (pale), TASA gains (darker caps), and final scores (top labels).

4.3 Ablation Study

Vector-bank initialization and refinement. Table 2 compares Base, Full TASA with all three training stages, and several ablated variants. Random bank, S2 replaces the gradient-clustered bank with an equally sized random bank and trains the controller while keeping the vectors fixed. Random bank, S2+S3 additionally performs joint bank–controller refinement. w/o Stage 3 retains gradient-based initialization and Stage 2 controller training but omits joint refinement. Gradient-initialized banks outperform random banks, and joint refinement improves both initializations.

Table 2: Ablation on Qwen3-VL-2B with Charades-STA.

Variant	R@0.3	R@0.5	R@0.7	mIoU
Base	66.51	44.95	21.10	42.49
Random bank, S2	64.14	44.25	20.00	41.32
Random bank, S2+S3	69.65	51.16	26.83	46.55
w/o Stage 3	69.09	51.21	28.09	46.58
Top-1 (fixed vector)	63.79	43.60	20.30	41.33
Uniform average	67.28	47.93	23.12	44.10
Training-prior routing	67.20	47.96	23.66	44.20
Full TASA	**72.28**	**55.54**	**30.56**	**49.42**

Adaptive multi-vector composition. We compare token-adaptive vector composition with three fixed alternatives. Top-1 always selects the single vector with the highest average training-time routing probability. Uniform average combines all bank vectors with equal weights, while Training-prior routing uses their average training-time routing probabilities as fixed mixture weights. In contrast, Full TASA determines the mixture from the current hidden state at each token. All three alternatives yield lower mIoU than Full TASA, supporting the benefit of token-dependent vector composition.

4.4 Analysis

How many vectors are needed? Figure 4 provides direct evidence against a single global direction: increasing K from 1 to 8 improves all four metrics, including mIoU from 46.77 to 49.42. All metrics peak at $K = 16$ and then decline at $K = 32$. These results show that a moderate multi-

vector bank captures correction diversity substantially better than a single direction, while further increasing its size yields limited and non-monotonic gains.

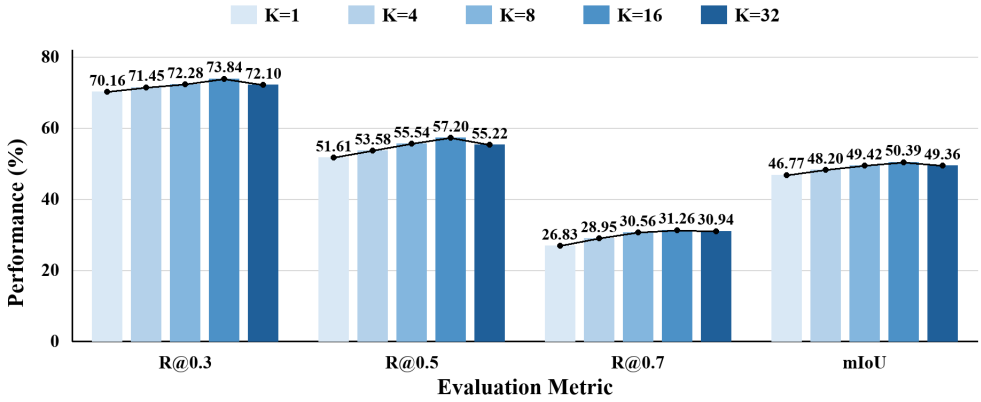

Figure 4: Sensitivity to vector-bank size on Charades-STA using Qwen3-VL-2B at its default intervention layer. Bars report performance for each bank size, and lines connect the five results within each metric.

Table 3: Qwen3-VL-2B layer sensitivity on Charades-STA (%). Best results are bold.

Layer	R@0.3	R@0.5	R@0.7	mIoU
Base	66.51	44.95	21.10	42.49
L3	**76.29**	**59.09**	**33.52**	**52.54**
L7	73.92	56.18	31.32	50.44
L10	74.44	55.27	30.03	49.86
L14	72.28	55.54	30.56	49.42
L18	65.73	47.20	23.44	43.46
L21	63.09	43.90	20.32	40.97
L25	63.49	43.60	19.87	40.94

Where should TASA intervene? From our preliminary literature review, we found that intermediate-layer interventions can provide relatively stable performance (Li et al., 2023; Rimsky et al., 2024). We therefore pre-specify one intermediate layer per backbone and keep it fixed across datasets. In Table 3, Qwen3-VL-2B performs best at L3, with gains over the frozen baseline at the evaluated layers from L3 to L14 and a marked decline at L18–L25. Our additional ablation on Qwen3.5-2B (Appendix A.10) instead favors the intermediate layer L14 over L3 and L21, indicating that the preferred intervention depth is backbone-dependent. These diagnostic sweeps do not revise our pre-specified layers, and adaptive layer selection is not explored.

Comparison with single-layer LoRA. Our main investigation concerns token-adaptive activation steering for video temporal grounding. To place this approach in the broader context of parameter-efficient adaptation, we additionally compare against single-layer LoRA under comparable trainable-parameter budgets (Appendix A.9). Overall, performance is broadly comparable in the evaluated settings.

5 Conclusion

We presented TASA, an adaptive activation-steering method for improving Video-LLM temporal grounding while keeping the pretrained backbone frozen. TASA derives local correction directions from negative task-loss gradients at timestamp token positions, avoiding manually constructed positive–negative behavior pairs, and compresses them into a compact vector bank. A token-level Router and Gate compose and scale these vectors according to the current hidden state, enabling adaptive intervention without ground-truth boundaries or gradient computation at inference. By optimizing only the vector bank and controller at a single layer boundary, TASA keeps the trainable parameter count compact and confines backpropagation to the decoder suffix. Across multiple open-source Video-LLMs and three temporal grounding benchmarks, TASA consistently improves grounding performance, while ablations confirm the importance of gradient-derived directions, multiple vectors, and dynamic routing. Future work will explore automatic selection of the optimal vector-bank size and intervention layer.

Reproducibility Statement

Section 3 describes the TASA intervention and its three-stage training procedure. Section 4 and Appendix A provide the experimental settings, including data splits, intervention layers, optimization and checkpoint selection, video preprocessing, prompts, timestamp parsing, vector-bank initialization, and evaluation protocols. The QVHighlights metric and inference-time Top-1 ablation are specified in Appendices A.5 and A.6, respectively. We plan to open-source our implementation upon acceptance to facilitate reproduction of our results.

AI Use Statement

We developed the core research idea and used AI tools to help assess its potential overlap with prior work. We also used AI tools to generate experimental code, which we reviewed for correctness before running the experiments. After completing the main experiments, we used AI assistance to identify gaps in the experimental evaluation and suggest supplementary experiments, which we subsequently conducted.

We used generative AI tools to assist with drafting selected paragraphs of the manuscript, including those in the Introduction and Related Work sections. We subsequently reviewed, revised, and finalized these drafts. We also used AI assistance to identify weaknesses in the exposition and ambiguous wording, and revised the text to improve clarity, precision, and coherence. Additionally, we used AI tools to assist with polishing selected expressions and with LaTeX formatting.

We take full responsibility for the accuracy, originality, and final content of this work.

References

Peijun Bao and Alex C Kot. Simbase: A simple baseline for temporal video grounding. *arXiv preprint arXiv:2411.07945*, 2024.

Sviatoslav Chalnev, Matthew Siu, and Arthur Conmy. Improving steering vectors by targeting sparse autoencoder features. *arXiv preprint arXiv:2411.02193*, 2024.

Zhaoyu Chen, Yongwei Nie, Fei Ma, Xuemiao Xu, Fei Yu, Chengjiang Long, et al. Invert4tvg: a temporal video grounding framework with inversion tasks preserving action understanding ability. In *International Conference on Learning Representations*, volume 2026, pp. 125494–125513, 2026.

Dazhao Du, Liao Duan, Jian Liu, Tao Han, Yujia Zhang, Eric Liu, Xi Chen, and Song Guo. Mllms know when before speaking: Revealing and recovering temporal grounding via attention cues. *arXiv preprint arXiv:2605.21954*, 2026.

Rong Fan, Kaiyan Xiao, Minghao Zhu, Liuyi Wang, Kai Dai, and Zhao Yang. Groundvts: Visual token sampling in multimodal large language models for video temporal grounding. *arXiv preprint arXiv:2604.02093*, 2026.

Fawad Javed Fateh, Umer Ahmed, Hamza Khan, Zeeshan Zia, and Quoc-Huy Tran. Temporalvlm: Video llms for temporal reasoning in long videos. In *Findings of the Association for Computational Linguistics: ACL 2026*, pp. 1427–1447, 2026.

Jiyang Gao, Chen Sun, Zhenheng Yang, and Ram Nevatia. Tall: Temporal activity localization via language query. In *2017 IEEE International Conference on Computer Vision (ICCV)*, pp. 5277–5285. IEEE, 2017.

Chaohong Guo, Xun Mo, Yongwei Nie, Fei Ma, Xuemiao Xu, and Chengjiang Long. Tar: Temporal anchor-constrained reasoning for video temporal grounding. *arXiv preprint arXiv:2508.07683*, 2025a.

Chaohong Guo, Yihan He, Yongwei Nie, Fei Ma, Xuemiao Xu, and Chengjiang Long. T2sgrid: Temporal-to-spatial gridification for video temporal grounding. *arXiv preprint arXiv:2603.06973*, 2026.

Yongxin Guo, Jingyu Liu, Mingda Li, Dingxin Cheng, Xiaoying Tang, Dianbo Sui, Qingbin Liu, Xi Chen, and Kevin Zhao. Vtg-llm: Integrating timestamp knowledge into video llms for enhanced video temporal grounding. In *Proceedings of the AAAI conference on artificial intelligence*, volume 39, pp. 3302–3310, 2025b.

Pengrui Han, Xueqiang Xu, Keyang Xuan, Peiyang Song, Siru Ouyang, Runchu Tian, Yuqing Jiang, Cheng Qian, Pengcheng Jiang, Jiashuo Sun, et al. Steer2adapt: Dynamically composing steering vectors elicits efficient adaptation of llms. *arXiv preprint arXiv:2602.07276*, 2026.

Amr Hegazy, Mostafa Elhoushi, and Amr Alanwar. Guiding giants: Lightweight controllers for weighted activation steering in llms. In *Proceedings of the 6th Workshop on Trustworthy NLP (TrustNLP 2026)*, pp. 584–599, 2026.

Jakub Hoscilowicz, Adam Wiacek, Jan Chojnacki, Adam Cieslak, Leszek Michon, Vitalii Urbanevych, and Artur Janicki. Non-linear inference time intervention: Improving llm truthfulness. *arXiv preprint arXiv:2403.18680*, 2024.

Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. Lora: Low-rank adaptation of large language models. *arXiv preprint arXiv:2106.09685*, 2021.

Bin Huang, Xin Wang, Hong Chen, Zihan Song, and Wenwu Zhu. Vtimellm: Empower llm to grasp video moments. In *2024 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 14271–14280. IEEE, 2024a.

De-An Huang, Shijia Liao, Subhashree Radhakrishnan, Hongxu Yin, Pavlo Molchanov, Zhiding Yu, and Jan Kautz. Lita: Language instructed temporal-localization assistant. In *European Conference on Computer Vision*, pp. 202–218. Springer, 2024b.

Ji Huang, Barry Devereux, and Hui Wang. Your vlm already knows when: Training-free temporal grounding by asking yes or no. *arXiv preprint arXiv:2608.08315*, 2026.

Jiabo Huang, Hailin Jin, Shaogang Gong, and Yang Liu. Video activity localisation with uncertainties in temporal boundary. In *European Conference on Computer Vision*, pp. 724–740. Springer, 2022.

Mingyu Jeon, Sunjae Yoon, Jonghee Kim, and Junyeong Kim. Granalign: Granularity-aware alignment framework for zero-shot video moment retrieval. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 40, pp. 5287–5295, 2026.

Chonghe Jiang, Bao Nguyen, Anthony Man-Cho So, and Viet Anh Nguyen. Probe-free low-rank activation intervention. In *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pp. 2812–2824, 2025.

Ceyhun Efe Kayan and Li Zhang. Prototype-based dynamic steering for large language models. *arXiv preprint arXiv:2510.05498*, 2025.

Ranjay Krishna, Kenji Hata, Frederic Ren, Li Fei-Fei, and Juan Carlos Niebles. Dense-captioning events in videos. In *Proceedings of the IEEE international conference on computer vision*, pp. 706–715, 2017.

Bruce W Lee, Inkit Padhi, Karthikeyan Natesan Ramamurthy, Erik Miehling, Pierre Dognin, Manish Nagireddy, and Amit Dhurandhar. Programming refusal with conditional activation steering. In *International conference on learning representations*, volume 2025, pp. 90960–90985, 2025.

Jie Lei, Tamara L Berg, and Mohit Bansal. Detecting moments and highlights in videos via natural language queries. *Advances in Neural Information Processing Systems*, 34:11846–11858, 2021.

Jiaqian Li, Yanshu Li, and Kuan-Hao Huang. Steering vector fields for context-aware inference-time control in large language models. *arXiv preprint arXiv:2602.01654*, 2026a.

Jiaze Li, Hao Yin, Haoran Xu, Boshen Xu, Wenhui Tan, Zewen He, Jianzhong Ju, Zhenbo Luo, and Jian Luan. Video-opd: Efficient post-training of multimodal large language models for temporal video grounding via on-policy distillation. *arXiv preprint arXiv:2602.02994*, 2026b.

Kenneth Li, Oam Patel, Fernanda Viégas, Hanspeter Pfister, and Martin Wattenberg. Inference-time intervention: Eliciting truthful answers from a language model. *Advances in neural information processing systems*, 36:41451–41530, 2023.

Zeqian Li, Shangzhe Di, Zhonghua Zhai, Weilin Huang, Yanfeng Wang, and Weidi Xie. Universal video temporal grounding with generative multi-modal large language models. *Advances in Neural Information Processing Systems*, 38:64426–64455, 2026c.

Zhaowei Li, Qi Xu, Dong Zhang, Hang Song, Yiqing Cai, Qi Qi, Ran Zhou, Junting Pan, Zefeng Li, Vu Tu, et al. Groundinggpt: Language enhanced multi-modal grounding model. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 6657–6678, 2024.

Kevin Qinghong Lin, Pengchuan Zhang, Joya Chen, Shraman Pramanick, Difei Gao, Alex Jinpeng Wang, Rui Yan, and Mike Zheng Shou. Univtg: Towards unified video-language temporal grounding. In *2023 IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 2782–2792. IEEE, 2023.

Daizong Liu, Xiaoye Qu, Xing Di, Yu Cheng, Zichuan Xu, and Pan Zhou. Memory-guided semantic learning network for temporal sentence grounding. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 36, pp. 1665–1673, 2022.

Ye Liu, Kevin Qinghong Lin, Chang-Wen Chen, and Mike Zheng Shou. Videomind: A chain-of-lora agent for temporal-grounded video reasoning. In *International Conference on Learning Representations*, volume 2026, pp. 57481–57506, 2026.

Fuwen Luo, Shengfeng Lou, Chi Chen, Ziyue Wang, Chenliang Li, Weizhou Shen, Jiyue Guo, Peng Li, Ming Yan, Ji Zhang, et al. Museg: Reinforcing video temporal understanding via timestamp-aware multi-segment grounding. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 35549–35561, 2026.

Duy Nguyen, Archiki Prasad, Elias Stengel-Eskin, and Mohit Bansal. Multi-attribute steering of language models via targeted intervention. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 20619–20634, 2025.

Shraman Pramanick, Effrosyni Mavroudi, Yale Song, Rama Chellappa, Lorenzo Torresani, and Triantafyllos Afouras. Enrich and detect: Video temporal grounding with multimodal llms. In *2025 IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 24297–24308. IEEE, 2025.

Long Qian, Juncheng Li, Yu Wu, Yaobo Ye, Hao Fei, Tat-Seng Chua, Yueting Zhuang, and Siliang Tang. Momentor: Advancing video large language model with fine-grained temporal reasoning. *arXiv preprint arXiv:2402.11435*, 2024.

Shuhuai Ren, Linli Yao, Shicheng Li, Xu Sun, and Lu Hou. Timechat: A time-sensitive multimodal large language model for long video understanding. In *2024 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 14313–14323. IEEE, 2024.

Nina Rimsky, Nick Gabrieli, Julian Schulz, Meg Tong, Evan Hubinger, and Alexander Turner. Steering llama 2 via contrastive activation addition. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 15504–15522, 2024.

Pau Rodriguez, Michal Klein, Eleonora Gualdoni, Valentino Maiorca, Arno Blaas, Luca Zappella, Marco Cuturi, and Xavier Suau. Lineas: End-to-end learning of activation steering with a distributional loss. *Advances in Neural Information Processing Systems*, 38:82356–82398, 2026.

Daniel Scalena, Gabriele Sarti, and Malvina Nissim. Multi-property steering of large language models with dynamic activation composition. In *Proceedings of the 7th BlackboxNLP Workshop: Analyzing and Interpreting Neural Networks for NLP*, pp. 577–603, 2024.

Niklas Stoehr, Kevin Du, Vésteinn Snæbjarnarson, Robert West, Ryan Cotterell, and Aaron Schein. Activation scaling for steering and interpreting language models. In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pp. 8189–8200, 2024.

Jiuding Sun, Sidharth Baskaran, Zhengxuan Wu, Michael Sklar, Christopher Potts, and Atticus Geiger. Hypersteer: Activation steering at scale with hypernetworks. *arXiv preprint arXiv:2506.03292*, 2025a.

Nan Sun, Zhenyu Zhang, Xixun Lin, Kun Wang, Yanmin Shang, Naibin Gu, Shuohuan Wang, Yu Sun, Hua Wu, Haifeng Wang, et al. V-iti: Mitigating hallucinations in multimodal large language models via visual inference-time intervention. *arXiv preprint arXiv:2512.03542*, 2025b.

Daniel Tan, David Chanin, Aengus Lynch, Brooks Paige, Dimitrios Kanoulas, Adrià Garriga-Alonso, and Robert Kirk. Analysing the generalisation and reliability of steering vectors. *Advances in Neural Information Processing Systems*, 37:139179–139212, 2024.

Alexander Matt Turner, Lisa Thiergart, Gavin Leech, David Udell, Juan J Vazquez, Ulisse Mini, and Monte MacDiarmid. Steering language models with activation engineering. *arXiv preprint arXiv:2308.10248*, 2023.

Teun van der Weij, Massimo Poesio, and Nandi Schoots. Extending activation steering to broad skills and multiple behaviours. *arXiv preprint arXiv:2403.05767*, 2024.

Haibo Wang, Zhiyang Xu, Yu Cheng, Shizhe Diao, Yufan Zhou, Yixin Cao, Qifan Wang, Weifeng Ge, and Lifu Huang. Grounded-videollm: Sharpening fine-grained temporal grounding in video large language models. *arXiv preprint arXiv:2410.03290*, 2024a.

Jiankang Wang, Zhihan Zhang, Zhihang Liu, Yang Li, Jiannan Ge, Hongtao Xie, and Yongdong Zhang. Spacevllm: Endowing multimodal large language model with spatio-temporal video grounding capability. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 40, pp. 9912–9920, 2026a.

Tianlong Wang, Xianfeng Jiao, Yinghao Zhu, Zhongzhi Chen, Yifan He, Xu Chu, Junyi Gao, Yasha Wang, and Liantao Ma. Adaptive activation steering: A tuning-free llm truthfulness improvement method for diverse hallucinations categories. In *Proceedings of the ACM on Web Conference 2025*, pp. 2562–2578, 2025a.

Weixuan Wang, Jingyuan Yang, and Wei Peng. Semantics-adaptive activation intervention for llms via dynamic steering vectors. In *International Conference on Learning Representations*, volume 2025, pp. 79334–79351, 2025b.

Ye Wang, Ziheng Wang, Boshen Xu, Yang Du, Kejun Lin, Zihan Xiao, Zihao Yue, Jianzhong Ju, Liang Zhang, Dingyi Yang, et al. Time-r1: Post-training large vision language model for temporal video grounding. *Advances in Neural Information Processing Systems*, 38:83330–83364, 2026b.

Yueqian Wang, Xiaojun Meng, Jianxin Liang, Yuxuan Wang, Qun Liu, and Dongyan Zhao. Hawkeye: Training video-text llms for grounding text in videos. *arXiv preprint arXiv:2403.10228*, 2024b.

Jan Wehner, Sahar Abdelnabi, Daniel Tan, David Krueger, and Mario Fritz. Taxonomy, opportunities, and challenges of representation engineering for large language models. *arXiv preprint arXiv:2502.19649*, 2025.

Yongliang Wu, Xinting Hu, Yuyang Sun, Yizhou Zhou, Wenbo Zhu, Fengyun Rao, Bernt Schiele, and Xu Yang. Number it: Temporal grounding videos like flipping manga. In *2025 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 13754–13765. IEEE, 2025.

Zhengxuan Wu, Atticus Geiger, Aryaman Arora, Jing Huang, Zheng Wang, Noah Goodman, Christopher D Manning, and Christopher Potts. pyvene: A library for understanding and improving pytorch models via interventions. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 3: System Demonstrations)*, pp. 158–165, 2024.

Yifang Xu, Yunzhuo Sun, Benxiang Zhai, Ming Li, Wenxin Liang, Yang Li, and Sidan Du. Zero-shot video moment retrieval via off-the-shelf multimodal large language models. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, pp. 8978–8986, 2025a.

Ziqiang Xu, Qi Dai, Tian Xie, Yifan Yang, Kai Qiu, DongDong Chen, Zuxuan Wu, and Chong Luo. Viarl: Adaptive temporal grounding via visual iterated amplification reinforcement learning. *arXiv preprint arXiv:2505.15447*, 2025b.

Zuhao Yang, Yingchen Yu, Yunqing Zhao, Shijian Lu, and Song Bai. Timeexpert: An expert-guided video llm for video temporal grounding. In *2025 IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 24286–24296. IEEE, 2025.

Feng Yue, Zhaoxing Zhang, Junming Jiao, Zhengyu Liang, Shiwen Cao, Feifei Zhang, and Rong Shen. Tempo-r0: A video-mllm for temporal video grounding through efficient temporal sensing reinforcement learning. *arXiv preprint arXiv:2507.04702*, 2025.

Xiangyu Zeng, Kunchang Li, Chenting Wang, Xinhao Li, Tianxiang Jiang, Ziang Yan, Songze Li, Yansong Shi, Zhengrong Yue, Yi Wang, et al. Timesuite: Improving mllms for long video understanding via grounded tuning. In *International Conference on Learning Representations*, volume 2025, pp. 38057–38081, 2025a.

Yingsen Zeng, Zepeng Huang, Yujie Zhong, Chengjian Feng, Jie Hu, Lin Ma, and Yang Liu. Distime: Distribution-based time representation for video large language models. In *2025 IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 21961–21971. IEEE, 2025b.

Hao Zhang, Aixin Sun, Wei Jing, and Joey Tianyi Zhou. Span-based localizing network for natural language video localization. In *Proceedings of the 58th annual meeting of the association for computational linguistics*, pp. 6543–6554, 2020a.

Hao Zhang, Aixin Sun, Wei Jing, Liangli Zhen, Joey Tianyi Zhou, and Siow Mong Rick Goh. Parallel attention network with sequence matching for video grounding. In *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, pp. 776–790, 2021.

Jinglei Zhang, Yuanfan Guo, Rolandos Alexandros Potamias, Jiankang Deng, Hang Xu, and Chao Ma. Vtimecot: Thinking by drawing for video temporal grounding and reasoning. In *2025 IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 24203–24213. IEEE, 2025.

Jun Zhang, Teng Wang, Yuying Ge, Yixiao Ge, Xinhao Li, and Limin Wang. Timelens: Rethinking video temporal grounding with multimodal llms. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 10419–10429, 2026.

Songyang Zhang, Houwen Peng, Jianlong Fu, and Jiebo Luo. Learning 2d temporal adjacent networks for moment localization with natural language. In *Proceedings of the AAAI conference on artificial intelligence*, volume 34, pp. 12870–12877, 2020b.

Xing Zhang, Jiaxi Gu, Haoyu Zhao, Shicong Wang, Hang Xu, Renjing Pei, Songcen Xu, Zuxuan Wu, and Yu-Gang Jiang. Autotvg: A new vision-language pre-training paradigm for temporal video grounding. *arXiv preprint arXiv:2406.07091*, 2024.

Minghang Zheng, Xinhao Cai, Qingchao Chen, Yuxin Peng, and Yang Liu. Training-free video temporal grounding using large-scale pre-trained models. In *European Conference on Computer Vision*, pp. 20–37. Springer, 2024.

Minghang Zheng, Zihao Yin, Yi Yang, Yuxin Peng, and Yang Liu. Omnivtg: A large-scale dataset and training paradigm for open-world video temporal grounding. *arXiv preprint arXiv:2604.25276*, 2026a.

Zelin Zheng, Xinyan Liu, Ruixin Li, Antoni B Chan, Guorong Li, Qingming Huang, and Laiyun Qing. Foresee-to-ground: From predictive temporal perception to evidence-driven reasoning for video temporal grounding. *arXiv preprint arXiv:2605.21973*, 2026b.

Andy Zou, Long Phan, Sarah Chen, James Campbell, Phillip Guo, Richard Ren, Alexander Pan, Xuwang Yin, Mantas Mazeika, Ann-Kathrin Dombrowski, et al. Representation engineering: A top-down approach to ai transparency. *arXiv preprint arXiv:2310.01405*, 2023.

A Experimental Details

A.1 Training Data and Splits

The intervention modules are trained on 9,210 annotated video-query pairs: 5,000 examples from Charades-STA and 4,210 from ActivityNet Captions. The ActivityNet examples are selected to broaden the duration distribution observed during training. We split the data by video into 8,759 training and 451 validation examples, ensuring that no video appears in both sets. Evaluation annotations from Charades-STA, QVHighlights, and ActivityNet Captions are never used for model selection.

A.2 Intervention Layers

Guided by our preliminary literature review (Li et al., 2023; Rimsky et al., 2024), we pre-specify one intermediate intervention boundary for each backbone from its decoder depth before conducting the depth sweep and keep it fixed across all evaluation datasets. We use layer 14 for Qwen3-VL-2B, InternVL3.5-2B, and Qwen3.5-2B; layer 18 for Qwen3-VL-8B; and layer 19 for Qwen3.5-9B. All layer indices are zero-based; for example, layer 14 denotes an intervention after `layers[14]` and before `layers[15]`. The subsequent depth sweeps in Tables 3 and 6 are diagnostic sensitivity analyses and are not used to select or revise the main configurations.

A.3 Optimization and Model Selection

Unless otherwise stated, the vector bank contains $K = 8$ directions. For the $K = 1$ ablation in Figure 4, the routing weight is fixed to $p_{t,1} = 1$, and we disable routing-entropy and diversity regularization by setting $\mathcal{L}_{\text{sparse}} = \mathcal{L}_{\text{div}} = 0$. The Gate is trained as usual.

In Stages 2 and 3, we use gradient accumulation over eight examples. The Router and Gate use a learning rate of 2×10^{-4}, while the vector bank uses 2×10^{-5} during joint refinement. Across all backbones and datasets, we fix the routing temperature to $\tau = 1.0$ and the global intervention scale to $\alpha = 5.0$. In Stages 2 and 3, the balance loss uses a FIFO window of $B_r = 32$ examples. We set $(\lambda_s, \lambda_g, \lambda_b) = (0.001, 0.001, 0.05)$ and $(\lambda_a, \lambda_d) = (0.1, 0.01)$. In Stage 2, the Gate is frozen for the first 100 optimization steps, after which λ_g is linearly increased to its target over the next 100 steps. In Stage 3, λ_g remains fixed at 0.001. Within each run at its pre-specified intervention layer, checkpoints are selected by free-generation mIoU on the fixed 451-example, video-disjoint validation split. For every Base–TASA comparison, we keep the video processing, prompt, decoding strategy, and timestamp parser identical and use greedy decoding.

A.4 Implementation Details

Video processing and precision. Videos are sampled uniformly over their full duration at a target 2 FPS, with 4–768 frames. Spatial resolution follows each backbone's default video preprocessing without a fixed-size override. The frozen backbone uses BF16; the Router, Gate, and vector bank use FP32.

Prompt and timestamp parsing. The prompt is (with {`query`} replaced by the event query):

> To accurately pinpoint the event '{query}' in the video, determine the precise timeperiod of the event. Provide the start and end times in the format: 'From x to y'.

Ground-truth answers use timestamps in seconds with one decimal place: `From {start:.1f} to {end:.1f}`. The parser extracts `x` and `y` as start and end times. All Base and TASA outputs are parseable; no evaluated samples are excluded for parsing failure.

Vector-bank initialization. Stage 1 applies spherical K-means to normalized correction directions, using a randomly selected first center and farthest-point initialization for subsequent centers. We run at most 100 iterations with seed 42, using one run without restarts.

A.5 QVHighlights Evaluation

On the official QVHighlights `val` split, Base and TASA each generate one interval for every one of the 1,550 query-video pairs: 1,020 have a single ground-truth interval and 530 (34.19%) have multiple intervals. This split is used only for evaluation.

For prediction $p_q = [\hat{s}_q, \hat{e}_q]$ and annotations $g_{q,j} = [s_{q,j}, e_{q,j}]$, we sum intersections and annotated durations to compute

$$I_q = \sum_j \max(0, \min(\hat{e}_q, e_{q,j}) - \max(\hat{s}_q, s_{q,j}))\,,$$

$$D_q = (\hat{e}_q - \hat{s}_q) + \sum_j (e_{q,j} - s_{q,j}) - I_q, \qquad u_q = \begin{cases} I_q/D_q, & D_q > 0, \\ 0, & D_q \le 0. \end{cases}$$

The code sums annotated intervals without merging them, equivalent to union IoU when ground-truth intervals do not overlap.

With $N_{\mathrm{QV}} = 1550$, we aggregate equally over query-video pairs:

$$R@a = \frac{100}{N_{\mathrm{QV}}} \sum_{q=1}^{N_{\mathrm{QV}}} \mathbf{1}[u_q > a], \qquad \mathrm{mIoU} = \frac{100}{N_{\mathrm{QV}}} \sum_{q=1}^{N_{\mathrm{QV}}} u_q, \quad a \in \{0.3, 0.5, 0.7\}.$$

The evaluator uses strict $>$ thresholds and outputs percentages rounded to two decimal places.

A.6 Inference-Time Top-1 Ablation

Top-1 in Table 2 loads the same best Stage 3 vector-bank and Router/Gate checkpoint as full TASA, trained with dynamic soft routing. Only the inference routing rule changes; no parameters are reoptimized. During Stage 3, we accumulate detached soft routing probabilities over all teacher-forced answer-predictor positions, including `From`, timestamp digits, `to`, and EOS. The saved prior is the token-wise mean $\bar{p}_k = M^{-1} \sum_{r=1}^{M} p_{r,k}$, where M counts these observed positions; it does not count token-wise argmax decisions. Unlike the timestamp-only balance loss, this statistic includes formatting tokens. The accumulators reset each epoch, and the prior is updated at every validation point. Thus the best checkpoint stores the prior over the current epoch's training prefix up to that point, not necessarily a full-dataset average.

At inference, $k^* = \arg\max_k \bar{p}_k$ selects one vector shared by all examples and generation positions. The Router no longer selects directions, while the learned Gate is recomputed from the current hidden state:

$$h_t^{\ell'} = h_t^{\ell} + \alpha g_t v_{k^*}, \qquad g_t = \sigma(w_g^\top z_t + b_g), \qquad \alpha = 5.$$

The vector-bank and Gate parameters remain unchanged. Intervention uses the same zero-based layer 14 boundary, acting on the final prompt position during prefill and the current position at each subsequent cached decoding step. For the reported run, the selected vector is the second bank entry (zero-based implementation index 1; saved prior weight 0.2233). Its prior was saved at the fourth validation point of Stage 3 epoch 2, after 3,504 training examples had been processed in that epoch.

A.7 Comparison with Prior VTG Methods

Table 4 places the controlled results in Table 1 in the context of representative published VTG systems. The reported methods use different backbones, video features, training corpora, and parameter-update strategies, so the table is intended for broader context rather than as a controlled head-to-head comparison.

A.8 Trainable Parameters and Computational Scope

For a hidden size d and a vector-bank size K, the Router contains $Kd + K$ parameters, while the Gate contains $d+1$. Stage 2 therefore trains $N_{\mathrm{Stage2}} = (K+1)d+K+1$ parameters. Making the Kd entries of the vector bank trainable in Stage 3 increases the count to $N_{\mathrm{Stage3}} = (2K+1)d+K+1$.

Table 4: Comparison with representative prior VTG methods on Charades-STA and ActivityNet Captions. Results are taken from the cited papers under potentially different backbones and training protocols; – denotes unreported metrics, † denotes a later reproduction, and PT/ZS denote pretraining/zero-shot transfer.

	Charades-STA				ActivityNet Captions			
Method	R@0.3	R@0.5	R@0.7	mIoU	R@0.3	R@0.5	R@0.7	mIoU
Training-free (no task-specific optimization)								
TFVTG (Zheng et al., 2024)	67.04	49.97	24.32	44.51	49.34	27.02	13.39	34.10
Moment-GPT (Xu et al., 2025a)	58.20	38.40	21.60	36.50	48.10	31.10	14.90	30.80
GranAlign (Jeon et al., 2026)	59.10	39.60	22.70	38.00	50.30	34.00	16.50	33.10
T2SGrid (Guo et al., 2026)	71.40	47.00	20.70	44.90	44.80	26.80	16.10	32.50
FV-Action (Huang et al., 2026)	70.10	56.80	29.70	–	48.60	26.70	11.90	–
Task-trained methods (with or without pretrained-parameter updates)								
Classic specialized VTG models								
VSLNet (Zhang et al., 2020a)	70.46	54.19	35.22	50.02	63.16	43.22	26.16	43.19
SeqPAN (Zhang et al., 2021)	73.84	60.86	41.34	53.92	61.65	45.50	28.37	45.11
UniVTG w/ PT (Lin et al., 2023)	72.63	60.19	38.55	52.17	–	–	–	–
Video-LLM and recent adaptation methods								
VideoChat2† (Wang et al., 2024b)	71.80	56.40	27.70	48.30	55.50	34.70	17.70	38.90
HawkEye (Wang et al., 2024b)	72.50	58.30	28.80	49.30	55.90	34.70	17.90	39.10
ED-VTG (Pramanick et al., 2025)	78.20	62.10	35.00	52.60	67.60	45.10	22.70	44.90
NumPro-FT (Wu et al., 2025)	63.80	42.00	20.60	41.40	55.60	37.50	20.60	38.80
OmniVTG (ZS) (Zheng et al., 2026a)	78.30	63.20	37.00	–	60.30	39.80	21.40	–
Foresee-to-Ground (Zheng et al., 2026b)	71.30	50.40	25.70	47.20	63.80	46.10	28.40	45.70
T2SGrid-FT (Guo et al., 2026)	76.80	60.60	35.90	53.20	64.40	48.40	29.50	46.70

With $K = 8$, Stages 2 and 3 respectively train $18{,}441$ and $34{,}825$ parameters for $d = 2048$, and $36{,}873$ and $69{,}641$ parameters for $d = 4096$.

Beyond its small parameter count, TASA confines backward propagation to the decoder suffix after a single intervention boundary. The hidden state at this boundary is detached from the frozen prefix, so the visual encoder and preceding decoder blocks require only forward computation. No gradients or optimizer states are maintained for the backbone parameters, and no prefix activations need to be retained for backpropagation. Although the decoder suffix still participates in backward computation, this restricted scope makes training highly memory-efficient.

A.9 Comparison with Single-Layer LoRA

Table 5 compares TASA with single-layer LoRA (Hu et al., 2021) on Charades-STA, pairing the same backbone and intervention layer under comparable trainable-parameter budgets. The Qwen3-VL-8B L18 setting corresponds to the main experiments; Qwen3-VL-2B L3 is an additional depth setting. This comparison is restricted to single-layer adaptation and does not evaluate multi-layer LoRA.

Table 5: Single-layer LoRA comparison on Charades-STA with comparable trainable-parameter budgets. Layer indices are zero-based; bold marks the better value within each same-backbone, same-layer pair. All values are percentages.

Backbone	Layer	Method	R@0.3	R@0.5	R@0.7	mIoU
Qwen3-VL-2B	L3	LoRA	**76.53**	58.41	31.51	51.80
		TASA	76.29	**59.09**	**33.52**	**52.54**
Qwen3-VL-8B	L18	LoRA	**80.78**	65.22	39.60	56.13
		TASA	80.19	**66.18**	**40.38**	**56.27**

TASA obtains higher R@0.5, R@0.7, and mIoU at Qwen3-VL-2B L3 and Qwen3-VL-8B L18, while LoRA has higher R@0.3 in both cases.

A.10 INTERVENTION-LAYER ABLATION ON QWEN3.5-2B

We extend the intervention-layer ablation to Qwen3.5-2B on Charades-STA. Table 6 reports the frozen baseline and three intervention depths. L14 achieves the highest scores across all four metrics among the tested layers, differing from the L3 optimum for Qwen3-VL-2B in Table 3.

Table 6: Intervention-layer sensitivity of Qwen3.5-2B on Charades-STA. Base denotes the frozen backbone, and layer indices are zero-based. All values are percentages; the best results are shown in bold.

Layer	R@0.3	R@0.5	R@0.7	mIoU
Base	74.97	56.32	30.65	50.73
L3	79.14	64.78	38.84	55.44
L14	**79.87**	**65.73**	**41.29**	**56.44**
L21	78.84	62.90	36.64	54.51