electronics

MDPI

Article

Real-Time Anomaly Detection on Edge Devices via VLM Prompt Optimization

Sungmin Yu [1], Jongwon Moon [2] and Hosub Yoon [2,*]

[1] Department of Mechatronics Engineering, Korea University of Technology and Education (KOREATECH), Cheonan 31253, Republic of Korea; smyu9150@koreatech.ac.kr
[2] Electronics and Telecommunications Research Institute (ETRI), Daejeon 34129, Republic of Korea; dudskrk@etri.re.kr
* Correspondence: yoonhs@etri.re.kr

Abstract

Real-time video anomaly detection (VAD) under realistic edge constraints—sub-second latency, ≤25 W power, no cloud dependency, and human-interpretable output—remains an open problem. Existing lightweight video convolutional neural networks (X3D, MoViNets) are bound to closed-set training distributions, while recent vision–language-model-based VAD methods (LAVAD, VERA, Holmes-VAD) achieve 80–89% area under the curve (AUC) but rely on datacenter-grade GPUs and Chain-of-Thought (CoT) reasoning that pushes per-segment latency well above one second. This paper reframes the design target from peak accuracy to *practical edge deployability* and contributes two tightly coupled designs: (i) an edge-optimized inference stack that compresses Qwen3-VL-2B with 4-bit Activation-aware Weight Quantization (INT4 AWQ) and serves it through a TensorRT-LLM C++ runtime on NVIDIA Jetson Orin NX (16 GB, 25 W); and (ii) a fully automatic, CoT-free verbalized prompt optimization in which an 8B optimizer iteratively refines a natural-language definition block D_t using class-balanced (stratified) development batches on a disjoint development subset, with no human editing and no runtime cost on the edge device. Three findings support this framing: (a) the inference stack reduces per-segment latency to 0.25 s, a 7.4× speed-up and 55% memory reduction over a Python/PyTorch baseline; (b) verbalized prompt optimization improves zero-shot AUC from 71.82% (manual prompt) to 76.39%, outperforming GPT-4- and Gemini-Pro-generated prompts (74.12% and 74.35%) under the same edge backbone; and (c) single-frame input attains the highest mean AUC among one-, five-, and eight-frame windows—statistically comparable to the five-frame setting—while offering the lowest latency, making it the preferred operating point under the edge budget. While the absolute AUC (76.39%) is below recent server-side methods (CLIP-TSA 87.58%, VadCLIP 88.02%, Holmes-VAD 89.51%), our framework is the only one in this comparison that operates entirely on a ≤25 W edge device, providing a deployment-oriented operating point on the accuracy–feasibility frontier of VLM-based VAD.

Keywords: anomaly detection; edge computing; Jetson Orin NX; streaming processing; TensorRT-LLM; verbalized learning; prompt optimization; vision–language model

Academic Editors: Young-Ho Seo and John A. Kalomiros

Received: 9 June 2026
Revised: 18 July 2026
Accepted: 24 July 2026
Published: 27 July 2026

1. Introduction

Real-time video anomaly detection (VAD) from closed-circuit television (CCTV) streams is a core component of modern security systems [1,2]. Transmitting raw footage to cloud servers incurs network latency and privacy risk [3], motivating on-device

edge inference that exposes only the analysis results. Existing edge VAD largely relies on lightweight video convolutional neural networks (CNNs) such as X3D [4] and MoViNets [5], but these closed-set models fail to generalize to the diverse, open-set anomalies encountered in deployment.

Vision–language models (VLMs) offer a natural remedy: pre-trained on web-scale image–text data, they perform zero-shot classification from natural-language prompts alone [6]. Yet their multi-billion-parameter footprint, autoregressive decoding cost, and reliance on CoT reasoning make sub-second streaming inference on a $\leq$25 W edge device an open problem.

1.1. Positioning of This Work

We deliberately frame this paper as a *deployment-constraint study* rather than an accuracy-competition study. Recent server-side VAD methods such as CLIP-TSA [7], VadCLIP [8], VERA [9], and Holmes-VAD [10] reach 86–89% AUC on UCF-Crime, but assume datacenter-grade GPUs, multi-second per-clip inference budgets, and, in several cases, fine-tuning or instruction-tuning on large custom corpora. In contrast, the operating regime targeted here is bounded by (i) $\leq$25 W on-device power, (ii) sub-second per-segment latency, (iii) no cloud connectivity, (iv) no fine-tuning, and (v) human-interpretable output. Within this regime, we report 76.39% AUC, lower than unconstrained server-side methods, but obtained on a single Jetson Orin NX board with 0.25 s per-segment latency.

1.2. Definition of Zero-Shot in This Work

Throughout this paper, *zero-shot* means that **no gradient-based learning or parameter fine-tuning is performed on the target VAD task**. Prompt optimization is conducted strictly offline on a disjoint development subset and updates only the natural-language definition block D_t; it never updates model weights, never sees the test split, and never runs on the edge device at inference time.

1.3. Why Prompt Optimization Is a First-Class Contribution

Our verbalized optimization lifts the AUC of an INT4-quantized 2B VLM from 71.82% (manual prompt) to 76.39% (final converged D_{final})—a 4.57 percentage-point (pp) gain on UCF-Crime—without any change to the underlying model weights, the runtime, or the windowing strategy. This is the single largest improvement reported in any of our ablations and motivates the detailed account of how D_t is refined (Section 4.2).

The main contributions are as follows.

1. **TensorRT-LLM-Based Edge Inference Stack:** We compress Qwen3-VL-2B [11,12] with INT4 AWQ [13] and serve it through a TensorRT-LLM C++ runtime, reducing memory footprint by 55% and per-segment latency by 7.4$\times$ over a Python/PyTorch baseline, satisfying the streaming bound at the 25 W power mode of Jetson Orin NX with some margin.
2. **CoT-Free Verbalized Prompt Optimization with Stratified Batch Sampling:** We refine the prompt offline via verbalized learning [14,15] with an 8B optimizer, removing CoT [16] and constraining the output to a single classification token. The optimizer is steered by iterative verbal feedback on a class-balanced (stratified) development batch, runs strictly offline with no human editing, and is fully automatic. We provide the $D_0 \rightarrow D_{\text{final}}$ trajectory with validation AUC at representative stages to make the procedure auditable.
3. **Prompt Quality Beyond Off-the-Shelf LLM Rewriting:** Under an identical edge backbone, our iteratively optimized prompt outperforms single-shot prompts generated by GPT-4 (GPT-4o, model version gpt-4o-2024-08-06) and Gemini Pro (Gemini

3.1 Pro, model version gemini-3.1-pro), both accessed via their respective APIs, by 2.04–2.27 pp, demonstrating a practical advantage over the common practice of one-shot LLM prompting. We note that this comparison is not step-matched; the extent to which the gain stems from iterative refinement per se, as opposed to raw LLM capability, is not isolated here (see Section 5).

4. **Empirical Finding—Single-Frame Sufficiency Under Edge VLM Constraints:** In the specific regime of an INT4-quantized VLM served with a tightly constrained output budget, a single-frame input attains the highest mean AUC among one-, five-, and eight-frame windows while requiring the lowest latency. A paired statistical test (Section 5.3) shows that the W1–W5 accuracy gap is within run-to-run noise; the case for W1 therefore rests on its structural latency advantage, not on a claim of accuracy dominance.

The remainder of this paper is organized as follows. Section 2 reviews related work. Section 3 provides technical background. Section 4 details the proposed framework. Section 5 presents experimental results. Section 6 concludes the paper. Appendices A–C provide the optimizer prompt template, quantization configuration, and full software/hardware environment.

2. Related Work

2.1. Video Anomaly Detection

For comprehensive surveys of the broader VAD literature, see [17,18].

2.1.1. Unsupervised and Weakly Supervised Approaches

Early VAD studies modeled normality through reconstruction or prediction errors using autoencoders, sparse representations, and future-frame prediction [1,2,19]. Subsequent weakly supervised methods based on Multiple Instance Learning (MIL) significantly improved performance on UCF-Crime—most notably Sultani et al. [20] (75.41% AUC) and RTFM [21] (84.30% AUC)—but they remained bound to closed-set anomaly distributions and could not explain decisions in natural language.

2.1.2. CLIP- and VLM-Based VAD

The advent of large-scale VLMs enabled open-vocabulary VAD [22] that exploits the pre-trained image–text alignment of Contrastive Language–Image Pre-training (CLIP) [6]. CLIP-TSA [7] and VadCLIP [8] adapted CLIP to the temporal domain, reaching 87.58% and 88.02% AUC respectively. These methods, however, rely on server-grade GPUs and provide no human-interpretable rationale.

2.1.3. Explainable VAD: LAVAD, Holmes-VAD, and VERA

A recent line of work pursues explainability alongside accuracy. LAVAD [23] prompts an LLM to caption frames and judge anomalies (80.28% AUC), but per-frame LLM calls incur prohibitive latency. Holmes-VAD [10] instruction-tunes a multi-modal LLM, reaching 89.51% AUC at a memory cost incompatible with edge deployment.

VERA [9] applies verbalized machine learning [14,15] to VAD, yielding 86.55% AUC with interpretable rationales. Our work is closest in spirit to VERA but differs in four respects motivated by the edge regime: (i) we remove CoT generation and force a single-token output; (ii) we apply the optimizer *strictly offline*; (iii) we make the optimization fully automatic with no human editing and a class-balanced (stratified) development batch; and (iv) we evaluate the resulting prompt against prompts produced by general-purpose LLMs under an identical edge backbone.

More recently, several works have extended VLM-based VAD toward tighter latency budgets and resource-constrained deployment. MemoVAD [24] introduces a dynamic semantic memory to reduce redundant VLM calls in edge computing scenarios; Cerberus [25] adopts a cascaded design that filters routine frames with a lightweight model before invoking a VLM, targeting real-time throughput; and a recent benchmarking study [26] systematically compares compact VLMs for clip-level surveillance anomaly detection under weak supervision, reporting both accuracy and per-clip latency. These works share our motivation of reconciling VLM-based explainability with deployment constraints, though none reports results on a $\leq$25 W embedded board under the sub-second per-segment budget targeted here.

2.2. Edge-Oriented VAD and Limitations

Recent surveys highlight that most VAD research overlooks real-time deployment on resource-constrained edge devices [3].

2.2.1. Existing Edge VAD and Lightweight Video CNNs

Edge VAD has historically relied on X3D [4] and MoViNets [5], which fit Jetson-class budgets via efficient 3D convolution design. Their effectiveness is fundamentally bounded by the closed-world action vocabulary of pre-training datasets, which contain virtually no real-world anomalies.

2.2.2. Why VLM-on-Edge Is Hard

VLMs would, in principle, lift this closed-set constraint, but their deployment on edge devices is blocked by three coupled factors [27]: (i) *memory*—multi-billion-parameter weights leave little room within the 16 GB unified pool of Jetson Orin NX; (ii) *latency*—Python/PyTorch autoregressive decoding takes 1–2 s per segment; and (iii) *reasoning overhead*—CoT prompting further inflates the token budget. To date, peer-reviewed accounts of zero-shot explainable VAD operating entirely on a $\leq$25 W edge device with sub-second latency are scarce, and the framework proposed here directly targets this gap.

3. Technical Background

3.1. Vision–Language Models and Prompt Optimization

VLMs are pre-trained on large image–text corpora to acquire joint visual–linguistic representations. CLIP [6] aligns image and text via contrastive learning; LLaVA [28] connects a CLIP-style encoder to an instruction-tuned LLM; and the Qwen-VL series [11,12] provides compact dense variants—including Qwen3-VL at 2B/4B/8B/32B—with strong image and video understanding, making the 2B class a practical edge choice.

Verbalized Machine Learning (VML) [14,15] treats the prompt itself as the optimization variable, situated within the broader landscape of automatic prompt optimization techniques [29]. VERA [9] adapts VML to VAD with a two-part prompt $Q_t = [I_{\text{fixed}}, D_t]$, where I_{fixed} specifies the fixed role/output format and D_t is an updatable behavior definition. We adopt this structure but remove the CoT component, specialize D_t for binary classification, and apply f_{opt} strictly offline.

3.2. Edge Computing and Model Lightweighting

The target platform, NVIDIA Jetson Orin NX (16 GB), is an Ampere-class embedded board with up to 100 tera-operations per second (TOPS) at INT8 and selectable 10/15/25 W power modes.

Activation-aware Weight Quantization (AWQ) [13] preserves the most activation-salient weight channels during INT4 quantization, allowing near-lossless compression with substantial memory reduction. **TensorRT-LLM** provides optimized inference through kernel-level acceleration and memory-aware runtime optimization. We adopted the C++ runtime of TensorRT-LLM for end-to-end deployment. Recent evaluations and surveys of on-device LLM deployment [30,31] corroborate that INT4 quantization combined with optimized runtimes such as TensorRT-LLM is central to meeting edge power and latency budgets.

3.3. Real-Time Video Processing and Evaluation Protocol

We adopted a segment-based streaming protocol: a video is divided into segments of duration T_{segment} with overlap T_{overlap}, and the system must produce a decision before the next non-overlapping portion arrives:

$$T_{\text{decision}} = T_{\text{segment}} - T_{\text{overlap}}. \tag{1}$$

Real-time operation requires

$$T_{\text{decision}} \leq 1\,\text{s} \quad \text{and} \quad T_{\text{process}} \leq T_{\text{decision}}, \tag{2}$$

where T_{process} is the measured wall-clock time per segment.

4. Proposed Method

4.1. Overview and Novelty

The framework rests on three design choices, each addressing a distinct failure mode of prior edge VAD:

- *CoT-free verbalized prompt optimization* (Section 4.2) replaces the multi-token reasoning of VERA [9] with an offline-refined definition block (+4.57 pp over the manual baseline).
- *Single-frame inference as the default operating point* (Section 4.3): W1 attains the highest mean AUC and the lowest latency in the INT4-quantized VLM regime; its accuracy is statistically comparable to the 5-frame setting (Section 5.3), making W1 the rational choice on the accuracy–latency frontier.
- *Edge deployment via INT4 AWQ and TensorRT-LLM* (Section 4.4): compresses Qwen3-VL-2B to 1.2 GB and brings per-segment latency below the streaming bound of Equation (2).

The end-to-end system, illustrated in Figure 1, runs entirely on the NVIDIA Jetson Orin NX without any cloud dependency. A camera stream is sampled by one of three windowing strategies and fed to the quantized Qwen3-VL-2B served via TensorRT-LLM. The model emits a single-token binary decision (*Normal*/*Abnormal*) per segment, rendered directly in the GUI. A light Gaussian smoothing filter (window $=$ 30 frames, $\sigma = 0.5$) is applied to the resulting frame-level anomaly scores to reduce boundary noise when propagating segment-level decisions to individual frames (Section 4.3.3); no other temporal post-processing (e.g., retrieval-based ensembling, hysteresis thresholding) is used.

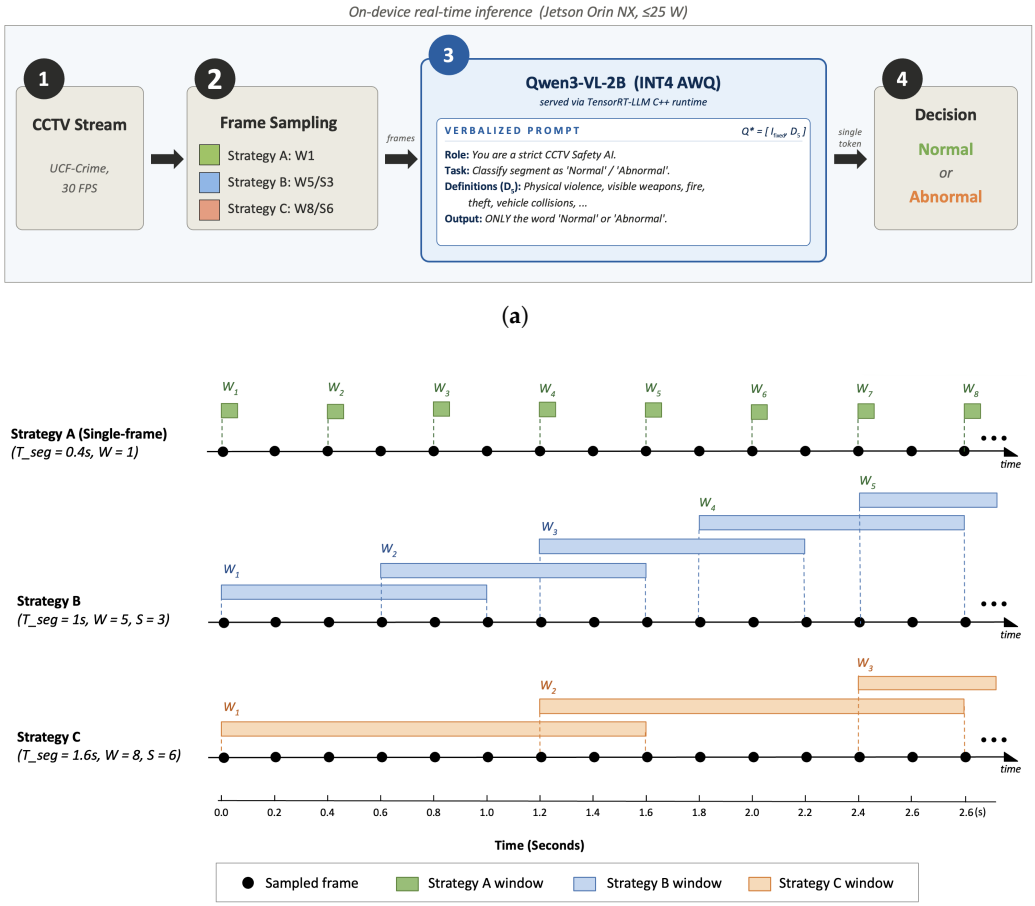


Figure 1. Proposed real-time streaming anomaly detection framework. (**a**) End-to-end pipeline running entirely on NVIDIA Jetson Orin NX. The CCTV stream is sampled by one of three windowing strategies, paired with the offline-optimized verbalized prompt $Q_t = [I_{\text{fixed}}, D_t]$, and fed to Qwen3-VL-2B (INT4 AWQ) served via the TensorRT-LLM C++ runtime. The model emits a single-token decision (*Normal*/*Abnormal*) per segment; a light Gaussian smoothing filter (window $= 30$, $\sigma = 0.5$) is applied to the resulting frame-level scores before thresholding, with no other temporal post-processing. Offline prompt optimization with an 8B optimizer (dashed block) is performed once before deployment and incurs no runtime cost. (**b**) Three windowing strategies on a common 0.2 s frame sampling grid: Strategy A (W1, $T_{\text{segment}} = 0.4$ s), Strategy B (W5/S3, $T_{\text{segment}} = 1.0$ s), and Strategy C (W8/S6, $T_{\text{segment}} = 1.6$ s).

4.2. Verbalized Prompt Optimization

4.2.1. Simple Prompt (Baseline)

The simplest instruction queries the presence of anomalies and constrains the output format (Table 1). Although broadly applicable, the term "abnormal events" is semantically ambiguous and induces frequent false positives in crowded or dynamic scenes. As shown in Section 5, this baseline reaches only 71.82% AUC on UCF-Crime.

Table 1. Detailed structure of the Simple Prompt (manual baseline).

System Instruction: Simple Prompt Structure
Instruction: Are there any abnormal events in the video? **Output Requirement:** Output ONLY the single word 'Normal' or 'Abnormal' without any punctuation.

4.2.2. Verbalized Prompt (Proposed)

The verbalized prompt assigns the model a strict CCTV-safety persona, enumerates concrete normal/abnormal categories, and *forbids* any output beyond a single classification token (Table 2). The definitions are not hand-crafted; they are the converged output (D_{final}) of the offline optimization procedure.

Table 2. Detailed structure of the Verbalized Prompt (deployed D_{final}).

System Instruction: Verbalized Prompt Structure
Role (I_{fixed}): You are a strict CCTV Safety AI that detects PROVEN anomalies only.
Task (I_{fixed}): Classify the video segment as 'Normal' or 'Abnormal'.
[Definitions (D_t)]
1. Normal: People shopping, working, or commuting; sports events, concerts, or public gatherings with crowd cheering; people walking normally through public spaces.
2. Abnormal: Physical violence, assault, or threats; visible weapons, fire, or explosion; actual theft, robbery, vandalism, or unauthorized access; vehicle collisions or significant property damage; crowd surges or panic that disrupt safety protocols.
Output Requirement (I_{fixed}): Output ONLY the single word 'Normal' or 'Abnormal' without any punctuation.

4.2.3. Output Format Design Considerations

Beyond the binary Normal/Abnormal format adopted here, recent work has explored richer output designs for VLMs. MVBench [32] formulates video understanding as multiple-choice question answering, converting open-ended reasoning into a closed answer set drawn from several candidate options; Vision-Semantics-Label [33] proposes a two-step mapping that first grounds visual content in natural-language semantic descriptions before mapping those descriptions to discrete action labels. Both designs could, in principle, be adapted to VAD by replacing the single Normal/Abnormal token with a small set of anomaly-category choices, or by inserting a semantics-then-label mapping stage. We did not adopt these formats here for two reasons tied directly to the edge regime targeted in this paper: (i) a multiple-choice format requires enumerating candidate categories in the prompt and lengthens the constrained output beyond a single token, increasing prefill and decode cost under the $\leq$25 W power budget; and (ii) a two-step semantics-then-label mapping introduces an additional generation pass, which is difficult to reconcile with the sub-second per-segment latency bound of Equation (2). Extending the verbalized optimization procedure to a constrained multiple-choice or two-step output format, and characterizing the resulting accuracy–latency trade-off, is a natural direction for future work.

4.2.4. Offline Optimization Procedure

Following the VML formulation [9,14,15], the prompt is decomposed as

$$Q_t = [I_{\text{fixed}}, D_t], \tag{3}$$

where I_{fixed} contains the role, task, and output constraint, and D_t contains the updatable definitions. An optimizer f_{opt} updates D_t from batch errors:

$$D_{t+1} = f_{\text{opt}}\big(D_t,\, V_{\text{batch}},\, \hat{Y}_{\text{batch}},\, Y_{\text{batch}};\, \psi\big), \tag{4}$$

where V_{batch} is a balanced batch of development clips, and $\hat{Y}_{\text{batch}}, Y_{\text{batch}}$ are the learner's predictions and ground-truth labels. We instantiate f_{opt} with **Qwen3-VL-8B** and the learner with **Qwen3-VL-2B (INT4 AWQ)** [11]. The full procedure is summarized in Algorithm 1.

Development subset construction. The development subset V_{dev} comprised 80 videos (40 normal, 40 abnormal) sampled uniformly from the UCF-Crime training split across all 13 anomaly categories, strictly disjoint from the 290-video official test split.

Algorithm 1 Offline verbalized prompt optimization.

Require: Learner f_{learn}, optimizer f_{opt}, fixed instruction I_{fixed}, development set V_{dev}, batch size B, max iterations $T_{\max}$, tolerance ϵ

Initialize D_0 via f_{opt} from I_{fixed} alone
Compute AUC_0 on V_{dev}
for $t = 0$ **to** $T_{\max} - 1$ **do**
  Sample stratified batch V_{batch} ($B/2$ normal, $B/2$ abnormal)
  $\hat{Y}_{\text{batch}} \leftarrow f_{\text{learn}}([I_{\text{fixed}}, D_t])$
  Propose $D_{t+1} \leftarrow f_{\text{opt}}(D_t, V_{\text{batch}}, \hat{Y}_{\text{batch}}, Y_{\text{batch}}; \psi)$
  Evaluate AUC_{t+1} on V_{dev}
  {ϵ flags practical convergence for reporting purposes only; the loop always runs the full $T_{\max}$ iterations}
end for
return D_t with highest validation AUC across full trajectory

Hyperparameters. We used $B = 16$, $T_{\max} = 16$, and $\epsilon = 0.2$ pp. The tolerance ϵ flagged practical convergence ($|\text{AUC}_{t+1} - \text{AUC}_t| < \epsilon$) for reporting purposes only; rather than terminating early at the first such flag, the full $T_{\max} = 16$ iterations were always run, and D_{final} was selected as the highest-validation-AUC variant across the complete trajectory $\{D_0, \ldots, D_{16}\}$ (Table 3).

Table 3. Progressive semantic refinement of the verbalized safety prompt across four representative stages ($W = 1$, single run on V_{dev}). AUC values reflect single-run validation on the development subset; the five-run test-split mean (76.39%) is measured separately on the held-out test set.

Stage	Semantic Characteristics	AUC (%)
D_0	*Simple ontology:* binary safety categories (violence, fire, theft)	71.82
D_5	*Semantic expansion:* public-event context, crowd panic, vehicle incidents added	72.54
D_{16}	*Context-aware:* scene-specific cues (casino, shoplifting, workstation)	74.78
D_{final}	*Converged policy:* focused threats + strict output constraint (deployed)	**76.24**

Bold indicates the deployed D_{final} configuration.

4.2.5. Optimization Signal: Stratified Batch Sampling

We steer f_{opt} using a class-balanced development batch: each iteration samples $B/2 = 8$ normal and $B/2 = 8$ abnormal clips (stratified batching), so that neither class dominates the verbal feedback given to the optimizer. The optimizer receives the batch predictions $\hat{Y}_{\text{batch}}$ and ground-truth labels Y_{batch} jointly (Equation (4)) and revises D_t accordingly; predictions are not separated into false-positive/false-negative subsets or diagnosed individually, and no rejection/retry mechanism is applied. Convergence is monitored via the overall validation AUC on V_{dev}.

4.2.6. Full $D_0 \rightarrow D_{\text{final}}$ Trajectory

Table 3 presents four representative stages illustrating qualitatively distinct semantic transitions. The validation-AUC trajectory is visualized in Figure 2.

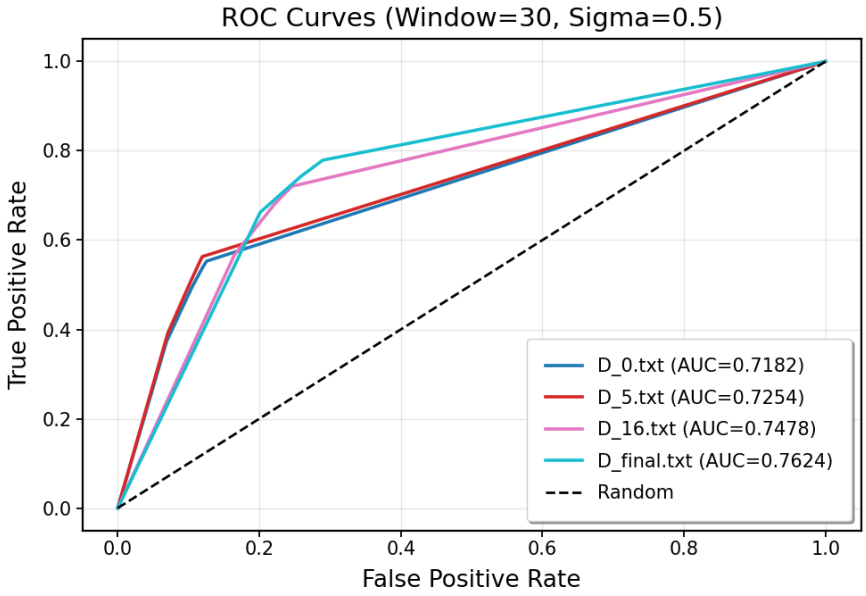


Figure 2. ROC curves for four representative prompt stages ($W = 1$, Qwen3-VL-2B INT4 AWQ, single run on V_{dev}). Each curve corresponds to a stage in the $D_0 \rightarrow D_{\text{final}}$ refinement trajectory; the legend reports the single-run validation AUC on V_{dev}. The five-run test-split mean AUC under D_{final} is 76.39% (Exp. A-2, Section 5.2).

4.2.7. No Human-in-the-Loop

All prompt refinements were generated automatically by the optimizer model. The authors fixed the optimization protocol once at the beginning of the procedure and did not edit D_t at any iteration, eliminating the risk that reported AUC gains came from authors' tacit knowledge of UCF-Crime.

4.3. Multi-Window Streaming Inference

4.3.1. Three Windowing Strategies

Three sliding-window configurations were evaluated (Table 4).

Table 4. Three sliding-window configurations evaluated under the edge VLM streaming protocol.

Strategy	T_{seg} (s)	T_{ovlp} (s)	T_{dec} (s)	Scope
W1 (1 frame)	0.4	0.0	0.4	Single keyframe
W5/S3 (5 frames)	1.0	0.4	0.6	Short-range
W8/S6 (8 frames)	1.6	0.4	1.2	Longer-range

All strategies shared a 0.2 s frame sampling interval. W1 and W5/S3 satisfied both conditions of Equation (2); W8/S6 satisfied $T_{\text{process}} \leq T_{\text{dec}}$ but its decision period ($T_{\text{dec}} = 1.2$ s) exceeded the 1 s real-time bound, and it was therefore included as a longer-range reference configuration rather than a deployable real-time setting.

4.3.2. Single-Frame Sufficiency: An Empirical Observation

In the specific regime studied here—an INT4-quantized 2B VLM served with a tightly constrained single-token output budget—a single-frame input (W1) attained the highest mean AUC and the lowest latency among the three windowing strategies across five repeated runs. The margin over W5 (0.12 pp) was small relative to the run-to-run standard deviations (0.16 and 0.07 pp); a paired statistical test (Section 5.3) confirmed that this accuracy difference was within run-to-run noise, whereas the latency advantage of W1 (0.25 s vs. 0.55 s per segment) was structural. We therefore characterized the finding as *single-frame sufficiency*: additional frames provided no measurable accuracy benefit in that regime while more than doubling latency. We emphasize that this finding is *regime-specific*;

the exact mechanism—whether related to visual token budget, prompt–visual alignment, or discriminative cues in UCF-Crime—was not directly measured and is left to future work.

4.3.3. Frame-Level Score Assignment and AUC Computation

The single-token decoder output was converted to a per-segment anomaly score $s_a \in [0, 1]$:

$$s_a = \frac{\exp(\ell_{\text{Abnormal}})}{\exp(\ell_{\text{Abnormal}}) + \exp(\ell_{\text{Normal}})}, \tag{5}$$

where ℓ_{Normal} and ℓ_{Abnormal} are the next-token logits of the first sub-token of "Normal" and "Abnormal". The per-segment score s_a was propagated to every frame covered by that segment; overlapping frames received the mean of covering segments' scores. To reduce frame-level boundary noise introduced by this propagation step, a Gaussian smoothing filter (window = 30 frames, $\sigma = 0.5$) was applied to the resulting frame-level score sequence prior to thresholding and AUC computation; no additional temporal post-processing (e.g., retrieval-based ensembling as in [9], or hysteresis thresholding) was applied. All AUC values reported in Section 5, including the headline 76.39% figure, were computed after this smoothing step. Frame-level AUC was computed against the official UCF-Crime per-frame ground truth [9,20,21,23].

4.4. Edge Deployment Optimization

4.4.1. INT4 AWQ Quantization

We compressed Qwen3-VL-2B with INT4 AWQ [13] using AutoAWQ. The language-model backbone was quantized to INT4 with group size 128, while the visual encoder was kept at FP16 to preserve fine-grained spatial features. Calibration used 200 short clips (∼1 s each) extracted from the 80 videos of V_{dev} (full configuration in Appendix B). The on-disk weight size shrank from ∼4.5 GB (FP16) to 1.2 GB, and total runtime memory stayed below 4 GB.

4.4.2. TensorRT-LLM C++ Runtime

The AWQ-quantized model was exported to ONNX (opset 17) and compiled into a TensorRT engine via the TensorRT-LLM toolchain, served through a C++ runtime benefiting from layer fusion, kernel auto-tuning, and Jetson-aware unified-memory allocation.

4.4.3. End-to-End Edge Performance

Measured end-to-end under W1 + verbalized (mean over five 290-video streaming passes, ∼12 h each):

- Latency: 1.85 s (Python/PyTorch FP16) → 0.25 s (TensorRT-LLM INT4), a 7.4× speed-up.
- Memory: 8.5 GB → 3.8 GB (55% reduction, peak resident set size via `tegrastats`).
- Throughput: ∼4 segments per second under W1.

5. Experimental Results

5.1. Evaluation Environment and Dataset Configuration

We evaluated on the official UCF-Crime [20] test split (150 normal, 140 anomalous videos). Since the framework is fully zero-shot, no training split was used for model optimization; the training split was only consulted to construct V_{dev} (80 videos), strictly disjoint from the 290-video test split.

Streaming evaluation. The CCTV stream was sampled at 30 FPS and partitioned into segments. The on-device pipeline processed each W1 segment in ∼0.25 s—a processing throughput of ∼4 segments per second—comfortably within the 0.4 s decision period of the

W1 configuration. The cumulative streaming time on Jetson Orin NX was approximately 12 h per pass, confirming sustained operation without thermal throttling.

Metric. The frame-level AUC of the receiver operating characteristic (ROC) curve, computed after the Gaussian smoothing step described in Section 4.3.3, is reported throughout, following standard practice [9,20,21,23].

Source of run-to-run variation. Although the learner used greedy decoding, the five repeated runs were not bit-identical because frame sampling offsets and segment boundaries varied slightly. Therefore, the resulting standard deviations (0.07–0.20 pp) characterize implementation-level noise.

5.2. Streaming Window and Prompt Optimization Performance Analysis

Table 5 reports the five-run results across all window–prompt combinations; Table 6 isolates the effect of quantization on the deployed configuration; and Table 7 reports quantization sensitivity across the three streaming windows. Verbalized prompt optimization was the dominant factor: AUC improved significantly under all window settings. The gain was especially pronounced under W5/S3 (+12.26 pp), confirming that a precisely worded D_t is essential for making compact edge VLMs feasible for VAD. Single-frame W1 (A-2, 76.39%) attained the highest mean AUC, though its margin over W5 (76.27%) was small relative to the run-to-run standard deviations (0.16 and 0.07 pp); Section 5.3 quantifies this comparison with a paired statistical test. Because W1 additionally more than halved the per-segment latency relative to W5 (0.25 s vs. 0.55 s), it remained the preferred operating point on the accuracy–latency frontier.

Table 5. Edge model repeated experimental results according to streaming window settings and prompt types (Qwen3-VL-2B INT4 AWQ, five-run validation). Mean latency denotes the average per-segment wall-clock time on Jetson Orin NX at 25 W. All AUC values were computed after the Gaussian smoothing step of Section 4.3.3.

Exp.	Window	Prompt	R1	R2	R3	R4	R5	Mean ± Std (%)	Lat.
A-1	W1 (0.4 s)	Simple	71.86	71.54	72.03	71.71	71.98	71.82 ± 0.20	0.20 s
A-2	**W1 (0.4 s)**	**Verbalized**	**76.24**	**76.20**	**76.45**	**76.49**	**76.55**	**76.39 ± 0.16**	**0.25 s**
B-1	W5/S3 (1.0 s)	Simple	64.01	63.88	64.15	63.92	64.09	64.01 ± 0.11	0.50 s
B-2	W5/S3 (1.0 s)	Verbalized	76.29	76.17	76.35	76.22	76.30	76.27 ± 0.07	0.55 s
C-1	W8/S6 (1.6 s)	Simple	63.25	63.11	63.39	63.20	63.28	63.25 ± 0.10	0.95 s
C-2	W8/S6 (1.6 s)	Verbalized	75.50	75.62	75.44	75.57	75.53	75.53 ± 0.07	0.96 s

The verbalized prompt slightly increased latency due to a longer input instruction (more visual–text cross-attention in the prefill phase) rather than longer generation, since the output was constrained to a single token. The bold row (A-2) marks the deployed configuration.

Table 6. Quantization performance comparison on the optimal configuration (A-2: W1 window, verbalized prompt). All conditions were identical except quantization precision.

Model Variant	Quant.	AUC (%)	Latency	Size	Retention
Qwen3-VL-2B	None (FP16)	76.52 ± 0.09	1.85 s	4.5 GB	—
Qwen3-VL-2B	**INT4 AWQ**	**76.39 ± 0.16**	**0.25 s**	**1.2 GB**	**99.83%**
Improvement		<0.2 pp loss	7.4× faster	3.75× smaller	—

All experiments were conducted under identical conditions (W1, verbalized prompt, five-run validation). Quantization achieved near-lossless accuracy with significant speed and memory improvements. Bold indicates the deployed INT4 AWQ configuration.

Table 7. Quantization sensitivity across streaming windows (verbalized prompt). Retention is INT4 AWQ AUC relative to FP16 AUC at the same window. W1 values represent five-run means (cf. Tables 5 and 6); W5/S3 and W8/S6 values denote single-run measurements.

Window	FP16 AUC (%)	INT4 AWQ AUC (%)	Retention (%)
W1 (deployed)	76.52	76.39	99.83
W5/S3	77.80	76.29	98.06
W8/S6	77.03	75.50	98.01

Quantization loss was the largest for the multi-frame windows (1.51 and 1.53 pp for W5/S3 and W8/S6, respectively) and smallest for the deployed single-frame setting (0.13 pp); retention remained ≥98% in all cases, indicating that INT4 AWQ compression did not catastrophically degrade any window, while the deployed W1 configuration was the most robust to quantization.

5.3. Statistical Comparison of W1 and W5

To assess whether the 0.12 pp mean-AUC gap between W1 (76.39% ± 0.16) and W5/S3 (76.27% ± 0.07) under the verbalized prompt reflected a genuine accuracy difference rather than run-to-run noise, we conducted a paired comparison across the five repeated runs reported in Table 5 (Exp. A-2 vs. Exp. B-2), pairing each run by its shared run index ($R1$–$R5$). The per-run differences (W1−W5) were −0.05, +0.03, +0.10, +0.27, and +0.25 pp, with mean 0.12 pp and standard deviation 0.14 pp. A paired t-test yielded $t(4) = 1.94$, $p = 0.12$, and an exact Wilcoxon signed-rank test (appropriate given the small sample size) yielded $W = 2$, $p = 0.19$. Neither test rejected the null hypothesis of equal AUC at the $\alpha = 0.05$ level, indicating that the accuracy difference between W1 and W5 was not statistically significant. The selection of W1 as the deployed configuration therefore rests on its structural latency advantage (0.25 s vs. 0.55 s per segment) and its marginally higher mean AUC, rather than on a claim of statistically established accuracy dominance.

5.4. AUC Comparison by Prompt Generation Source

Table 8 shows a 4.57 pp gain over the manual baseline and a further 2.04–2.27 pp gain over single-shot LLM prompts. We emphasize that this is not a step-matched, controlled comparison: GPT-4 and Gemini Pro were each queried once (single shot), reflecting how these models are typically used in practice for prompt drafting, whereas our approach performed iterative, learner-error-driven refinement over multiple rounds. The comparison therefore demonstrates the practical advantage of adopting verbalized learning over the common practice of single-shot LLM prompting, rather than isolating the effect of iteration count from the effect of the optimization method itself; disentangling these two factors (e.g., by running GPT-4 or Gemini Pro as the optimizer under the same number of iterative steps) is left to future work.

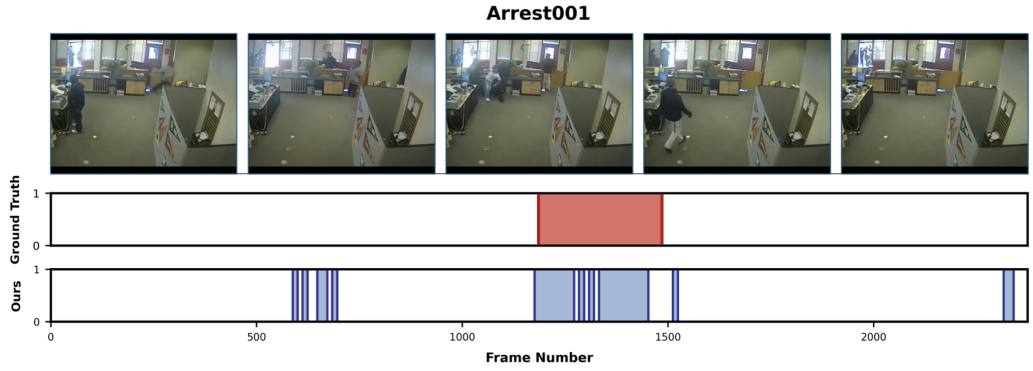
Table 8. AUC performance comparison according to prompt optimization method and generation source (UCF-Crime test split, edge backbone held constant).

Prompt Origin/Method	Optimizer	AUC (%)
Manual (simple prompt)	—	71.82
GPT-4 generated (single shot)	GPT-4	74.12
Gemini Pro generated (single shot)	Gemini Pro	74.35
Verbalized learning (ours, iterative)	**Qwen3-VL-8B**	**76.39**

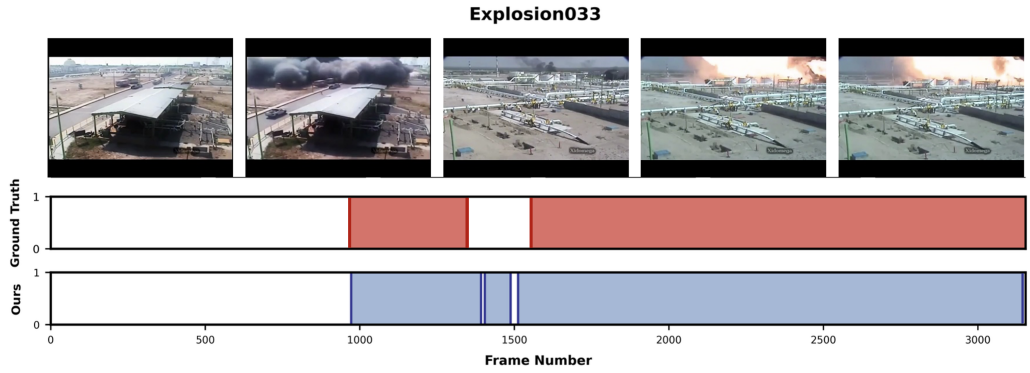
Bold indicates the proposed method. GPT-4 refers to GPT-4o (model version gpt-4o-2024-08-06) and Gemini Pro refers to Gemini 3.1 Pro (model version gemini-3.1-pro), each queried once via its API.

5.5. Qualitative Verification of Real-Time Operation

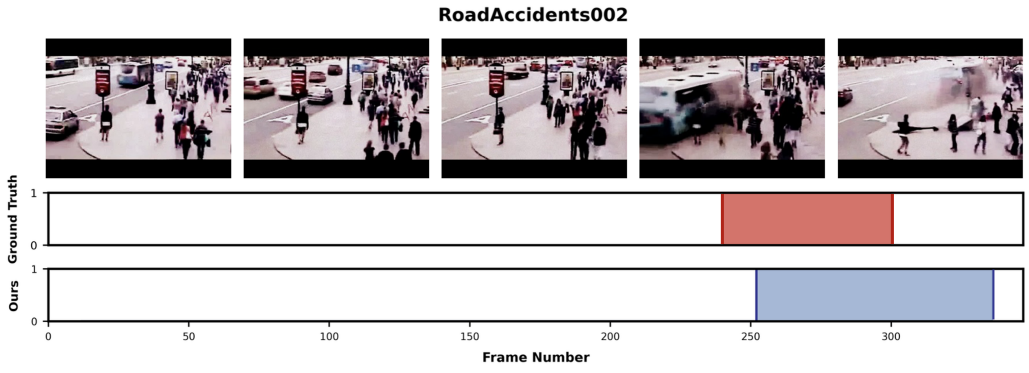
Figure 3 illustrates real-time stream processing on representative UCF-Crime samples. The system accurately recognizes situational context through internal VLM inference alone, sustaining ∼0.25 s per-segment latency without frame drops.



(**a**)



(**b**)



(**c**)

Figure 3. Ground-truth and predicted anomaly intervals on representative UCF-Crime samples. (**a**) Arrest001. (**b**) Explosion033. (**c**) RoadAccidents002. Red regions indicate ground-truth anomalous segments; blue regions represent predicted anomaly intervals, including the Gaussian smoothing described in Section 4.3.3. Results demonstrate temporally consistent anomaly localization across Arrest, Explosion, and Road Accident scenarios. All predictions were produced on Jetson Orin NX in the W1 + verbalized configuration.

5.6. Per-Category AUC Analysis

To identify which anomaly types the deployed configuration (W1, verbalized prompt) handles well and which it misses, and to reveal whether the AUC gain from prompt

optimization is consistent across anomaly categories, Table 9 reports the frame-level AUC for each of the 13 UCF-Crime anomaly categories at $W = 1$ for both the simple prompt (baseline) and the verbalized prompt (deployed), computed by restricting the anomalous test videos to a single category while retaining all 150 normal test videos, following common practice for per-category breakdowns [21]. These category-level values were measured on a single run and are reported here to characterize relative, cross-category behavior; the headline 76.39% figure remains the five-run mean of Table 5.

Table 9 shows that the AUC gain from prompt optimization is far from uniform across anomaly categories. The largest improvements occur in Robbery (+10.17 pp) and Stealing (+5.82 pp), categories centered on a clear possession-related action (an object being taken or handled) that the verbalized prompt's explicit category enumeration appears well suited to describe. In contrast, Vandalism (−10.85 pp), Assault (−8.72 pp), and Abuse (−6.77 pp) show a net decline under the verbalized prompt: for Vandalism and Assault, whose baseline AUC under the simple prompt was already comparatively high, the added category enumeration appears to remove rather than add discriminative signal, whereas Abuse performs poorly under both prompts. The remaining categories (Shooting, Road Accidents, Shoplifting, Fighting, Arrest, Explosion, Burglary, Arson) show comparatively small changes in either direction ($|\Delta| \leq 5$ pp). These results indicate that prompt optimization is not a uniformly beneficial intervention across anomaly types, and that categories whose anomalous cues were already captured well by the simple prompt may benefit less, or regress, from the added category enumeration. We emphasize that the specific mechanism behind this category-dependent behavior was not directly measured and is left to future work; the Abuse and Assault categories in particular warrant cautious interpretation given their small sample sizes ($n = 2$ and $n = 3$, respectively).

Table 9. Per-category frame-level AUC on the UCF-Crime test split at $W = 1$, comparing the simple prompt (baseline) and the verbalized prompt (deployed configuration, single-run measurement). Δ is the AUC gain from prompt optimization. n denotes the number of test videos in each category; categories with $n \leq 5$ should be interpreted with caution due to limited sample size.

Category	*n*	Simple (%)	Verbalized (%)	Δ (pp)
Robbery	5	67.53	77.70	+10.17
Stealing	5	72.94	78.76	+5.82
Shooting	23	54.86	55.97	+1.11
Road Accidents	23	62.65	63.62	+0.97
Shoplifting	21	60.43	60.34	−0.09
Fighting	5	55.20	54.00	−1.20
Arrest	5	69.37	67.62	−1.75
Explosion	21	65.89	63.17	−2.72
Burglary	13	69.42	65.78	−3.64
Arson	9	58.37	53.63	−4.74
Abuse	2	46.39	39.62	−6.77
Assault	3	82.40	73.68	−8.72
Vandalism	5	64.75	53.90	−10.85

5.7. Comparison with State-of-the-Art Methods on the Deployment Axis

We position 76.39% AUC at 0.25 s on a ≤25 W Jetson Orin NX as a distinct operating point on the accuracy–deployability frontier. Our system is the only method in Table 10 that simultaneously satisfies all five deployment constraints.

Table 10. Comparison of VAD methods on the deployment axis. Power figures are order-of-magnitude estimates inferred from reported pipelines, not direct measurements. FT: fine-tuning required; CF: cloud-free; Expl.: human-interpretable output; IT: instruction tuning. **Ours** is measured on Jetson Orin NX (25 W, W1). The proposed system is the only entry satisfying all five constraints simultaneously.

Method	Power	FT	CF	Expl.	Lat./Seg	AUC (%)
GCL [34]	>200 W	Yes	No	No	<1 s (CNN)	71.04
Sultani [20]	>200 W	Yes	No	No	<1 s (CNN+MIL)	75.41
LAVAD [23]	>200 W	No	No	Yes	≫1 s (VLM+LLM)	80.28
GCN [35]	>200 W	Yes	No	No	<1 s (CNN+MIL)	82.12
RTFM [21]	>200 W	Yes	No	No	<1 s (CNN+MIL)	84.30
MSL [36]	>200 W	Yes	No	No	<1 s (CNN+MIL)	85.30
VERA [9]	>200 W	No	No	Yes	>1 s (VLM+CoT)	86.55
CLIP-TSA [7]	>200 W	Yes	No	No	<1 s (CLIP)	87.58
VadCLIP [8]	>200 W	Yes	No	No	<1 s (CLIP)	88.02
Holmes-VAD [10]	>200 W	IT	No	Yes	≫1 s (MLLM)	89.51
Ours	**≤25 W**	**No**	**Yes**	**Yes**	**0.25 s (VLM)**	**76.39**

Bold indicates the proposed system.

6. Conclusions

This paper presented a deployment-constraint study of zero-shot video anomaly detection on edge devices. Within the target regime (≤25 W, sub-second latency, cloud-free, no fine-tuning, human-interpretable output), two contributions carried the framework. First, INT4 AWQ quantization combined with a TensorRT-LLM C++ runtime resolved the memory and latency bottlenecks of Qwen3-VL-2B on NVIDIA Jetson Orin NX, reducing per-segment latency to 0.25 s (7.4× over PyTorch) and memory footprint by 55%. Second, CoT-free verbalized prompt optimization—fully automatic, driven by a class-balanced (stratified) development batch, applied strictly offline, and auditable through the $D_0 \rightarrow D_{\text{final}}$ trajectory—was the single largest source of accuracy improvement (+4.57 pp over manual prompting, +2.04–2.27 pp over common-practice single-shot GPT-4/Gemini-Pro prompting).

A secondary finding is that single-frame input was *sufficient* in this regime: it attained the highest mean AUC (76.39% vs. 76.27% and 75.53% for five- and eight-frame windows) at the lowest latency, with the W1–W5 accuracy gap not statistically significant under a paired test (Section 5.3). The absolute AUC (76.39%) was below recent server-side methods, but those methods operate under fundamentally different deployment assumptions. We position our result as a distinct operating point on the accuracy–deployability frontier, suited to surveillance settings in which latency, power, privacy, and explainability are the binding constraints.

Limitations and Future Work

Sustained-runtime thermal measurements are planned. Cross-dataset transfer of D_{final} to ShanghaiTech and XD-Violence remains an open empirical question; the scope of this paper is deliberately limited to a single-dataset, single-device deployability study (see "Positioning of This Work," Section 1). The per-category analysis of Section 5.6 indicates category-dependent, non-uniform gains from prompt optimization whose underlying mechanisms warrant dedicated study, and the category-level measurements reported there are based on a single run rather than the five-run protocol used for the headline results; multi-run per-category evaluation is left to future work. We also plan to introduce dynamic window scheduling and extend the framework to multi-camera streams and adverse-weather conditions.

Author Contributions: Conceptualization, S.Y. and H.Y.; methodology, S.Y.; software, S.Y.; validation, S.Y. and J.M.; formal analysis, S.Y.; investigation, S.Y.; resources, H.Y.; data curation, S.Y.; writing—original draft preparation, S.Y.; writing—review and editing, J.M. and H.Y.; visualization, S.Y.; supervision, H.Y.; project administration, H.Y.; funding acquisition, H.Y. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by the Korea Planning & Evaluation Institute of Industrial Technology (KEIT) grant funded by the Korean government (MOTIE) (RS-2024-00442120, Development of AI technology capable of robustly recognizing abnormal and dangerous situations and behaviors during night and bad weather conditions). The APC was funded by KEIT/MOTIE.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: The UCF-Crime dataset is publicly available at https://www.crcv.ucf.edu/projects/real-world/ (accessed on 23 July 2026). The deployed optimized prompt D_{final} is provided in full in Table 2; the optimizer prompt template and the AWQ quantization configuration are provided in Appendices A and B, respectively.

Acknowledgments: The authors thank the ETRI Social Robotics Research Laboratory for providing the Jetson Orin NX hardware platform used in this study.

Conflicts of Interest: Authors Jongwon Moon and Hosub Yoon were employed by the Electronics and Telecommunications Research Institute (ETRI). The remaining authors declare that the research was conducted in the absence of any commercial or financial relationships that could be construed as potential conflicts of interest.

Abbreviations

AUC	Area Under the ROC Curve
AWQ	Activation-aware Weight Quantization
CCTV	Closed-Circuit Television
CLIP	Contrastive Language–Image Pre-training
CNN	Convolutional Neural Network
CoT	Chain-of-Thought
FP16	16-bit Floating Point
FPS	Frames Per Second
GUI	Graphical User Interface
INT4	4-bit Integer
LLM	Large Language Model
MIL	Multiple Instance Learning
MLLM	Multi-modal Large Language Model
ONNX	Open Neural Network Exchange
ROC	Receiver Operating Characteristic
TOPS	Tera-Operations Per Second
VAD	Video Anomaly Detection
VLM	Vision-Language Model
VML	Verbalized Machine Learning

Appendix A. Optimizer Prompt Template ψ

Table A1 reports the optimizer prompt template ψ used by the 8B optimizer at every iteration of Algorithm 1. The template was fixed throughout optimization; only the slot values ($\{D_t\}$, $\{V_{\text{batch}}\}$, $\{\hat{Y}_{\text{batch}}\}$, $\{Y_{\text{batch}}\}$) changed between iterations.

Table A1. Optimizer prompt template ψ used to drive f_{opt} at every iteration of Algorithm 1.

Optimizer Instruction: ψ
You are a senior CCTV safety policy designer. A small learner model uses the following anomaly-definition block to classify video segments as "Normal" or "Abnormal": **[Current definition block D_t]:** $\{D_t\}$
On a class-balanced batch of development clips ($B/2$ normal, $B/2$ abnormal), the learner produced the following predictions. **[Batch predictions]:** $\{\hat{Y}_{\text{batch}}\}$ **[Batch ground-truth labels]:** $\{Y_{\text{batch}}\}$
Your task is to propose a revised definition block D_{t+1} that improves classification accuracy on this batch and generalizes to unseen clips. **Constraints:** (1) Consider both missed anomalies and false alarms visible in the batch. (2) Do not relax the "PROVEN anomalies only" constraint. (3) Keep the block structurally identical to D_t (two sections: Normal and Abnormal, bullet-style enumeration). (4) Output ONLY the revised block. Do not include rationale, headers, or commentary.

Appendix B. AWQ Calibration Configuration

Table A2 lists the full AWQ calibration configuration used to quantize the deployed backbone.

Table A2. AWQ calibration configuration for Qwen3-VL-2B-Instruct.

Setting	**Value**
Quantization scheme	INT4, symmetric, per-group
Group size	128
Visual encoder	FP16 (not quantized)
Language backbone	All linear layers, INT4
Calibration source	V_{dev} (disjoint from test split)
Calibration clips	100 normal + 100 abnormal ($\sim$1 s each, 30 FPS)
Frames per clip	8 (uniformly sampled)
Calibration prompt	Identical to deployed Q_t (Table 2)
AWQ search	Default AutoAWQ (scale grid = 20)
Output format	GPTQ-style "awq"-packed weights

Appendix C. Software and Hardware Environment

Table A3 summarizes the software and hardware environment used for all experiments.

Table A3. Software and hardware environment used for all experiments.

Component	**Specification**
Device	NVIDIA Jetson Orin NX (16 GB)
GPU architecture	NVIDIA Ampere
Memory	16 GB LPDDR5 (unified CPU/GPU)
Power mode	MAXN_25W (25 W used)
JetPack	6.0 (L4T 36.3)
CUDA	12.2
cuDNN	8.9
TensorRT	8.6
TensorRT-LLM	v0.10 (C++ runtime)
Python (calib. only)	3.10
PyTorch (baseline only)	2.2
AutoAWQ	v0.2.5
Backbone model	Qwen3-VL-2B-Instruct (INT4 AWQ)
Optimizer model	Qwen3-VL-8B-Instruct (FP16, offline only)
Camera input rate	30 FPS, 320 $\times$ 240

References

1. Hasan, M.; Choi, J.; Neumann, J.; Roy-Chowdhury, A.K.; Davis, L.S. Learning temporal regularity in video sequences. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Las Vegas, NV, USA, 26 June–1 July 2016; pp. 733–742.
2. Liu, W.; Luo, W.; Lian, D.; Gao, S. Future frame prediction for anomaly detection—A new baseline. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Salt Lake City, UT, USA, 18–23 June 2018; pp. 6536–6545.
3. Le, H.; Lu, C.-K.; Hsu, C.-C. Video anomaly detection for edge-based IoT systems: A survey of input modalities and real-time applications. *Intell. Syst. Appl.* **2026**, *29*, 200635. [CrossRef]
4. Feichtenhofer, C. X3D: Expanding architectures for efficient video recognition. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Seattle, WA, USA, 13–19 June 2020; pp. 203–213.
5. Kondratyuk, D.; Yuan, L.; Li, Y.; Zhang, L.; Tan, M.; Brown, M.; Gong, B. MoViNets: Mobile video networks for efficient video recognition. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Nashville, TN, USA, 19–25 June 2021; pp. 16020–16030.
6. Radford, A.; Kim, J.W.; Hallacy, C.; Ramesh, A.; Goh, G.; Agarwal, S.; Sastry, G.; Askell, A.; Mishkin, P.; Clark, J.; et al. Learning transferable visual models from natural language supervision. In Proceedings of the International Conference on Machine Learning (ICML), Virtual, 18–24 July 2021; pp. 8748–8763.
7. Joo, H.K.; Vo, K.; Yamazaki, K.; Le, N. CLIP-TSA: CLIP-assisted temporal self-attention for weakly-supervised video anomaly detection. In Proceedings of the IEEE International Conference on Image Processing (ICIP), Kuala Lumpur, Malaysia, 8–11 October 2023; pp. 3230–3234.
8. Wu, P.; Zhou, X.; Pang, G.; Zhou, L.; Yan, Q.; Wang, P.; Zhang, Y. VadCLIP: Adapting vision-language models for weakly supervised video anomaly detection. In *Proceedings of the AAAI Conference on Artificial Intelligence*; AAAI Press: Washington, DC, USA, 2024; Volume 38, pp. 6074–6082.
9. Ye, M.; Liu, W.; He, P. VERA: Explainable video anomaly detection via verbalized learning of vision-language models. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Nashville, TN, USA, 11–15 June 2025; pp. 8679–8688.
10. Zhang, H.; Xu, X.; Wang, X.; Zuo, J.; Han, C.; Huang, X.; Gao, C.; Wang, Y.; Sang, N. Holmes-VAD: Towards unbiased and explainable video anomaly detection via multi-modal LLM. *arXiv* **2024**, arXiv:2406.12235.
11. Bai, S.; Cai, Y.; Chen, R.; Chen, K.; Chen, X.; Cheng, Z.; Deng, L.; Ding, W.; Gao, C.; Ge, C.; et al. Qwen3-VL technical report. *arXiv* **2025**, arXiv:2511.21631.
12. Bai, J.; Bai, S.; Yang, S.; Wang, S.; Tan, S.; Wang, P.; Lin, J.; Zhou, C.; Zhou, J. Qwen-VL: A versatile vision-language model for understanding, localization, text reading, and beyond. *arXiv* **2023**, arXiv:2308.12966.
13. Lin, J.; Tang, J.; Tang, H.; Yang, S.; Chen, W.-M.; Wang, W.-C.; Xiao, G.; Dang, X.; Gan, C.; Han, S. AWQ: Activation-aware weight quantization for LLM compression and acceleration. In Proceedings of the Machine Learning and Systems (MLSys), Santa Clara, CA, USA, 22–25 April 2024.
14. Xiao, T.Z.; Bamler, R.; Schölkopf, B.; Liu, W. Verbalized machine learning: Revisiting machine learning with language models. *Trans. Mach. Learn. Res.* **2025**. Available online: https://openreview.net/forum?id=k3Ab6RuJE9 (accessed on 10 July 2026). [CrossRef]
15. Yuksekgonul, M.; Bianchi, F.; Boen, J.; Liu, S.; Lu, P.; Huang, Z.; Guestrin, C.; Zou, J. Optimizing generative AI by backpropagating language model feedback. *Nature* **2025**, *639*, 609–616. [CrossRef] [PubMed]
16. Wei, J.; Wang, X.; Schuurmans, D.; Bosma, M.; Ichter, B.; Xia, F.; Chi, E.; Le, Q.; Zhou, D. Chain-of-thought prompting elicits reasoning in large language models. In *Advances in Neural Information Processing Systems (NeurIPS)*; Curran Associates: Red Hook, NY, USA, 2022; Volume 35, pp. 24824–24837.
17. Liu, J.; Liu, Y.; Lin, J.; Li, J.; Cao, L.; Sun, P.; Hu, B.; Song, L.; Boukerche, A.; Leung, V.C.M. Networking systems for video anomaly detection: A tutorial and survey. *ACM Comput. Surv.* **2025**, *57*, 1–37. [CrossRef]
18. Abdalla, M.; Javed, S.; Al Radi, M.; Ulhaq, A.; Werghi, N. Video anomaly detection in 10 years: A survey and outlook. *Neural Comput. Appl.* **2025**, *37*, 26321–26364. [CrossRef]
19. Lu, C.; Shi, J.; Jia, J. Abnormal event detection at 150 FPS in MATLAB. In Proceedings of the IEEE International Conference on Computer Vision (ICCV), Sydney, NSW, Australia, 1–8 December 2013; pp. 2720–2727.
20. Sultani, W.; Chen, C.; Shah, M. Real-world anomaly detection in surveillance videos. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Salt Lake City, UT, USA, 18–23 June 2018; pp. 6479–6488.
21. Tian, Y.; Pang, G.; Chen, Y.; Singh, R.; Verjans, J.W.; Carneiro, G. Weakly-supervised video anomaly detection with robust temporal feature magnitude learning. In Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV), Montreal, QC, Canada, 10–17 October 2021; pp. 4975–4986.
22. Wu, P.; Zhou, X.; Pang, G.; Sun, Y.; Liu, J.; Wang, P.; Zhang, Y. Open-vocabulary video anomaly detection. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Seattle, WA, USA, 16–22 June 2024; pp. 18297–18307.

23. Zanella, L.; Menapace, W.; Mancini, M.; Wang, Y.; Ricci, E. Harnessing large language models for training-free video anomaly detection. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Seattle, WA, USA, 16–22 June 2024; pp. 18527–18536.
24. Li, G.; Zeng, J.; Li, Y.; Peng, Z.; Chen, K.; Wang, T. MemoVAD: Resource-efficient video anomaly detection via dynamic semantic memory in edge computing scenarios. *arXiv* **2026**, arXiv:2606.07669.
25. Zheng, Y.; Shi, X.; Chen, J.; Shu, Y. Cerberus: Real-time video anomaly detection via cascaded vision-language models. *arXiv* **2025**, arXiv:2510.16290.
26. Borodin, K.; Kondrashov, K.; Vasiliev, N.; Gladkova, K.; Larina, I.; Gorodnichev, M.; Mkrtchian, G. Benchmarking compact VLMs for clip-level surveillance anomaly detection under weak supervision. *J. Imaging* **2025**, *11*, 400. [CrossRef] [PubMed]
27. Sharshar, A.; Khan, L.U.; Ullah, W.; Guizani, M. Vision-language models for edge networks: A comprehensive survey. *IEEE Internet Things J.* **2025**, *12*, 32701–32724. [CrossRef]
28. Liu, H.; Li, C.; Wu, Q.; Lee, Y.J. Visual instruction tuning. In *Advances in Neural Information Processing Systems (NeurIPS)*; Curran Associates: Red Hook, NY, USA, 2023; Volume 36.
29. Ramnath, K.; Zhou, K.; Guan, S.; Mishra, S.S.; Qi, X.; Shen, Z.; Wang, S.; Woo, S.; Jeoung, S.; Wang, Y.; et al. A systematic survey of automatic prompt optimization techniques. In Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing (EMNLP), Suzhou, China, 4–9 November 2025; pp. 33078–33110. [CrossRef]
30. Song, Q.; Liu, R.; Lin, W.; Liao, P.; Zhao, W.; Wang, Y.; Hu, S.; Jiang, Y.; Long, M.; Zhen, H.-L.; et al. A systematic evaluation of on-device LLMs: Quantization, performance, and resources. *arXiv* **2025**, arXiv:2505.15030.
31. Kristiani, E.; Verma, V.K.; Yang, C.-T. Deploying LLM transformer on edge computing devices: A survey of strategies, challenges, and future directions. *AI* **2026**, *7*, 15. [CrossRef]
32. Li, K.; Wang, Y.; He, Y.; Li, Y.; Wang, Y.; Liu, Y.; Wang, Z.; Xu, J.; Chen, G.; Luo, P.; et al. MVBench: A comprehensive multi-modal video understanding benchmark. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Seattle, WA, USA, 16–22 June 2024; pp. 22195–22206.
33. Chen, X.; Xu, W.; Kan, S.; Zhang, L.; Jin, Y.; Cen, Y. Vision-Semantics-Label: A new two-step paradigm for action recognition with large language model. *IEEE Trans. Circuits Syst. Video Technol.* **2026**, *36*, 1313–1327. [CrossRef]
34. Zaheer, M.Z.; Mahmood, A.; Khan, M.H.; Segù, M.; Yu, F.; Lee, S.-I. Generative cooperative learning for unsupervised video anomaly detection. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), New Orleans, LA, USA, 18–24 June 2022; pp. 14724–14734.
35. Zhong, J.-X.; Li, N.; Kong, W.; Liu, S.; Li, T.H.; Li, G. Graph convolutional label noise cleaner: Train a plug-and-play action classifier for anomaly detection. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Long Beach, CA, USA, 15–20 June 2019; pp. 1237–1246.
36. Li, S.; Liu, F.; Jiao, L. Self-training multi-sequence learning with transformer for weakly supervised video anomaly detection. In Proceedings of the AAAI Conference on Artificial Intelligence, Virtual, 22 February–1 March 2022; pp. 1395–1403.