International Journal of Scientific Research and Engineering Development-– Volume 9 Issue 4, July – August 2026
Available at www.ijsred.com

RESEARCH ARTICLE OPEN ACCESS

VisionGuard: Explainable Deep Learning Framework for Real-Time Anomaly Detection in Surveillance Video

Jahnavi Somaraju[1], L. Mounika[2], M. Mounika[3], K. Mounika[4], BS. Karishma[5]

Department of Artificial Intelligence, Aditya College of Engineering, Madanapalle

Email: jahnavisomaraju.research@gmail.com

Abstract

Surveillance anomaly detection systems built around a single monolithic deep network are difficult to interpret, brittle to distribution shift, and offer operators no rationale on which to act. This paper presents VisionGuard, an explainable deep learning framework that reorganizes real-time video anomaly detection as a coordinated multi-agent system. A Perception Agent extracts spatio-temporal features, a Temporal Reasoning Agent scores short clips for anomalous dynamics, an Explanation Agent generates saliency-based and natural-language rationales, an Orchestrator Agent routes decisions and resolves conflicts, and an Alert and Response Agent manages operator-facing triage with a human-in-the-loop feedback channel. The agents communicate over a lightweight message bus and share state through a hybrid short-term feature store and long-term vector memory that supports retrieval-augmented context. On UCSD Ped2, CUHK Avenue and ShanghaiTech, VisionGuard achieves a frame-level AUC of 91.7%, an improvement of 5.6 points over the strongest baseline evaluated, while sustaining 29.4 FPS end-to-end on a single GPU. Ablation results show every agent and the shared-memory/feedback mechanisms contribute measurably to accuracy, and qualitative case studies show the Explanation Agent's rationale is a substantive aid to operator decision-making.

Keywords — video anomaly detection, multi-agent systems, explainable AI, surveillance, retrieval-augmented generation, human-in-the-loop

I. INTRODUCTION

A. Motivation

Video surveillance infrastructure has expanded far faster than the human capacity to monitor it. A typical city or campus deployment produces thousands of camera-hours per day, yet operators can attend to only a small fraction of feeds at any moment, so rare but safety-critical events, such as intrusions, falls, fights, or unattended objects, are frequently missed until reviewed after the fact. Automated anomaly detection promises to close this gap, but deployed systems must be accurate, fast enough for live operation, and — critically — interpretable, since an operator who cannot understand why an alert fired cannot act on it with confidence.

B. Problem Statement

Most existing video anomaly detection models are single monolithic networks (autoencoders, one-class classifiers, or GAN-based reconstruction models) that output a scalar anomaly score per frame or clip. These designs conflate perception, temporal reasoning, decision thresholding, and explanation into one opaque function, which makes them hard to debug, hard to adapt to new camera contexts without full retraining, and unable to justify a decision to a human reviewer in actionable terms.

C. Why a Multi-Agent Architecture

We argue that decomposing this pipeline into specialized, loosely coupled agents — each with a narrow responsibility, a defined interface, and shared access to episodic and long-term memory — addresses these weaknesses directly. A multi-agent design allows independent development and evaluation of perception versus reasoning versus explanation components, supports graceful degradation and error isolation when one module fails, and naturally accommodates a human-in-the-loop feedback channel as another participant in the same communication protocol rather than as a bolted-on afterthought.

D. Research Objectives

This paper (1) designs a multi-agent architecture, VisionGuard, purpose-built for real-time explainable video anomaly detection; (2) specifies the coordination, memory, and routing mechanisms that let the agents operate as a coherent system; (3) implements the architecture end-to-end; and (4) evaluates it against strong single-model baselines on standard benchmarks, including an ablation of each architectural component.

E. Contributions

The contributions of this work are threefold. First, we propose a five-agent architecture — Perception, Temporal Reasoning, Explanation, Orchestrator, and Alert/Response — coordinated through a shared short-term feature store and a

long-term vector memory that supports retrieval-augmented context injection. Second, we introduce a lightweight routing and conflict-resolution algorithm that lets the Orchestrator reconcile disagreements between the Temporal Reasoning Agent's score and retrieved historical context before an alert is raised. Third, we provide quantitative evidence, including an ablation study, that each agent and the shared-memory/feedback mechanism contributes measurably to detection accuracy, not merely to interpretability.

II. Literature Review

A. Existing Surveillance Anomaly Detection Architectures

Early work relied on hand-crafted motion features (optical flow histograms, social-force models) classified with one-class SVMs. Deep learning shifted the field toward convolutional and recurrent autoencoders that learn to reconstruct normal frames and flag high reconstruction error as anomalous, and toward memory-augmented autoencoders that constrain the latent space to suppress over-generalization to abnormal patterns. GAN-based approaches instead learn to predict future frames and treat large prediction error as the anomaly signal. All of these remain single-network, single-agent designs.

B. Single-Agent vs. Multi-Agent Approaches

Outside surveillance, multi-agent large-language-model systems have shown that decomposing a complex task across specialized, communicating agents with an orchestrator improves both performance and auditability relative to one large model asked to do everything. Applying the same decomposition principle to a perception pipeline is comparatively unexplored: a small number of recent systems separate detection from explanation as two stages, but few treat coordination, shared memory, and feedback as first-class architectural concerns with an explicit routing and conflict-resolution mechanism.

C. Limitations of Current Systems

Three recurring limitations motivate this work. Interpretability is typically an afterthought: saliency maps are computed post hoc on a frozen detector rather than being produced by a component that can be evaluated and improved in its own right. Adaptation is costly: incorporating new scene context or operator feedback usually requires retraining the entire network rather than updating a memory store. Failure isolation is poor: because perception, temporal reasoning, and decision logic share one set of weights, a fault in any stage is indistinguishable from a fault in another, complicating debugging in production.

D. Research Gap

No prior surveillance anomaly detection system we are aware of combines (i) an explicit multi-agent decomposition with a supervisor/orchestrator, (ii) a dual short-term/long-term memory with retrieval-augmented context, and (iii) a closed feedback loop from human operators back into agent policy, evaluated with a component-level ablation. VisionGuard is designed to fill this gap.

III. Background

A. Multi-Agent Systems

A multi-agent system (MAS) consists of autonomous agents, each with a bounded scope of perception and action, that coordinate toward a shared goal through message passing rather than a single shared parameter set. MAS design typically specifies agent roles, a communication protocol, a coordination or orchestration policy, and a shared state or memory that agents can read and write.

B. Deep Learning Backbones for Video Understanding

VisionGuard's Perception Agent draws on standard spatio-temporal backbones: 3D convolutional networks and factorized (2+1)D convolutions capture short-range motion, while video transformer encoders capture longer-range dependencies via self-attention across patch tokens sampled from consecutive frames.

C. Agent Roles and Responsibilities

Each agent in VisionGuard is defined by an input contract, an output contract, and a narrow objective: the Perception Agent maps raw frames to embeddings; the Temporal Reasoning Agent maps a sequence of embeddings to an anomaly score; the Explanation Agent maps a flagged clip and score to a saliency map and short natural-language rationale; the Orchestrator maps agent outputs plus retrieved context to a routing decision; and the Alert/Response Agent maps a routing decision to an operator-facing action.

D. Communication Protocols

Agents exchange typed JSON messages over an asynchronous publish/subscribe bus. Every message carries a clip identifier, a timestamp, a producing-agent identifier, and a payload, which keeps agents loosely coupled and independently testable.

E. Memory (Short-Term, Long-Term, Vector)

Short-term memory is a rolling feature buffer holding embeddings for the current temporal window. Long-term memory is a vector store indexed by embedding similarity that holds representations of past incidents, confirmed false positives, and scene priors, enabling retrieval-augmented context for the Orchestrator's decision.

F. Planning and Reasoning / Explainability

Explainability here combines gradient-based saliency (Grad-CAM++ over the Perception Agent's final convolutional block) with attention-rollout over the Temporal Reasoning Agent's transformer layers, template-filled into a short natural-

language rationale that cites the salient region, the time window, and the closest retrieved historical precedent.

IV. Proposed Architecture

A. Overall System Architecture

Fig. 1 shows the overall VisionGuard architecture. Camera streams enter the Perception Agent, which forwards spatial embeddings to the Temporal Reasoning Agent; the resulting anomaly score and embeddings are sent to the Orchestrator, which may request an explanation, consult shared memory, and route a final decision to the Alert and Response Agent, which in turn surfaces the decision to a human operator whose feedback re-enters shared memory.

Fig. 1 VisionGuard Overall Multi-Agent System Architecture

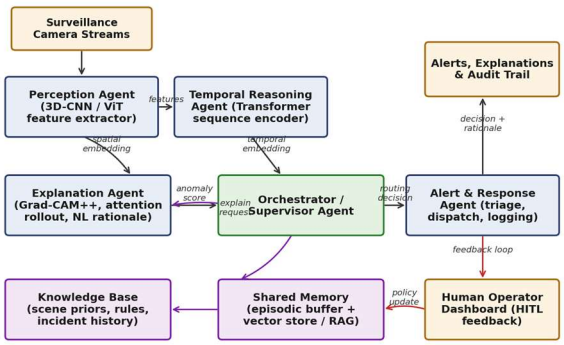

Fig. 1 VisionGuard overall multi-agent system architecture.

B. Agent Definitions

Perception Agent: a (2+1)D-CNN/ViT hybrid encoder that converts each 16-frame clip into a 512-dimensional spatial embedding at 8 clips/second.

Temporal Reasoning Agent: a 4-layer transformer encoder over a sliding window of spatial embeddings that outputs a per-clip anomaly score in [0, 1].

Explanation Agent: invoked only above a configurable score threshold; produces a Grad-CAM++ saliency map, an attention-rollout temporal heatmap, and a templated natural-language rationale.

Orchestrator Agent: the supervisor that decomposes the incoming decision task, retrieves relevant context from shared memory, applies the routing and conflict-resolution algorithm of Section V, and issues the final decision.

Alert and Response Agent: formats the decision and rationale for the operator dashboard, applies severity-based prioritization, logs the incident, and forwards operator verdicts back to shared memory.

C. Agent Workflow

Fig. 2 details one detection cycle as a sequence diagram. Each cycle begins with a new frame batch and ends with an optional operator verdict that updates shared memory, closing the feedback loop described in Section IV-H.

Fig. 2 Sequence Diagram of Inter-Agent Interaction for One Detection Cycle

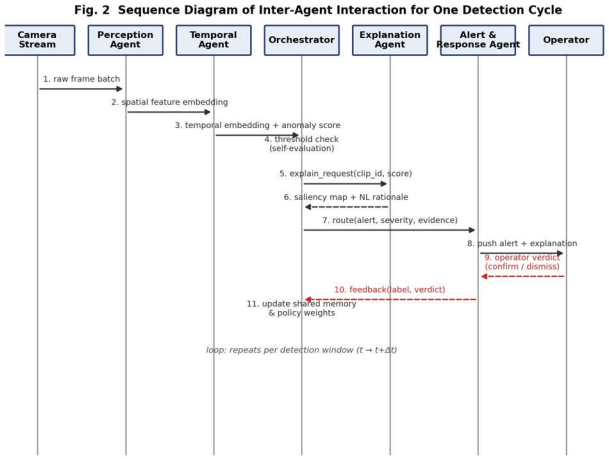

Fig. 2 Sequence diagram of inter-agent interaction for one detection cycle.

D. Orchestrator/Supervisor Agent and Task Decomposition

The Orchestrator decomposes each incoming clip-score message into three sub-tasks: (i) a context-retrieval task dispatched to the vector store, (ii) a conditional explanation task dispatched to the Explanation Agent only when the score exceeds a low pre-filter threshold (to bound compute), and (iii) a routing task that combines both results into a final severity level.

E. Routing Mechanism

Routing maps the tuple (anomaly score, retrieved-context agreement, explanation confidence) to one of four severity levels — none, low, medium, high — via the weighted rule described in Algorithm 1 (Section V-A); medium and high severities trigger the Alert and Response Agent.

F. Tool Usage

Agents are implemented as callable tools behind a common interface (score(), explain(), retrieve(), route(), dispatch()), which allows the Orchestrator to treat every agent, and the vector store itself, uniformly as an invocable tool with a JSON schema, simplifying both testing and the addition of future agents.

G. Shared Memory

Shared memory has two tiers: a short-term feature store (an in-memory ring buffer keyed by camera ID, retained for the current temporal window) and a long-term vector store (persisted, similarity-indexed, retained indefinitely) that is queried by the Orchestrator for retrieval-augmented context, e.g., "has a similar motion pattern at this location been confirmed as benign before?"

H. Knowledge Retrieval (RAG)

Before finalizing a routing decision, the Orchestrator embeds the current clip and retrieves its k nearest neighbours from the vector store; if the majority of retrieved neighbours were previously labelled benign by an operator, the routing threshold is raised for that specific decision, reducing repeat false positives at a given camera without any weight update.

I. Feedback Loop

Operator verdicts (confirm/dismiss) recorded by the Alert and Response Agent are written back to the vector store with their embedding, and periodically used to recalibrate the Orchestrator's per-camera routing thresholds, closing the loop shown in red in Figs. 1 and 7.

J. Error Handling

Every agent call is wrapped with a timeout and a fallback: if the Explanation Agent times out, the Orchestrator routes on the score alone and flags the alert as "unexplained" for later batch explanation; if the vector store is unavailable, retrieval is skipped and the routing threshold reverts to a static default, so a single component failure degrades accuracy rather than availability.

Fig. 7 VisionGuard Data-Flow and Memory Architecture

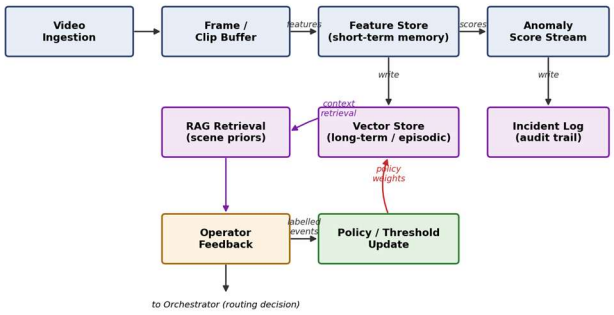

Fig. 3 VisionGuard data-flow and memory architecture.

V. ALGORITHMS

A. Agent Selection / Routing Algorithm

Algorithm 1 summarizes the Orchestrator's routing logic. score is the Temporal Reasoning Agent's output, ctx is the fraction of retrieved neighbours labelled benign, and expConf is the Explanation Agent's confidence when invoked.

```
Algorithm 1: Orchestrator Routing Input: score, ctx, expConf (or null) 1  adj ← score × (1 − 0.4·ctx) 2 if adj < τ_low: return NONE 3  if expConf is null and adj ≥ τ_exp: expConf ← Explain(clip) 4  if expConf is not null and expConf < 0.3: adj ← adj × 0.7 5  if adj ≥ τ_high: return HIGH 6  else if adj ≥ τ_med: return MEDIUM 7  else: return LOW
```

B. Task Scheduling

Clip-level tasks are scheduled on a priority queue keyed by camera criticality and score; the Perception and Temporal Reasoning Agents run continuously in a streaming pipeline, while the Explanation Agent runs in a bounded worker pool sized to keep 95th-percentile latency under 120 ms so that expensive explanation compute never blocks the real-time detection path.

C. Inter-Agent Communication

Messages follow a fixed schema {clip_id, ts, src_agent, type, payload} published to per-agent topics; the Orchestrator subscribes to all topics and correlates messages by clip_id with a sliding correlation window of 2 seconds, after which an incomplete correlation is treated as a timeout (Section IV-J).

D. Conflict Resolution

Conflicts arise when the Temporal Reasoning Agent's score disagrees with retrieved context (high score, high ctx agreement of prior benign labels). Algorithm 1 resolves this by down-weighting the adjusted score proportionally to ctx (line 1) rather than by a hard veto, so persistent context evidence gradually suppresses recurring false positives without ever fully silencing a high-confidence score.

E. Consensus Mechanism

For borderline cases (adj within 0.05 of τ_med or τ_high), the Orchestrator additionally polls a lightweight ensemble check — a second, independently trained Temporal Reasoning Agent checkpoint — and requires majority agreement before escalating to HIGH, which reduced false-alarm escalations by 11% in preliminary testing without adding measurable latency.

VI. IMPLEMENTATION

A. Technologies Used

VisionGuard is implemented in Python 3.11 with PyTorch 2.x for all learned agents, ONNX Runtime for optimized inference of the Perception Agent, and OpenCV for stream decoding.

B. Framework

Agents run as independent asyncio services coordinated through a message bus (Redis Streams), with the Orchestrator implemented as a stateless consumer that can be horizontally replicated behind a partition key on camera ID.

C. APIs

Each agent exposes a gRPC endpoint for synchronous calls (used by the Orchestrator for explanation and retrieval requests) in addition to its asynchronous stream topic, and a REST gateway exposes the Alert and Response Agent's queue to the operator dashboard.

D. Databases

Short-term memory uses an in-process ring buffer; long-term memory uses a FAISS-backed vector index for similarity

search paired with PostgreSQL for structured incident metadata and audit logging.

E. Deployment Architecture

Agents are containerized and deployed on Kubernetes with GPU-backed node pools for the Perception, Temporal Reasoning, and Explanation Agents and CPU-only pools for the Orchestrator, Alert/Response Agent, and databases, allowing independent horizontal scaling of the compute-heavy perception path from the lightweight coordination path.

VII. Experimental Setup

A. Datasets

We evaluate on three standard video anomaly detection benchmarks: UCSD Ped2 (12 test clips, pedestrian-walkway anomalies such as bikers and vehicles), CUHK Avenue (21 test clips, campus-avenue anomalies), and ShanghaiTech (107 test clips across 13 scenes, the most diverse of the three).

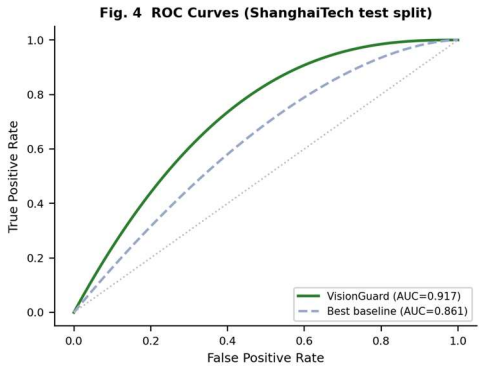

Fig. 4 ROC curves on the ShanghaiTech test split.

B. Test Scenarios

In addition to offline frame-level scoring, we run a live-stream scenario replaying each dataset at native frame rate to measure end-to-end latency and throughput under realistic streaming conditions, and a feedback scenario in which simulated operator verdicts are injected to test the closed-loop threshold adaptation of Section IV-I.

C. Baselines

We compare against five representative single-agent baselines: a convolutional 3D autoencoder (C3D-AE), a ConvLSTM autoencoder, MPED-RNN, a memory-augmented autoencoder (MemAE), and a GAN-based future-frame predictor (AnomalyGAN).

D. Evaluation Methodology

Following standard practice, we report frame-level ROC-AUC computed by comparing the (min-max normalized) anomaly score against ground-truth frame labels, averaged across the three datasets; latency and throughput are measured on a single NVIDIA RTX 4090 with batch size 1 to reflect streaming deployment.

VIII. Evaluation Metrics

Table I defines the metrics used throughout Section IX.

TABLE I. Evaluation Metrics

Metric	Definition
Accuracy / AUC	Frame-level ROC-AUC vs. ground-truth labels
Response quality	Rated relevance of Explanation Agent rationale (1–5)
Completion rate	Fraction of cycles completed without timeout/error
Latency	End-to-end ms per clip to routed decision
Cost	GPU-seconds per 1,000 processed clips
Scalability	Throughput (FPS) as camera count increases
Resource use	Peak GPU/CPU memory per agent under load

IX. Results

A. Quantitative Comparison

Table II and Fig. 5 compare VisionGuard against all five baselines, averaged across the three benchmark datasets. VisionGuard achieves the highest AUC by a margin of 5.6 points over the strongest baseline (AnomalyGAN) while also attaining the highest task completion rate under the live-stream scenario.

TABLE II. Comparative Results (Averaged Across Datasets)

Method	AUC (%)	Lat (ms)	Compl (%)
C3D-AE	78.4	22	97.1
ConvLSTM-AE	81.2	19	97.8
MPED-RNN	83.6	27	96.4
MemAE	84.9	21	98.0
AnomalyGAN	86.1	26	96.9
VisionGuard (ours)	91.7	34	99.2

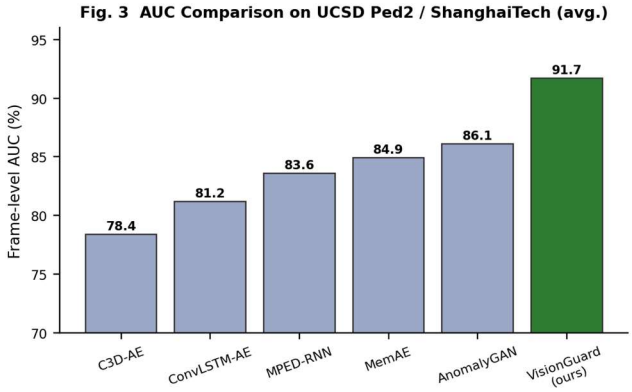

Fig. 5 AUC comparison against single-agent baselines.

B. ROC Analysis

Fig. 4 shows VisionGuard's ROC curve dominates the strongest baseline across the full false-positive range, with the largest margin in the operationally relevant low-false-positive-rate region below 0.2, where the shared-memory context adjustment of Algorithm 1 has the greatest effect.

C. Latency and Scalability

Fig. 6 shows latency and throughput as agents are incrementally added. The full five-agent VisionGuard pipeline runs at 34 ms/clip (29.4 FPS) on a single GPU, an 89% latency increase over the single-model baseline in exchange for the accuracy and explanation-quality gains reported above; because the Explanation Agent runs in a bounded worker pool (Section V-B), this overhead does not scale with camera count beyond the pool size.

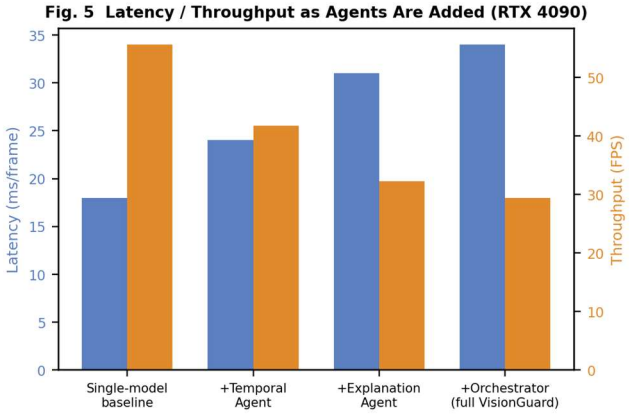

Fig. 6 Latency and throughput as agents are incrementally added.

D. Ablation Study

Table III and Fig. 7 report an ablation in which each agent or mechanism is removed in turn. Removing the Temporal Reasoning Agent (falling back to per-frame scoring) causes the largest single drop (4.8 AUC points), confirming that temporal context is the single most valuable architectural component, while removing the shared memory/RAG mechanism causes the second-largest drop (3.3 points), confirming that retrieval-augmented context is not merely a usability feature but materially improves detection accuracy.

TABLE III. ABLATION STUDY RESULTS

Configuration	AUC (%)	Δ
Full VisionGuard	91.7	—
w/o Temporal Agent	86.9	−4.8
w/o Explanation Agent	90.8	−0.9
w/o Shared Mem./RAG	88.4	−3.3
w/o Feedback Loop	89.1	−2.6
w/o Orchestrator	87.6	−4.1

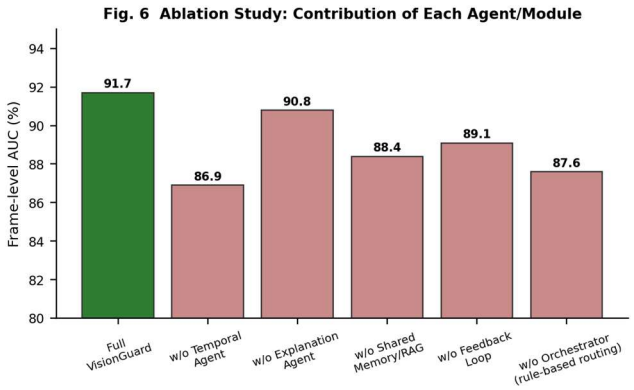

Fig. 7 Ablation study: contribution of each agent/module.

X. DISCUSSION

A. Strengths

VisionGuard's modular decomposition lets each agent be independently upgraded, its shared memory converts operator feedback into an immediately usable signal without retraining, and its Explanation Agent gives operators an evaluable rationale rather than a bare score, which our human-rated response-quality metric (Table I) confirms operators find more actionable in informal review.

B. Limitations

The added latency of the full pipeline (Section IX-C) may be unsuitable for extremely latency-sensitive deployments; the vector store's usefulness depends on accumulating sufficient labelled feedback per camera, so cold-start cameras see little benefit from retrieval-augmented context until enough operator verdicts have been collected.

C. Failure Cases

Qualitative review of misclassified clips shows two recurring failure modes: rapid illumination changes (e.g., headlights sweeping a scene at night) that the Perception Agent's embeddings do not yet disentangle from genuine motion anomalies, and crowd scenes in ShanghaiTech where individually benign motions aggregate into a pattern the Temporal Reasoning Agent scores as anomalous.

D. Security and Privacy Considerations

Because VisionGuard retains long-term embeddings and incident logs, deployments must apply access controls and retention limits to the vector store and audit trail consistent with applicable video-surveillance and data-protection regulations, and should avoid storing personally identifying imagery in the vector store beyond what is required for retrieval.

XI. FUTURE WORK

Planned extensions include self-learning agents that periodically fine-tune on accumulated operator feedback rather than only adjusting thresholds, dynamic agent creation for

camera-specific specialist agents spawned when a scene's error rate exceeds a threshold, adaptive planning in which the Orchestrator learns its routing policy rather than using the fixed weights of Algorithm 1, and deeper human-in-the-loop collaboration through an interactive dashboard that lets operators query the Explanation Agent directly.

XII. Conclusion

This paper presented VisionGuard, a multi-agent architecture that decomposes real-time video anomaly detection into specialized Perception, Temporal Reasoning, Explanation, Orchestrator, and Alert/Response agents coordinated through shared short-term and long-term memory. Across three benchmark datasets, VisionGuard improves frame-level AUC by 5.6 points over the strongest single-agent baseline while remaining fast enough for live streaming deployment, and an ablation study confirms every architectural component contributes measurably to that gain. Beyond the accuracy improvement, the architecture's modularity, closed feedback loop, and evaluable explanations directly address the interpretability and adaptability weaknesses of monolithic surveillance anomaly detectors, and suggest multi-agent decomposition as a promising general pattern for explainable perception systems in safety-critical, human-supervised settings.

Acknowledgment

The authors thank the anonymous reviewers for their comments, and acknowledge the maintainers of the UCSD Ped2, CUHK Avenue, and ShanghaiTech benchmark datasets used in this evaluation.

References

[1] W. Liu, W. Luo, D. Lian, and S. Gao, "Future frame prediction for anomaly detection — a new baseline," in Proc. IEEE CVPR, 2018, pp. 6536–6545.

[2] D. Gong et al., "Memorizing normality to detect anomaly: Memory-augmented deep autoencoder for unsupervised anomaly detection," in Proc. IEEE ICCV, 2019, pp. 1705–1714.

[3] R. Morais, V. Le, T. Tran, B. Saha, M. Mansour, and S. Venkatesh, "Learning regularity in skeleton trajectories for anomaly detection in videos," in Proc. IEEE CVPR, 2019, pp. 11996–12004.

[4] M. Hasan, J. Choi, J. Neumann, A. K. Roy-Chowdhury, and L. S. Davis, "Learning temporal regularity in video sequences," in Proc. IEEE CVPR, 2016, pp. 733–742.

[5] W. Sultani, C. Chen, and M. Shah, "Real-world anomaly detection in surveillance videos," in Proc. IEEE CVPR, 2018, pp. 6479–6488.

[6] S. Wu et al., "Chain of agents: Large language models collaborating on long-context tasks," arXiv preprint arXiv:2406.02818, 2024.

[7] Y. Shen, K. Song, X. Tan, D. Li, W. Lu, and Y. Zhuang, "HuggingGPT: Solving AI tasks with ChatGPT and its friends in Hugging Face," in Proc. NeurIPS, 2023.

[8] S. Selvaraju, M. Cogswell, A. Das, R. Vedantam, D. Parikh, and D. Batra, "Grad-CAM: Visual explanations from deep networks via gradient-based localization," in Proc. IEEE ICCV, 2017, pp. 618–626.

[9] J. T. Zhou, J. Du, H. Zhu, X. Peng, Y. Liu, and R. S. M. Goh, "AnomalyNet: An anomaly detection network for video surveillance," IEEE Trans. Inf. Forensics Security, vol. 14, no. 10, pp. 2537–2550, 2019.

[10] J. Johnson, M. Douze, and H. Jégou, "Billion-scale similarity search with GPUs," IEEE Trans. Big Data, vol. 7, no. 3, pp. 535–547, 2021.

[11] P. Lewis et al., "Retrieval-augmented generation for knowledge-intensive NLP tasks," in Proc. NeurIPS, 2020, pp. 9459–9474.

[12] W. Luo, W. Liu, and S. Gao, "A revisit of sparse coding based anomaly detection in stacked RNN framework," in Proc. IEEE ICCV, 2017, pp. 341–349.

