Technical Report: Real-Time Violence Detection System

Author: Gireesh Chowdary
Date: 2025-08-28

Executive Summary

This document describes an end-to-end real-time violence detection system that processes RTSP CCTV streams to detect violent events and send timely alerts. The system extracts a compact 30-dimensional feature vector per frame (visual + motion + audio cues), uses a BiLSTM with attention to model temporal sequences (16 frames), and applies a robust decision layer (EMA + consecutive confirmations) before issuing alerts. The report explains each architectural layer in detail, lists training and evaluation metrics, and highlights significance, limitations, and future work.

Dataset & Training Summary

The model was trained using a balanced dataset with approximately 1,000 violent videos and 1,000 non-violent videos (≈2,000 total). A held-out test set of 400 samples was used for final evaluation (metrics below). Key points:

1. Balanced dataset: ~1,000 violent, ~1,000 non-violent videos
2. Training/validation/test split – standard practice (held-out test set used for final results)
3. Per-frame features were extracted consistently for training and inference to avoid distribution mismatch.

Training & Evaluation Metrics (Test set: 400 samples)

Metric	Score / Count
Accuracy	82%
F1 score	0.83
ROC-AUC	0.82
True Positives (TP)	172
True Negatives (TN)	156
False Positives (FP)	44
False Negatives (FN)	28

System Architecture Overview

High-level flow:
RTSP CCTV Stream → Threaded Frame Reader → YOLOv8-Pose Detection → 30D Feature Extraction (per frame) → Sequence Queue (16 frames) → BiLSTM + Attention → Probability of Violence → Decision Layer (EMA + thresholds + consecutive confirmation) → Alert Module (snapshot + email)

Layer-by-Layer Detailed Explanation

1. Input Layer / Frame Reader (RTSP)

Role: Connects to live RTSP CCTV streams and provides a continuous frame source for downstream processing. How it works: - A threaded reader uses OpenCV to grab frames in a background thread, preventing processing delays from blocking capture. - The reader handles reconnection on network disruptions and can optionally skip frames to maintain throughput. Significance: - Ensures robust, low-latency ingestion of live video feeds for continuous monitoring. - Decouples frame capture from processing to prevent backlog and reduce jitter.

2. Human Detection & Pose Estimation (YOLOv8-Pose / MediaPipe fallback)

Role: Identify people in each frame and compute skeletal keypoints (landmarks) for pose features. How it works: - YOLOv8-Pose runs per-frame detection to return bounding boxes and keypoints for each person. - If YOLOv8-Pose is not available, MediaPipe Pose provides a fast fallback for landmark extraction. Outputs: - Per person: bounding box (x_center, y_center, width, height), confidence, and keypoint coordinates. Significance: - Localizes human actions to compute targeted motion and posture features, improving signal-to-noise for violence cues. - Pose keypoints help distinguish ambiguous interactions (e.g., hugging vs. fighting) by analyzing limb positions and velocities.

3. Feature Extraction (30-D Per Frame)

Role: Convert raw pixels, detection boxes, pose landmarks, optical flow and (optional) audio into a compact 30-dimensional vector representing that frame. Why a 30D vector: - Compact yet descriptive: encodes geometry, motion, appearance and audio cues used by the LSTM. - Keeps the temporal model lightweight and invariant to raw image dimensions. Feature list & significance:

1. **Bounding box (6):** x_center_norm, y_center_norm, w_norm, h_norm, area_norm, aspect_ratio. (captures person location & scale)
2. **Num persons (1):** count of detected people in the frame (helps flag crowd interactions).
3. **Pose stats (5):** mean_x, mean_y, std_x, std_y, pose_count_norm (captures body configuration).
4. **Optical flow (3):** mean magnitude, std magnitude, fraction_nonzero (measures motion intensity).
5. **Color stats (3):** mean_R, mean_G, mean_B (scene appearance cues).
6. **Intensity (2):** mean_gray, std_gray (lighting & contrast changes).
7. **Edge density (1):** Canny edge count / area (texture / motion edges).
8. **BBox motion deltas (3):** dx, dy, darea relative to previous frame (short-term movement).
9. **YOLO confidence stats (2):** mean_conf, max_conf (detection reliability).
10. **Audio (3):** rms, zcr, spectral_flatness (optional; 0 if no audio).
11. **Frame variance (1):** pixel variance (general motion indicator).

Significance: - Combining pose, motion, and audio features improves discrimination between violent and non-violent interactions. - Temporal deltas (flow and bbox motion) capture dynamics not visible in static frames. - Audio features, when available, help detect loud or abrupt sound patterns correlated with violence.

4. Sequence Formation (Temporal Window)

Role: Stack consecutive per-frame 30D vectors into a sequence (length = 16) to capture temporal context over ~0.5-1.5 seconds depending on frame rate. Why sequences? - Violence is temporal: one frame rarely contains enough evidence; patterns over several frames matter (e.g., repeated hits, sudden falls). - The LSTM learns temporal dependencies and transitions from pre- to post-event frames.

5. Sequence Model: BiLSTM + Attention

Role: Convert the input sequence (1 × 16 × 30) into a compact representation that indicates the presence of violent behavior. Architecture details (layer-by-layer): - **Input Layer:** receives a sequence shaped (batch=1, seq_len=16, feat_dim=30). - **LSTM Stack (4 layers, bidirectional):** each layer processes the sequence and passes hidden states to the next layer. Bidirectionality lets the network consider both past and future context within the sequence window (useful for capturing actions that have multi-frame onset and aftermath). - **Hidden size 256:** each LSTM layer outputs 256 features per direction; for bidirectional, outputs are concatenated (512-dimensional per timestep). - **Attention Layer:** computes a scalar score per timestep by passing the LSTM output through a linear layer, then softmax across time to obtain attention weights. The final 'context' vector is a weighted sum of timestep representations. This lets the model focus on the most informative frames (e.g., the moment of impact). - **Dropout:** applied to reduce overfitting and make predictions more robust in noisy streams. - **Fully Connected (FC) Classifier:** the context vector is fed to an FC layer producing 2 logits (non-violent, violent). - **Softmax:** converts logits to class probabilities. Significance: - BiLSTM captures temporal patterns and reversibility in short windows; attention highlights critical frames where the event occurs. - This combination increases sensitivity to temporal cues while keeping the model computationally efficient for real-time inference.

6. Decision Layer (EMA + Confirmations + Cooldown)

Role: Stabilize raw LSTM predictions and only trigger alerts for sustained, confident detections. Components: - **Sequence Probability (seq_prob):** p_violence from the LSTM for each 16-frame window. - **Exponential Moving Average (EMA):** ema = α * seq_prob + (1-α) * ema. EMA smooths sudden spikes due to transient motion. - **Rolling mean / Windowing (optional):** a longer rolling average can further reduce noise if desired. - **Consecutive Confirmations:** require K consecutive windows where EMA exceeds a threshold before alerting (e.g., K=3). -

Cooldown: after triggering, wait a minimum separation time to avoid email spam. Significance: - Reduces false positives caused by brief ambiguous frames (e.g., hugging, handshakes, sudden camera noise). - Ensures alerts are meaningful and actionable.

7. Alerting & Evidence Module

Role: Provide human operators with immediate notice and evidence for verification. Functionality: - On confirmed alert: save a high-resolution snapshot and a sampled short clip (pre + post-event) to the evidence folder. - Send an email to a configured recipient with the snapshot attached. Uses a single sender account (Gmail) and an application-specific password for security. - Maintain logs (CSV) of sequences and probabilities for later auditing and model improvement. Significance: - Human-in-the-loop verification: operators can quickly view evidence to validate and escalate. - Forensics: saved clips provide a record for investigations.

Training Methodology

- Input: sequences of 16 frames, each represented by the 30D feature vector. - Loss: Cross-entropy loss for binary classification. - Optimizer: (Typical choice: Adam or AdamW) – adjust learning rate and weight decay during experiments. - Regularization: Dropout between LSTM output and classifier; early stopping on validation loss. - Validation & Test: use separate hold-out sets; the test set reported here contains 400 samples. Hyperparameters (used in the reported model):

1. Sequence length = 16 frames
2. LSTM layers = 4 (bidirectional)
3. Hidden size = 256
4. Dropout = 0.3
5. Batch size = depends on GPU memory (commonly 16–64)
6. Loss = CrossEntropyLoss
7. Evaluation metrics = Accuracy, F1, ROC-AUC, Confusion Matrix

Evaluation & Analysis

The model achieved the following on the held-out test set (400 samples): - Accuracy: 82% (correct classification rate across all samples) - F1 score: 0.83 (balance between precision and recall) - ROC-AUC: 0.82 (ability to separate classes across thresholds) Confusion matrix breakdown: - True Positives (TP): 172 (violent events correctly detected) - True Negatives (TN): 156 (non-violent correctly classified) - False Positives (FP): 44 (non-violent flagged as violent) — causes unnecessary alerts - False Negatives (FN): 28 (violent events missed) — critical failure (missed detection) Interpretation & significance: - False Negatives are most critical: missed detections represent real-world safety risks and must be minimized with targeted improvements. - False Positives are disruptive but less severe; mitigated via EMA, consecutive confirmations, and thresholds.

Recommendations & Next Steps

1. Augment training data with 'hard negatives' (hugging, handshakes, sports) to reduce false positives.
2. Collect and label additional violent edge cases (low-light, occlusions) to reduce false negatives.
3. Experiment with adding explicit pose-motion features (joint velocities/angles) and retraining.
4. Consider a small lightweight re-training where pose_motion is included as an additional input dimension.
5. Deploy a lightweight edge inference pipeline (quantized model) for low-latency multi-camera deployments.
6. Continuous monitoring & logging for model drift; periodic re-evaluation and retraining.

Appendix: Quickstart & File Paths

- LSTM weights: /content/drive/MyDrive/VIRAT_KOHLI_Models_v2/best_lstm_attn.pt - YOLOv8-pose model: /content/models/yolov8n-pose.pt (or your chosen pose checkpoint) - Evidence folder: /content/evidence/ - To run live inference: use the RTSP threaded reader with the provided config and ensure GPU (CUDA) is available.