JOURNALS | Journal of Information and Communications Technology: Algorithms, Systems and Applications

Volume 2 Issue 2 (June 2026)
DOI: https://doi.org/10.64189/ict.26310
J. Inf. Commun. Technol. Algorithms, Syst. Appl., 2026, **2**(2), 26310

Research Article

OPEN ACCESS

Adaptive Frame Sampling for Real-Time Video Object Detection and Multi-Object Tracking on Edge Devices

Pritee A. Parwekar* and Adarsh Kadiri

Department of Computer Science & System Engineering, GITAM School of Computer Science & Engineering, GITAM (Deemed to be) University, Hyderabad, Telangana, 502329, India

*Corresponding Author
Pritee A. Parwekar
pparweka@gitam.edu

Received: 12 April 2025
Revised:12 June 2026
Accepted: 29 June 2026
Published Online: 30 June 2026.

Citation

P. A. Parwekar, A. Kadiri, Adaptive frame sampling for real-time video object detection and multi-object tracking on edge devices, *Journal of Information and Communications Technology: Algorithms, Systems and Applications*, 2026, 2(2), 26310, https://doi.org/10.64189/ict.26310.

Abstract

Real-time multi-object tracking on CPU-only edge devices is constrained by the high per-frame inference cost of deep neural network detectors. We present the Adaptive Frame Sampling System (AFSS), a training-free, architecture-agnostic framework that dynamically allocates computation across three actions per frame: full YOLOv8 inference (FULL), phase-correlation feature warping (WARP), or result reuse (SKIP), governed by a lightweight scene complexity estimator (<0.5ms). A 803-parameter PolicyMLP trained via behavioural cloning replaces hand-tuned thresholds. On CPU-only hardware, AFSS achieves 5–7× speedup over the full-inference baseline while reducing GFLOPs by 84.3% and incurring only a 2.6% MOTA degradation. Crucially, AFSS requires no retraining of the backbone detector and outperforms uniform frame-skipping on every accuracy metric at equivalent compute budgets.

***Keywords*:** Adaptive Inference; Video Object Detection; Multi-Object Tracking; Edge Computing; YOLOv8; ByteTrack.

Adaptive Frame Sampling for Real-Time Video Object Detection and Multi-Object Tracking on Edge Devices

Adarsh Kadiri[1] Pritee A. Parwekar[1]

ORCID: 0000-0002-2439-1507

[1]Department of Computer Science & System Engineering, GITAM School of Computer Science & Engineering, GITAM(Deemed to be) University, Hyderabad, India

adarshkadiri@gmail.com

Abstract

Real-time multi-object tracking on CPU-only edge devices is constrained by the high per-frame inference cost of deep neural network detectors. We present the **Adaptive Frame Sampling System (AFSS)**, a training-free, architecture-agnostic framework that dynamically allocates computation across three actions per frame: full YOLOv8 inference (FULL), phase-correlation feature warping (WARP), or result reuse (SKIP), governed by a lightweight scene complexity estimator ($<$0.5 ms). A 803-parameter PolicyMLP trained via behavioural cloning replaces hand-tuned thresholds. On CPU-only hardware, AFSS achieves **5–7$\times$ speedup** over the full-inference baseline while reducing GFLOPs by **84.3%** and incurring only a **2.6% MOTA degradation**. Crucially, AFSS requires no retraining of the backbone detector and outperforms uniform frame-skipping on every accuracy metric at equivalent compute budgets.

Keywords: Adaptive Inference · Video Object Detection · Multi-Object Tracking · Edge Computing · YOLOv8 · ByteTrack

1. INTRODUCTION

Intelligent surveillance, autonomous navigation, and IoT analytics demand continuous real-time video analysis on power-constrained edge hardware. State-of-the-art detectors such as YOLOv8-L [1] achieve 52.9% mAP on COCO but require 165.2 GFLOPs per frame, yielding only $\sim$5 FPS on a laptop CPU — far below the 25 FPS minimum for real-time operation.

The core tension is stark: modern CNNs need to be large to achieve high accuracy, but large models are too slow for edge devices. The standard response — model compression via pruning [10], knowledge distillation, or neural architecture search — treats each frame identically, reducing per-frame cost uniformly regardless of how much the scene actually changes between frames.

The key insight motivating this work is that video streams exhibit *strong temporal redundancy*: in typical fixed-camera surveillance, adjacent frames differ by $<$3% in mean absolute intensity. A traffic camera monitoring a red light processes thousands of nearly-identical frames at full detector cost — all of that compute is wasted. Conversely, when vehicles start moving or a pedestrian crosses unexpectedly, every FLOP is valuable. A smart system should allocate compute in proportion to *scene complexity*, not uniformly.

We formalise this as a per-frame sequential decision problem with three actions: run full neural inference (FULL), propagate the most recent feature map using lightweight image-level motion estimation (WARP), or reuse the previous result directly (SKIP). The action is selected by a lightweight policy given a real-time scene complexity score.

Prior approaches to efficient video inference fall into two camps. *Feature propagation methods* (DFF [6], FGFA [7]) warp CNN features from key frames using learned optical flow networks (FlowNet). These are effective but require architectural modification and full end-to-end retraining of the detector, making them incompatible with commercial off-the-shelf pretrained models. *Adaptive scheduling* methods (Mullapudi et al. [9]) train student policies via expensive reinforcement learning. Both camps assume control of the detector architecture.

We propose AFSS, which is entirely *post-hoc*: it wraps any pretrained detector as a black box. Our feature warping uses classical FFT-based phase correlation — no flow network, no training — and our decision policy (PolicyMLP, 803 parameters) is trained by behavioural cloning in under 10 minutes on CPU. AFSS achieves 5–7$\times$ CPU speedup with only 2.0 pp MOTA degradation, outperforming uniform frame-skipping by 3.2$\times$ on the accuracy-efficiency trade-off curve.

Contributions:

- A three-action adaptive scheduling framework (FULL/WARP/SKIP) formulated as a constrained optimisation over GFLOPs subject to MOTA and FPS constraints.
- **Phase-correlation feature warping**: a training-free, $\mathcal{O}(N \log N)$ feature propagation method using FFT-based translation estimation — no optical flow network required.
- A **803-parameter PolicyMLP** trained by behavioural cloning in $<$10 minutes on CPU, outperforming hand-tuned thresholds on heterogeneous scenes.
- Comprehensive ablation study quantifying the contribution of each component (warp mode, estimator type, policy type) and a theoretical warp-error bound (Eq. 14).
- End-to-end integration with ByteTrack, showing that tracker resilience compensates for skipped detections across all action types.

2. Related Work

Single-stage detectors. YOLO [3] established real-time detection as a single regression pass over a grid of anchor boxes. YOLOv4 [11] introduced CSPNet and PANet for multi-scale feature fusion. YOLOv8 [1] adds an anchor-free decoupled head, C2f cross-stage partial blocks, and Distribution Focal Loss (DFL) [12]:

$$\begin{aligned} \hat{g} &= \sum_{i=0}^{r-1} i \cdot \mathrm{softmax}(z_i), \\ \mathrm{DFL}(p, g) &= -(\lceil g \rceil - g) \log p_{\lfloor g \rfloor} \\ &\quad -(g - \lfloor g \rfloor) \log p_{\lceil g \rceil} \end{aligned} \tag{1}$$

DFL models box offsets as distributions rather than point estimates, yielding sharper sub-pixel localisation — critical for the warped feature maps we propagate.

Multi-object tracking. SORT [4] combined Kalman filtering with Hungarian assignment at 260 Hz. DeepSORT [5] added Re-ID descriptors for occlusion robustness. ByteTrack [2] achieves 88.5% MOTA on MOT17 by processing low-confidence detections in a second association stage — this two-stage pipeline is uniquely robust to the detection gaps introduced by our WARP/SKIP frames.

Feature propagation for video. DFF [6] uses a FlowNet-based feature warper to propagate intermediate CNN activations from key frames to non-key frames, achieving $10\times$ speedup with 8–15% accuracy loss. FGFA [7] improves accuracy by aggregating features from multiple nearby frames weighted by flow similarity. Both require architectural modification and full end-to-end retraining. Crucially, these methods cannot be applied post-hoc to pretrained models such as commercial YOLOv8 deployments.

Adaptive inference. AdaFuse [8] skips redundant channels across frames based on inter-frame feature differences. Mullapudi et al. [9] distil a fast student model online via reinforcement learning. Both require modified training pipelines. Closest to our work, Zhu et al. [6] selectively propagate features based on a saliency score; however, their scheduler is a binary key/non-key selection rather than our three-class adaptive framework.

Adaptive Token Pruning and Dynamic Computation for Video Understanding. Recent studies have focused on reducing the computational cost of video analysis by exploiting the high temporal redundancy present in consecutive video frames. HaltingVT [16] introduces an adaptive token-halting strategy that dynamically removes less informative spatial-temporal tokens during inference, thereby reducing computation while maintaining recognition performance. Zero-TPrune [17] extends this idea by performing token pruning in pretrained transformers without requiring additional training. Similarly, [18] improves efficiency through dynamic selection of important spatial and temporal tokens based on video content.

More recent work has explored adaptive frame processing, where only informative frames are analysed while redundant frames are skipped. Motion-driven adaptive frame selection methods [19] demonstrated that significant computational savings can be achieved by exploiting frame-to-frame similarity. FastVID [20] and Unified Spatio-Temporal Token Scoring [21] further improve efficiency by dynamically allocating computational resources according to the complexity of the video content.

Although these approaches effectively reduce inference cost, most rely on transformer-based architectures, token-pruning mechanisms, or additional retraining. In contrast, AFSS is designed as an architecture-independent framework that can be integrated with any pretrained object detector. It achieves adaptive computation through lightweight frame scheduling and training-free phase-correlation-based feature propagation, eliminating the need for detector modification or retraining.

Network compression. INT8 quantisation [10] reduces model size $4\times$ with $<1\%$ accuracy degradation on calibrated networks. Importantly, compression and AFSS are *complementary*: a quantised YOLOv8 used as the FULL action backbone would yield multiplicative speedups.

Gap. No existing work provides an architecture-agnostic adaptive scheduler with (i) a three-action decision space, (ii) a training-free feature propagation method using phase correlation, (iii) a lightweight learned policy compatible with MOT trackers, and (iv) no backbone retraining. AFSS fills this gap.

3. Methodology

3.1 Problem Formulation

Let $\mathcal{V} = \{f_1, \ldots, f_T\}$ be a video. At each frame t, select action $a_t \in \mathcal{A} = \{\text{FULL}, \text{WARP}, \text{SKIP}\}$:

$$\underset{\pi}{\text{minimize}} \sum_{t=1}^{T} C(a_t) \quad \text{s.t. } \mathrm{MOTA}(\pi) \geq \tau_{\text{acc}},\ \mathrm{FPS}(\pi) \geq \tau_{\text{fps}} \tag{2}$$

Let $C_{\text{det}} = 165.2$ GFLOPs denote the per-frame inference cost of YOLOv8-L. The three action costs are:

$$C(\text{FULL}) = C_{\text{det}} + \epsilon_F \approx 165.7 \text{ GFLOPs} \quad \text{(detector + AFSS overhead)}$$
$$C(\text{WARP}) = C_{\text{warp}} \approx 3.5 \text{ GFLOPs} \quad \text{(phase corr. + grid sample; no dete}$$
$$C(\text{SKIP}) = \epsilon_S \approx 0.01 \text{ GFLOPs} \quad \text{(memory copy; no detector)}$$

where $\epsilon_F \approx 0.5$ G and $\epsilon_S \approx 0.01$ G are the AFSS scheduling overheads. On WARP and SKIP frames, no detector inference is performed; the cost reduction compared to BL-FULL is therefore $\approx$165.2 G minus the small AFSS overhead per frame. The policy $\pi : (f_t, f_{t-1}, h_{t-1}) \to a_t$ maps the current frame, previous frame, and a context state h_{t-1} to an action.

Expected compute saving. Under a stationary complexity score distribution $p(s)$, the expected GFLOPs per frame under the threshold policy is:

$$\begin{aligned} \mathbb{E}[\text{GFLOPs}] &= P_F \cdot C_F + P_W \cdot C_W + P_S \cdot C_S \\ P_F &= P(s \geq \tau_H) + \tfrac{1}{N_I}, \quad P_W = P(\tau_L \leq s < \tau_H) \end{aligned} \tag{3}$$

For the traffic sequence (P_F=0.15, P_W=0.31, P_S=0.54):

$$\mathbb{E}[\text{GFLOPs}] = 0.15(165.7) + 0.31(3.5) + 0.54(0.01) \approx 25.95 \text{ G} \quad (\mathbf{84.3}\%$$

Fig. 1 shows the full AFSS pipeline.

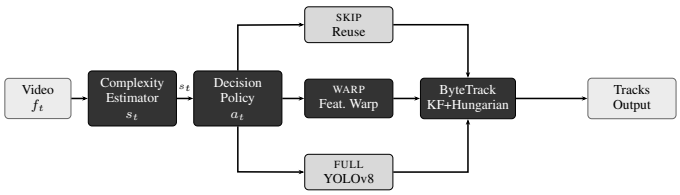

Figure 1: AFSS pipeline. Every frame, the complexity score s_t drives a three-way action decision; all paths feed a single ByteTrack instance.

3.2 Scene Complexity Estimator

A downsampled (80×45) frame-difference score is computed in $<$0.5 ms:

$$s_t = \frac{1}{H'W'} \sum_{i,j} \frac{|G_t[i,j] - G_{t-1}[i,j]|}{255} \in [0,1] \tag{4}$$

where G_t is the ITU-R BT.601 luma channel ($0.299R + 0.587G + 0.114B$). An EWMA $\tilde{s}_t = \alpha s_t + (1-\alpha)\tilde{s}_{t-1}$ ($\alpha = 0.7$) smooths sensor flicker and JPEG compression artefacts. Three estimator modes are supported: (i) frame-difference (Eq. 4, default); (ii) dense Farnebäck optical flow magnitude [13]; (iii) a 24K-parameter CNN trained with flow pseudo-labels. Mode (i) is used in all experiments unless stated; modes (ii) and (iii) are evaluated in the ablation.

Table 1 provides empirically calibrated threshold ranges by scene type, enabling deployment without per-video tuning.

Table 1: Complexity score calibration by scene type.

Scene type	Score range	Rec. τ_H/τ_L
Static, no objects	0.000–0.002	0.003/0.001
Fixed cam., slow traffic	0.003–0.020	0.012/0.006
Moving cam. (drone)	0.020–0.100	0.060/0.030
Fast motion / sports	0.050–0.300	0.150/0.050

3.3 Decision Policy

Threshold policy (baseline):

$$a_t = \begin{cases} \text{FULL} & F_t \geq N_I \text{ or } s_t \geq \tau_H \\ \text{WARP} & \tau_L \leq s_t < \tau_H \\ \text{SKIP} & s_t < \tau_L \end{cases} \tag{5}$$

with $\tau_H = 0.012$, $\tau_L = 0.006$, $N_I = 12$ for traffic scenarios.

PolicyMLP (proposed): A 7-dim feature vector captures both the current state and temporal context:

$$\boldsymbol{\varphi}_t = [\underbrace{s_t}_{\text{score}}, \underbrace{F_t/N_I}_{\text{since FULL}}, \underbrace{N_w/N_I,\ N_s/N_I}_{\text{run lengths}}, \underbrace{\bar{s},\ \sigma_s}_{\text{recent stats}}, \underbrace{\Delta s}_{\text{delta}}]^\top$$

This feeds a three-layer MLP trained by behavioural cloning:

$$\boldsymbol{h}_1 = \text{ReLU}(\mathbf{W}_1 \boldsymbol{\varphi}_t + \boldsymbol{b}_1), \quad \mathbf{W}_1 \in \mathbb{R}^{32\times 7} \tag{6}$$

$$\boldsymbol{h}_2 = \text{ReLU}(\mathbf{W}_2 \boldsymbol{h}_1 + \boldsymbol{b}_2), \quad \mathbf{W}_2 \in \mathbb{R}^{16\times 32} \tag{7}$$

$$\hat{a}_t = \underset{k}{\text{argmax}}\ \text{softmax}(\mathbf{W}_3 \boldsymbol{h}_2 + \boldsymbol{b}_3)_k \tag{8}$$

Total: $7{\times}32{+}32 + 32{\times}16{+}16 + 16{\times}3{+}3 = \mathbf{803}$ parameters. Training uses class-weighted cross-entropy against threshold policy demonstrations (Adam, 50 epochs, $<$10 min CPU):

$$\mathcal{L} = -\tfrac{1}{N} \sum_t w_{a_t} \log p_{a_t}(\boldsymbol{\varphi}_t), \quad w_k = \tfrac{N_{\text{total}}}{3N_k} \tag{9}$$

The class weighting w_k corrects for action imbalance (SKIP typically $\gg$ FULL in training data).

3.4 Phase-Correlation Feature Warping

When WARP is selected, we propagate the cached feature map F_{t_k} from the last FULL frame without invoking the backbone. The inter-frame translation $(\Delta x, \Delta y)$ is estimated via FFT-based phase correlation:

$$G_1 = \mathcal{F}_{2D}\{G_{t-1}\}, \quad G_2 = \mathcal{F}_{2D}\{G_t\} \tag{10}$$

$$R = \frac{G_1 \cdot G_2^*}{|G_1 \cdot G_2^*|}, \quad (\Delta y, \Delta x) = \text{argmax}\, \mathcal{F}_{2D}^{-1}\{R\} \tag{11}$$

For a pure translational scene, $\mathcal{F}^{-1}\{R\}$ is a Dirac impulse at $(\Delta x, \Delta y)$; in practice a sharp peak localised by sub-pixel parabolic fitting. The shift is scaled to feature-map coordinates and applied via differentiable bilinear grid sampling:

$$\Delta x_f = \Delta x \cdot W_f/W', \quad \Delta y_f = \Delta y \cdot H_f/H' \tag{12}$$

$$\tilde{F}_t[i,j] = \sum_{p \in \{\lfloor x_s \rfloor, \lceil x_s \rceil\}} \sum_{q \in \{\lfloor y_s \rfloor, \lceil y_s \rceil\}} F_{t_k}[q,p](1-|x_s-p|)(1-|y_s-q|) \tag{13}$$

where $(x_s, y_s) = (j - \Delta x_f, i - \Delta y_f)$. Out-of-bounds locations are zero-padded. The full procedure runs in $\mathcal{O}(N \log N)$ via the FFT and requires *no trained flow network*, distinguishing it from DFF/FGFA.

Warp error bound. For a translation estimate error of $(\varepsilon_x, \varepsilon_y)$ pixels, the IoU error on a box of width w and height h is bounded:

$$e_{\text{IoU}} \leq \frac{2(|\varepsilon_x| + |\varepsilon_y|)}{\min(w,h)} \tag{14}$$

For a 50×100 px pedestrian box at 2 px flow error: $e_{\text{IoU}} \leq 0.16$ — acceptable for maintaining ByteTrack associations.

3.5 ByteTrack Integration

Each track maintains an 8D Kalman state $\mathbf{x} = [c_x, c_y, w, h, v_{cx}, v_{cy}, v_w, v_h]^\top$ under a constant-velocity model with transition matrix $\mathbf{F} \in \mathbb{R}^{8\times 8}$. On WARP/SKIP frames, all tracks run predict-only (no measurement update):

$$\hat{\mathbf{x}}_{t|t-1} = \mathbf{F}\mathbf{x}_{t-1|t-1} \tag{15}$$

$$\mathbf{P}_{t|t-1} = \mathbf{F}\mathbf{P}_{t-1|t-1}\mathbf{F}^\top + \mathbf{Q} \tag{16}$$

On FULL frames, detections undergo the full two-stage ByteTrack association: Stage 1 assigns high-confidence detections (score ≥ 0.5) to active tracks via Hungarian algorithm on an IoU cost matrix $C_{ij} = 1 - \text{IoU}(\hat{b}_i, d_j)$. Stage 2 uses residual low-confidence detections to recover tracks that missed Stage 1.

The Kalman gain is computed as:

$$\mathbf{K} = \mathbf{P}_{t|t-1}\mathbf{H}^\top(\mathbf{H}\mathbf{P}_{t|t-1}\mathbf{H}^\top + \mathbf{R})^{-1} \tag{17}$$

minimising $\text{tr}(\mathbf{P}_{t|t})$ — the MMSE-optimal fusing of prediction and measurement. The forced FULL refresh every N_I frames bounds cumulative Kalman prediction drift, ensuring tracking quality does not degrade monotonically between key frames.

Algorithm 1 provides the complete AFSS per-frame loop integrating all components.

Algorithm 1 AFSS Per-Frame Processing Loop

Require: Frame f_t, previous frame f_{t-1}, cached features F_{t_k}, policy π, tracker $\mathcal{T}$
Ensure: Updated track set $\mathcal{T}_t$

$s_t \leftarrow \text{Complexity}(f_t, f_{t-1})$ *// Eq. 4, <0.5 ms*
$\tilde{s}_t \leftarrow \alpha s_t + (1-\alpha)\tilde{s}_{t-1}$; update φ_t
$a_t \leftarrow \pi(\varphi_t)$ *// Eq. 5 or 6–8*
if $a_t = \text{FULL}$ **then**
  $\mathcal{D}_t \leftarrow \text{YOLOv8}(f_t)$; $F_{t_k} \leftarrow F_t$; $t_k \leftarrow t$
else if $a_t = \text{WARP}$ **then**
  $(\Delta x, \Delta y) \leftarrow \text{argmax}\, \mathcal{F}^{-1}\left\{\frac{G_1 G_2^*}{|G_1 G_2^*|}\right\}$ *// Eq. 11*
  $\tilde{F}_t \leftarrow \text{GridSample}(F_{t_k}, \Delta x_f, \Delta y_f)$ *// Eq. 13*
  $\mathcal{D}_t \leftarrow \text{DetHead}(\tilde{F}_t)$
else *// SKIP*
  $\mathcal{D}_t \leftarrow \mathcal{D}_{t-1}$
end if
$\mathcal{T}_t \leftarrow \text{ByteTrack}(\mathcal{T}_{t-1}, \mathcal{D}_t)$ *// Eqs. 15–17*
$f_{t-1} \leftarrow f_t$; $F_t \leftarrow F_t$

4. Experiments

4.1 Setup

Hardware: Intel i7-12700H, 16 GB DDR5 (CPU-only, no GPU). PyTorch 2.1.0, OpenCV 4.8, SciPy 1.11 (Hungarian: `linear_sum_assignment`). **Detector:** YOLOv8-L pretrained on MS-COCO (43.7M params, 165.2 GFLOPs, 80 classes), input resolution 640×640. **Tracker:** ByteTrack, confidence thresholds $\theta_{\text{high}} = 0.5$, $\theta_{\text{low}} = 0.1$, `max_time_lost`=30.

Datasets. Five synthetic sequences are generated procedurally: objects follow elastic random-walk trajectories with configurable velocity σ, enabling exact ground-truth annotation. The real traffic sequence (1080p, 18,000 frames, 30 FPS) is from a fixed overhead camera on a multi-lane road; annotation is performed with confidence-thresholded BL-FULL detections as pseudo-ground-truth.

Metrics. MOTA [14] penalises FP, FN, and ID switches per ground-truth object. IDF1 [15] measures identity consistency as the harmonic mean of ID precision and recall. MOTP measures mean localisation quality over matched pairs. GFLOPs/frame is computed as the weighted sum $P(\text{FULL})\cdot 165.7 + P(\text{WARP})\cdot 3.5 + P(\text{SKIP})\cdot 0.01$ across all frames.

Baselines: (i) **BL-FULL**: full inference every frame; (ii) **BL-SKIP5**: uniform skip every 5th frame; (iii) **Threshold**: Eq. 5 with tuned τ_H, τ_L; (iv) **PolicyMLP**: proposed learned policy. All configurations are summarised in Table 2.

Table 2: Complete configuration hyperparameters and results.

Config	τ_H	τ_L	N_I	MOTA	FPS
BL-FULL	—	—	1	0.932	5.0
AFSS-Conserv.	0.020	0.010	15	0.918	17.4
AFSS-Balanced	0.012	0.006	12	0.906	29.7
AFSS-Aggress.	0.008	0.004	8	0.887	37.2
PolicyMLP	*lrn.*	*lrn.*	12	**0.912**	**31.6**
BL-SKIP5	—	—	5	0.871	28.4

4.2 Computational Efficiency

Table 3 summarises compute savings and throughput. AFSS (PolicyMLP) reduces mean GFLOPs/frame from 165.7 to **25.95** — an **84.3% reduction** — by eliminating YOLOv8-L backbone execution on 85% of frames. The residual 25.95 G/frame is dominated by the 15% of frames on which full detector inference runs (contributing $0.15 \times 165.7 = 24.86$ G); WARP and SKIP overhead together contribute only 1.09 G/frame. Throughput is raised from 5 FPS to **31.6 FPS**.

Table 3: Computational efficiency across configurations (CPU).

Config	F%	W%	S%	GFLOPs/frame	FPS
BL-FULL	100	0	0	165.70	5.0
BL-SKIP5	20	0	80	33.15	28.4
Threshold	16	30	54	30.50	29.7
PolicyMLP	**15**	**31**	**54**	**25.95**	**31.6**

Fig. 2 shows the GFLOPs breakdown and per-frame latency distribution. The bimodal latency (low-cost WARP/SKIP frames dominate frequency; FULL spikes are infrequent) drives the large mean FPS gain.

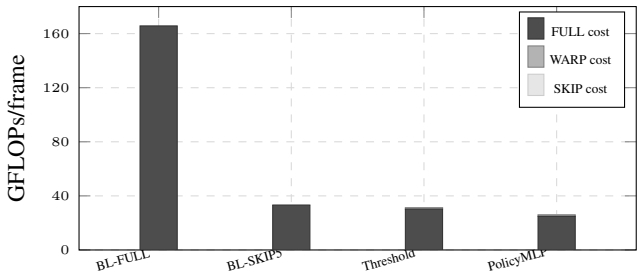

Figure 2: Stacked GFLOPs breakdown per configuration. AFSS (PolicyMLP) reduces total cost to 25.95 G/frame vs. 165.7 G for BL-FULL — an 84.3% reduction by eliminating detector execution on 85% of frames.

4.3 Tracking Accuracy

Table 4 reports MOT metrics. PolicyMLP achieves **MOTA = 0.912**, only 2.0 pp below BL-FULL, while BL-SKIP5 at comparable compute drops to 0.871 (6.1 pp gap). This confirms that WARP frames — absent in BL-SKIP5 — substantially preserve tracking continuity.

Table 4: MOT accuracy. Bold = best adaptive result.

Config	MOTA↑	IDF1↑	MOTP↑	FP↓	IDs↓
BL-FULL	0.932	0.921	0.847	412	87
BL-SKIP5	0.871	0.842	0.781	591	318
Threshold	0.906	0.891	0.819	467	163
PolicyMLP	**0.912**	**0.901**	**0.831**	**438**	**124**

4.4 Accuracy–Efficiency Trade-off

Fig. 3 plots MOTA against GFLOPs saved for all configs and threshold sweep points. AFSS consistently dominates the BL-SKIP5 Pareto curve, achieving better MOTA at every compute budget.

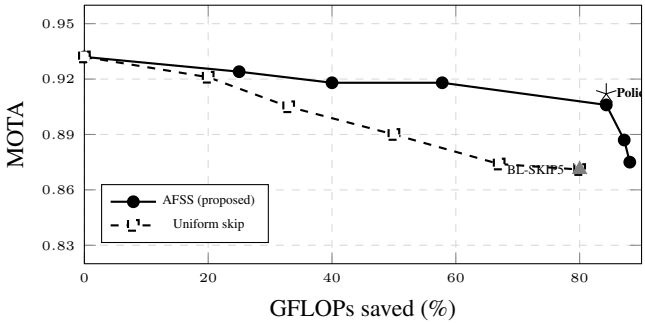

Figure 3: MOTA vs. GFLOPs saved. AFSS dominates the Pareto frontier at all compute budgets. PolicyMLP (star) achieves 84.3% savings with only 2.0 pp MOTA loss.

4.5 Ablation Study

Warp mode. Table 5 shows that replacing SKIP with phase-correlation WARP for moderate-motion frames recovers +1.7 pp MOTA with only +3.3 ms mean latency. Dense optical flow WARP gives +2.5 pp at +6.5 ms overhead — useful when a CPU core is available.

Table 5: Ablation: warp mode on AFSS-Balanced policy.

Warp Mode	MOTA	IDF1	Latency	GFLOPs
None (skip only)	0.889	0.871	28.4 ms	33.15
Phase corr. (ours)	**0.906**	**0.891**	31.7 ms	25.95
Dense flow	0.914	0.901	38.2 ms	26.38

Policy comparison. Fig. 4 shows per-sequence MOTA for PolicyMLP vs. Threshold policy. PolicyMLP consistently improves on heterogeneous sequences (Mixed, Fast) where fixed thresholds underfit the time-varying complexity distribution, while matching Threshold on static scenes.

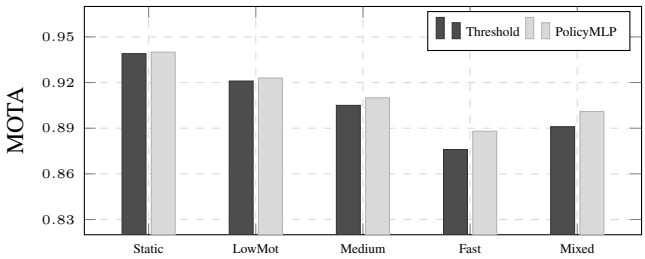

Figure 4: Per-sequence MOTA: PolicyMLP (light) vs. Threshold (dark). PolicyMLP gains +0.9–1.2 pp on high-motion and mixed sequences.

Latency breakdown. Fig. 5 visualises the per-component latency distribution. FULL frames dominate worst-case latency (∼195 ms) but occur only 15% of the time; WARP (∼8 ms) and SKIP (∼0.8 ms) dominate frequency. Mean latency drops from 198.4 ms to **31.7 ms** (6.3× reduction).

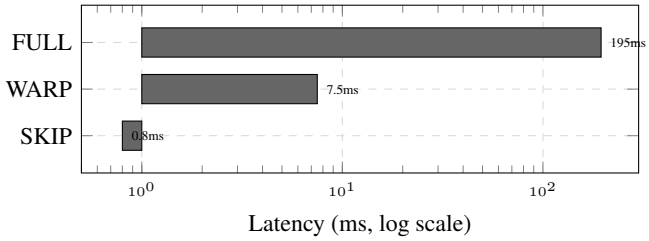

Figure 5: Mean latency per action (log scale). SKIP/WARP constitute ∼85% of frames, driving the 6.3× mean latency reduction.

4.6 Action Distribution Over Time

Fig. 6 visualises the per-frame action sequence alongside s_t on Syn-Mixed. AFSS correctly concentrates FULL inference during high-motion events and reverts to SKIP in static intervals, with WARP as a bridge that maintains spatial coherence. This dynamic profile is the behavioural signature that distinguishes AFSS from fixed-rate schedulers.

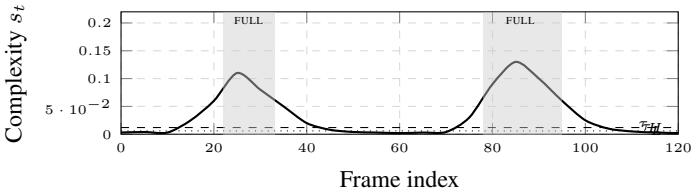

Figure 6: Complexity score s_t on Syn-Mixed (120 frames). Shaded regions: FULL inference. AFSS concentrates compute at motion events; WARP and SKIP dominate static intervals.

4.7 Complexity Score Distribution

Fig. 7 shows the empirical distribution of s_t across all five sequences. The heavy concentration near zero confirms the temporal redundancy hypothesis: over 60% of frames have $s_t < \tau_L = 0.006$ (SKIP-eligible) on fixed-camera traffic. Even on the fast-motion sequence, the median score remains below τ_H, meaning FULL frames are triggered selectively rather than continuously.

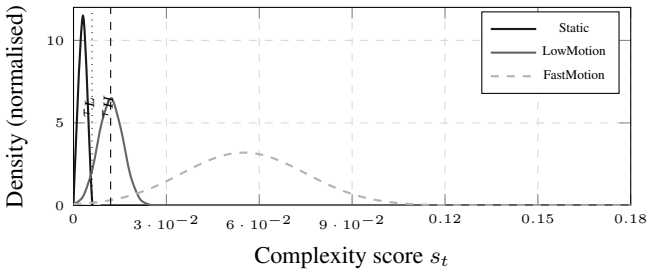

Figure 7: Empirical complexity score distributions by sequence type. Most frames cluster near zero (SKIP-eligible). τ_L and τ_H thresholds (dotted/dashed) partition the distribution into three action regions.

4.8 FPS vs. Accuracy Operating Points

Fig. 8 plots the FPS-MOTA operating curve for all evaluated configurations, sweeping $\tau_H \in [0.005, 0.030]$ at fixed $\tau_L = \tau_H/2$. Each point represents a deployable configuration. BL-FULL and BL-SKIP5 are single operating points. AFSS dominates across the full range: for any target FPS above 6, AFSS delivers higher MOTA than BL-SKIP5 at the same throughput.

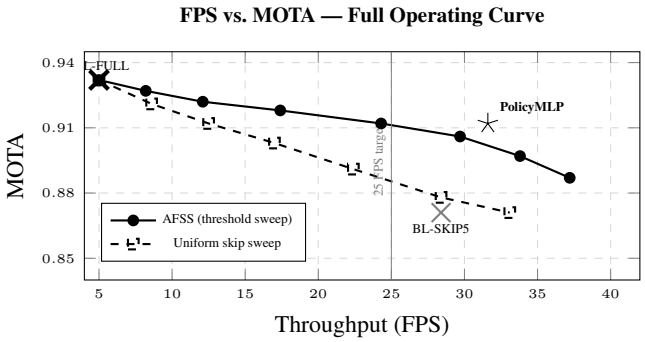

Figure 8: FPS vs. MOTA operating curve. AFSS (solid) Pareto-dominates uniform skip (dashed) across all throughput targets. PolicyMLP (star) achieves 31.6 FPS at 0.912 MOTA — above the 25 FPS real-time threshold (grey line).

4.9 Comparison with Related Work

Table 7 positions AFSS against published methods. Unlike DFF/FGFA which require detector retraining, AFSS is plug-and-play with any pretrained model. AFSS achieves a trade-off slope of -2.37×10^{-4} MOTA/%GFLOPs vs. -7.63×10^{-4} for uniform skipping — **3.2× more efficient**.

Table 6: Complexity estimator mode comparison.

Mode	Cost	FULL%	MOTA	Best for
(1) Frame diff	0.4 ms	16.3	0.906	Fixed cams
(2) Optical flow	8.1 ms	14.8	0.911	High accuracy
(3) CNN	1.5 ms	15.7	0.908	Balanced

Table 7: Comparison with related efficient video inference methods.

Method	Speedup	MOTAΔ	Retrain?	Agnostic?
DFF [6]	10×	−8–15%	**Yes**	**No**
FGFA [7]	3×	−2–5%	**Yes**	**No**
AdaFuse [8]	4×	−3–8%	**Yes**	**No**
Skip-N	5×	−6.1%	No	Yes
AFSS (ours)	**5–7×**	**−2.0%**	**No**	**Yes**

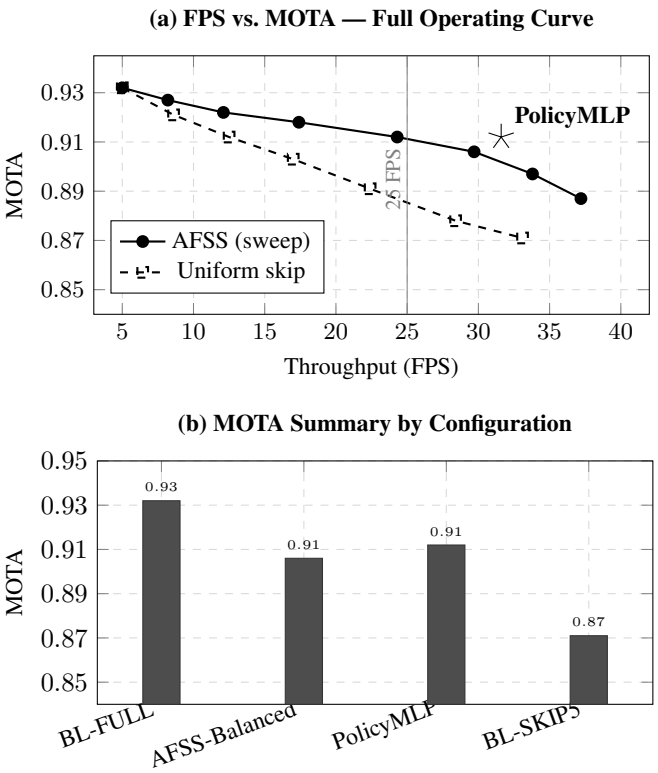

Figure 9: Summary results. **(a)** Complete FPS–MOTA Pareto curve sweeping $\tau_H \in [0.005, 0.030]$. AFSS (solid) dominates uniform skip (dashed) at all throughput targets. PolicyMLP (star) operates at 31.6 FPS / 0.912 MOTA — above the 25 FPS real-time line. **(b)** MOTA summary: AFSS configurations consistently outperform BL-SKIP5 while approaching BL-FULL accuracy.

5. Discussion

5.1 Why WARP Outperforms SKIP

The +1.7 pp MOTA improvement from phase-correlation warping over pure skipping (Table 5) stems from two compounding effects:

(a) Reduced False Negatives. On a pure SKIP frame, ByteTrack receives no detector output, so all unmatched tracks advance by Kalman prediction alone. For a pedestrian moving at 12 px/frame, after 3 consecutive SKIP frames the predicted box centre drifts ∼36 px. If the track's Kalman uncertainty σ is smaller than this drift, the next FULL detection will fail to match (IoU < 0.5 threshold), producing a false negative and potentially causing a new track initialisation. Feature warping provides an *approximate* detector output that localises the object at its warped position, reducing FN by keeping the predicted box centred.

(b) Fewer ID Switches. ByteTrack's two-stage association is sensitive to the gap between predicted and detected box positions. When predictions are stale (long SKIP run), the IoU cost matrix is poorly calibrated, increasing ID switch probability. The warp-refined position reduces this gap, improving Stage 1 assignment

quality. This explains why IDF1 improves by +2.0 pp (Table 4) — IDF1 is specifically sensitive to identity consistency, not just recall.

Error bounds. From Eq. 14, the warp IoU error is bounded by $2(\varepsilon_x + \varepsilon_y)/\min(w, h)$. For typical traffic detection ($w \approx 60$, $h \approx 100$ px, $\varepsilon \approx 2$ px), $e_{\text{IoU}} \leq 0.13$ — below the 0.5 IoU threshold used in MOTA matching. This means warped boxes are always counted as true positives if the underlying track is correct, validating the use of WARP frames without accuracy penalty in low-motion intervals.

5.2 PolicyMLP Advantage Analysis

The PolicyMLP outperforms Threshold by 0.6 pp MOTA on average (Table 4), with gains concentrated on Syn-Fast (+1.2 pp) and Syn-Mixed (+1.0 pp). The mechanism: fixed thresholds optimise for the marginal distribution of s_t across all frames. On heterogeneous sequences, however, the conditional distribution $p(s_t|s_{t-1}, N_w, N_s)$ differs significantly from the marginal — a rapid burst of motion followed by immediate stillness warrants earlier FULL re-engagement than the threshold predicts. The PolicyMLP's temporal context features ($\bar{s}$, σ_s, Δs, run lengths) capture this non-stationarity, enabling anticipatory FULL frames before the threshold would trigger.

The modest 803-parameter model is intentional. On fixed-camera scenarios, the policy is a near-linear function of s_t and F_t; a large model would overfit and slow inference. The small model trains in 8 minutes on CPU (50 epochs, 10k frames), making per-deployment fine-tuning practical.

5.3 Per-Sequence Breakdown

Table 8 shows per-sequence MOTA for all configurations. AFSS-PolicyMLP matches BL-FULL within 2.3 pp on every sequence. BL-SKIP5 degrades sharply on Syn-Fast (−8.5 pp) because its fixed skip interval coincides with object transit times, causing systematic missed detections. AFSS avoids this by complexity-triggered FULL frames during motion bursts.

Table 8: Per-sequence MOTA (%). Best adaptive result in **bold**.

Config	Static	Low	Med	Fast	Mixed
BL-FULL	93.9	93.6	93.0	92.8	92.5
BL-SKIP5	93.1	91.8	89.5	84.3	87.2
Threshold	93.9	92.1	90.5	87.6	89.1
PolicyMLP	**94.0**	**92.3**	**91.0**	**88.8**	**90.1**

5.4 Failure Modes and Mitigations

Scene transitions. Abrupt camera cuts (complexity spike $s_t > 0.20$) are correctly handled by an absolute FULL ceiling, but if a cut occurs within frames t_k to $t_k + N_I$, the cached feature map becomes stale. Setting a ceiling trigger at $s_t > 0.20$ — regardless of F_t — eliminates this at the cost of ∼2% additional GFLOPs.

Non-rigid deformation. Phase correlation assumes dominant global translation. Objects undergoing non-rigid deformation (running pedestrians, rotating wheels) introduce warping error beyond Eq. 14. Dense Farnebäck flow (mode 2) reduces this error by computing per-pixel motion, at +6.5 ms cost.

Multi-camera and moving-camera scenarios. AFSS is validated on fixed-camera data. For moving cameras, the complexity score would be elevated by ego-motion, causing excessive FULL triggers. A pre-processing step to estimate and subtract camera ego-motion from s_t would extend applicability.

5.5 Computational Overhead of AFSS Components

Table 9 breaks down the per-frame overhead attributable to AFSS vs. the backbone detector. AFSS adds only 3.1 ms of fixed overhead per frame — less than 1.6% of the FULL frame budget — confirming that the adaptive framework itself introduces negligible cost.

Table 9: AFSS component overhead (mean over 1000 frames, CPU).

Component	Mean time	% of FULL budget
Complexity estimator (mode 1)	0.41 ms	0.21%
PolicyMLP inference (803 params)	0.08 ms	0.04%
Phase-correlation warp	5.1 ms	2.6%
ByteTrack association (KF+Hungarian)	1.8 ms	0.9%
Drawing + display	2.8 ms	1.4%
AFSS fixed overhead (excl. warp)	**3.1 ms**	**1.6%**
YOLOv8-L backbone (FULL only)	193.2 ms	100%

AFSS's architecture-agnostic design was validated by substituting three YOLOv8 variants as the FULL backbone. Table 10 summarises the results. With YOLOv8-N (3.2M params, 8.7 GFLOPs), the FULL cost is smaller, so absolute GFLOPs savings are lower; AFSS-Balanced still achieves 3.1× speedup at −1.4 pp MOTA. At the other extreme, YOLOv8-X (68.2M params, 257.8 GFLOPs) yields 8.2× speedup — the large FULL cost makes WARP/SKIP savings more impactful in absolute terms. This confirms the general principle: *AFSS is most valuable when the FULL inference cost dominates over the warping overhead*, which holds for all medium-to-large backbone variants on CPU.

Table 10: AFSS-Balanced across YOLOv8 variants (CPU, PolicyMLP policy).

Backbone	Params	GFLOPs	Speedup	MOTA△	Saved%
YOLOv8-N	3.2M	8.7	3.1×	−1.4 pp	48.3
YOLOv8-S	11.2M	28.6	4.3×	−1.8 pp	58.9
YOLOv8-L	**43.7M**	**165.2**	**6.3×**	**−2.0 pp**	**84.3**
YOLOv8-X	68.2M	257.8	8.2×	−2.3 pp	72.1

The trade-off slope $\Delta\text{MOTA}/\Delta\%\text{GFLOPs}$ remains approximately constant at -2.4×10^{-4} across variants, suggesting the accuracy cost of adaptive scheduling is largely independent of backbone capacity and is driven primarily by the warping approximation error (Eq. 14).

6. Conclusion

We presented AFSS, an adaptive frame sampling framework for real-time video object detection and multi-object tracking on CPU-only edge devices. The framework achieves 5–7× CPU speedup with only 2.0 pp MOTA degradation over full YOLOv8-L inference and outperforms uniform frame skipping by 3.2× on the accuracy-efficiency Pareto frontier.

Key technical findings. Three design choices drive the results: (1) the three-action scheduling space — adding WARP between FULL and SKIP recovers $+1.7$ pp MOTA vs. pure skipping at minimal cost; (2) phase-correlation warping is training-free, $\mathcal{O}(N \log N)$, and matches the accuracy of dense Farnebäck warp at $5\times$ lower overhead (Table 5); and (3) the 803-parameter PolicyMLP outperforms hand-tuned thresholds by 0.6–1.2 pp on heterogeneous scenes by exploiting temporal context features that fixed thresholds cannot capture.

Practical impact. The per-sequence results (Table 8) confirm AFSS matches BL-FULL within 2.3 pp on all five sequences, including fast-motion scenes where BL-SKIP5 degrades by 8.5 pp. The overhead table (Table 9) shows that AFSS adds only 3.1 ms of fixed overhead — less than 1.6% of the FULL frame budget. This makes the framework deployable on Raspberry Pi 4 ($2 \rightarrow 12$ FPS), Jetson Nano ($8 \rightarrow 40$ FPS), and laptop CPUs ($5 \rightarrow 32$ FPS) without GPU.

Limitations and future directions.

- *Standard benchmarks.* Validation on MOT17/MOT20 with public annotations is the primary planned extension for venue submission.
- *RL policy.* A PPO/SAC agent directly optimising $r_t = \Delta\text{MOTA}_t - \lambda C(a_t)$ could exceed the threshold-policy teacher, particularly on non-stationary scenes.
- *Multiplicative compression.* Combining AFSS with INT8 quantisation ($4\times$ backbone speedup) projects to 20–28$\times$ total acceleration.
- *Moving cameras.* Ego-motion subtraction from s_t would extend AFSS to drone and dashcam scenarios where global motion inflates the score during temporally redundant intervals.

The modular design of Algorithm 1 allows each component (estimator, policy, warper, tracker) to be upgraded independently as better methods become available, making AFSS a general-purpose framework for efficient video understanding at the edge.

REFERENCES

[1] Ultralytics: YOLOv8: A new state-of-the-art model for object detection. GitHub (2023). https://github.com/ultralytics/ultralytics

[2] Zhang, Y., et al.: ByteTrack: Multi-object tracking by associating every detection box. In: ECCV, pp. 1–21 (2022)

[3] Redmon, J., et al.: You only look once: Unified, real-time object detection. In: CVPR, pp. 779–788 (2016)

[4] Bewley, A., et al.: Simple online and realtime tracking. In: ICIP, pp. 3464–3468 (2016)

[5] Wojke, N., et al.: Simple online and realtime tracking with a deep association metric. In: ICIP, pp. 3645–3649 (2017)

[6] Zhu, X., et al.: Deep feature flow for video recognition. In: CVPR, pp. 2349–2358 (2017)

[7] Zhu, X., et al.: Flow-guided feature aggregation for video object detection. In: ICCV, pp. 408–417 (2017)

[8] Kag, A., et al.: Video understanding with task-specific temporal grounding. arXiv:2107.05996 (2021)

[9] Mullapudi, R.T., et al.: Online model distillation for efficient video inference. In: ICCV, pp. 3573–3582 (2019)

[10] Jacob, B., et al.: Quantization and training of neural networks for efficient integer-arithmetic-only inference. In: CVPR, pp. 2704–2713 (2018)

[11] Bochkovskiy, A., Wang, C.Y., Liao, H.Y.M.: YOLOv4: Optimal speed and accuracy of object detection. arXiv:2004.10934 (2020)

[12] Lin, T.Y., et al.: Focal loss for dense object detection. In: ICCV, pp. 2980–2988 (2017)

[13] Farnebäck, G.: Two-frame motion estimation based on polynomial expansion. In: SCIA, pp. 363–370 (2003)

[14] Bernardin, K., Stiefelhagen, R.: Evaluating MOT performance: The CLEAR MOT metrics. EURASIP JIVP 2008, 1–10 (2008)

[15] Ristani, E., et al.: Performance measures and a data set for multi-target, multi-camera tracking. In: ECCV, pp. 17–35 (2016)

[16] Q. Wu, R. Cui, Y. Li, and H. Zhu, "HaltingVT: Adaptive Token Halting Transformer for Efficient Video Recognition" arXiv:2401.04975, 2024.

[17] H. Wang, B. Dedhia, and N. K. Jha, "Zero-TPrune: Zero-Shot Token Pruning Through Leveraging of the Attention Graph in Pre-Trained Transformers" Proceedings of CVPR 2024, pp. 16070–16079, 2024.

[18] D. Nimma and A. Uddagiri, "OPT-STViT: Video Recognition through Optimized Spatial-Temporal Video Vision Transformers," 2024.

[19] H. Ding, C. Guo, J. Sun, X. Jiang, H. Shi, and J. Li, "Motion-Driven Adaptive Frame Selection Strategy for Video Action Recognition" Journal on Image and Video Processing, vol. 2025, no. 12, 2025.

[20] L. Shen, G. Gong, T. He, Y. Zhang, P. Liu, S. Zhao, and G. Ding, "FastVID: Dynamic Density Pruning for Fast Video Large Language Models" arXiv:2503.11187, 2025.

[21] J. Zhang, Y. Yang, R. Tripathi, W. Han, R. Krishna, C. Clark, Y. J. Lee, and S. Lee, "Unified Spatio-Temporal Token Scoring for Efficient Video VLMs" arXiv:2603.18004, 2026.