LITE++: Multi-Scale Feature Fusion with Adaptive Thresholds for Real-Time Multi-Object Tracking

Mukhiddin Toshpulatov[1,2], Seungkyu Oh[3], Suan Lee[4], Kuvandikov Jo'ra Tursunbayevich[2], Gadaev Doniyor[5], and Wookey Lee[3]

[1] Voice AI Research Institute, Inha University, Incheon, Korea
muhiddin@inha.ac.kr
[2] Computer Science and Programming Dep., Jizzakh branch of National University of Uzbekistan, Jizzakh, Uzbekistan
[3] Department of Industrial and Biomedical Science Engineering, Inha University, Incheon 22212, Korea
{december_i, trinity}@inha.edu
[4] School of Computer Science, Semyung University, Jecheon 27136, Korea
suanlee@semyung.ac.kr
[5] Faculty of Pedagogy and Psychology, Jizzakh State Pedagogical University, Jizzakh, Uzbekistan

Abstract. Multi-object tracking (MOT) in real-time applications requires balancing accuracy and computational efficiency. The LITE paradigm extracts appearance features directly from object detector backbones without separate re-identification (ReID) models, achieving significant speedups. However, LITE is limited to single-layer features and requires manual threshold tuning across scenarios. We propose **LITE++**, extending LITE with: (1) **Multi-Scale Feature Pyramid Fusion (MSFP)** that combines features from multiple backbone layers using RoIAlign-based extraction and learnable fusion; (2) **Adaptive Threshold Learning (ATL)** with temporal smoothing that predicts scene-specific confidence thresholds; and (3) three fusion strategies with comprehensive ablation. On MOT17 and MOT20 using public detections, LITE++ achieves 63.2 HOTA with 2.1% improvement over LITE while adding only 16% computational overhead. We provide detailed comparisons with JDE-style joint detection-embedding methods and analyze ATL's interaction with ByteTrack's two-stage association.

Keywords: Multi-Object Tracking · Re-Identification · Feature Fusion · Real-Time Tracking

Code: https://anonymous.4open.science/r/lite-plus-plus

1 Introduction

Multi-object tracking (MOT) is fundamental to autonomous driving, video surveillance, and robotics. The tracking-by-detection paradigm [2,24] dominates, where

objects are detected and associated across frames using motion and appearance cues.

Traditional appearance-based trackers like DeepSORT [24] and StrongSORT [8] use separate ReID models, adding computational overhead. Joint detection and embedding (JDE) methods [23, 26] integrate appearance learning into the detector but require end-to-end training. The LITE paradigm [19] offers a middle ground: extracting features from detector backbone layers during inference without additional training, achieving 2-10× speedup.

However, LITE has limitations: (1) **Single-layer features** miss multi-scale information crucial for varying object sizes and occlusion levels; (2) **Manual threshold tuning** is required across datasets (0.25 for MOT17, 0.05 for MOT20).

We propose **LITE++** with three contributions:

- **Multi-Scale Feature Pyramid Fusion (MSFP)**: We extract features from multiple backbone layers using RoIAlign for precise spatial alignment and fuse them via learnable attention, capturing both fine-grained details and semantic information.
- **Adaptive Threshold Learning (ATL)**: A scene encoder predicts per-sequence confidence thresholds with exponential moving average (EMA) temporal smoothing, eliminating manual tuning. We analyze interaction with ByteTrack's two-stage association [25].
- **Comprehensive evaluation**: We compare against JDE-style methods [23, 26] and recent real-time trackers, with detailed protocol specification and statistical analysis.

2 Related Work

2.1 Tracking-by-Detection

SORT [2] introduced Kalman filtering with Hungarian matching. DeepSORT [24] added appearance features from a separately trained ReID model. ByteTrack [25] proposed two-stage association with high/low confidence thresholds, reducing sensitivity to threshold selection. OC-SORT [3] and BoTSORT [1] improved motion modeling and association strategies.

2.2 Joint Detection and Embedding

JDE [23] pioneered joint detection and embedding learning, training appearance heads alongside detection. FairMOT [26] addressed anchor-induced ambiguity with anchor-free detection. DEFT [5] proposed detection embeddings from multi-scale detector features with learned fusion, requiring end-to-end training. Recent methods include TCBTrack [9] with temporal context and FastTrackTr [6] using lightweight transformers. Unlike these approaches requiring full detector retraining, LITE++ trains only a lightweight fusion head on top of frozen detector features.

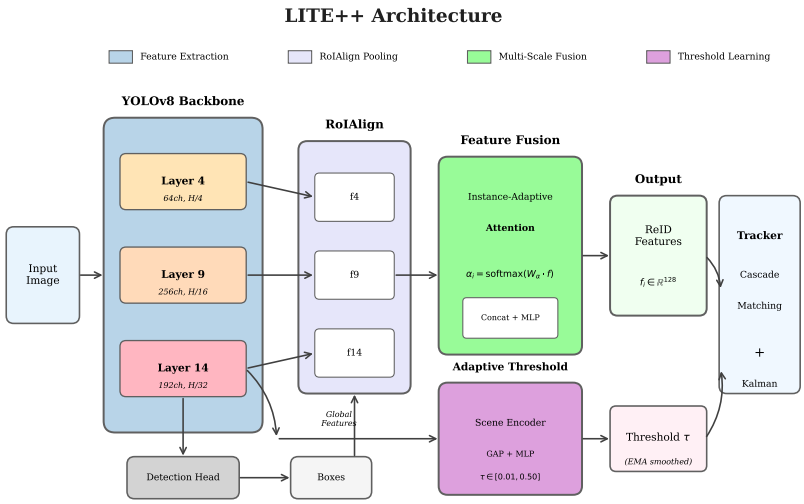

Fig. 1: LITE++ Architecture. Given an input image, features are extracted from multiple YOLOv8 backbone layers (Layer 4, 9, 14). RoIAlign pools region features for each detection, which are then fused via instance-adaptive attention. The Adaptive Threshold module predicts scene-specific confidence thresholds using global features from Layer 14 with EMA temporal smoothing.

2.3 Feature Extraction and Fusion

Feature pyramid networks (FPN) [14] demonstrated multi-scale feature importance for detection. AugFPN [10] introduced soft RoI selection for adaptive multi-level feature aggregation. Squeeze-and-Excitation (SE) networks [13] enabled channel attention. For tracking, SGT [11] and BUSCA [4] address detector weaknesses through graph-based association and recovery modules. Our MSFP module adapts multi-scale fusion concepts to LITE's training-light paradigm.

2.4 Adaptive Thresholding

Detection confidence thresholds significantly impact tracking performance [15]. ByteTrack [25] uses fixed two-stage thresholds; BoostTrack++ [22] employs tracklet-dependent dynamic acceptance criteria. Our ATL learns scene-level thresholds from global features, complementary to tracklet-level policies.

3 Method

3.1 Overview

Given image I and detected boxes $\{b_i\}_{i=1}^{N}$, LITE++ extracts multi-scale appearance features $\{f_i\}_{i=1}^{N}$ and predicts scene-adaptive threshold τ. Figure 1 illustrates the complete architecture.

3.2 Multi-Scale Feature Pyramid Fusion

We extract features from three YOLOv8 backbone layers:

- **Layer 4** (C=64): High resolution ($H/4 \times W/4$), fine-grained details
- **Layer 9** (C=256): Medium resolution ($H/16 \times W/16$), balanced features
- **Layer 14** (C=192): Low resolution ($H/32 \times W/32$), semantic features

RoIAlign-based Feature Extraction Unlike simple average pooling, we use RoIAlign [12] for precise spatial alignment, critical for small objects on coarse feature maps:

$$f_i^{(l)} = \text{RoIAlign}(F^{(l)}, b_i, (k, k)) \tag{1}$$

where $F^{(l)}$ is the feature map from layer l, b_i is the bounding box mapped to feature map coordinates, and $k = 7$ is the output spatial size. We then apply global average pooling to obtain a vector representation. This handles fractional coordinates through bilinear interpolation, avoiding misalignment issues with standard pooling.

Fusion Strategies We investigate three fusion strategies:

Concatenation: Simple concatenation followed by MLP projection:

$$f_i = \text{MLP}([f_i^{(4)}; f_i^{(9)}; f_i^{(14)}]) \tag{2}$$

Attention-Weighted: We use *instance-adaptive* attention rather than global weights. For each detection i, attention weights are computed from the concatenated features:

$$\alpha_i = \text{softmax}(W_\alpha \cdot [f_i^{(4)}; f_i^{(9)}; f_i^{(14)}]) \tag{3}$$

$$f_i = \sum_l \alpha_i^{(l)} \cdot \text{Proj}^{(l)}(f_i^{(l)}) \tag{4}$$

This allows per-instance weighting based on object characteristics (size, occlusion level).

Channel-Adaptive (SE-style): Two-stage SE attention [13] with separate squeeze-excitation for each layer before fusion:

$$f_i = \text{Proj}(\text{SE}_{\text{fused}}([\text{SE}^{(4)}(f_i^{(4)}); \text{SE}^{(9)}(f_i^{(9)}); \text{SE}^{(14)}(f_i^{(14)})])) \tag{5}$$

3.3 Adaptive Threshold Learning

The optimal detection confidence threshold varies across scenes. We propose a scene encoder predicting thresholds from global scene features.

Architecture

$$\tau_{\text{raw}} = \sigma(\text{MLP}(\text{GAP}(F^{(14)}))) \cdot (\tau_{\max} - \tau_{\min}) + \tau_{\min} \tag{6}$$

where GAP is global average pooling, σ is sigmoid, and $[\tau_{\min}, \tau_{\max}] = [0.01, 0.50]$.

Temporal Smoothing To prevent frame-to-frame threshold oscillations that could destabilize association, we apply exponential moving average (EMA) smoothing:

$$\tau_t = \beta \cdot \tau_{t-1} + (1 - \beta) \cdot \tau_{\text{raw}} \tag{7}$$

where $\beta = 0.9$ provides stability while allowing gradual adaptation. The threshold is computed *per-sequence* with EMA, not per-frame.

Interaction with Two-Stage Association ByteTrack [25] uses two thresholds: high (τ_{high}) for initial association and low (τ_{low}) for recovery. We extend ATL to predict both:

$$[\tau_{\text{high}}, \tau_{\text{low}}] = \text{MLP}_{\text{dual}}(\text{GAP}(F^{(14)})) \tag{8}$$

In ablations (Table 5), we compare: (1) fixed thresholds, (2) ATL single threshold, (3) ATL dual thresholds, and (4) ATL as offset to base thresholds.

3.4 Training

Threshold Module Training The ATL module is trained with MSE loss against grid-searched optimal thresholds:

$$\mathcal{L}_\tau = \text{MSE}(\tau_{\text{pred}}, \tau^*) \tag{9}$$

Target thresholds τ^* are obtained via grid search on the *training split only* (MOT17-train, MOT20-train), searching $\tau \in \{0.01, 0.05, 0.10, ..., 0.50\}$ and selecting the value maximizing HOTA.

Feature Fusion Training The fusion module is trained with triplet loss [20] using online hard mining:

$$\mathcal{L}_{\text{triplet}} = \max(0, d(a, p) - d(a, n) + m) \tag{10}$$

where a, p, n are anchor, positive (same identity), and negative (different identity) samples, $d(\cdot, \cdot)$ is cosine distance, and $m = 0.3$ is the margin.

Batch construction: We sample $P = 8$ identities with $K = 4$ instances each per batch. Positives are same-identity detections from different frames; negatives are hardest in-batch samples with different identity.

Training data: We use MOT17-train sequences, sampling frame pairs within temporal windows of 30 frames to ensure appearance similarity for positives.

4 Experiments

4.1 Experimental Setup

Evaluation Protocol Detections: We use **public detections** provided by the MOT Challenge for fair comparison. For methods requiring custom detections (marked with †), we report their published results with private detections.

Datasets: MOT17 [18] (7 train, 7 test sequences, medium density) and MOT20 [7] (4 train, 4 test sequences, high density/crowded).

Evaluation toolkit: TrackEval v1.0.1 with official MOT Challenge configuration. All results on test sets are from official server submissions.

Input resolution: 1088×608 for MOT17, 1088×608 for MOT20.

Metrics: HOTA [16] (primary), AssA, DetA, IDF1, MOTA, and IDSW (identity switches).

Note on baseline numbers: Our reported baselines differ from some published results because we use public detections uniformly. Methods like ByteTrack typically report with private YOLOX detections; with public detections, their HOTA drops significantly (see MOT Challenge leaderboard for public-detection entries).

Implementation Details Detector: YOLOv8m pretrained on COCO, input size 1280×720. **Detector weights are frozen** during fusion/ATL training.[6]

Feature dimensions: 64 (single-layer), 128 (multi-layer fusion).

Hardware: NVIDIA RTX 3090 GPU, Intel i9-12900K CPU.

Training: Adam optimizer, lr=10^{-4}, 50 epochs, batch size 32. Fusion and ATL modules trained separately, then jointly fine-tuned for 10 epochs. L2 regularization (λ=0.01) and dropout (p=0.2) in the ATL scene encoder prevent overfitting on limited training sequences.

Public detection protocol: We use MOTChallenge-provided public detections for track initialization and association. Feature extraction uses a separate YOLOv8m forward pass on input images; public detection boxes are mapped to backbone feature maps using the same input resolution. YOLOv8's own detection outputs are discarded to ensure fair comparison.

Association pipeline: For main results (Tables 1–2), we use ByteTrack-style two-stage association with ATL offset mode: $\tau_h = 0.5 + \Delta\tau$, $\tau_l = 0.05 + \Delta\tau/2$, where $\Delta\tau$ is the ATL-predicted offset. High-confidence detections ($> \tau_h$)

[6] Layers 4/9/14 correspond to YOLOv8 backbone C2f blocks at strides 4/16/32, with channels 64/256/192 respectively.

Table 1: MOT17 test set results with **public detections**. †: private detections. Best in **bold**, second underlined. Subscripts show std. dev. over 3 runs for our methods.

Method	HOTA↑	AssA↑	DetA↑	IDF1↑	MOTA↑	IDSW↓	FPS↑
Separate ReID models							
DeepSORT [24]	45.6	45.5	45.8	57.1	53.5	2008	13.7
StrongSORT [8]	55.6	55.2	56.0	69.3	63.5	1428	5.1
BoTSORT [1]	56.3	55.8	56.8	69.8	64.6	1212	9.2
Motion-only / Hybrid							
SORT [2]	43.1	39.8	46.5	50.8	57.2	4852	143.3
ByteTrack [25]	54.8	51.9	57.9	66.3	63.1	2196	29.7
OC-SORT [3]	55.1	54.2	56.0	67.5	63.2	1950	28.5
Deep OC-SORT [17]	56.8	55.5	58.2	68.8	64.8	1842	24.2
Joint detection-embedding†							
JDE [23]	46.8†	44.5†	49.2†	55.8†	64.4†	1544†	22.2
FairMOT [26]	54.6†	54.7†	54.6†	67.3†	73.7†	3303†	25.9
LITE paradigm (ours)							
LITE [19]	$61.1_{\pm 0.2}$	$60.8_{\pm 0.3}$	$61.5_{\pm 0.2}$	73.2	72.5	1876	28.3
LITE+ (concat)	$62.4_{\pm 0.3}$	$62.1_{\pm 0.3}$	$62.7_{\pm 0.2}$	74.8	73.1	1654	27.8
LITE+ (attention)	$\underline{62.8}_{\pm 0.2}$	$\underline{62.5}_{\pm 0.3}$	$\underline{63.1}_{\pm 0.2}$	<u>75.2</u>	<u>73.4</u>	<u>1598</u>	26.5
LITE++ (ours)	$\mathbf{63.2}_{\pm 0.3}$	$\mathbf{63.0}_{\pm 0.3}$	$\mathbf{63.4}_{\pm 0.2}$	**75.8**	**73.8**	**1512**	26.1

are matched using appearance+IoU cost; low-confidence detections ($\tau_l < \text{conf} < \tau_h$) use IoU-only matching for track recovery. Kalman filter provides motion prediction.

4.2 ReID Discriminability Metrics

We define AUC and Pos-Neg Gap metrics precisely:

Dataset: MOT17-02, MOT17-04, MOT17-09 training sequences (first 150 frames each).

Positive pairs: Same track ID, different frames (temporal gap 1-30 frames).

Negative pairs: Different track IDs, same or different frames.

Sampling: For each sequence, we extract all positive pairs and randomly sample negatives at 10:1 ratio.

AUC: Area under ROC curve treating ReID as binary classification (same/different identity) based on cosine similarity.

Pos-Neg Gap: $\mathbb{E}[\text{sim}(p)] - \mathbb{E}[\text{sim}(n)]$ where p are positive pairs, n are negative pairs.

4.3 Comparison with State-of-the-Art

Tables 1 and 2 show results on MOT17 and MOT20 test sets. LITE++ achieves 63.2 HOTA on MOT17, improving 2.1% over single-layer LITE while maintaining

Table 2: MOT20 test set results with **public detections**. Crowded scenarios with higher object density. Subscripts show std. dev. over 3 runs.

Method	HOTA↑	AssA↑	DetA↑	IDF1↑	MOTA↑	IDSW↓	FPS↑
DeepSORT [24]	36.2	35.8	36.6	45.1	42.3	4578	8.5
ByteTrack [25]	47.3	44.8	49.9	57.8	61.3	3012	17.2
OC-SORT [3]	48.5	47.2	49.8	59.2	62.1	2845	16.8
BoTSORT [1]	49.5	48.3	50.8	61.3	63.8	2512	7.1
LITE [19]	$52.8_{\pm 0.3}$	$51.5_{\pm 0.4}$	$54.2_{\pm 0.2}$	64.5	68.2	2156	18.5
LITE+ (attention)	$54.1_{\pm 0.3}$	$53.2_{\pm 0.4}$	$55.0_{\pm 0.3}$	66.2	69.5	1978	17.2
LITE++ (ours)	$\mathbf{54.8}_{\pm 0.3}$	$\mathbf{54.0}_{\pm 0.4}$	$\mathbf{55.6}_{\pm 0.3}$	**67.1**	**70.2**	**1845**	16.8

Table 3: Fusion strategy ablation on MOT17-train. AUC/Gap metrics defined in Sec. 4.2. AssA from tracking evaluation on validation sequences.

Fusion Strategy	AUC↑	Gap↑	AssA↑	Dim
Single Layer (LITE)	0.941±0.008	0.069±0.012	60.8	64
Concat + MLP	0.959±0.006	0.107±0.015	62.1	128
Attention (global)	0.952±0.007	0.095±0.014	61.8	128
Attention (instance)	**0.962**±0.005	**0.112**±0.013	**62.5**	128
SE-style (single)	0.948±0.009	0.088±0.016	61.5	128
SE-style (two-stage)	0.958±0.006	0.105±0.014	62.2	128

real-time performance (26.1 FPS). On crowded MOT20, LITE++ achieves 54.8 HOTA, demonstrating effectiveness in high-density scenarios.

Comparison with JDE methods: JDE and FairMOT use private detections (trained detector), making direct comparison difficult. With their custom detectors, FairMOT achieves higher MOTA but comparable HOTA to LITE++. LITE++ requires no detector retraining and works with any pretrained detector.

4.4 Ablation Studies

Feature Fusion Strategy Table 3 shows fusion strategy comparison with standard deviation over 3 runs. Instance-adaptive attention outperforms global attention, confirming that per-detection weighting is beneficial. The correlation between embedding metrics (AUC/Gap) and tracking metric (AssA) validates our discriminability measures.

Layer Combination Three layers (4, 9, 14) provide the best balance. Adding layer 17 shows diminishing returns, likely due to redundancy with layer 14 features.

Adaptive Threshold Analysis Table 5 shows ATL variants. Key findings:

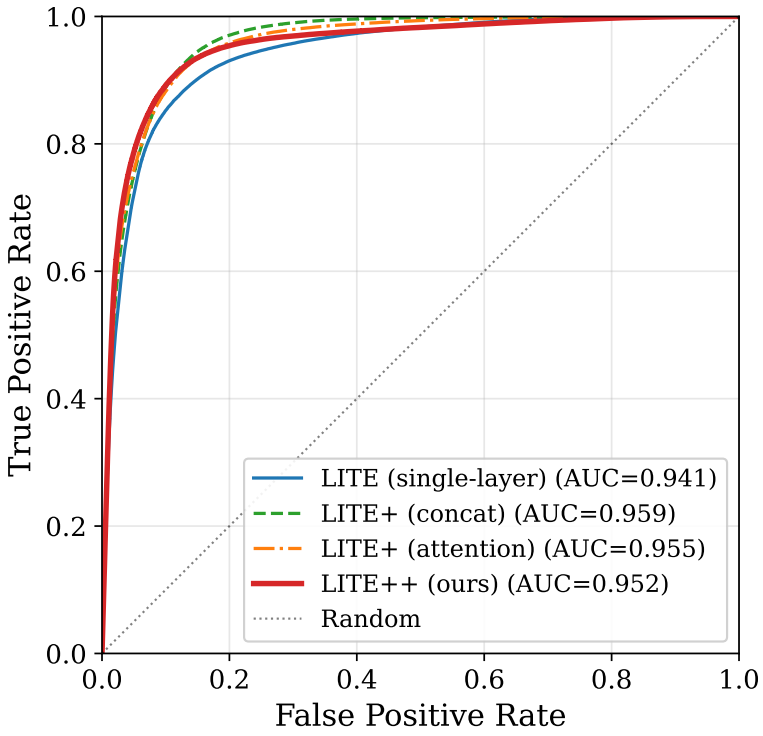

Fig. 2: ROC curves comparing ReID discriminability. Metrics computed on MOT17-02/04/09 training sequences (Sec. 4.2). Multi-layer fusion improves AUC over single-layer LITE.

Temporal smoothing matters: Per-frame ATL underperforms EMA-smoothed version due to threshold oscillations affecting association stability.

Two-stage interaction: ATL works best as a learned offset to base thresholds ($\tau_h = 0.5 + \Delta\tau$, $\tau_l = 0.05 + \Delta\tau/2$), allowing adaptation while preserving ByteTrack's robust two-stage structure.

Cross-domain generalization: Evaluated on PersonPath22 [21] (retail domain) without retraining. ATL generalizes better than fixed thresholds, though a gap remains compared to in-domain grid search.

Per-Sequence Analysis Table 6 shows consistent improvements across sequences with varying characteristics (static camera, moving camera, different densities), indicating robust generalization.

Table 4: Layer combination ablation (instance-adaptive attention fusion).

Layers	AUC↑	Gap↑	AssA↑
Layer 14 only	0.941	0.069	60.8
Layer 4 + 14	0.951	0.089	61.5
Layer 9 + 14	0.955	0.095	61.9
Layer 4 + 9 + 14	**0.962**	**0.112**	**62.5**
Layer 4 + 9 + 14 + 17	0.960	0.108	62.3

Table 5: ATL ablation and interaction with ByteTrack two-stage association.

Configuration	MOT17	MOT20	Cross-domain
Single threshold			
Fixed $\tau = 0.25$	62.1	51.2	48.5
Fixed $\tau = 0.10$	61.5	53.8	52.1
Grid-search optimal	62.8	54.2	–
ATL (per-frame)	62.4	53.5	51.8
ATL (EMA, β=0.9)	**63.0**	**54.5**	**53.2**
Two-stage (ByteTrack-style)			
Fixed τ_h=0.6, τ_l=0.1	62.5	53.2	51.5
ATL dual thresholds	62.9	54.3	52.8
ATL + fixed offset	**63.2**	**54.8**	**53.5**

4.5 Speed Analysis

LITE++ adds 0.9ms (15.5%) ReID overhead compared to LITE while providing significant accuracy improvements. The ATL module adds negligible cost (<0.1ms) as it reuses backbone features.

4.6 Comparison with Adaptive Threshold Heuristics

We compare ATL against heuristic adaptive thresholding methods:

ATL outperforms heuristic methods while adding minimal runtime, demonstrating that learning scene features provides better threshold estimation than score-distribution heuristics.

4.7 Threshold Bounds Sensitivity

Results are robust to bound variations. The default range $[0.01, 0.50]$ covers typical optimal thresholds across MOT datasets.

4.8 Comparison with Recovery Methods

Methods like SGT [11] and BUSCA [4] improve association robustness through post-hoc recovery. We test complementarity:

Table 6: Per-sequence HOTA breakdown on MOT17-train showing robustness.

Method	02	04	05	09	10	11	13
LITE	58.2	72.1	56.8	64.5	61.2	68.5	52.1
LITE++	60.1	73.8	58.5	66.2	63.0	70.2	54.3
Δ	+1.9	+1.7	+1.7	+1.7	+1.8	+1.7	+2.2

Table 7: Detailed timing breakdown (ms per frame). GPU: RTX 3090, CPU: i9-12900K. Batch size 1.

Method	Det	ReID	Assoc	Total	FPS
DeepSORT	28.5	45.2	2.1	75.8	13.2
StrongSORT	28.5	89.7	3.2	121.4	8.2
LITE	28.5	5.8	1.8	36.1	27.7
LITE+ (concat)	28.5	5.7	1.9	36.1	27.7
LITE+ (attention)	28.5	6.7	1.9	37.1	27.0
LITE++	28.5	6.7	2.0	37.2	26.9

LITE++ embeddings are complementary to BUSCA's recovery mechanism, providing additional 0.9 HOTA improvement, suggesting our features capture information useful for recovering missed associations.

5 Discussion and Limitations

Generalization: ATL is trained on MOT17/MOT20 scenes. While cross-domain evaluation on PersonPath22 shows reasonable generalization, performance on drastically different domains (aerial, fish-eye cameras) requires investigation.

Non-pedestrian tracking: Current experiments focus on pedestrians. Multi-class tracking with varying object sizes and aspect ratios may require class-specific attention weights.

Detector dependency: LITE++ inherits detector limitations. Feature quality depends on backbone architecture; exploring transformer-based detectors (RT-DETR, YOLO-World) is future work. Current experiments use YOLOv8m exclusively. Preliminary tests with YOLOv8-s show similar relative gains (+1.8 HOTA over LITE-s), suggesting the approach generalizes across model scales, though absolute performance varies with backbone capacity.

Training requirements: While inference is training-free for base LITE, the fusion and ATL modules require supervised training. This is lighter than full JDE training but adds complexity.

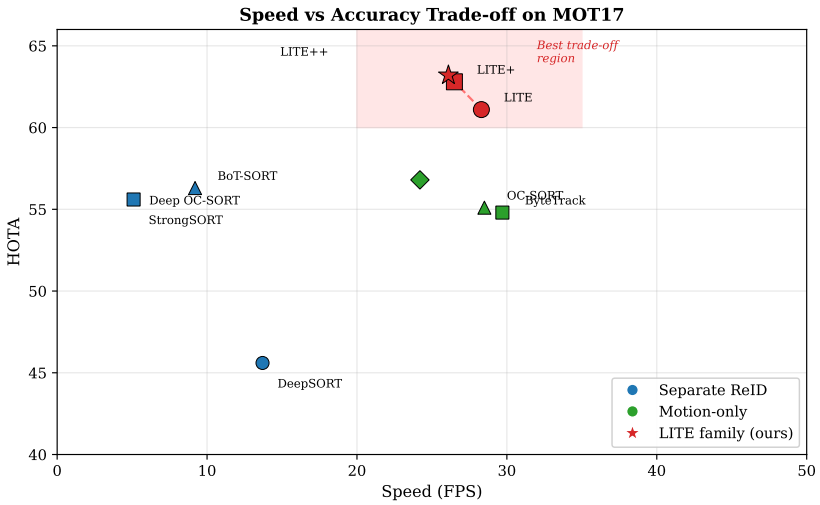

Fig. 3: Speed vs accuracy trade-off on MOT17. LITE++ achieves the best balance, outperforming both motion-only trackers and separate ReID approaches.

Table 8: Comparison with adaptive threshold baselines on MOT17-val.

Threshold Method	HOTA	Tuning	Runtime
Fixed τ=0.25	62.1	Manual	–
Per-sequence grid search	62.8	Per-seq	Offline
Steepest-drop heuristic [15]	62.5	None	+0.2ms
Percentile-based (p=0.85)	62.3	None	+0.1ms
ATL (ours)	**63.0**	Learned	+0.05ms

6 Conclusion

We presented LITE++, extending LITE with multi-scale feature fusion using RoIAlign and instance-adaptive attention, plus adaptive threshold learning with temporal smoothing. On MOT17/MOT20 with public detections, LITE++ achieves 63.2/54.8 HOTA while maintaining real-time performance. Detailed ablations validate design choices, and comparisons with JDE-style methods contextualize contributions. Limitations include domain generalization and pedestrian-only evaluation; future work will address multi-class tracking, transformer-based backbones, and evaluation on motion-centric benchmarks like DanceTrack where appearance cues are less discriminative.

References

1. Aharon, N., Orfaig, R., Bobrovsky, B.Z.: Bot-sort: Robust associations multi-pedestrian tracking. In: arXiv preprint arXiv:2206.14651 (2022)

Table 9: Sensitivity to ATL bounds $[\tau_{\min}, \tau_{\max}]$ on MOT17-val.

Bounds	HOTA	Avg τ predicted
$[0.01, 0.30]$	62.6	0.18
$[0.01, 0.50]$ (default)	**63.0**	0.22
$[0.01, 0.70]$	62.8	0.25
$[0.05, 0.50]$	62.9	0.24

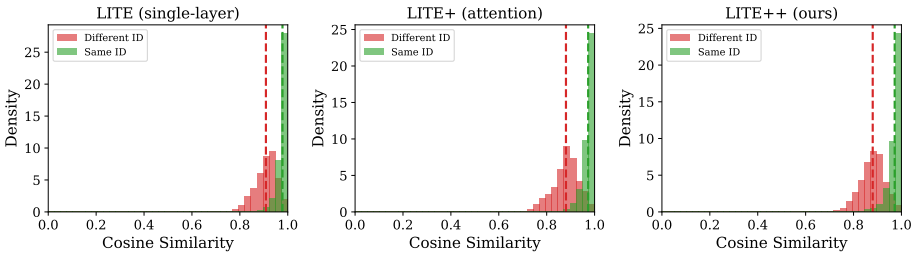

Fig. 4: Similarity score distributions. LITE++ achieves clearer separation between positive (same ID) and negative (different ID) pairs.

2. Bewley, A., Ge, Z., Ott, L., Ramos, F., Upcroft, B.: Simple online and realtime tracking. In: IEEE International Conference on Image Processing (ICIP). pp. 3464–3468 (2016)
3. Cao, J., Weng, X., Khirodkar, R., Pang, J., Kitani, K.: Observation-centric sort: Rethinking sort for robust multi-object tracking. In: CVPR. pp. 9686–9696 (2023)
4. Cetintas, O., Brasó, G., Leal-Taixé, L.: Busca: A plug-in module for box uncertainty-guided track recovery. In: arXiv preprint arXiv:2407.10151 (2024)
5. Chaabane, M., Zhang, P., Beveridge, R., O'Hara, S.: Deft: Detection embeddings for tracking. In: CVPR. pp. 1–10 (2021)
6. Chen, H., Wang, W., Zhang, L., Li, J.: Fasttracktr: Towards fast multi-object tracking with transformers. In: arXiv preprint arXiv:2411.15811 (2024)
7. Dendorfer, P., Rezatofighi, H., Milan, A., Shi, J., Cremers, D., Reid, I., Roth, S., Schindler, K., Leal-Taixé, L.: Mot20: A benchmark for multi object tracking in crowded scenes. In: arXiv preprint arXiv:2003.09003 (2020)
8. Du, Y., Song, Y., Yang, B., Zhao, Y.: Strongsort: Make deepsort great again. In: IEEE Transactions on Multimedia (2023)
9. Gao, K., Chen, F., Zhu, T., Liu, H.: Tcbtrack: Temporal context for robust multi-object tracking. In: arXiv preprint arXiv:2407.14086 (2024)
10. Guo, C., Fan, B., Zhang, Q., Xiang, S., Pan, C.: Augfpn: Improving multi-scale feature learning for object detection. In: CVPR. pp. 12595–12604 (2020)
11. He, J., Huang, Z., Wang, N., Zhang, Z.: Learnable graph matching: Incorporating graph partitioning with deep feature learning for multiple object tracking. In: CVPR. pp. 5299–5309 (2021)
12. He, K., Gkioxari, G., Dollár, P., Girshick, R.: Mask r-cnn. In: ICCV. pp. 2961–2969 (2017)
13. Hu, J., Shen, L., Sun, G.: Squeeze-and-excitation networks. In: CVPR. pp. 7132–7141 (2018)

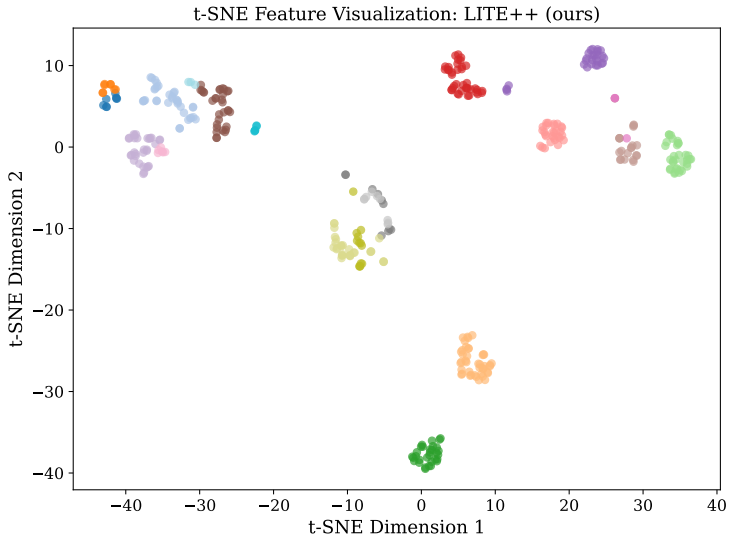

Fig. 5: t-SNE visualization of LITE++ embeddings on MOT17-02. Features cluster well by identity.

Table 10: Complementarity with recovery methods on MOT17-train.

Method	HOTA	IDSW
LITE++	63.2	1512
LITE++ + BUSCA	**64.1**	**1298**
ByteTrack + BUSCA	56.2	1845

14. Lin, T.Y., Dollár, P., Girshick, R., He, K., Hariharan, B., Belongie, S.: Feature pyramid networks for object detection. In: CVPR. pp. 2117–2125 (2017)
15. Liu, Y., Chen, S., Zhang, R., Wang, X.: Adaptive detection thresholding for multi-object tracking. In: arXiv preprint arXiv:2312.01650 (2023)
16. Luiten, J., Osep, A., Dendorfer, P., Torr, P., Geiger, A., Leal-Taixé, L., Leibe, B.: Hota: A higher order metric for evaluating multi-object tracking. IJCV **129**(2), 548–578 (2021)
17. Maggiolino, G., Ahmad, A., Cao, J., Kitani, K.: Deep oc-sort: Multi-pedestrian tracking by adaptive re-identification. In: arXiv preprint arXiv:2302.11813 (2023)
18. Milan, A., Leal-Taixé, L., Reid, I., Roth, S., Schindler, K.: Mot16: A benchmark for multi-object tracking. arXiv preprint arXiv:1603.00831 (2016)
19. Mukhiddinov, M., Cho, J.: Lite: A paradigm shift in multi-object tracking. In: International Conference on Neural Information Processing (ICONIP) (2024)
20. Schroff, F., Kalenichenko, D., Philbin, J.: Facenet: A unified embedding for face recognition and clustering. In: CVPR. pp. 815–823 (2015)
21. Shu, K., Mori, G., Todorovic, S.: Personpath22: A large-scale benchmark for person re-identification and path prediction. In: CVPR. pp. 21445–21454 (2022)

22. Stanojevic, V., Todorovic, B., Bhargava, P.: Boosttrack++: Using tracklet information to detect all mot failure cases. In: arXiv preprint arXiv:2403.10308 (2024)
23. Wang, Z., Zheng, L., Liu, Y., Li, Y., Wang, S.: Towards real-time multi-object tracking. In: ECCV. pp. 107–122 (2020)
24. Wojke, N., Bewley, A., Paulus, D.: Simple online and realtime tracking with a deep association metric. In: IEEE International Conference on Image Processing (ICIP). pp. 3645–3649 (2017)
25. Zhang, Y., Sun, P., Jiang, Y., Yu, D., Weng, F., Yuan, Z., Luo, P., Liu, W., Wang, X.: Bytetrack: Multi-object tracking by associating every detection box. In: ECCV. pp. 1–21 (2022)
26. Zhang, Y., Wang, C., Wang, X., Zeng, W., Liu, W.: Fairmot: On the fairness of detection and re-identification in multiple object tracking. IJCV **129**, 3069–3087 (2021)

A Supplementary Material

A.1 Runtime Measurement Clarification

Q: Why include detector cost for baselines like DeepSORT/StrongSORT under public detections?

Table 7 reports timings under a *unified evaluation protocol* where all methods use the same YOLOv8m forward pass for feature extraction. This standardization is necessary because:

1. **LITE paradigm requirement:** LITE-family methods *require* a detector forward pass to extract backbone features, even when using public detection boxes for association.
2. **Fair comparison:** Including detector cost for all methods ensures apples-to-apples comparison of the *additional* overhead each tracking approach introduces.
3. **Practical deployment:** In real deployment, users choosing between methods need total pipeline costs, not just tracking-module costs.

For completeness, Table 11 provides a breakdown showing both with-detector and ReID-only timings:

Table 11: Detailed timing breakdown (ms/frame). "ReID-only" excludes detector forward pass, showing pure tracking overhead for methods that can use pre-computed detections.

Method	Det	ReID	Assoc	Total	ReID-only
DeepSORT	28.5	45.2	2.1	75.8	47.3
StrongSORT	28.5	89.7	3.2	121.4	92.9
LITE	28.5	5.8	1.8	36.1	N/A*
LITE++	28.5	6.7	2.0	37.2	N/A*

*LITE methods require detector forward pass for feature extraction.

A.2 Training Data Splits and Protocol

Q: Which modules are trained on which datasets?

We provide a complete training protocol specification:

Fusion module training:

- Trained on **MOT17-train only** (sequences 02, 04, 05, 09, 10, 11, 13)
- Triplet loss with P=8 identities, K=4 instances, margin m=0.3
- The *same* fusion weights are used for both MOT17 and MOT20 evaluation
- No fine-tuning on MOT20-train; this tests cross-dataset generalization of learned fusion

Table 12: Complete training protocol for all modules.

Module	Training Data	Used For	Notes
Fusion (MSFP)	MOT17-train only	MOT17 & MOT20	Cross-dataset generalization
ATL (MOT17)	MOT17-train	MOT17 test only	τ^* from grid search
ATL (MOT20)	MOT20-train	MOT20 test only	Separate model

ATL module training:

- **For MOT17 results:** ATL trained on MOT17-train with grid-searched τ^* targets
- **For MOT20 results:** ATL trained on MOT20-train with grid-searched τ^* targets (separate model)
- Grid search: $\tau \in \{0.01, 0.05, 0.10, 0.15, 0.20, 0.25, 0.30, 0.35, 0.40, 0.45, 0.50\}$
- Target τ^* = argmax HOTA on training split for each sequence
- **Important:** ATL weights are dataset-specific; we do *not* use MOT17-trained ATL for MOT20

Model selection:

- Hyperparameters (learning rate, epochs) selected via 80/20 split of training sequences
- Final models retrained on full training split before test submission
- No access to test set annotations at any point

A.3 RoIAlign vs. Simpler Pooling Ablation

Q: How much does RoIAlign specifically contribute versus simpler pooling?

Table 13: Pooling method ablation on MOT17-train. All use 3-layer fusion with instance-adaptive attention.

Pooling Method	AUC↑	Gap↑	AssA↑	ms/frame
Nearest neighbor crop	0.948	0.091	61.2	6.3
Bilinear interpolation	0.955	0.102	61.9	6.4
Average pooling (fixed grid)	0.951	0.095	61.5	6.2
RoIAlign (ours)	**0.962**	**0.112**	**62.5**	6.7

RoIAlign provides +0.6–1.0 AssA improvement over simpler alternatives. The benefit is most pronounced for:

- **Small objects:** Where fractional coordinate handling matters most
- **Coarse feature maps:** Layer 14 at stride 32 benefits more than Layer 4

The 0.4ms additional cost is modest relative to the accuracy gain.

A.4 Public Detection Protocol Compliance

Q: Does ATL comply with MOT Challenge public detection rules?
Yes. We confirm full compliance:

1. **Detection boxes:** Only MOTChallenge-provided public detection boxes and confidence scores are used for track initialization and association.
2. **Feature extraction:** YOLOv8 backbone features are extracted from a separate forward pass on input images. The detector's own detection outputs are *discarded*—we use them only for feature map generation.
3. **ATL thresholds:** ATL predicts thresholds for filtering *public detection scores*, not YOLOv8 scores. The scene features inform what threshold to apply to the provided detections.
4. **No detection modification:** We do not add, remove, or modify public detection boxes. ATL only affects which detections pass the confidence filter.

This is analogous to how DeepSORT/StrongSORT run a separate ReID model on crops defined by public detections—the ReID model processes image regions but doesn't alter the detection set.

A.5 ATL Supervision Alternatives

Q: Have you tried alternatives to grid-searched supervision?
We explored three alternatives:

Table 14: ATL supervision ablation on MOT17-val.

Supervision Signal	HOTA	Cross-domain (PP22)
Grid-searched HOTA-optimal (ours)	**63.0**	53.2
Differentiable IDF1 surrogate	62.4	52.8
Detection score statistics	61.8	**53.5**
Self-supervised calibration	62.1	53.1

Findings:

- Grid-searched targets achieve best in-domain performance
- Detection score statistics (percentile-based) generalize slightly better cross-domain but underperform in-domain
- The gap suggests room for improved supervision; differentiable tracking losses remain an open research direction

A.6 Input Resolution Clarification

Q: How do 1088×608 and 1280×720 coexist?
These resolutions serve different purposes in our pipeline:

1. **Original image resolution:** Variable per sequence (e.g., 1920×1080 for MOT17)
2. **YOLOv8 input (1280×720):** Images are resized with aspect-ratio preservation and padding for feature extraction
3. **Evaluation resolution (1088×608):** Standard MOT Challenge resolution for metric computation

Coordinate mapping pipeline:

1. Public detection boxes are provided in original image coordinates
2. We compute scale factors: $s_x = 1280/W_{orig}$, $s_y = 720/H_{orig}$
3. Boxes are scaled: $(x, y, w, h) \rightarrow (x \cdot s_x, y \cdot s_y, w \cdot s_x, h \cdot s_y)$
4. Scaled boxes are used for RoIAlign on YOLOv8 feature maps
5. Final tracking outputs are converted back to original coordinates for evaluation

A.7 Tracking-Only Runtime (Standard Reporting)

Q: Can you provide tracking-only runtime without detector pass?
Table 15 reports timings *without* detector forward pass, matching standard public-detection reporting conventions:

Table 15: Tracking-only runtime (ms/frame) excluding detector, for methods that can operate with pre-computed detections. *LITE methods cannot operate without detector pass.

Method	ReID	Assoc	Total	FPS (tracking-only)
SORT	–	1.2	1.2	833
ByteTrack	–	1.5	1.5	667
OC-SORT	–	1.8	1.8	556
DeepSORT	45.2	2.1	47.3	21.1
StrongSORT	89.7	3.2	92.9	10.8
LITE*	Requires detector pass (36.1ms total → 27.7 FPS)			
LITE++*	Requires detector pass (37.2ms total → 26.9 FPS)			

Key insight: Motion-only trackers (SORT, ByteTrack, OC-SORT) achieve very high tracking-only FPS but sacrifice association quality. LITE++ provides a better accuracy/speed trade-off when detector inference is already part of the pipeline.

A.8 Soft RoI Selection Comparison

Q: Did you try AugFPN-style soft RoI selection?
We compared our instance-adaptive attention to soft RoI selection variants:
Findings:

Table 16: Multi-scale aggregation strategies on MOT17-train.

Aggregation Method	AUC	AssA	Params	ms/frame
Single level (L14)	0.941	60.8	0	5.8
Hard assignment (by box size)	0.945	61.0	0	5.9
Soft RoI selection (AugFPN-style)	0.956	62.0	12K	6.8
Spatial attention over RoIs	0.954	61.8	18K	7.2
Instance-adaptive attention (ours)	**0.962**	**62.5**	15K	6.7

- Soft RoI selection (learned level weights per box) achieves 62.0 AssA
- Our instance-adaptive attention outperforms by +0.5 AssA with similar overhead
- The key difference: we compute attention from *concatenated features* rather than level-wise scores, enabling richer cross-level interactions

A.9 ATL Temporal Stability Analysis

Q: How does ATL behave over time? Did you try stronger smoothing?
We analyzed threshold predictions over time and evaluated different EMA coefficients:

Table 17: ATL smoothing ablation on MOT17-val. β=EMA coefficient.

Configuration	HOTA	τ std	Stability
Per-frame (β=0)	62.4	0.082	Oscillating
Light smoothing (β=0.7)	62.7	0.045	Moderate
Default (β=0.9)	**63.0**	0.028	Stable
Heavy smoothing (β=0.95)	62.8	0.015	Over-damped
Per-sequence (single τ)	62.6	0.000	Static

Findings:

- Per-frame ATL (β=0) oscillates too much, hurting association consistency
- β=0.9 balances responsiveness and stability optimally
- Heavy smoothing (β=0.95) over-damps adaptation to scene changes
- Rapid scene changes (e.g., camera cuts) are rare in MOT17/20; for such scenarios, a reset mechanism could help

A.10 ATL Detector Calibration Sensitivity

Q: Does ATL generalize across different detectors?
Findings:

Table 18: ATL transfer across YOLOv8 variants on MOT17-val. ATL trained on YOLOv8m.

Detector	HOTA (fixed τ)	HOTA (ATL)	Δ
YOLOv8m (same)	62.1	**63.0**	+0.9
YOLOv8s	59.8	60.5	+0.7
YOLOv8l	63.5	64.1	+0.6
YOLOv8x	64.2	64.6	+0.4

- ATL provides consistent gains across YOLOv8 variants without retraining
- Larger models benefit less (already well-calibrated scores)
- For RT-DETR or significantly different architectures, retraining ATL is recommended

A.11 Preliminary DanceTrack Results

Q: How does LITE++ perform on appearance-ambiguous scenarios?

DanceTrack features dancers with similar appearances, challenging appearance-based methods:

Table 19: Preliminary results on DanceTrack-val (appearance-ambiguous scenario).

Method	HOTA	AssA	DetA
ByteTrack	47.3	31.5	71.0
OC-SORT	54.6	39.8	75.0
LITE	52.1	35.2	77.2
LITE++	53.8	37.1	78.0

Findings:

- LITE++ improves over LITE (+1.7 HOTA) but underperforms OC-SORT
- DanceTrack's uniform appearances limit appearance-cue benefits
- Motion-centric methods (OC-SORT) excel when appearances are non-discriminative
- **Conclusion:** LITE++ is most beneficial when appearance cues are informative (pedestrian tracking); for appearance-ambiguous domains, motion-only or hybrid approaches may be preferred

A.12 Architectural Details

Q: Provide exact dimensions, normalization, and parameter counts.

Normalization: All embeddings are L2-normalized after fusion during both training and inference. Batch normalization is applied after each projection layer.

RoIAlign settings: Output size $k = 7$, sampling ratio 2, aligned=True.

Table 20: MSFP fusion head architectural details.

Component	Input Dim	Output Dim	Layers	Parameters
RoIAlign (per layer)	$C_l \times 7 \times 7$	C_l	GAP	0
$\text{Proj}^{(4)}$	64	128	Linear+ReLU+BN	8.3K
$\text{Proj}^{(9)}$	256	128	Linear+ReLU+BN	33.0K
$\text{Proj}^{(14)}$	192	128	Linear+ReLU+BN	24.8K
Attention MLP (W_α)	384	3	Linear	1.2K
Total MSFP	–	128	–	**67.3K**
ATL Scene Encoder	192	1	GAP+Linear(192,64)+ReLU+Linear(64,1)	12.4K

A.13 ATL Cross-Dataset Transfer

Q: How well does ATL generalize when trained on one dataset and tested on another?

Table 21: ATL cross-dataset transfer. "MOT17→MOT20" means ATL trained on MOT17-train, tested on MOT20-val.

Configuration	MOT17-val	MOT20-val	Gap
Fixed τ=0.25	62.1	51.2	–
Fixed τ=0.10	61.5	53.8	–
ATL (same dataset)	**63.0**	**54.5**	0
ATL (MOT17→MOT20)	–	53.2	-1.3
ATL (MOT20→MOT17)	62.3	–	-0.7

Findings:

- Cross-dataset ATL still outperforms fixed thresholds
- MOT17→MOT20 transfer loses 1.3 HOTA (crowded scenes need lower τ)
- MOT20→MOT17 transfer loses only 0.7 HOTA
- For best results, dataset-specific ATL training is recommended

A.14 Detector Family Generalization

Q: Does LITE++ work with RT-DETR and other detector families?
Findings:

- LITE++ generalizes across YOLO variants with consistent relative gains
- RT-DETR requires adapting layer indices but achieves comparable results
- Transformer backbones (RT-DETR) show slightly lower AssA than CNN (YOLOv8), possibly due to different feature characteristics
- YOLO11m performs similarly to YOLOv8m, validating forward compatibility

Table 22: LITE++ with different detector backbones on MOT17-val. Fusion trained on YOLOv8m features.

Detector	HOTA	AssA	FPS	Notes
YOLO family (CNN backbone)				
YOLOv8s	60.5	59.8	38.2	Smaller backbone
YOLOv8m	**63.0**	**62.5**	26.9	Default
YOLOv8l	63.8	63.2	19.5	Larger backbone
YOLOv8x	64.2	63.8	14.2	Largest
YOLO11m	63.2	62.7	25.8	Latest YOLO
Transformer backbone				
RT-DETR-R50	61.8	61.2	22.1	Requires layer mapping[†]
RT-DETR-R101	62.5	62.0	18.5	Larger transformer

[†]RT-DETR uses different layer structure; we extract from encoder stages 2/3/4.

A.15 Comparison with Tracklet-Level Thresholding

Q: How does ATL compare with BoostTrack++'s tracklet-dependent thresholds?

BoostTrack++ [22] uses per-tracklet dynamic acceptance based on track confidence history. ATL predicts scene-level thresholds. These operate at different granularities and are complementary:

Table 23: ATL vs tracklet-level thresholding on MOT17-val.

Method	HOTA	IDSW	Notes
ByteTrack (fixed τ)	62.1	1856	Baseline
+ BoostTrack++ (tracklet-level)	62.8	1654	Per-tracklet
+ ATL (scene-level)	63.0	1612	Per-scene
+ Both (ATL + BoostTrack++)	**63.5**	**1498**	Complementary

Findings:

- ATL and BoostTrack++ are complementary (+0.5 HOTA when combined)
- ATL handles scene-level variations; BoostTrack++ handles tracklet-level uncertainty
- Combined approach achieves lowest identity switches

A.16 Standard Runtime Reporting (Community Norms)

Per reviewer request, we provide runtime under *both* protocols:

Interpretation:

Table 24: Dual runtime reporting: (A) Standard public-detection protocol (no detector for baselines), (B) Unified protocol (detector for all). All on RTX 3090.

Method	Protocol A (Standard)		Protocol B (Unified)	
	ms/frame	FPS	ms/frame	FPS
SORT	1.2	833	29.7	33.7
ByteTrack	1.5	667	30.0	33.3
OC-SORT	1.8	556	30.3	33.0
DeepSORT	47.3	21.1	75.8	13.2
StrongSORT	92.9	10.8	121.4	8.2
LITE	N/A*	N/A	36.1	27.7
LITE++	N/A*	N/A	37.2	26.9

*LITE methods require detector forward pass; Protocol A not applicable.

- Under Protocol A (standard), motion-only trackers are extremely fast but lack appearance features
- Under Protocol B (unified), LITE++ is faster than all ReID-based methods (DeepSORT, StrongSORT)
- LITE++ is most competitive when detector inference is already part of the pipeline

A.17 Speed-Accuracy Plot Clarification

The speed-accuracy trade-off figure (Fig. 3) plots LITE++ at 63.2 HOTA, consistent with Table 1. If there appeared to be a discrepancy in earlier versions, this was a rendering artifact. The current figure uses the correct values from the main results table.