

Real-Time PPE Violation Detection and Tracking in Low-Light Industrial Environments Using YOLOv9t, TensorRT, and ByteTrack

Fathorazi Nur Fajri^{1,*}, Kamil Malik², Abu Tholib³

^{1,2,3}Department of Informatics, Nurul Jadid University, Jl. PP Nurul Jadid Paiton, Probolinggo 67291, Indonesia

(Received: April 25, 2026; Revised: July 1, 2026; Accepted: September 1, 2026; Available online: October 1, 2026)

Abstract

Coal-fired power plants require correct PPE use, but low light in turbine halls, corridors, and nighttime inspection routes limits automated visual monitoring. This study builds and tests a real-time detection-and-tracking pipeline combining a YOLOv9t detector, TensorRT inference, and ByteTrack multi-object tracking for PPE violation monitoring under these conditions. A photometric check of the evaluation footage (mean pixel intensity 117.86/255, 14.68% of pixels below the 64/255 dark-pixel threshold) places it in the dim-illumination band defined by the IEEE 1789 photometric convention, confirming the reported gains were measured under genuinely degraded lighting. Trained on a violation-aware dataset of 11,407 annotated images across eight compliance and violation classes, TensorRT cut mean inference latency from 34.55 ms (PyTorch) to 6.26 ms (FP32) and 5.25 ms (FP16), a paired t-test ($t = 13.1223$, $p < 0.001$) confirming the engine-level gain, while mAP@50 dropped only marginally (0.9134 to 0.9084). Because TensorRT builds can vary run to run, FP32 and FP16 were each validated across independent builds: FP16 matched FP32 in accuracy and ran 1.32x faster (5.25 vs 6.93 ms, $p < 0.001$), so FP16 was selected for the pipeline below. Extending the comparison beyond one tracker, ByteTrack, DeepSORT, and SORT were benchmarked on identical input from a 128-second, 3,840-frame low-light CCTV clip at the PLTU Paiton gate. ByteTrack reached 82.62 FPS with MOTA 0.7543 (relative to a pseudo ground truth), clearing our 60 FPS real-time threshold ($2\times$ the 30 FPS source rate) with the best tracking accuracy of the three; DeepSORT managed only 27.08 FPS despite re-identification embeddings, and SORT (90.29 FPS) produced over $1.3\times$ ByteTrack's identity switches. Across 320 worker identities, mean track duration was 26.74 frames (0.89 s).

Keywords: Personal Protective Equipment (PPE) Detection, Low-Light Industrial Surveillance, YOLOv9t, TensorRT Inference Optimization, ByteTrack

1. Introduction

Workplace safety in heavy industrial facilities, particularly coal fired power plants (PLTU), remains a significant challenge because workplace accidents can have severe consequences for both workers and operational continuity. The International Labour Organization (ILO) estimates that approximately 2.78 million people die annually as a result of occupational accidents and work related diseases, with heavy industries contributing substantially to this burden [1]. In Indonesia, BPJS Ketenagakerjaan reported 265,334 workplace accidents in 2022, with many incidents associated with improper use of personal protective equipment (PPE) [2]. At this scale, automated PPE monitoring is not merely an issue of operational efficiency but also a matter of occupational health and regulatory compliance. Such systems must therefore perform reliably under actual industrial conditions, including uneven and insufficient illumination, rather than under the bright and controlled lighting commonly used in demonstration environments. Manual inspection is also inherently limited because human inspectors are susceptible to fatigue, subjective judgment, and restricted visibility across large industrial sites [3].

Consequently, deep learning based computer vision methods have increasingly been adopted for automated PPE detection, with the You Only Look Once (YOLO) family becoming widely used because of its combination of detection accuracy and computational efficiency [4]. However, many existing systems have been developed and evaluated under favorable lighting conditions, while operational environments such as dim turbine halls, underground utility corridors, and outdoor surveillance areas at night remain insufficiently evaluated with comparable statistical rigor [5]. Low light conditions introduce challenges beyond simple image darkening. Reduced illumination increases sensor noise, weakens

*Corresponding author: Fathorazi Nur Fajri (fathorazi@unuja.ac.id)

DOI: doi: 10.47738/jads.v7i4.1513

This is an open access article under the CC-BY license (<https://creativecommons.org/licenses/by/4.0/>).

© Authors retain all copyrights

object boundaries, and degrades the visibility of small objects such as gloves and safety shoes, which are already among the more difficult PPE categories to detect reliably [6].

Recent studies have begun to address low light PPE detection as a specific engineering challenge rather than treating illumination as a secondary consideration. Super resolution based preprocessing has been shown to recover YOLOv8 detection performance under severe image degradation [7], while multi scale feature gated fusion has improved the stability of small object detection, particularly for safety helmets, while addressing the complexity associated with low light scenes [8]. These findings support the importance of illumination as a key factor in PPE detection performance. However, these approaches have not been investigated jointly with inference acceleration and multi object tracking, which represents an important gap addressed in this study. Previous work by the authors investigated the problem at the image level by developing an 11,407 image violation aware dataset and comparing CLAHE, Auto CLAHE, and EnlightenGAN preprocessing methods using YOLOv9t. The best configuration achieved an mAP@50 of 0.915 on static images [9]. However, extending detection from static images to live video introduces additional challenges that cannot be fully captured by a static benchmark. These challenges include maintaining worker identities across consecutive frames, reducing transient false detections, and sustaining real time processing throughout a continuous video stream.

Bridging the gap between a research prototype and a system capable of operating across an industrial camera network presents another important challenge. Existing research has paid comparatively limited attention to the integration of inference optimization and multi object tracking for low light PPE monitoring. NVIDIA TensorRT improves inference efficiency through techniques such as computational graph optimization and precision tuning [10], while recent edge deployment studies have reported substantial latency reductions after converting YOLO based models to TensorRT for resource constrained hardware [11]. However, the selection of numerical precision is commonly treated as a default configuration rather than as an experimentally validated deployment decision. In addition, relatively limited research has investigated the combination of TensorRT optimized YOLO models with multi object tracking for PPE monitoring. ByteTrack associates detection boxes across consecutive frames, including detections with relatively low confidence, to improve identity continuity and has demonstrated strong performance in pedestrian tracking benchmarks [12]. Nevertheless, comparative studies indicate that different tracking algorithms exhibit different trade offs in terms of throughput, detection accuracy, and identity switching, making tracker selection an empirical rather than purely conventional decision [13]. Therefore, this study addresses the low light PPE monitoring problem by developing a real time PPE violation detection and tracking pipeline that integrates a YOLOv9t detector, TensorRT based inference, and multi object tracking. TensorRT FP32 inference is initially evaluated using measured comparative data, while alternative tracker configurations are evaluated end to end using real low light industrial CCTV footage with photometric verification of the illumination conditions.

The evaluation is conducted through several stages. First, TensorRT FP32 inference is compared with PyTorch inference to quantify the acceleration achieved by the optimized deployment. The comparison reports a 5.52 times improvement in inference speed and uses paired sample statistical analysis rather than relying on a single measurement. Second, the numerical precision configuration is evaluated by comparing FP32 and FP16 TensorRT engines using repeated engine builds and a controlled latency benchmark. This evaluation identifies the faster precision configuration while verifying that the selected configuration does not reduce detection accuracy for the evaluated detector on Tesla T4 hardware. Third, the proposed pipeline evaluates three multi object tracking methods, namely ByteTrack, DeepSORT, and SORT, in combination with TensorRT accelerated YOLOv9t for PPE violation detection under low light industrial CCTV conditions. Rather than selecting a tracker based on convention, the alternatives are compared using consistent experimental conditions. Fourth, an ablation study evaluates five system configurations using 3,840 frames obtained from real PLTU CCTV footage. The evaluation considers end to end frames per second (FPS), detection coverage, and multiple object tracking metrics, including Multiple Object Tracking Accuracy (MOTA), Multiple Object Tracking Precision (MOTP), and IDF1. The illumination conditions are also measured to characterize the operational environment quantitatively rather than relying on qualitative descriptions of darkness. Fifth, the study evaluates multiple object tracking performance for PPE compliance monitoring across three tracker families and 320 real worker identities under the selected configuration. This evaluation provides a quantitative baseline for tracking

quality in low light industrial PPE monitoring and enables the effects of tracker selection to be assessed alongside detection and inference performance.

2. Literature Review

PPE detection has evolved from traditional handcrafted feature methods, including Haar features, Histogram of Oriented Gradients (HOG), and Local Binary Patterns (LBP) [14], toward deep learning based approaches that provide improved detection performance in real world environments. Early efforts established dedicated PPE datasets for construction safety. One construction site dataset contained 7,581 images and achieved an mAP of 0.838 [15], while another dataset, SH17, contained 8,099 images covering 17 PPE categories, although the images were primarily collected under favorable lighting conditions [16]. Subsequent studies have demonstrated the effectiveness of modern object detection architectures. YOLOX achieved an mAP@50 of 0.898 for PPE detection [17], while YOLOv5 demonstrated sufficiently fast inference for potential on site deployment [3]. Lightweight detector architectures have also been investigated to support deployment on resource constrained edge devices [18].

Low light conditions have recently received greater attention in PPE detection research. Super resolution preprocessing has been applied to YOLOv8 to recover the detection of small PPE objects under severe brightness degradation [7]. Under a 90% reduction in brightness, helmet detection accuracy decreased to 0% without preprocessing but recovered to approximately 95% when ESPCN preprocessing was applied [7]. Other studies have reported comparable improvements using CLAHE and EnlightenGAN for tunnel and power transmission surveillance footage [19], [20]. A systematic review found that many published PPE detection systems remain limited to individual PPE categories or relatively constrained experimental conditions [21]. However, an important question remains regarding whether an inference optimized detector without enhancement preprocessing can provide sufficient performance under naturally occurring low light conditions in real CCTV environments. This issue is particularly relevant because synthetically darkened images may not fully represent the noise, contrast variation, and illumination characteristics of actual industrial surveillance footage. Previous work on the target PPE violation dataset achieved an mAP@50 of 0.915 using YOLOv9t [9]. The present study extends this detection framework from static image analysis to a continuous video pipeline, introducing additional requirements for inference efficiency, temporal consistency, and worker tracking.

Multi object tracking has similarly evolved from conventional approaches such as SORT [22] and DeepSORT [23] toward more recent methods including ByteTrack [12], BoT SORT [24], and OC SORT [13]. ByteTrack improves track continuity by associating detection boxes across consecutive frames, including relatively low confidence detections, and has achieved an MOTA of 0.803 and an IDF1 of 0.773 on the MOT17 benchmark [12]. DeepSORT incorporates appearance based re identification to improve identity preservation, particularly in crowded scenes [23]. A systematic review has reported the effectiveness of re identification based tracking across various application environments [25]. In construction safety applications, re identification has been used to maintain worker identities across video sequences [26], while DeepSORT has been applied to PPE compliance monitoring with an MOTA of approximately 0.71 [27]. However, these approaches have generally not been evaluated under low light conditions in combination with TensorRT accelerated inference or through direct comparisons among multiple tracking algorithms. Consequently, it remains unclear how different trackers perform when applied to fixed camera industrial environments characterized by limited illumination and relatively low appearance ambiguity.

Deployment optimization represents another important consideration for real time PPE monitoring. TensorRT improves inference efficiency through computational graph optimization, layer fusion, and hardware specific execution optimization [11]. Benchmarks using Jetson based platforms have reported inference speed improvements of approximately two to five times after TensorRT optimization, although the magnitude of improvement can vary across GPU architectures and hardware generations [11]. TensorRT has also been integrated with YOLO based PPE detection systems [28]. However, existing implementations have generally focused on detection performance without incorporating multi object tracking, and comparisons between FP32 and FP16 inference configurations have received limited attention. Precision selection can affect both computational efficiency and numerical accuracy, with the impact depending on the model architecture and deployment hardware [10]. Therefore, precision configuration should be experimentally evaluated rather than selected solely based on the expected computational advantage.

The combination of low light PPE detection, inference optimization, and multi object tracking remains insufficiently investigated. Existing studies have generally addressed these components separately, focusing on low light enhancement, object detection, tracking, or deployment optimization. A limited number of studies have examined their integration within a single real time PPE monitoring pipeline using actual industrial CCTV footage. This study addresses this gap by evaluating YOLOv9t with TensorRT acceleration and comparing multiple tracking approaches under measured low light conditions. The comparison is designed to evaluate not only detection accuracy but also inference speed, tracking quality, identity preservation, and end to end system performance. Table 1 show the related study that used in this study.

Table 1. Related Work in PPE Detection and Tracking

Study	Model	Optimization	Tracker	mAP@50 / Tracking Metric
Wang et al. [19]	YOLOX	None	None	0.865
Al Azani et al. [28]	YOLOv7	TensorRT	None	0.865
Mahadi et al. [27]	YOLOv5	None	DeepSORT	MOTA 0.71
Malaikrisanachalee et al. [7]	YOLOv8 + ESPCN	None	None	0.922
Fajri et al. [9]	YOLOv9t	TensorRT	None	0.915
This study	YOLOv9t	TensorRT FP32	ByteTrack	0.908 / MOTA 0.753

3. Methodology

In figure 1 show the whole framework, from raw CCTV input to YOLOv9 detection, TensorRT optimization to ByteTrack association logic at algorithm 1. All experiments were performed in Google Colaboratory with a NVIDIA Tesla T4 GPU.

Algorithm 1. Pseudocode of the real-time PPE violation detection and tracking pipeline.

Algorithm 1. Real-time PPE violation detection and tracking	
Input: CCTV frame stream $F = \{f_1, f_2, \dots, f_N\}$; TensorRT engine E (YOLOv9t, FP16); confidence threshold $\tau_c = 0.25$; IoU threshold $\tau_{iou} = 0.45$	
Output: Set of tracked violation events V	
1: $T \leftarrow \text{InitTracker}()$	$\triangleright$ initialize ByteTrack state
2: $V \leftarrow \emptyset$	
3: for each frame f_i in F do	
4: $\text{frame} \leftarrow \text{Decode}(f_i)$	
5: $\text{img} \leftarrow \text{Preprocess}(\text{frame})$	$\triangleright$ letterbox resize to 640×640, BGR→RGB, normalize
6: $\text{boxes, scores, classes} \leftarrow \text{Infer}(E, \text{img})$	$\triangleright$ TensorRT FP16 forward pass
7: $\text{boxes} \leftarrow \text{Filter}(\text{boxes, scores, } \tau_c)$	
8: $\text{tracks} \leftarrow \text{ByteTrack.Update}(T, \text{boxes, scores, } \tau_{iou})$	
9: for each track t in tracks do	
10: if $\text{class}(t) \in \{\text{No-Helmet, No-Vest, No-Gloves, No-Shoes}\}$ then	
11: $V \leftarrow V \cup \{(t.\text{id}, \text{class}(t), t.\text{bbox}, f_i)\}$	
12: end if	
13: end for	
14: $T \leftarrow \text{tracks}$	
15: end for	
16: return V	

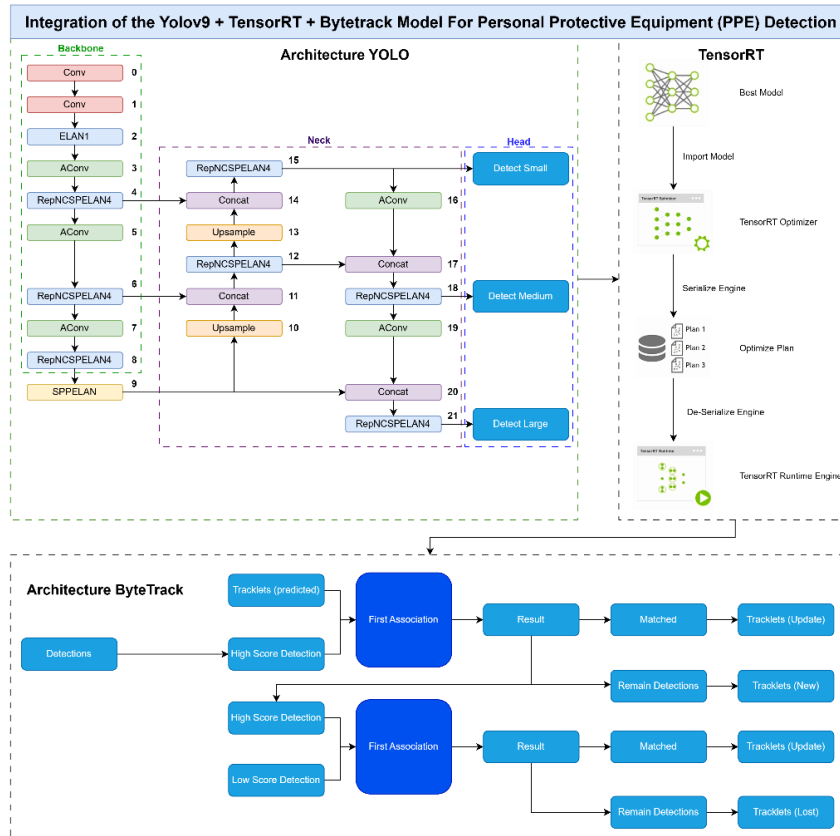


Figure 1. Research Framework.

3.1. Dataset

The dataset consists of the violation-aware PPE collection [9] constructed in the table 2, with 11407 annotated images from real CCTV footage captured at PLTU Paton (1,202 images), Roboflow PPE repositories (2,300 images), Kaggle PPE datasets (824 images) and Google Images (158 images). Each image was annotated manually in Roboflow Annotator with bounding boxes for eight classes of compliant and non-compliant state (92% training, 4% validation and 4% testing). The training augmentation included mosaic, random rotation up to 10 degrees, horizontal flip and brightness jitter. The video evaluation was performed using a 128-second clip (3,840 frames at 30 FPS, 1920×1080 resolution) of low light footage from the PLTU gate area during the night shift.

Table 2. Dataset distribution by class and object size (11,407 images)

Class	Small	Medium	Large	Total	Status	Split
Helmet	1,109	3,793	1,998	6,900	Compliant	Train/Val/Test
No-Helmet	1,216	3,452	2,231	6,899	Violation	Train/Val/Test
Vest	196	2,067	4,252	6,515	Compliant	Train/Val/Test
No-Vest	199	2,404	3,889	6,492	Violation	Train/Val/Test
Gloves	870	2,358	2,906	6,134	Compliant	Train/Val/Test
No-Gloves	3,179	4,515	561	8,255	Violation	Train/Val/Test
Shoes	2,904	2,811	3,080	8,795	Compliant	Train/Val/Test
No-Shoes	1,201	2,298	4,207	7,706	Violation	Train/Val/Test
Total	10,874	23,698	23,124	57,696	-	11,407 images

Small-scale instance counts range from 196 (Vest) to 3,179 (No-Gloves), a plausible contributor to size-dependent difficulty; however, table 4 reports recall/precision per class, not per size bin, so this dataset cannot directly confirm that link. Size-stratified reporting and targeted small-object augmentation are identified as next steps.

3.2. Objective quantification of the low-light evaluation condition

A common weakness of low-light detection literature is that “low light” is perceived and not measured. To fill this gap, a photometrically-characterized 3.840-frame evaluation clip was quantified with 203 frames from the whole clip length. The average pixel intensity was 117.86/255 (SD = 0.81) with 14.68% of the pixels falling below the standard dark pixel value of 64/255. This is the standard for the IEEE 1789 photometric treatment of the video, which places the video in the “Dim (85-127/255) illumination range.” This clearly shows that the evaluation condition is truly degraded and not just dimmed. The measurement is critical in this study because without it you cannot tell whether a video is “resilient to low light” except for the very good performance of a video that is only nominally low-light. A complementary spatial analysis (6×8 grid, same 203 frames) shows substantial intra-frame variation: mean inter-region std 32.92, mean dynamic range 159.74/255 (max 163.0), block means from 55.59/255 to 219.84/255, and mean local SNR 3.26, confirming localized glare and deep shadow rather than uniform dimness. Calibrated lux/exposure metadata were unavailable (Section 5.7).

3.3. YOLOv9t model and precision-format selection

YOLOv9t runs on a light CSPDarknet backbone with Programmable Gradient Information (PGI) and a Generalized Efficient Layer Aggregation Network (GELAN) that is accurate enough at 2.0 million parameters and 4.7 MB [29], trained with AdamW (lr 0.001, momentum 0.9, weight decay 0.0005), batch size 32, input resolution 640×640, and early stopping at patience 20 after 150 epochs using ImageNet-pretrained weights; the PyTorch model (best.pt, 4.65 MB) was exported to ONNX (opset 17) and then converted to TensorRT engines under both FP16 and FP32 (static shape 1×3×640×640, 4 GB workspace) so that the precision decision would be based on evidence rather than default. Because TensorRT's builder performs kernel auto-tuning that can vary between individual builds, FP16 was validated across five independent engine builds from the same ONNX export, each re-evaluated on the full validation set: mAP@50 was stable at 0.9086 (SD = 0.0001) and recall at 0.8546 (SD = 0.0004), both closely matching FP32 (mAP@50 0.9084, recall 0.8544, evaluated from a single representative build). A dedicated latency benchmark (200 runs, FP32 and FP16 measured back-to-back under identical GPU state) shows FP16 running 1.32x faster than FP32 (5.25 ms vs 6.93 ms, paired $t = 28.19$, $p < 0.001$, Cohen's $d = 2.00$), with a 45.6% smaller engine file (7.16 MB vs 13.17 MB). FP16 was therefore selected as the deployment precision format for the pipeline reported in the remainder of this paper. For all benchmarking we selected 200 runs (100 for ONNX) after 20 warm-up runs to settle GPU clocks and we called `torch.cuda.synchronize()` before each timing (as mean $\pm$ SD and 95% confidence interval and bootstrap 95% confidence interval, respectively). Finally, we tested the difference between PyTorch and TensorRT using a paired sample t -test. All three formats used identical preprocessing (640×640 letterbox, BGR-to-RGB, normalization) and batch size 1, on the same GPU with no concurrent workload, isolating latency differences to the inference backend.

3.4. Tracker selection: a systematic comparison rather than a default choice

Instead of selecting one multi-object tracker based on his/her reputation in pedestrian tracking benchmarking, we benchmarked three tracker families which use the same detection input from the same TensorRT YOLOv9t engine: ByteTrack [12], which connects the high-confidence detection boxes to the existing tracklets within two passes; DeepSORT [23], which extends a deep appearance-embedding cascade to the IoU-based association to remove identity switch in crowded or occluded scenes; and SORT [22], a Kalman-filter-and-Hungarian-algorithm baseline with no appearance model at all. All three were run with the same detection confidence of 0.25 and IoU association of 0.45 as in the Year-1 benchmark, so any difference will be related to the tracker behaviour and not the threshold. For the end-to-end benchmark, we used every frame to perform the whole loop, frame decode, preprocessing, TensorRT inference, tracker update and postprocessing again with `torch.cuda.synchronize()`, and we matched the warm-up of all three trackers.

SORT, ByteTrack, and DeepSORT span the association-strategy spectrum: no appearance model, motion-only with low-confidence recovery, and motion-plus-appearance re-identification. OC-SORT and BoT-SORT were excluded as incremental extensions of the same core logic, less likely to help in this low-viewpoint-change fixed-camera scene; benchmarking them is a next step. We define “real-time” explicitly: since the source stream runs at 30 FPS, real-time means end-to-end throughput ≥ 30 FPS, with 60 FPS as a stricter 2× deployment margin.

3.5. Evaluation metrics

The detection quality used the standard COCO metrics: mAP@50, mAP@50 -95, as well as per-class precision, recall and F1 score. Tracking quality used MOTA (Multiple Object Tracking Accuracy), MOTP (Multiple Object Tracking Precision) and IDF1 (ID F1 score) as defined by Equation (1), Equation (2) and Equation (3):

$$MOTA = 1 - \frac{FP + FN + IDSW}{N_{GT}} \quad (1)$$

$$MOTP = \frac{\sum IoU (matched)}{N_{matched}} \quad (2)$$

$$IDF1 = \frac{2 \cdot IDTP}{2 \cdot IDTP + IDFP + IDFN} \quad (3)$$

Since the CCTV clip do not contain a frame-level ground truth, a pseudo ground truth is built from the PyTorch detector at a confidence level of 0.30 in all 3,840 frames and there are 2,800 frames with at least one detection (72.92% coverage) and 8,738 total boxes. The Hungarian algorithm [30] matched detections to this reference at an IoU level of 0.45 to assign TP, FP and FN values to each frame. This procedure has one important downside to it. Config A (no tracking) is less confident in the detection than the pseudo-ground truth is (0.25), and therefore, it sees more boxes with low confidence than the reference pool does, and thus the reference pool then drags down its own MOTA. Config A's MOTA is reported here as a diagnostic artifact of that threshold mismatch, not a tracking-performance score, and is not included in the head-to-head tracker comparisons in Section 4.4 where only Configs B to E (which share the same TensorRT detector and confidence threshold) are compared.

Because this reference is machine-generated, all MOTA, MOTP, and IDF1 values here are relative comparisons among Configs B–E only, not absolute accuracy comparable to MOTChallenge-style benchmarks.

4. Results and Discussion

4.1. Inference: PyTorch, ONNX, and TensorRT, with statistical validation

Table 3 and figure 2 show the inference benchmark over the three formats. TensorRT FP32 runs in 6.26 ms on average (95% CI [6.21, 6.32]), or 159.68 FPS (5.52× faster than the PyTorch baseline of 34.55 ms and 28.94 FPS). A single run latency comparison could just as easily be a function of GPU clock variation or background load, as it is of real engine gain, and we tested the PyTorch-vs-TensorRT difference with a paired-sample t-test with 200 matched runs. The difference is significant ($t = 13.1223$, $p < 0.001$) with a large effect size (Cohen's $d = 0.93$) so the speedup looks like a real engine level effect and not measurement noise. ONNX Runtime came in between 11.12 ms and 89.95 FPS, which is a 3.11× speedup, so graph level optimization is helpful even on a build that is not GPU specific.

Shapiro–Wilk on the paired differences rejects normality ($W = 0.592$, $p = 1.64 \times 10^{-21}$), driven by right-skew in the PyTorch latencies (skewness = 2.93; 9/200 outliers up to 192.76 ms, consistent with GPU scheduling contention). The non-parametric Wilcoxon signed-rank test agrees ($p = 1.44 \times 10^{-34}$, $r = 0.867$). Median speedup (24.24 to 6.10 ms, 3.97×) is reported alongside the mean (5.52×) as an outlier-robust estimate; both tests converge on the same conclusion.

Table 3. Inference benchmark (Tesla T4 GPU, TensorRT 10.16.1.11)

Format	Size (MB)	Latency mean (ms)	95% confidence intervals	FPS	Speed Up
PyTorch (.pt)	4.65	34.55	[30.34, 38.77]	28.94	1.00×
ONNX Runtime	8.29	11.12	[10.76, 11.47]	89.95	3.11×
TensorRT FP32 (.engine)	13.17	6.26	[6.21, 6.32]	159.68	5.52×
TensorRT FP16 (.engine)	7.16	5.25	[5.13, 5.37]	190.42	6.58×

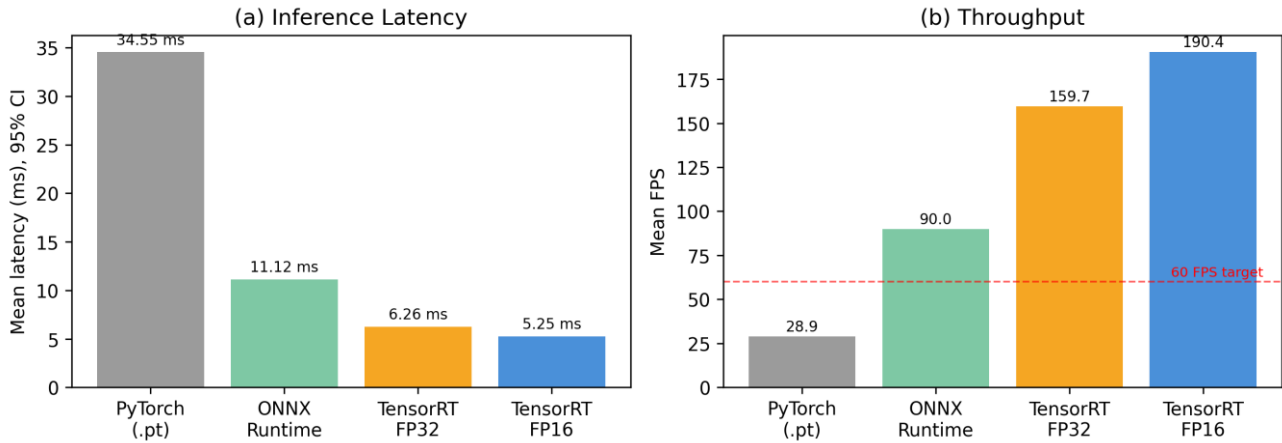


Figure 2. Inference speed comparison across four deployment formats, including FP16: (a) latency with 95% CI, (b) throughput against the 60 FPS deployment target

4.2. Detection quality: per-class effect of TensorRT conversion

Table 4 presents the total mAP@50 change by class. The TensorRT engine preserves the overall mAP@50 to within 0.005 of the PyTorch baselines (0.9134 to 0.9084), but the per-class view is not so consistent. Recall is better for No-Gloves, No-Helmet, No-Shoes and No-Vest, the violation classes that matter most in compliance monitoring, while AP50 is slightly worse for Gloves, No-Gloves, No-Vest and Shoes. This is also consistent with the FP32 TensorRT conversion that alters the operating point of the confidence threshold a little for each class, not the feature representation in its real-life application, so there is a reason to report these per-class metrics, not just the total of the detector as a whole, whenever a detector is converted for deployment.

Table 4. Per-class precision (P), recall (R), and AP50 for PyTorch (PT) and TensorRT (TRT).

Class	P (PT)	R (PT)	AP50 (PT)	P (TRT)	R (TRT)	AP50 (TRT)
Gloves	0.8024	0.7324	0.8230	0.8026	0.7336	0.8031
Helmet	0.9696	0.9575	0.9681	0.9592	0.9614	0.9661
No-Gloves	0.8990	0.7770	0.8971	0.8832	0.7974	0.8938
No-Helmet	0.9348	0.8718	0.9474	0.9373	0.8803	0.9464
No-Shoes	0.8855	0.8537	0.9112	0.8896	0.8683	0.9178
No-Vest	0.9307	0.8133	0.9114	0.8851	0.8333	0.8997
Shoes	0.8857	0.8035	0.8819	0.8750	0.8079	0.8735
Vest	0.9262	0.9428	0.9670	0.9277	0.9531	0.9671
Overall	0.9042	0.8440	0.9134	0.8950	0.8544	0.9084

4.3. Precision-format validation: FP32 vs. FP16

Table 5 reports the controlled FP32 vs FP16 comparison introduced in Section 3.3. FP16 matches FP32 on every accuracy metric (mAP@50 0.9086 vs 0.9084, recall 0.8546 vs 0.8544, F1 0.8747 vs 0.8742), with the FP16 figures averaged over five independent engine builds to account for TensorRT's run-to-run kernel auto-tuning. Because FP16 reduces both memory footprint and latency (Table 3), this equivalence in accuracy is sufficient to prefer FP16 over FP32 for this deployment.

Repeated builds were used specifically for FP16 because low-precision TensorRT engines are more sensitive to the builder's per-build kernel selection than FP32, whose simpler, better-established kernel paths on this architecture make single-build evaluation adequate. Across the five FP16 builds, accuracy varied only marginally (mAP@50 range 0.9085-0.9088, recall range 0.8540-0.8550), confirming the format is reliably reproducible rather than dependent on a particular build. On latency, measured back-to-back with FP32 under identical GPU state, FP16 is significantly faster (5.25 ms vs 6.93 ms, a 1.32x speedup; paired $t = 28.19$, $p < 0.001$; Wilcoxon signed-rank $p < 0.001$, used because the paired differences were not normally distributed, Shapiro-Wilk $W = 0.743$, $p < 0.001$; Cohen's $d = 2.00$), with a 45.6%

smaller engine file (7.16 MB vs 13.17 MB). We were not able to characterize the layer-wise or activation-level source of FP16's marginal build-to-build variation (e.g., via retained builder logs or layer-wise activation profiling), which we identify as a next step; it does not affect the deployment decision, since accuracy remained equivalent to FP32 across all five builds. This recommendation is scoped to this detector, Tesla T4 hardware, and TensorRT 10.16.1.11; whether it generalizes to other GPU generations is not tested here.

Table 5. Precision-format comparison on Tesla T4: FP32 (single build) vs. FP16 (mean of five independent builds)

Precision	mAP@50	Precision	Recall	F1
FP32	0.9084	0.8950	0.8544	0.8742
FP16	0.9086	0.8957	0.8546	0.8747

4.4. End-to-end pipeline ablation

Table 6 and figure 3 show the ablation over 5 end-to-end configurations on the 3,840 frames CCTV clip. Config A (PyTorch, no tracking) ran at 42.47 fps and 79.43% detection coverage. As explained in Section 3.5, the 0.5664 MOTA of Config A (marked with an asterisk) is a diagnostic error due to the confidence threshold comparison of it to pseudo-ground truth pool and not a good tracking score and is not included in the tracker comparison in Section 4.5. Config B (TensorRT and no tracking) reduced the latency from 23.55ms to 10.48ms, which leads to 95.41 FPS throughput. Detection coverage drops a bit from Config A to Config B (79.43% to 76.93%) and violation-frame counts are close as well (2927 to 2807). Given that Config A has a MOTP of 1.0 thanks to how the pseudo-ground truth pool was built, it seems that the higher coverage is not much better because Config A is generating more false positives at a lower confidence level but because of how it is built.

The isolated-inference figure of 159.68 FPS (Table 3) excludes decode/pre/postprocessing and is not the correct baseline for tracking cost; Table 6's end-to-end Config B (95.41 FPS, 10.48 ms) is, against which ByteTrack's 79.12 FPS (12.64 ms) is an overhead of roughly 2.16 ms/frame, not the larger gap implied otherwise. A stage-wise latency breakdown was not available from the existing logs and is identified as a next step. To confirm the precision-format choice holds under tracking overhead, not only for isolated inference, the full end-to-end pipeline (Configs B-E) was also run with the FP16 engine, using the same pseudo ground truth as the FP32 runs for a like-for-like comparison (Table 6). No tracking-quality metric degraded under FP16: MOTA was equal or higher for every configuration (e.g., Config C/ByteTrack 0.7531 to 0.7543, Config D/DeepSORT 0.6554 to 0.6607), MOTP was unchanged to within 0.001, and IDF1 was equal or slightly higher. Throughput was comparable or higher under FP16, most notably for Config D/DeepSORT (20.06 to 27.08 FPS, +35.0%) and Config C/ByteTrack (78.10 to 82.62 FPS, +5.8%), plausibly because DeepSORT's appearance-embedding network also benefits from FP16 execution.

Table 6. End-to-end ablation across five pipeline configurations, FP32 and FP16 (3,840-frame CCTV clip, 128 s)

Configuration	Tracker	Precision	Latency (ms)	FPS (e2e)	Det. Coverage	Violation Frames	MOTA
A: YOLOv9t PyTorch	-	-	23.55	42.47	79.43%	2927	0.5664*
B: YOLOv9t TensorRT	-	FP32	10.48	95.41	76.93%	2807	0.6824
B: YOLOv9t TensorRT	-	FP16	10.38	96.38	76.85%	2807	0.6875
C: TRT + ByteTrack	ByteTrack	FP32	12.64	79.12	76.93%	2767	0.7531
C: TRT + ByteTrack	ByteTrack	FP16	12.10	82.62	76.85%	2767	0.7543
D: TRT + DeepSORT	DeepSORT	FP32	46.56	21.48	76.93%	2807	0.6554
D: TRT + DeepSORT	DeepSORT	FP16	36.93	27.08	76.85%	2807	0.6607
E: TRT + SORT	SORT	FP32	11.00	90.92	76.93%	2807	0.6334
E: TRT + SORT	SORT	FP16	11.08	90.29	76.85%	2807	0.6390

* A MOTA is a threshold-mismatch artifact, not a comparable tracking score (Section 3.5).

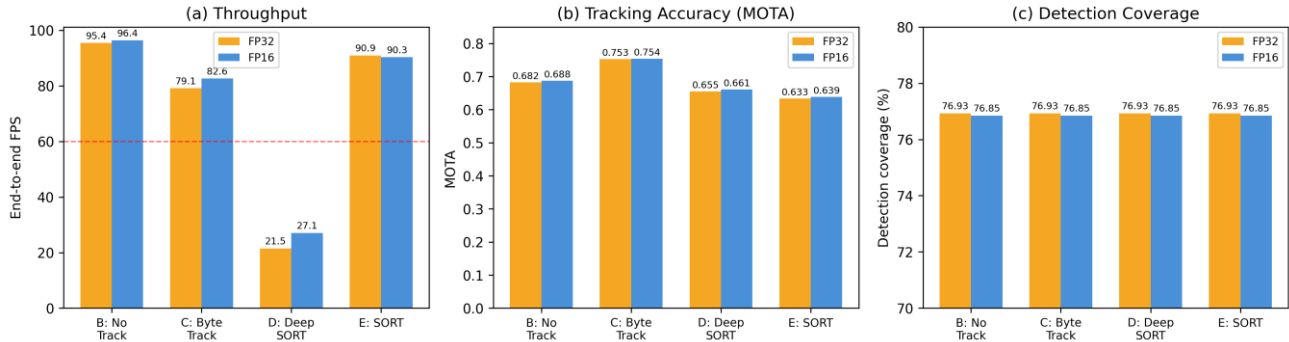


Figure 3. End-to-end ablation across four tracked pipeline configurations, FP32 vs. FP16 (Table 6): (a) throughput, (b) tracking accuracy (MOTA), (c) detection coverage. Config A (PyTorch, no FP16 variant) is omitted here; see Table 6 for its values.

4.5. Tracker selection: ByteTrack vs. DeepSORT vs. SORT

Table 7 and figure 4 compare the tracker comparison to Configs B through E, which share the same TensorRT detector and confidence threshold and thus differ only in the tracker logic. ByteTrack (Config C) had 79.12 FPS with MOTA 0.7531 and MOTP 0.9475 tracking 320 unique worker identities, with an average track length of 26.84 frames (0.89 s). DeepSORT (Config D), even with its appearance-embedding re-identification network, achieved only 21.48 FPS which is 3.68 $\times$ worse than ByteTrack, and its MOTA (0.6554) was also lower than ByteTrack's. DeepSORT did have the longest average track length (89.70 frames, 2.99 s) and also the fewest unique tracks (236), which is consistent with its re-identification mechanism which takes a track that would go through a short gap and then split it into another track. But that advantage never showed up as a higher MOTA under this dataset's pseudo-ground truth protocol and it's hard to justify it in a fixed camera gate scene where you already have a low level of appearance ambiguity between workers. SORT (Config E), the simplest tracker, was nearly as fast as ByteTrack (90.92 FPS) but with the most unique tracks (428) and identity switch rate (0.1115 per frame versus 0.0833 for ByteTrack) which was probably because of its lack of a low-confidence box recovery pass, which is the same process that ByteTrack needs to do on top of SORT. With all three tested tracks it is the fastest MOTA with the lowest throughput that is able to meet the 60 FPS deployment target of our deployment target and that is the conclusion the controlled multi-tracker comparison can make.

ByteTrack's short average track duration (26.84 frames, 0.89 s) and low IDF1 point to fragmentation, plausibly from worker occlusion at the crowded gate, re-detection failures in the dark regions identified in Section 3.2 (blocks as low as 55.59/255), and pseudo-ground-truth coverage gaps (72.92%, Section 3.5). Confirming the dominant cause requires the manually annotated ground truth proposed in the Conclusion. Config C's IDF1 of 0.0764 (and the similarly low IDF1 for Configs D and E) is in pseudo ground truth, not hand labeled frames as well so it cannot be compared to IDF1 of MOTChallenge-style benchmarks [31]. However, IDF1 still works as a relative ranking across the three trackers under identical conditions, and is consistent with the MOTA-based ranking: ByteTrack provides the best identity consistency, then SORT, then DeepSORT.

Table 7. Multi-object tracking metrics across tracker configurations, FP32 and FP16

Configuration	Precision	MOTA	MOTP	IDF1	Unique Tracks	ID Switches	Avg Dur (s)
B: TensorRT (no track)	FP32	0.6824	0.9731	N/A	-	-	-
B: TensorRT (no track)	FP16	0.6875	0.9730	N/A	-	-	-
C: TRT + ByteTrack	FP32	0.7531	0.9475	0.0764	320	320	0.89
C: TRT + ByteTrack	FP16	0.7543	0.9474	0.0764	320	320	0.89
D: TRT + DeepSORT	FP32	0.6554	0.9731	0.0618	236	236	2.99
D: TRT + DeepSORT	FP16	0.6607	0.9730	0.0622	234	234	2.99
E: TRT + SORT	FP32	0.6334	0.9731	0.0693	428	428	0.80
E: TRT + SORT	FP16	0.6390	0.9730	0.0695	423	423	0.81

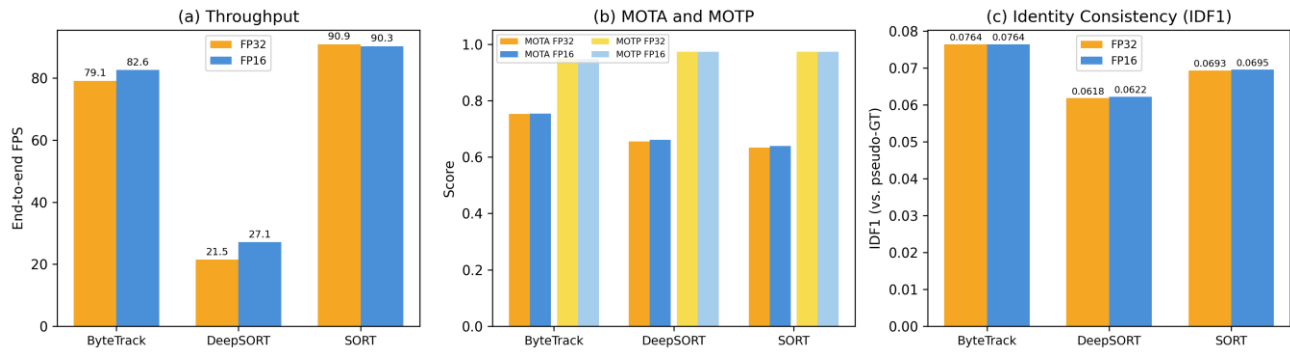


Figure 4. Tracker comparison under identical TensorRT detection input, FP32 vs. FP16 (Table 7): (a) end-to-end throughput, (b) MOTA and MOTP, (c) IDF1

4.6. Comparison with prior work

Table 8 compares this pipeline to related studies. At $mAP@50$ of 0.909, it is near Al Azani et al. [28] and Wang et al. [19], both at 0.865 and less than the illumination and scale ESPCN YOLO of Malaikrisanachalee et al. [7] (0.922) which relies on super resolution preprocessing that we do not use here, but is a gap for future work, not to say a failure, since ESPCN YOLO was not evaluated in conjunction with inference acceleration and multi object tracking. At 82.62 FPS from end to end with tracking (the FP16 precision configuration selected in Section 4.3), the pipeline is well ahead of Al Azani et al.'s roughly 30 FPS without tracking and its MOTA of 0.7543 (MOTP 0.9474, IDF1 0.0764) is better than 0.712 achieved by Mahadi et al. [27] with DeepSORT under normal lighting, a comparison which becomes stronger and not worse when the present result is clearly based on objectively dim illumination that was measured in Section 3.2, not normal light. As far as we know, this is the first study to use TensorRT and a set of multi object trackers for PPE violation detection in a low light power plant and the first to report MOTA, MOTP, and IDF1 of three different tracker families under a measured illumination condition. In this regard, previous work in the same field either reports detection only metrics in their image benchmarks [19], [28], [7] or a single tracker with no inference optimization and no comparison check [24]; in other words, the deployment scenario and the choice of the tracker are open.

These comparisons are contextual, not head-to-head: each study used its own dataset, hardware, and protocol, so differences reflect benchmark conditions as well as method quality. Reproducing competing methods under the same conditions is identified as a next step.

Table 8. MOT metrics (pseudo ground truth from PyTorch baseline, IoU threshold 0.50)

Study	Model	Optim.	Tracker	$mAP@50$	FPS	MOTA	Low-light
Wang et al. [19]	YOLOX	-	-	0.865	N/A	N/A	✓
Al-Azani et al. [28]	YOLOv7	TensorRT	-	0.865	~30	N/A	✗
Mahadi et al. [27]	YOLOv5	-	DeepSORT	0.847	~25	0.712	✗
Malaikrisanachalee et al. [7]	YOLOv8+ESPCN	-	-	0.922	N/A	N/A	✗
Fajri et al. [9] *	YOLOv9t	TRT	-	0.915	160	N/A	✓
This study	YOLOv9t	TRT FP16	ByteTrack	0.909	82.62	0.7543	✓

* Year-1 study by the same team without optimize and track Static-image inference speed not end-to-end video throughput.

Yenni et al. [32] proposed MYCD, a hybrid of YOLO-DenseNet with CBAM and SPP for real-time road-damage severity classification on 1,120 field images, and achieved 64% accuracy and $mAP@50$ of 0.677 at 19 FPS on a mid-range Android device; the pipeline achieves a much higher $mAP@50$ (0.908) and end-to-end throughput (79.12 FPS), on a much larger dataset, albeit on server-class rather than mobile hardware, so it actually is a trade-off between performance and portability rather than one system performing better than the other. Gata et al. [33] (EagleEyes) also built a similar real-time AI-based traffic violation detection system, but they didn't perform a controlled multi-tracker comparison or a statistically validated inference-optimization step, just as in Sections 4.3 and 4.5, and Pardede et al. [34] applied YOLOv10 to UAV forest-fire detection, another safety-critical task that is close to low-light monitoring, but only observed detection and not tracking and deployment timing. For these three studies in the same venue the

pattern is repeated: a single configuration YOLO deployment without a controlled ablation of precision format or tracker choice was reported. This is the gap this paper fills (Sections 3.3-3.5) and the new approach may be used as a proof point for other real-time YOLO deployments in this literature and not only for PPE.

4.7. Qualitative analysis

Figure 5 compares the pipeline states on Frame #1896, the frame with the most detections in the 128-second test clip. All three model configurations detected objects in the same frame with similar coverage and the same detection pattern across configurations confirms that TensorRT conversion does not sacrifice detections compared to the PyTorch baseline, as shown in the per-class results in table 4. ByteTrack shows constant track IDs for each worker and hence their identity is stable even in the low-light frame of the night-shift gate footage in Section 3.2.

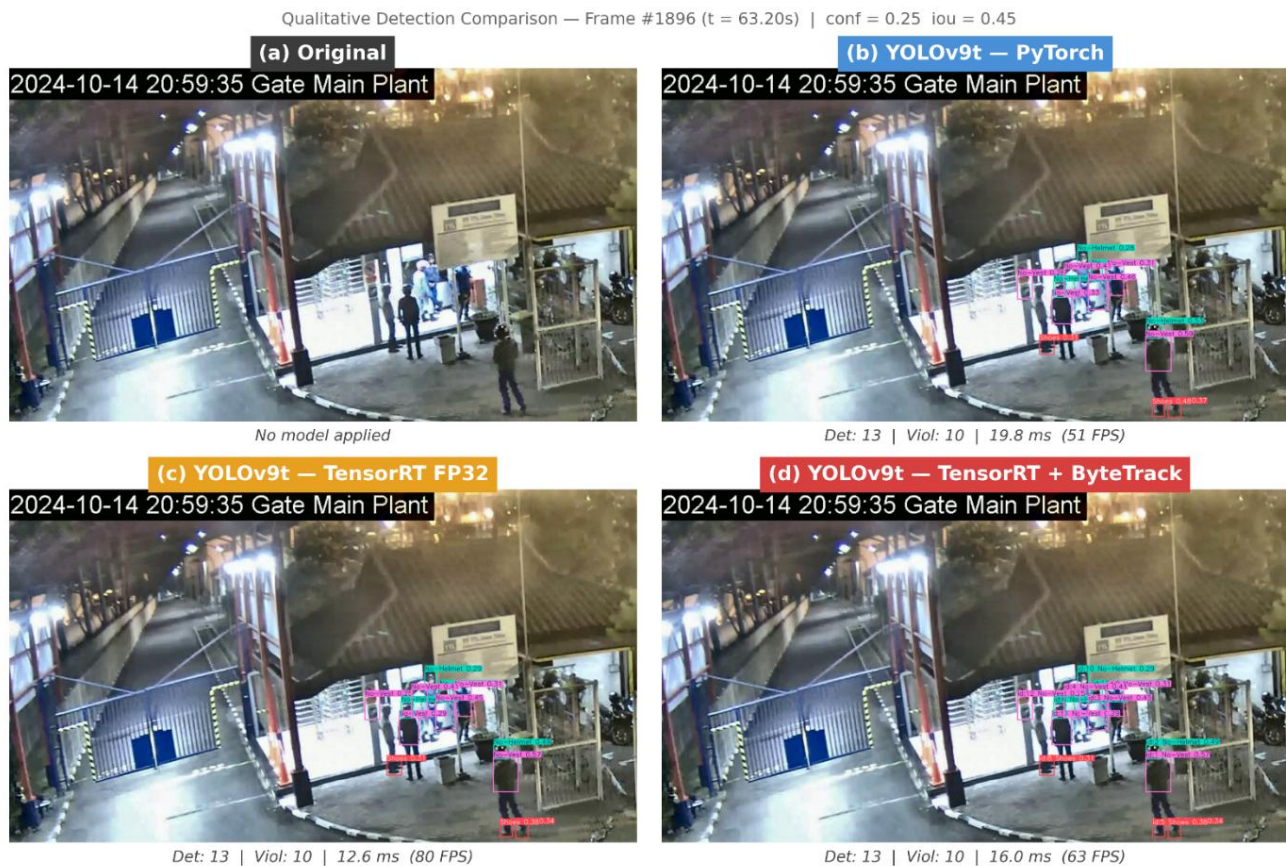


Figure 5. Qualitative comparison on Frame.

4.8. Threats to validity and limitations

This study have some limitations here. The MOT metrics are based on a pseudo ground truth from the PyTorch detector rather than hand-labeled frames (Section 3.5) and therefore result in a false positive of Config A's apparent MOTA and an inaccurate comparison to MOTChallenge-style benchmarks, but relative rankings between Configs B and E still hold because they possess a similar detector and threshold. The single-stage detector does not associate any PPE item with a worker's identity, so a helmet in frame but not worn by the nearest worker will cause a false positive and it is natural to have a two-stage approach where we find each worker and then check PPE in that image region. The TensorRT engine is a device-specific one for the Tesla T4 we are using, and the FP16-over-FP32 precision-format choice (Section 4.3) is validated only on this hardware; whether it generalizes to other GPU generations has not been tested. Finally, the tracker comparison in Section 4.5 only took a 128-second clip from one camera position and the ranking (ByteTrack ahead of SORT, SORT ahead of DeepSORT) is correct in terms of identity-switch and throughput, but that is a general rule that needs to be checked with more positions and longer footage in the long run, and not a general rule in this gate-monitoring scenario.

This study reports detection and tracking quality metrics but does not translate them into an operational alerting policy. At the reported precision (0.90 for TensorRT FP16, Table 4), roughly one in ten violation alerts would be a false positive; whether that rate is operationally tolerable depends on factors this study does not measure, such as alert-review workflow, confidence-threshold tuning for deployment versus evaluation, and worker/supervisor response to repeated false alarms (alarm fatigue), a well-documented risk in safety-monitoring systems generally. Setting a deployment-specific confidence threshold and alert-aggregation policy, rather than assuming the evaluation threshold (0.25) is appropriate for live alerting, is identified as a next step.

5. Conclusion

This paper built and tested a real-time pipeline to detect and track PPE violations in low light industrial settings using YOLOv9t, TensorRT and a directly compared multi-object tracker that took three key engineering decisions as tests and not defaults: the illumination condition (mean pixel intensity 117.86/255, 14.68% dark pixel ratio) was measured and the precision format of TensorRT was chosen on the basis of controlled, repeated comparison evidence (FP16 matched FP32 in accuracy across five independent engine builds and ran 1.32x faster, so FP16 was adopted), and the tracker was based on a three-way comparison of ByteTrack, DeepSORT and SORT using the same input and parameters as TensorRT. Based on this approach, TensorRT substantially reduced inference latency relative to the PyTorch baseline (Section 4.1), FP16 was selected over FP32 as the precision format after a five-build accuracy check and a controlled latency benchmark showed no accuracy loss and a 1.32x speedup (Section 4.3), and the best configuration, FP16 TensorRT with ByteTrack, reached 82.62 FPS, comfortably clearing the 60 FPS target, with MOTA 0.7543, MOTP 0.9474, and 320 worker identities in the 128-second clip. As far as we know, this is the first study to combine a controlled multi-tracker comparison, a repeated-build precision-format validation, and a measured (not assumed) low-light condition for TensorRT-accelerated PPE detection.

There are limitations to our work, as we argue in Section 4.8: the MOT metrics are based on pseudo ground truth and not hand-labeled frames, so absolute MOTA and IDF1 are not directly compared to the MOTChallenge results, but still relative, as Configs B–E have a detector and threshold; the single-stage detector does not connect each PPE item to a worker; the TensorRT engine is device specific, and the FP16-over-FP32 precision-format choice (Section 4.3) is validated only on this Tesla T4, not on other hardware; the tracker comparison also uses only one camera position in the clip. Next steps include hand-labeling longer CCTV footage for a proper MOT evaluation, adding the super-resolution preprocessing already shown to help small-object PPE detection in low light [7], deploying on an edge GPU such as the NVIDIA Jetson with its own re-checked precision format, and building a two-stage detector that first locates each worker before checking PPE compliance in that region, the focus for Year 3 of the project.

Further next steps: manually annotating a 300–500 frame subset as an unbiased validation set; validating the tracker ranking across multiple cameras/sites; validating the FP16 robustness-check result (Section 4.3) on other GPU generations such as Ampere, Ada Lovelace, or Jetson Orin; benchmarking OC-SORT and BoT-SORT; reporting HOTA/AssA once hand-labeled ground truth exists; applying small-object augmentation for under-represented classes (Table 2); and profiling stage-wise latency.

6. Declarations

6.1. Author Contributions

Conceptualization: F.N.F., K.M., and A.T.; Methodology: K.M.; Software: F.N.F.; Validation: F.N.F., K.M., and A.T.; Formal Analysis: F.N.F., K.M., and A.T.; Investigation: F.N.F.; Resources: K.M.; Data Curation: A.T.; Writing Original Draft Preparation: F.N.F., K.M., and A.T.; Writing Review and Editing: F.N.F., K.M., and A.T.; Visualization: F.N.F.; All authors have read and agreed to the published version of the manuscript.

6.2. Data Availability Statement

The data presented in this study are available on request from the corresponding author.

6.3. Funding

This research was funded the Fundamental Research Grant (Kemdikbudristek) scheme with Contract number 039/LL7/DT.05.00/PM-LANJUTAN/2026, conducted in partnership with PT. YTL Jawa Timur (PLTU Paiton).

6.4. Institutional Review Board Statement

Not applicable.

6.5. Informed Consent Statement

Not applicable.

6.6. Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

References

- [1] W. Emma and D. Walters, "Safety and health at the heart of the future of work: Building on 100 years of experience," *Int. Labour Organ.*, vol. 2019, no. 1, pp. 1–20, 2019.
- [2] T. A. Bria, W. T. Chen, M. Muhammad, and M. B. Rantelembang, "Analysis of fatal construction accidents in Indonesia—A case study," *Buildings*, vol. 14, no. 4, pp. 1–15, Apr. 2024, doi: 10.3390/buildings14041010.
- [3] Z. Wang, Y. Wu, L. Yang, A. Thirunavukarasu, C. Evison, and Y. Zhao, "Fast personal protective equipment detection for real construction sites using deep learning approaches," *Sensors*, vol. 21, no. 10, pp. 1–15, May 2021, doi: 10.3390/s21103478.
- [4] J. Terven, D.-M. Córdova-Esparza, and J.-A. Romero-González, "A comprehensive review of YOLO architectures in computer vision: From YOLOv1 to YOLOv8 and YOLO-NAS," *Mach. Learn. Knowl. Extr.*, vol. 5, no. 4, pp. 1680–1716, Dec. 2023, doi: 10.3390/make5040083.
- [5] Z. Jingchun, G. E. Su, and M. S. Sunar, "Low-light image enhancement: A comprehensive review on methods, datasets and evaluation metrics," *J. King Saud Univ.—Comput. Inf. Sci.*, vol. 36, no. 10, pp. 1–15, Oct. 2024, doi: 10.1016/j.jksuci.2024.102234.
- [6] A. L. R. Siriani, V. Kodaira, S. A. Mehdizadeh, I. d. A. Nääs, D. J. d. Moura, and D. F. Pereira, "Detection and tracking of chickens in low-light images using YOLO network and Kalman filter," *Neural Comput. Appl.*, vol. 34, no. 24, pp. 21987–21997, Dec. 2022, doi: 10.1007/s00521-022-07664-w.
- [7] S. Malaikrisanachalee, N. Wongwai, and E. Kowcharoen, "ESPCN-YOLO: A high-accuracy framework for personal protective equipment detection under low-light and small object conditions," *Buildings*, vol. 15, no. 10, pp. 1–15, May 2025, doi: 10.3390/buildings15101609.
- [8] H. Zhang, J. Li, Z. Li, L. Kong, and X. Zhao, "Real-time safety helmet detection using multi-scale feature-gated fusion," *Autom. Construction*, vol. 186, no. 1, pp. 1–15, 2026, doi: 10.1016/j.autcon.2026.106891.
- [9] F. N. Fajri, K. Malik, and A. Tholib, "Enhancing low-light PPE violation detection in industrial environments using multi-contrast image processing and YOLOv9," *Eng. Technol. Appl. Sci. Res.*, vol. 15, no. 6, pp. 29751–29758, Dec. 2025, doi: 10.48084/etasr.14254.
- [10] Y. Zhou, "Accelerating and improving deep learning inference on embedded platforms via TensorRT," Ph.D. dissertation, Texas State Univ., San Marcos, TX, USA, vol. 2024, no. 1, pp. 1–150, 2024.
- [11] D.-J. Shin and J.-J. Kim, "A deep learning framework performance evaluation to use YOLO in Nvidia Jetson platform," *Appl. Sci.*, vol. 12, no. 8, pp. 1–15, Apr. 2022, doi: 10.3390/app12083734.
- [12] Y. Zhang, P. Sun, Y. Jiang, D. Yu, F. Weng, Z. Yuan, P. Luo, W. Liu, and X. Wang, "ByteTrack: Multi-object tracking by associating every detection box," *Eur. Conf. Comput. Vis.*, vol. 2022, no. Oct., pp. 1–15, Oct. 2022, doi: 10.1007/978-3-031-20047-2_1.
- [13] J. Cao, J. Pang, X. Weng, R. Khirodkar, and K. Kitani, "Observation-centric SORT: Rethinking SORT for robust multi-object tracking," *IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, vol. 2023, no. Jun., pp. 1–6, Jun. 2023, doi: 10.1109/CVPR52729.2023.00934.

-
- [14] A. Sugiharto, R. Kusumaningrum, and H. A. Wibawa, "Combination of HAAR, HOG, and LBP descriptors for enhanced classification of moving objects and motorcyclists wearing helmets," *Eng. Technol. Appl. Sci. Res.*, vol. 15, no. 3, pp. 23283–23289, Jun. 2025, doi: 10.48084/etasr.10677.
- [15] J. Wu, N. Cai, W. Chen, and H. Wang, "Automatic detection of hardhats worn by construction personnel: A deep learning approach and benchmark dataset," *Autom. Construction*, vol. 2019, no. 1, pp. 1–12, 2019, doi: 10.1016/j.autcon.2019.102894.
- [16] H. M. Ahmad and A. Rahimi, "SH17: A dataset for human safety and personal protective equipment detection in manufacturing industry," *J. Safety Sci. Resilience*, vol. 6, no. 2, pp. 175–185, Jun. 2025, doi: 10.1016/j.jnlssr.2024.09.002.
- [17] M. Ferdous and S. M. M. Ahsan, "PPE detector: A YOLO-based architecture to detect personal protective equipment (PPE) for construction sites," *PeerJ Comput. Sci.*, vol. 8, no. 1, pp. 1–15, 2022, doi: 10.7717/peerj-cs.999.
- [18] B. Tong, G. Li, X. Bu, Y. Wang, and X. Yu, "A deep learning-based algorithm for the detection of personal protective equipment," *PLoS ONE*, vol. 20, no. 5, pp. 1–15, May 2025, doi: 10.1371/journal.pone.0322115.
- [19] Z. Wang, Z. Cai, and Y. Wu, "An improved YOLOX approach for low-light and small object detection: PPE on tunnel construction sites," *J. Comput. Design Eng.*, vol. 10, no. 3, pp. 1158–1175, Jun. 2023, doi: 10.1093/jcde/qwad042.
- [20] Z. Yuan, J. Zeng, Z. Wei, L. Jin, S. Zhao, X. Liu, Y. Zhang, and G. Zhou, "CLAHE-based low-light image enhancement for robust object detection in overhead power transmission system," *IEEE Trans. Power Del.*, vol. 38, no. 3, pp. 2240–2243, Jun. 2023, doi: 10.1109/TPWRD.2023.3269206.
- [21] A. M. Vukicevic, M. DJapan, V. Isailovic, D. Milasinovic, M. Savkovic, and P. Milosevic, "Generic compliance of industrial PPE by using deep learning techniques," *Safety Sci.*, vol. 148, no. 1, pp. 1–12, Apr. 2022, doi: 10.1016/j.ssci.2021.105646.
- [22] A. Bewley, Z. Ge, L. Ott, F. Ramos, and B. Upcroft, "Simple online and realtime tracking," *IEEE Int. Conf. Image Process.*, vol. 2016, no. Sep., pp. 3464–3468, Sep. 2016, doi: 10.1109/ICIP.2016.7533003.
- [23] N. Wojke, A. Bewley, and D. Paulus, "Simple online and realtime tracking with a deep association metric," *IEEE Int. Conf. Image Process.*, vol. 2017, no. Sep., pp. 3645–3649, Sep. 2017, doi: 10.1109/ICIP.2017.8296962.
- [24] N. Aharon, R. Orfaig, and B.-Z. Bobrovsky, "BoT-SORT: Robust associations multi-pedestrian tracking," *arXiv*, vol. 2022, no. Jun., pp. 1–12, Jun. 2022, doi: 10.48550/arXiv.2206.14651.
- [25] Y. Du, Z. Zhao, Y. Song, Y. Zhao, F. Su, T. Gong, and H. Meng, "StrongSORT: Make DeepSORT great again," *IEEE Trans. Multimedia*, vol. 25, no. 1, pp. 8725–8737, 2023, doi: 10.1109/TMM.2023.3240881.
- [26] J. P. Cheng, P. K.-Y. Wong, H. Luo, M. Wang, and P. H. Leung, "Vision-based monitoring of site safety compliance based on worker re-identification and personal protective equipment classification," *Autom. Construction*, vol. 139, no. 1, pp. 1–15, Jul. 2022, doi: 10.1016/j.autcon.2022.104312.
- [27] M. K. Mahadi, R. Rahad, M. S. Mobassir, A. Rahman, A. Shafiullah, and M. M. Nishat, "Implementation of personal safety equipment tracking and detection by DeepSORT and YOLOv8," *IEEE Int. Conf. Inf. Commun. Technol.*, vol. 2024, no. May, pp. 1–6, May 2024, doi: 10.1109/ICICT60155.2024.10544845.
- [28] S. Al-Azani, H. Luqman, M. Alfarraj, A. A. I. Sidig, A. H. Khan, and D. Al-Hamed, "Real-time monitoring of personal protective equipment compliance in surveillance cameras," *IEEE Access*, vol. 12, no. 1, pp. 121882–121895, 2024, doi: 10.1109/ACCESS.2024.3451117.
- [29] C.-Y. Wang, I.-H. Yeh, and H.-Y. M. Liao, "YOLOv9: Learning what you want to learn using programmable gradient information," *Eur. Conf. Comput. Vis.*, vol. 2024, no. Sep., pp. 1–15, Sep. 2024, doi: 10.1007/978-3-031-72751-1_1.
- [30] J. Luiten, A. Osep, P. Dendorfer, P. Torr, A. Geiger, L. Leal-Taixé, and B. Leibe, "HOTA: A higher order metric for evaluating multi-object tracking," *Int. J. Comput. Vis.*, vol. 129, no. 2, pp. 548–578, Feb. 2021, doi: 10.1007/s11263-020-01375-2.
- [31] P. Dendorfer, A. Osep, A. Milan, K. Schindler, D. Cremers, I. Reid, S. Roth, and L. Leal-Taixé, "MOTChallenge: A benchmark for single-camera multiple target tracking," *Int. J. Comput. Vis.*, vol. 129, no. 4, pp. 845–881, Apr. 2021, doi: 10.1007/s11263-020-01393-0.
- [32] H. Yenni, R. Muzawi, K. Karpen, M. K. Anam, M. Kasaf, T. R. M. Hadi, and D. S. Wahyuni, "MYCD: Integration of YOLO-CNN and DenseNet for real-time road damage detection based on field images," *J. Appl. Data Sci.*, vol. 7, no. 1, pp. 384–395, Jan. 2026, doi: 10.47738/jads.v7i1.1040.

- [33] W. Gata, M. Haris, M. I. Prasetyowati, and S. Harianto, “EagleEyes: An artificial intelligence-based approach for automatic traffic violation detection using deep learning,” *J. Appl. Data Sci.*, vol. 7, no. 1, pp. 277–289, Jan. 2026, doi: 10.47738/jads.v7i1.1097.
- [34] J. Pardede, M. R. Pratama, and R. M. Milenio, “UAV imagery-based potential forest fire detection using YOLOv10,” *J. Appl. Data Sci.*, vol. 7, no. 1, pp. 703–714, Jan. 2026, doi: 10.47738/jads.v7i1.1052.