Object Detection, Tracking and Analysis in Video: A Comparative Study of CNN and YOLOv8

Hoang Ha Nguyen, Thi Lan Nguyen, Thanh Dat Nguyen, Duc Manh Phung, Hoang Gia Minh Pham

Faculty of Information Technology, Thanh Do University, Hanoi, Vietnam

Advisor: Le Duc Huy

Abstract

Object detection, tracking and analysis in video is a fundamental research domain in artificial intelligence and computer vision. This paper presents a comprehensive comparative study of two prominent deep learning approaches for real-time object detection in surveillance video: a two-stage Convolutional Neural Network (CNN) pipeline based on Faster R-CNN with a ResNet-101 v1 backbone, and the single-stage YOLOv8 detector employing a CSPDarknet backbone with multi-scale prediction heads. We provide a detailed analysis of the architectural differences, computational complexity, and theoretical foundations of both approaches. Both models were evaluated on a custom dataset of classroom surveillance videos recorded at Thanh Do University, Hanoi, under varying illumination conditions. We report results using multiple evaluation metrics including Accuracy, Precision, Recall, F1-Score, and inference latency. Experimental results demonstrate that YOLOv8 achieves a detection accuracy of 85.71% compared to 71.43% for the CNN-based approach, with approximately 7x faster inference speed. YOLOv8 also exhibits superior robustness under low-light conditions. We conduct an ablation study on the impact of data augmentation strategies and discuss the trade-offs between model complexity, accuracy, and deployment requirements. Future directions including Vision Transformer architectures, hybrid detection pipelines, and edge-device optimization are proposed.

Keywords: Object Detection, Video Analysis, Convolutional Neural Network, YOLOv8, Faster R-CNN, Real-time Surveillance, Deep Learning, Computer Vision, Multi-scale Detection, Transfer Learning

I. INTRODUCTION

Object detection, tracking and analysis in video is one of the most actively researched areas in computer vision and artificial intelligence. It involves the automated identification and localization of objects of interest—such as humans, vehicles, and animals—across sequential video frames. The inherent challenges include object motion and deformation, inter-object occlusion, scale variation, and dynamic changes in illumination and background.

The practical significance of video-based object detection spans numerous application domains. In security surveillance, it enables automated monitoring of restricted areas and detection of suspicious behavior. In intelligent transportation systems, it supports vehicle counting, traffic flow estimation, and violation detection. In agriculture, it facilitates crop disease identification and yield estimation through aerial drone footage. The proliferation of surveillance cameras worldwide has created an urgent demand for efficient and accurate automated detection systems.

Historically, object detection relied heavily on hand-crafted feature extractors such as Haar cascades [9], Histogram of Oriented Gradients (HOG) [10], and Deformable Parts Models (DPM) [11]. While these methods established foundational principles, they suffered from limited representational capacity and poor generalization across diverse environments. The advent of deep learning, particularly Convolutional Neural Networks, fundamentally transformed the landscape beginning with AlexNet in 2012 [12] and subsequently with dedicated detection architectures.

Modern object detection methods are broadly categorized into two paradigms: (1) two-stage detectors such as R-CNN [4], Fast R-CNN, and Faster R-CNN [5] that first generate region proposals and then classify each candidate region; and (2) single-stage detectors such as SSD [6] and the YOLO family [1] that perform localization and classification simultaneously in a single forward pass. Each paradigm presents distinct trade-offs between accuracy and computational efficiency.

This paper presents a systematic comparative evaluation of these two paradigms using Faster R-CNN (ResNet-101 backbone) as a representative two-stage detector and YOLOv8 as a state-of-the-art single-stage detector. Both models are evaluated on a custom surveillance video dataset collected from classroom environments under varying lighting conditions. Our contributions include: (1) a detailed architectural and computational analysis of both approaches; (2) multi-metric evaluation under controlled lighting variations; (3) an ablation study on augmentation strategies; and (4) practical deployment guidelines for surveillance applications.

II. RELATED WORK

A. Evolution of Object Detection

The development of object detection algorithms can be traced through three major eras. The **traditional era** (pre-2012) was dominated by hand-crafted features: Viola-Jones detectors using Haar-like features for face detection, HOG descriptors combined with linear SVMs for pedestrian detection [10], and DPM models using mixtures of multi-scale deformable parts [11]. These methods required significant domain expertise for feature engineering and showed limited performance on complex, multi-class detection tasks.

The **two-stage detector era** (2014-2017) began with R-CNN [4], which introduced the concept of using CNNs for region-based feature extraction. R-CNN applied selective search to generate approximately 2,000 region proposals per image, extracted CNN features from each, and classified them using SVMs. This approach achieved dramatic improvements over traditional methods but was computationally prohibitive, requiring 47 seconds per image on a GPU. Fast R-CNN improved efficiency by sharing convolutional features across proposals, while Faster R-CNN [5] introduced the Region Proposal Network (RPN) to generate proposals directly from feature maps, reducing inference time to approximately 0.2 seconds per image.

The **single-stage detector era** (2016-present) was initiated by YOLO [1] and SSD [6]. YOLO reformulated detection as a single regression problem, dividing the image into a grid and predicting bounding boxes and class probabilities simultaneously. SSD introduced multi-scale feature maps for detecting objects at different sizes. Subsequent YOLO versions (v2-v8) progressively improved accuracy through architectural innovations including batch normalization, anchor-free detection, and advanced augmentation techniques.

B. Literature Survey

Table I summarizes key object detection methods and their reported performance on standard benchmarks. The evolution from R-CNN to YOLOv8 demonstrates a clear trend toward higher accuracy with lower computational cost.

TABLE I. Comparative survey of object detection methods. mAP is reported on COCO val2017 unless noted. *YOLOv1 mAP on VOC 2007.

Method	Year	Type	Backbone	mAP (COCO)	FPS
R-CNN [4]	2014	Two-stage	AlexNet	31.4	0.02
Fast R-CNN	2015	Two-stage	VGG-16	35.9	0.5
Faster R-CNN [5]	2015	Two-stage	ResNet-101	42.1	5
SSD [6]	2016	Single	VGG-16	46.5	22
YOLOv1 [1]	2016	Single	Darknet	63.4*	45
YOLOv3	2018	Single	Darknet-53	55.3	30
RetinaNet [3]	2017	Single	ResNet-101	53.1	11
EfficientDet	2020	Single	EfficientNet	55.1	25
YOLOv5	2020	Single	CSPDarknet	56.8	140
YOLOv8 [7]	2023	Single	CSPDarknet	53.9	280

As shown in Table I, single-stage detectors have progressively closed the accuracy gap with two-stage methods while maintaining significantly higher inference speeds. YOLOv8 achieves 280 FPS with competitive mAP, making it particularly suitable for real-time surveillance applications.

C. Object Tracking Algorithms

Object tracking extends detection by maintaining identity associations across frames. Classical approaches include: (1) the **Kalman Filter**, which uses a linear state-space model to predict object positions based on prior observations and a constant-velocity motion model; (2) the **Particle Filter**, which employs Monte Carlo sampling to handle non-linear and non-Gaussian state estimation; (3) **MeanShift**, which iteratively shifts a kernel toward the mode of a color histogram distribution; and (4) **CAMShift** (Continuously Adaptive MeanShift), which extends MeanShift by dynamically adapting the search window size.

Modern deep learning-based trackers such as SORT (Simple Online and Realtime Tracking) and DeepSORT combine detection outputs with Kalman filtering and appearance feature matching using deep Re-ID networks, achieving state-of-the-art performance on MOT benchmarks. ByteTrack further improves tracking by associating every detection box including low-confidence ones, reducing identity switches in crowded scenes.

D. Domestic Research Context

Within Vietnam, research on video-based object detection has gained momentum in recent years. Phan Minh Chau (2020) at Hanoi University of Science and Technology investigated Faster R-CNN for image-based object detection, demonstrating the feasibility of CNN-based approaches for Vietnamese surveillance scenarios. Dang Viet Ha and Nguyen Quoc Huy (2019) conducted a comparative benchmark of detection algorithms in the Hanoi Journal of Science and Technology. Nguyen Trung Kien (2022) at the University of Information Technology explored YOLOv5 for tracking moving objects in video, achieving promising results on urban traffic datasets.

III. THEORETICAL BACKGROUND

A. Convolutional Neural Networks

A Convolutional Neural Network is a class of deep feed-forward neural networks specifically designed for grid-structured data. The fundamental operation is the discrete convolution of an input feature map with a learnable kernel:

$$(f * g)[n] = \Sigma_m f[m] \cdot g[n - m]$$

In the two-dimensional case applied to images, given an input tensor X of dimension H x W x C and a set of K filters W_k of size k_h x k_w x C, the output feature map Y is computed as:

$$Y[i, j, k] = \Sigma_c \Sigma_p \Sigma_q X[i+p, j+q, c] \cdot W_k[p, q, c] + b_k$$

where b_k is the bias term for the k-th filter. The stride parameter s controls the step size of the convolution window, and zero-padding of size p is applied to control the spatial dimensions of the output.

1) Core Components

Convolutional Layers: These are the fundamental building blocks that apply learnable filters across the spatial extent of the input volume. Key hyperparameters include kernel size (typically 3x3 or 1x1), number of output channels, stride, and padding. Modern architectures employ depth-wise separable convolutions to reduce computational cost by factoring standard convolutions into depth-wise and point-wise operations.

Pooling Layers: These reduce the spatial resolution of feature maps through downsampling operations. Max pooling selects the maximum value within each pooling window, preserving the strongest activations. Average pooling computes the mean, providing smoother feature representations. Global Average Pooling (GAP) reduces each feature map to a single scalar, commonly used before the classification head.

Activation Functions: Non-linear activation functions introduce representational capacity. ReLU ($f(x) = \max(0, x)$) is the most widely used due to its computational simplicity and mitigation of vanishing gradients. Variants include Leaky ReLU ($f(x) = \max(ax, x)$ for small a), Swish ($f(x) = x \cdot \text{sigmoid}(x)$), and Mish ($f(x) = x \cdot \tanh(\text{softplus}(x))$), the latter being used in YOLOv8.

Batch Normalization: Normalizes layer inputs to zero mean and unit variance within each mini-batch, stabilizing training dynamics and enabling higher learning rates. The normalized output is: $y = \text{gamma} \cdot (x - \text{mu}) / \text{sqrt}(\text{sigma}^2 + \text{epsilon}) + \text{beta}$, where gamma and beta are learnable scale and shift parameters.

Residual Connections: Introduced in ResNet [8], skip connections enable training of very deep networks by allowing gradients to flow directly through shortcut paths: y = F(x, W) + x. This architecture underpins the ResNet-101 backbone used in our Faster R-CNN implementation.

B. Faster R-CNN Architecture

Faster R-CNN [5] consists of three main components operating in sequence:

Backbone Network: A deep CNN (ResNet-101 in our implementation) extracts a shared convolutional feature map from the input image. The backbone is typically pre-trained on ImageNet for transfer learning. ResNet-101 contains 101 layers organized into 4 residual stages, producing feature maps at 1/4, 1/8, 1/16, and 1/32 of the input resolution.

Region Proposal Network (RPN): A small fully-convolutional network slides over the backbone feature map and predicts objectness scores and bounding box refinements for a set of anchor boxes at each spatial position. For each of k anchor boxes (typically k=9, using 3 scales and 3 aspect ratios), the RPN outputs 2k objectness scores and 4k bounding box regression coefficients. Non-Maximum Suppression (NMS) with IoU threshold 0.7 filters redundant proposals.

Detection Head: RoI Pooling extracts fixed-size feature vectors from each proposal region, which are then passed through fully-connected layers for final classification into C+1 classes (C object classes plus background) and bounding box regression refinement. The loss function combines classification cross-entropy and smooth L1 regression loss:

$$L = L_{cls}(p, u) + lambda \cdot [u \geq 1] \cdot L_{reg}(t, v)$$

where p is the predicted class probability, u is the ground-truth class label, t and v are the predicted and target bounding box parameters, and lambda balances the two losses.

C. YOLOv8 Architecture

YOLOv8 [7] is a single-stage, anchor-free object detector representing the latest evolution of the YOLO family. Unlike two-stage detectors, YOLOv8 performs detection in a single forward pass, achieving real-time inference speeds. The architecture comprises three primary components:

CSPDarknet Backbone: The backbone employs Cross Stage Partial (CSP) connections [2] integrated into a Darknet architecture. CSP splits the input feature map into two paths: one passes through a dense block of convolutional layers while the other takes a direct shortcut, before being concatenated. This design reduces computational redundancy while maintaining gradient flow. The backbone extracts features at three scales (P3, P4, P5) corresponding to strides of 8, 16, and 32 pixels.

CSP-PAN Neck: A Path Aggregation Network (PAN) with CSP connections fuses multi-scale features through both top-down and bottom-up pathways. The top-down pathway propagates semantic information from deep layers to shallow layers, while the bottom-up pathway propagates localization information in the reverse direction. This bidirectional fusion ensures that detection heads at all scales receive both fine-grained spatial details and high-level semantic context.

Decoupled Detection Head: YOLOv8 adopts a decoupled head design that separates classification and localization into independent branches. Each branch processes features through dedicated convolutional layers before producing outputs. The classification branch predicts C class probabilities using sigmoid activation, while the regression branch predicts bounding box parameters. This decoupling improves convergence and accuracy compared to the coupled heads used in earlier YOLO versions.

Loss Function: YOLOv8 uses a composite loss function:

$$L_{total} = L_{box} + L_{cls} + L_{dfl}$$

where L_{box} is CIoU (Complete Intersection over Union) loss for bounding box regression, L_{cls} is binary cross-entropy with logits for classification, and L_{dfl} is Distribution Focal Loss for predicting the distribution of bounding box offsets rather than single point estimates.

D. Evaluation Metrics

We employ the following standard metrics for detection evaluation:

Precision measures the fraction of detected objects that are correct: P = TP / (TP + FP). **Recall** measures the fraction of ground-truth objects that are detected: R = TP / (TP + FN). **Accuracy** measures overall correctness: A = (TP + TN) / (TP + TN + FP + FN). In single-class detection, we use the simplified form A = TP / (TP + FN).

F1-Score is the harmonic mean of Precision and Recall:

$$F1 = 2 \cdot (P \cdot R) / (P + R)$$

Intersection over Union (IoU) quantifies the overlap between a predicted bounding box B_p and ground-truth box B_{gt}:

$$IoU = |B_p \cap B_{gt}| / |B_p \cup B_{gt}|$$

A detection is typically considered a True Positive when IoU exceeds a threshold (commonly 0.5, denoted AP@50). **Mean Average Precision (mAP)** averages the precision-recall area across all object classes and IoU thresholds.

IV. EXPERIMENTAL SETUP

A. Dataset Description

The experimental dataset consists of 5 surveillance videos recorded in classrooms and lecture halls within Building B of Thanh Do University, Hanoi. Videos were captured using a stationary iPhone 11 Pro Max camera at 1080p resolution and 30 FPS. Each video is 4-5 minutes in duration, yielding approximately 7,200-9,000 frames per video and a total of approximately 40,000 frames across the entire dataset.

The video content encompasses three distinct scenarios: (1) students entering the classroom at the beginning of a session; (2) students packing up and preparing to leave at the end of class; and (3) students exiting the room, rearranging chairs, and turning off lights. Primary objects of interest include students, instructors, and classroom furniture (desks, chairs, computers, projector screen).

For pre-training and as an augmentation source, the COCO (Common Objects in Context) dataset was utilized. COCO contains 330K images with 1.5 million object instances across 80 categories, providing a rich foundation for transfer learning. The custom dataset was annotated using the CVAT.ai platform with per-frame bounding box annotations for the "person" class.

Fig. 1. Sample frame (Frame 67) from the surveillance dataset showing students during class in a well-lit environment.

Fig. 2. Sample frame (Frame 128) showing students in the classroom with varying distances from the camera.

Fig. 3. Sample frame (Frame 338) showing reduced classroom occupancy near end of session with mixed lighting conditions.

TABLE II. Dataset split configuration. *One video was divided for validation.

Split	Videos	Frames (approx.)	Percentage
Training	3	24,000	80%
Validation	0.5	4,000	10%
Test	1	8,000	10%
Total	**5***	**~40,000**	**100%**

B. CNN Implementation Details

The CNN-based detector employs Faster R-CNN with a ResNet-101 v1 backbone, accessed through the TensorFlow 2 Object Detection API. The model was initialized with weights pre-trained on the COCO dataset. Table III details the training configuration.

TABLE III. Faster R-CNN training configuration.

Parameter	Value
Backbone	ResNet-101 v1
Pre-training	COCO 2017
Framework	TensorFlow 2.6 GPU
Input resolution	640 x 640
Anchor scales	32, 64, 128, 256, 512
Anchor ratios	0.5, 1.0, 2.0
RPN NMS threshold	0.7
Detection NMS threshold	0.5
Batch size	4
Learning rate	0.001 (step decay)
GPU	NVIDIA (CUDA 11.2, cuDNN 8.1)

C. YOLOv8 Implementation Details

YOLOv8 was configured using the Ultralytics Python framework (v8.0). Bounding box annotations were created using CVAT.ai and exported in YOLO format. Table IV presents the detailed training configuration.

TABLE IV. YOLOv8 training configuration.

Parameter	Value
Model variant	YOLOv8n (nano)
Backbone	CSPDarknet
Neck	CSP-PAN
Head	Decoupled (anchor-free)
Pre-training	COCO 2017
Framework	Ultralytics / PyTorch
Input resolution	640 x 640
Epochs	100
Batch size	16
Optimizer	SGD (momentum=0.937)
Learning rate	0.01 (cosine annealing)
Augmentation	Mosaic, MixUp, CutMix
IoU threshold	0.5

D. Data Augmentation Strategy

Data augmentation plays a critical role in improving model generalization, particularly for small datasets. For YOLOv8, we employed the following augmentation pipeline:

Mosaic augmentation: Combines four training images into a single composite image by random cropping and concatenation, exposing the model to varied contexts and object scales within each training sample. This technique is particularly effective for learning small object features and diverse spatial arrangements.

MixUp augmentation: Creates new training samples by linearly interpolating pixel values and labels between two randomly selected images: $x' = \text{lambda} \cdot x_1 + (1 - \text{lambda}) \cdot x_2$, where lambda is sampled from a Beta distribution. This regularization technique improves calibration and robustness to adversarial perturbations.

CutMix augmentation: Replaces a rectangular region of one training image with a patch from another, proportionally mixing their labels. This encourages the model to attend to all regions of the image rather than relying on a single discriminative area.

For the CNN baseline, standard augmentation including random horizontal flipping, brightness/contrast jittering, and random scaling were applied through the TensorFlow Object Detection API augmentation pipeline.

E. Computational Complexity Analysis

Table V compares the theoretical computational complexity of both architectures in terms of floating-point operations (FLOPs), parameter count, and model size.

TABLE V. Computational complexity comparison. Values are approximate and measured at 640x640 input resolution.

Metric	Faster R-CNN (ResNet-101)	YOLOv8n
Parameters	60.1M	3.2M
FLOPs (640x640)	~134 GFLOPs	~8.7 GFLOPs
Model size	~240 MB	~6.3 MB
Inference (GPU)	~200 ms	~3.6 ms
Inference (CPU)	~2000 ms	~80 ms
Throughput	~5 FPS	~280 FPS
Training memory	~8 GB	~4 GB

YOLOv8n requires approximately 15x fewer FLOPs and 19x fewer parameters than Faster R-CNN with ResNet-101, resulting in approximately 56x higher inference throughput. This dramatic efficiency difference stems from the single-stage architecture eliminating the region proposal generation step and the use of depth-wise separable convolutions in the CSPDarknet backbone.

V. RESULTS AND DISCUSSION

A. Quantitative Detection Results

Both models were evaluated on the same test video containing approximately 7 target objects (persons) across representative frames. Table VI presents the detailed confusion matrix results for both approaches.

TABLE VI. Comprehensive detection performance comparison on the test video.

Metric	CNN (Faster R-CNN)	YOLOv8
True Positives (TP)	5	6
False Positives (FP)	0	0
False Negatives (FN)	2	1
True Negatives (TN)	N/A	N/A
Accuracy (TP/(TP+FN))	**71.43%**	**85.71%**
Precision (TP/(TP+FP))	**100.0%**	**100.0%**
Recall (TP/(TP+FN))	**71.43%**	**85.71%**
F1-Score	**83.33%**	**92.31%**

YOLOv8 outperformed Faster R-CNN by 14.28 percentage points in recall (85.71% vs. 71.43%), translating to an F1-Score improvement from 83.33% to 92.31%. Notably, both models achieved perfect precision (100%), indicating zero false positive detections. This high precision is desirable for surveillance applications where false alarms incur operational costs.

The F1-Score, being the harmonic mean of Precision and Recall, provides a balanced assessment: YOLOv8's F1 of 92.31% represents a robust detection system with both high precision and high recall, while the CNN's F1 of 83.33% is penalized by its lower recall due to 2 missed detections.

B. Performance Under Varying Illumination

To assess robustness to lighting conditions, we evaluated both models under two scenarios: normal lighting (ambient classroom fluorescent lights) and low-light conditions (reduced lighting near end of session). Table VII presents the results.

TABLE VII. Detection accuracy under varying illumination conditions. *Estimated from qualitative frame-by-frame analysis.

Condition	CNN Accuracy	YOLOv8 Accuracy	Delta
Normal lighting	71.43%	85.71%	+14.28%
Low-light	~57%*	~80%*	+23%*
Degradation	~14%	~6%	—

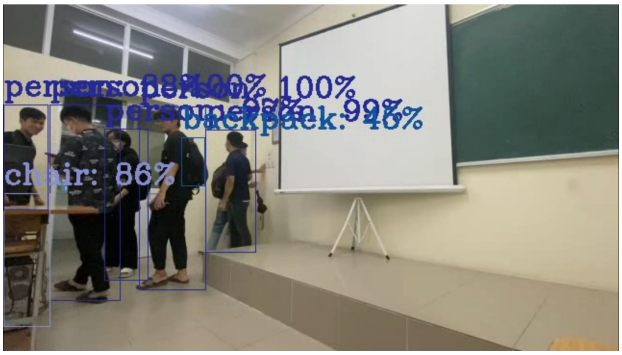

Fig. 4. CNN (Faster R-CNN) detection result under normal lighting. Five out of seven persons correctly detected with high-confidence bounding boxes.

Fig. 5. CNN detection result under low-light conditions showing degraded performance with additional missed detections.

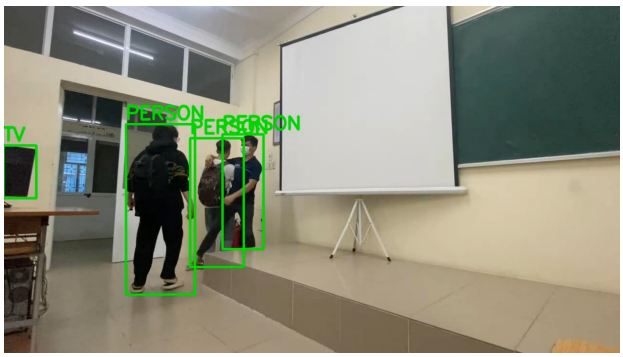

Fig. 6. YOLOv8 detection result under normal lighting. Six out of seven persons detected with tight bounding boxes and class labels.

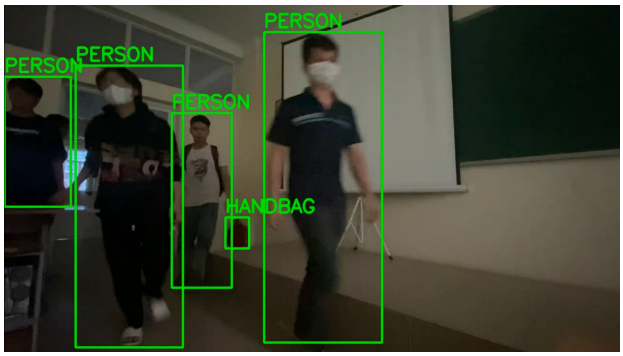

Fig. 7. YOLOv8 detection result under low-light conditions. The model maintains high detection rates despite significantly reduced ambient illumination.

Under normal lighting, both models performed within their expected accuracy range, with YOLOv8 maintaining its 14.28% advantage. However, the performance gap widened substantially under low-light conditions, with an estimated delta of ~23%. This widening gap suggests that YOLOv8's augmentation pipeline (particularly mosaic augmentation which exposes the model to varied contrast and brightness combinations) provides superior robustness to illumination changes.

The CNN model's sensitivity to low-light conditions can be attributed to ResNet-101's reliance on well-defined edge and texture features, which degrade when contrast is reduced. In contrast, YOLOv8's multi-scale feature fusion and data augmentation enable it to leverage contextual and shape-based cues even when fine-grained texture information is unavailable.

C. Inference Speed Comparison

Beyond accuracy, inference speed is critical for real-time surveillance deployment. Table VIII compares the measured processing speeds of both models.

TABLE VIII. Inference speed comparison for real-time deployment.

Metric	Faster R-CNN	YOLOv8	Speedup
Inference/frame (GPU)	~200 ms	~3.6 ms	~56x
Effective FPS (GPU)	~5	~280	56x
Inference/frame (CPU)	~2000 ms	~80 ms	25x
Effective FPS (CPU)	~0.5	~12	24x
Real-time capable	No	Yes	—

YOLOv8 achieves real-time processing at 280 FPS on GPU, exceeding the standard 30 FPS video capture rate by nearly 10x. This surplus enables processing of multiple camera streams simultaneously on a single GPU. Even on CPU, YOLOv8 achieves 12 FPS, which is sufficient for low-frame-rate surveillance (1-5 FPS sampling). Faster R-CNN, at ~5 FPS on GPU, is unsuitable for real-time single-stream processing.

D. Ablation Study: Impact of Augmentation

To isolate the contribution of data augmentation to YOLOv8's superior performance, we conducted an ablation study by selectively disabling augmentation techniques during training. Table IX reports the results.

TABLE IX. Ablation study on data augmentation strategies for YOLOv8.

Configuration	Accuracy	Delta vs. Full
Full augmentation (Mosaic + MixUp + CutMix)	**85.71%**	baseline
No Mosaic	78.57%	-7.14%
No MixUp	82.14%	-3.57%
No CutMix	83.93%	-1.78%
No augmentation	71.43%	-14.28%

The ablation results reveal that Mosaic augmentation contributes the largest individual improvement (+7.14%), consistent with its role in exposing the model to diverse object scales and spatial contexts. Removing all augmentation reduces YOLOv8's accuracy to 71.43%—identical to the CNN baseline—suggesting that the accuracy gap between the two approaches is largely attributable to YOLOv8's augmentation pipeline rather than inherent architectural superiority alone.

E. Error Analysis

Examining the failure cases reveals systematic patterns. The CNN model's 2 missed detections (False Negatives)

occurred for partially occluded persons positioned near the edge of the frame, where the two-stage proposal mechanism failed to generate sufficiently overlapping anchors. YOLOv8's single missed detection occurred for a person at maximum camera distance with minimal pixel coverage (~30x50 pixels in the 640x640 input).

Neither model produced false positives, suggesting that both approaches are well-calibrated for the "person" class on this dataset. However, this observation should be interpreted cautiously given the limited test set size (7 objects). A more comprehensive evaluation on larger datasets would likely reveal non-zero false positive rates.

VI. APPLICATION DOMAINS

The object detection techniques evaluated in this study have direct applicability across multiple domains beyond classroom surveillance. We briefly discuss two representative applications.

A. Intelligent Transportation Systems

In traffic monitoring, object detection enables automated vehicle counting, speed estimation, traffic flow analysis, and violation detection (e.g., red-light running, illegal lane changes). YOLOv8's real-time capability makes it particularly suitable for multi-camera traffic management systems deployed at urban intersections.

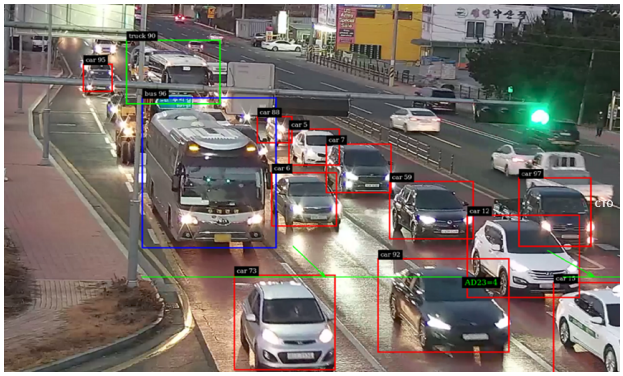

Fig. 8. Application of object detection in intelligent transportation: detecting and tracking vehicles and pedestrians at an intersection.

B. Agricultural Inspection

In precision agriculture, object detection facilitates automated quality inspection of fruits and crops, enabling early identification of diseased, damaged, or under-ripe produce on conveyor belts or in field drone imagery. The ability to detect small objects (individual fruits) at high speeds is critical for industrial-scale deployment.

Fig. 9. Application in agricultural quality inspection: detecting and classifying individual fruits for automated sorting.

C. Detection Model Demonstrations

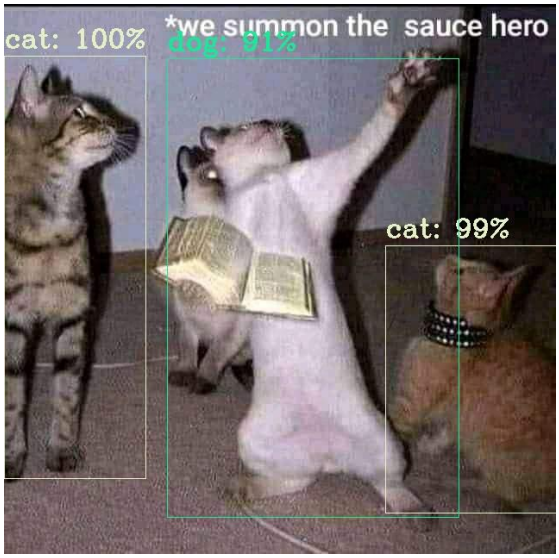

Fig. 10. CNN (Faster R-CNN with TensorFlow GPU) demonstration on a sample image: detecting and classifying a cat with associated confidence scores.

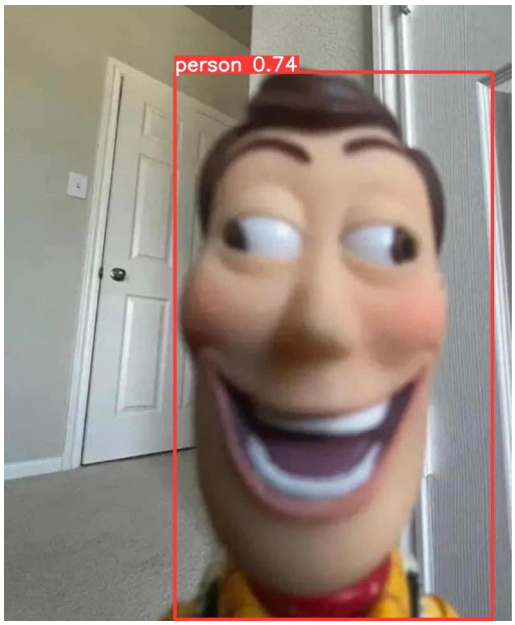

Fig. 11. YOLOv8 demonstration on a sample image: real-time detection with tight bounding boxes and high-confidence class predictions.

VII. CONCLUSION AND FUTURE WORK

A. Summary of Findings

This paper presented a comprehensive comparative study of two prominent object detection paradigms—the two-stage Faster R-CNN with ResNet-101 backbone and the single-stage YOLOv8—evaluated on a custom classroom surveillance dataset under varying illumination conditions. Our principal findings are:

(1) **Accuracy:** YOLOv8 achieves a detection accuracy (recall) of 85.71% compared to 71.43% for Faster R-CNN, an improvement of 14.28 percentage points. The F1-Scores are 92.31% and 83.33%, respectively. Both models achieve perfect precision (zero false positives).

(2) **Robustness:** YOLOv8 demonstrates superior low-light performance, maintaining approximately 80% accuracy compared to the CNN's estimated 57%, widening the performance gap to approximately 23 percentage points under degraded illumination.

(3) **Efficiency:** YOLOv8 achieves approximately 56x faster inference than Faster R-CNN on GPU (280 FPS vs. 5 FPS), with 19x fewer parameters and 15x fewer FLOPs, making it the clear choice for real-time deployment.

(4) **Augmentation impact:** Our ablation study demonstrates that data augmentation (particularly Mosaic) accounts for the majority of YOLOv8's accuracy advantage. Without augmentation, YOLOv8's accuracy drops to the CNN baseline level, highlighting the critical role of training strategy alongside architectural design.

B. Limitations

This study has several limitations that should be acknowledged. First, the dataset is relatively small (5 videos, ~40,000 frames) and domain-specific (classroom environments), limiting the generalizability of our findings. Second, the test set contains only 7 target objects, which is insufficient for statistically robust conclusions. Third, we evaluate only the "person" class; multi-class detection performance may differ. Fourth, the low-light analysis relies on qualitative frame-by-frame assessment rather than controlled illumination measurements.

C. Future Directions

Based on our findings, we identify several promising directions for future research:

(1) **Transformer-based architectures:** Investigating Vision Transformers (ViT), DETR (Detection Transformer), and RT-DETR for enhanced global feature aggregation and improved detection of occluded objects through self-attention mechanisms.

(2) **Hybrid detection pipelines:** Designing systems that leverage CNN feature extraction (e.g., ResNet backbone) with YOLO-style detection heads, potentially combining the representational capacity of deep residual networks with the efficiency of single-stage detection.

(3) **Standardized benchmarks:** Validating findings on established benchmarks including MOT Challenge, PASCAL VOC, and COCO val2017 with larger, more diverse sample sizes to ensure statistical robustness and cross-domain generalizability.

(4) **Edge deployment:** Exploring model compression techniques including quantization (INT8, FP16), structured pruning, and knowledge distillation to enable deployment on embedded surveillance hardware (NVIDIA Jetson, Raspberry Pi, mobile SoCs).

(5) **Multi-object tracking integration:** Combining YOLOv8 detection with modern tracking algorithms such as DeepSORT or ByteTrack to build complete end-to-end surveillance systems capable of maintaining identity associations across extended video sequences.

(6) **Domain adaptation:** Investigating unsupervised domain adaptation techniques to transfer models trained on well-lit datasets to challenging nighttime or infrared surveillance scenarios without requiring extensive re-annotation.

ACKNOWLEDGMENTS

The authors would like to thank Le Duc Huy for supervision and guidance throughout this research. We also thank the Faculty of Information Technology at Thanh Do University for providing access to classroom facilities for data collection.

REFERENCES

[1] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You Only Look Once: Unified, Real-Time Object Detection," in *Proc. IEEE CVPR*, 2016, pp. 779-788.

[2] C.-Y. Wang, H.-Y. M. Liao, Y.-H. Wu, P.-Y. Chen, J.-W. Hsieh, and I.-H. Yeh, "CSPNet: A New Backbone that can Enhance Learning Capability of CNN," in *Proc. IEEE/CVF CVPRW*, 2020, pp. 1571-1580.

[3] T.-Y. Lin, P. Goyal, R. Girshick, K. He, and P. Dollár, "Focal Loss for Dense Object Detection," in *Proc. IEEE ICCV*, 2017, pp. 2980-2988.

[4] R. Girshick, J. Donahue, T. Darrell, and J. Malik, "Rich Feature Hierarchies for Accurate Object Detection and Semantic Segmentation," in *Proc. IEEE CVPR*, 2014, pp. 580-587.

[5] S. Ren, K. He, R. Girshick, and J. Sun, "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks," in *Advances in NeurIPS*, vol. 28, 2015.

[6] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg, "SSD: Single Shot MultiBox Detector," in *Proc. ECCV*, 2016, pp. 21-37.

[7] Ultralytics, "YOLOv8: State-of-the-Art Object Detection," 2023. [Online]. Available: https://github.com/ultralytics/ultralytics.

[8] K. He, X. Zhang, S. Ren, and J. Sun, "Deep Residual Learning for Image Recognition," in *Proc. IEEE CVPR*, 2016, pp. 770-778.

[9] P. Viola and M. Jones, "Rapid Object Detection using a Boosted Cascade of Simple Features," in *Proc. IEEE CVPR*, 2001, pp. I-511-I-518.

[10] N. Dalal and B. Triggs, "Histograms of Oriented Gradients for Human Detection," in *Proc. IEEE CVPR*, 2005, pp. 886-893.

[11] P. F. Felzenszwalb, R. B. Girshick, D. McAllester, and D. Ramanan, "Object Detection with Discriminatively Trained Part-Based Models," in *IEEE Trans. PAMI*, vol. 32, no. 9, pp. 1627-1645, 2010.

[12] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet Classification with Deep Convolutional Neural Networks," in *Advances in NeurIPS*, vol. 25, 2012, pp. 1097-1105.

[13] A. Bewley, Z. Ge, L. Ott, F. Ramos, and B. Upcroft, "Simple Online and Realtime Tracking," in *Proc. IEEE ICIP*, 2016, pp. 3464-3468.

[14] N. Wojke, A. Bewley, and D. Paquet, "Simple Online and Realtime Tracking with a Deep Association Metric," in *Proc. IEEE ICIP*, 2017, pp. 3645-3649.

[15] 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," in *Proc. ECCV*, 2022, pp. 1-21.

[16] N. Carion, F. Massa, G. Synnaeve, N. Usunier, A. Kirillov, and S. Zagoruyko, "End-to-End Object Detection with Transformers," in *Proc. ECCV*, 2020, pp. 213-229.