Track Intrusion Detection and Track Integrity Evaluation – Year 2

Yu Qian, PhD
Principal Investigator
Department of Civil and Environmental Engineering
University of South Carolina

Youzhi Tang
Graduate Research Assistant
Department of Civil and Environmental Engineering
University of South Carolina

Dimitrios C. Rizos, PhD
Associate Director, UTCRS, Co-Principal Investigator
Department of Civil and Environmental Engineering
University of South Carolina

Nikolaos Vitzilaios
Co-Principal Investigator
Department of Mechanical Engineering
University of South Carolina

A Report on Research Sponsored by

University Transportation Center for Railway Safety (UTCRS)

Molinaroli College of Engineering and Computing
University of South Carolina

September 2025

Technical Report Documentation Page

1. Report No. UTCRS-USC-O7CY24	2. Government Accession No.	3. Recipient's Catalog No.
4. Title and Subtitle Track Intrusion Detection and Track Integrity Evaluation – Year 2		5. Report Date September 19, 2025
		6. Performing Organization Code UTCRS-USC
7. Author(s) Yu Qian, Youzhi Tang, Dimitrios C. Rizos, and Nikolaos Vitzilaios		8. Performing Organization Report No. UTCRS-USC-O7CY24
9. Performing Organization Name and Address University Transportation Center for Railway Safety (UTCRS) University of South Carolina (USC) Columbia, SC 29208		10. Work Unit No. (TRAIS)
		11. Contract or Grant No. 69A3552348340
12. Sponsoring Agency Name and Address U.S. Department of Transportation (USDOT) University Transportation Centers Program 1200 New Jersey Ave. SE Washington, DC, 20590		13. Type of Report and Period Covered June 1, 2024 – August 31, 2025
		14. Sponsoring Agency Code USDOT UTC Program
15. Supplementary Notes		
16. Abstract Track intrusion, especially trespassing within railroad rights-of-way, poses a major safety risk, causing more fatalities than train-vehicle collisions. Existing methods often struggle with real-time detection on edge devices. This study introduces YOLO-RCNN, a novel model integrating YOLO-FG for comprehensive object detection and segmentation, optimized for real-time, resource-constrained environments. YOLO-RCNN achieves superior performance, with deployment optimizations boosting inference speed from 4.69 FPS to 54.79 FPS on desktop and from 3.39 FPS to 29.19 FPS on Jetson AGX Orin, highlighting its potential for efficient, reliable railroad crossing monitoring.		
17. Key Words Railroad Tracks, Trespassers, Artificial Intelligence, Neural Networks, Computer Vision	18. Distribution Statement This report is available for download from https://www.utrgv.edu/railwaysafety/research/operations/index.htm	
19. Security Classification (of this report) None	20. Security Classification (of this page) None	21. No. of Pages 19	22. Price

Table of Contents

List of Figures 4
List of Tables 4
List of Abbreviations 4
Disclaimer 5
Acknowledgements 5
1. **SUMMARY** 6
2. **BACKGROUND** 7
3. **METHODOLOGY** 9
4. **EXPERIMENTS** 11
5. **CONCLUSIONS** 18
6. **REFERENCE** 19

List of Figures

Figure 1: YOLO-RCNN architecture 6
Figure 2: YOLO-FG architecture 10
Figure 3: Data collecting sites of Railroad Crossing Dataset (RCD) 11
Figure 4: Data collecting equipment for Railroad Crossing Dataset 12
Figure 5: Example detection results on the railroad crossing dataset (For each subfigure, from top to bottom: YOLO-RCNN, YOLO-CLIP-DeepSORT and GTR) 15

List of Tables

Table 1. The comparison of YOLO-RCNN, YOLO-CLIP-DeepSORT and GTR on RCD dataset 14
Table 2. The main components latencies (ms) of YOLO-RCNN, YOLO-CLIP-DeepSORT and GTR on A6000 GPU 16
Table 3. Latency (ms) comparison of pipelines within desktop environment 17
Table 4. Latency (ms) comparison of pipelines within Jetson AGX Orin 17

List of Abbreviations

AI	Artificial Intelligence
CLIP	Contrastive Language-Image Pretraining
CNNs	Convolutional Neural Networks
COCO	Common Objects in Context
CV	Computer Vision
GTR	Global Tracking Transformer
LVIS	Large Vocabulary Instance Segmentation
IoU	Intersection over Union
JDT	Joint Detection and Tracking
RCNN	Region-based Convolutional Network
ROI	Region of Interest
RPN	Region Proposal Network
TAO	Tracking Any Object
USDOT	U.S. Department of Transportation
UTC	University Transportation Center
ViT	Vision Transformer

Disclaimer

The contents of this report reflect the views of the authors, who are responsible for the facts and the accuracy of the information presented herein. This document is disseminated under the sponsorship of the U.S. Department of Transportation's University Transportation Centers Program, in the interest of information exchange. The U.S. Government assumes no liability for the contents or use thereof.

Acknowledgements

The authors wish to acknowledge the University Transportation Center for Railway Safety (UTCRS) for funding this project under the USDOT UTC Program Grant No. 69A3552348340.

1. SUMMARY

Track intrusion, especially trespassing, which encompasses unauthorized entry and lingering within the railroad right-of-way, is a significant safety concern. It has been associated with a higher number of fatalities compared to incidents involving collisions between vehicles and trains. This stark statistic underscores the urgent need for advanced surveillance and detection systems at rail crossings to ensure track integrity and enhance overall railway safety. This research aims to develop a novel YOLO-RCNN that innovatively merges foreground segmentation and object detection methodologies to form a comprehensive railroad crossing surveillance system, as shown in **Figure 1**.

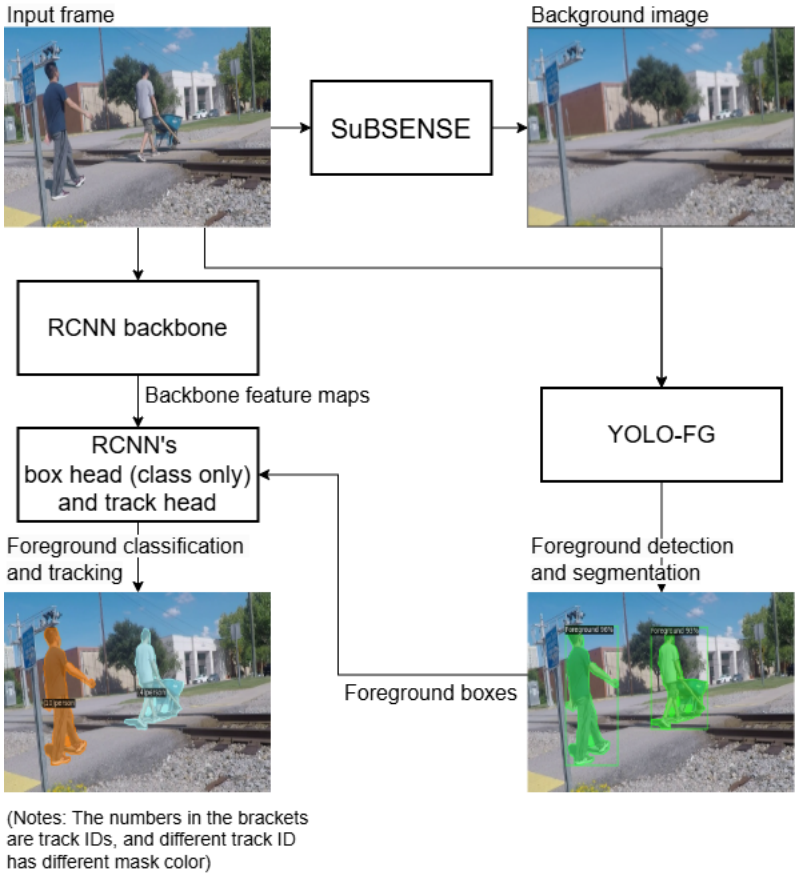

Figure 1: YOLO-RCNN architecture

The design of the proposed YOLO-RCNN model is strategically tailored to perform multiple functions simultaneously: foreground detection, segmentation, classification, and tracking of objects. This multifaceted approach allows for a more precise and efficient monitoring system adept at identifying and responding to any non-compliant objects or unauthorized activities within

the railroad area. The integrated system not only detects but also classifies various types of objects, making it possible to differentiate between harmless elements and potential hazards.

Our evaluation and testing phase highlights the effectiveness of the YOLO -RCNN model. The findings from these experiments validate the robustness of the network, demonstrating its superior capability to accurately identify and track unauthorized or non-compliant objects at railroad crossings. This enhanced detection is critical as it significantly surpasses the performance of traditional object detection models such as Mask-RCNN. Through a series of comparative analyses, the YOLO -RCNN consistently outperformed existing models, proving its potential as a pivotal technology in railway safety systems.

2. BACKGROUND

In recent years, the application of Computer Vision (CV) and Artificial Intelligence (AI) to enhance railroad safety and track resilience has become increasingly prevalent [1]. CNN-based models have significantly improved detection efficiency and accuracy, reducing human errors and aiding auxiliary decision-making. For instance, Zaman et al. [2] utilized Mask R-CNN to detect intrusion events on railroads. Additionally, Guo et al. [3] developed an automated video analysis, detection, and tracking system to assess traffic conditions at rail crossings. Among these deep learning approaches, object detection and foreground detection are critical for video surveillance, playing an essential role in ensuring safety at railroad crossings.

Object detection involves identifying and locating instances of specific object classes within images or video frames [4]. This technique, typically powered by neural networks, recognizes and classifies objects such as cars, pedestrians, and animals while also determining their boundaries or locations in the scene. Despite its widespread use and advancements across various fields, object detection still faces two significant challenges in the context of railroad crossing monitoring. First, conventional object detectors are limited to recognizing objects they have been explicitly trained on. While open-world object detectors offer more flexibility [5], they still cannot guarantee the detection of all track intrusions, which are often the cause of accidents. These intrusions could involve a wide range of objects, from animals and dropped parcels to collapsed catenary lines and more. Second, railroad crossing areas often contain static objects, such as traffic lights, barriers, and other infrastructure. These static elements can sometimes be falsely detected as intruding objects, leading to false-positive errors. Such errors increase the burden on post-processing

systems, which must differentiate between genuine intrusions and false alarms, ultimately creating stress and inefficiencies in monitoring systems. On the other hand, foreground detection, also known as change detection or background subtraction, is a technique designed to differentiate moving elements, referred to as the foreground, from the static scene, or background [6]. It analyzes the differences between the current frame and background images to identify regions with significant changes, which are then classified as foreground objects. This feature is particularly useful for railroad crossing monitoring, as it can accurately detect both static and moving outliers within the crossing area without mistakenly classifying background objects as intrusions. However, while foreground detection is effective at highlighting outliers within a scene as the Region of Interest (ROI), it falls short in two key areas necessary for a comprehensive railroad crossing monitoring system. First, its classification ability is limited. While it does not require precise classification of every object, it must at least distinguish between the train and other objects. This distinction is crucial to prevent false alarms caused by the train itself being incorrectly flagged as an intruder. Second, its tracking capability is lacking. It is essential to continuously track each pedestrian or object across multiple frames and trigger an alarm if anything or anyone remains in the crossing area for an extended period.

To extend foreground detection to include classification and tracking, a natural approach is to use a "classification and tracking by detection" method [7]. After identifying foreground objects in the current frame, their bounding boxes are cropped from the input image and fed into a Convolutional Neural Network (CNN) for feature extraction. These features can then be classified using a conventional classifier or a Contrastive Language-Image Pretraining (CLIP) model [8] and tracked using algorithms like DeepSORT [7] or its enhanced versions. However, this approach presents significant challenges. When numerous objects appear in a single frame, the resulting increase in batch size for the feature extraction CNN imposes a substantial computational burden. Additionally, the varied input sizes required by the feature extraction CNN introduce model dynamics, making it less compatible with static model inference frameworks like TensorRT. These challenges are particularly problematic for deployment on edge-computing devices with limited processing power.

To address the challenges of railroad crossing monitoring on edge computing platforms, this study proposes a YOLO-RCNN network that integrates foreground detection, segmentation, classification, and tracking. Foreground detection and segmentation are achieved through YOLO-

FG, which is built on YOLOv8-seg [9], a well-known network recognized for its high detection speed and accuracy, making it suitable for real-time applications. Inspired by the RoIAlign mechanism, classification and tracking are handled by Region-based Convolutional Neural Networks (RCNN). Unlike the classical "classification and tracking by detection" approach, in which each object is processed separately, the RCNN backbone extracts features from the entire input image, and the RoIAlign mechanism directly crops the backbone features for each object. This ensures that the batch size of the RCNN backbone is fixed at one, significantly reducing model dynamics and improving computational efficiency, particularly on resource-constrained edge computing platforms. This design enhances the ability to handle multiple objects without compromising real-time performance.

3. METHODOLOGY

Figure 1 illustrates the architecture of the proposed YOLO-RCNN network, a hybrid model that integrates two prominent deep learning frameworks: YOLO [9] and RCNN [4]. This model is specifically designed to perform foreground detection, segmentation, classification, and tracking in a unified framework. In this study, the YOLOv8-seg model, referred to as YOLO-FG, is extended for foreground detection and segmentation. It achieves this by distinguishing objects in the current frame from those in the background, which is generated using the SuBSENSE algorithm. Once foreground objects are identified, their bounding boxes are processed by the RCNN module, which handles classification and tracking to ensure accurate object identification and trajectory consistency.

3.1 Foreground detection and segmentation using YOLO-FG

As illustrated in **Figure 2**, YOLO-FG enhances the well-established YOLOv8-seg architecture [9] by incorporating a specialized input head designed for foreground detection and segmentation. This modification enables YOLO-FG to process both the current video frame and the background image simultaneously, facilitating accurate foreground detection. YOLOv8-seg extends YOLOv8 by integrating segmentation capabilities, leveraging the Protonet architecture from YOLACT [10]. This enhancement allows YOLOv8-seg to perform not only object detection but also precise instance-level segmentation, making it highly effective for detailed object delineation.

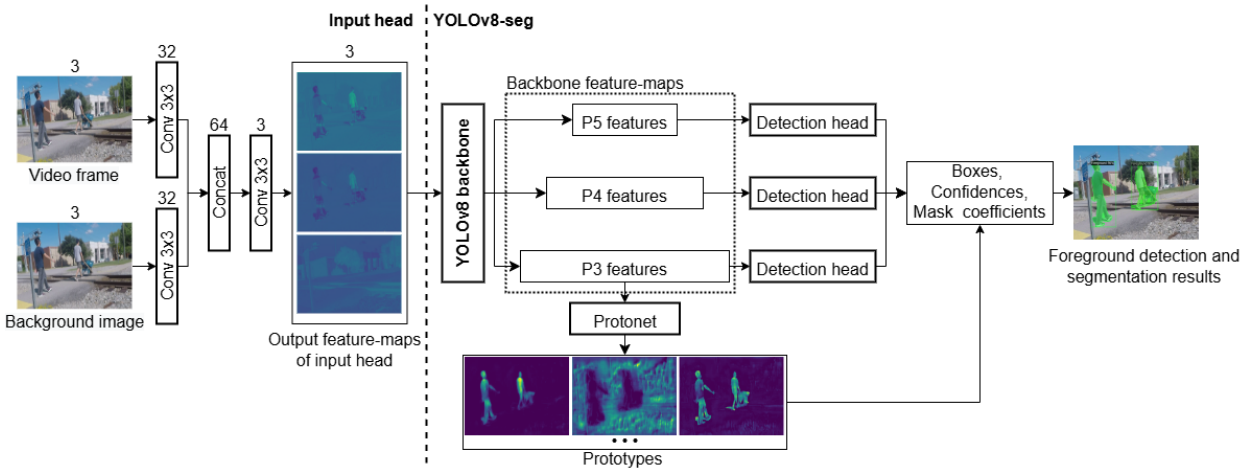

Figure 2: YOLO-FG architecture

To maintain modularity and avoid modifying the YOLO backbone, YOLO-FG introduces a dedicated input head. This input head consists of two distinct branches, each equipped with a single convolutional layer. The first branch processes the input frame, and the second branch processes the background image. Each branch transforms its respective input into a 32-channel feature map. The resulting feature maps are then concatenated, producing a 64-channel fused feature map. This concatenated representation is subsequently passed through a final convolutional layer, compressing it into a 3-channel output. As shown in **Figure 2**, this compressed output preserves essential information while ensuring compatibility with the input format of YOLOv8-seg. By employing this modular input head, YOLO-FG seamlessly integrates with YOLOv8-seg without requiring further architectural modifications.

3.2 Foreground classification and tracking using RCNN

As discussed in the introduction section, this study employs RCNN for foreground classification and tracking, aiming to replace the conventional "classification and tracking by detection" approach. This modification enhances computational efficiency, particularly in edge devices where resource constraints are a critical factor.

During the training phase, the RCNN follows standard training procedures without requiring architectural modifications. In this study, the RCNN architecture from GTR [11], is adopted. As a Joint Detection and Tracking (JDT) method, GTR introduces a built-in, trainable tracker head in addition to standard box heads, enabling object association across frames. When training the track head, both the track head and the RCNN backbone are fine-tuned, ensuring that

the features extracted by the backbone are optimized for trackability.

In contrast, the inference stage of YOLO-RCNN follows a different approach. Here, RCNN does not retain all network components. Specifically, the Region Proposal Network (RPN), which generates proposal boxes, is omitted during inference. Instead, only the backbone, box head, and track head remain operational. The backbone serves as the primary feature extractor, processing input images to generate feature representations. The box head and track head classify and track foreground objects by cropping feature maps produced by the backbone. Foreground boxes predicted by YOLO-FG guide the cropping process, ensuring efficient foreground classification and tracking. By adopting this streamlined inference strategy, YOLO-RCNN reduces computational overhead while maintaining high tracking accuracy, making it well-suited for real-time applications on edge devices.

4. EXPERIMENTS

4.1 Railroad Crossing Dataset

Railroad Crossing Dataset

To evaluate the performance of the YOLO-RCNN model, the Railroad Crossing Dataset (RCD) is established. As illustrated in **Figure 3**, data collection sites were strategically selected across various locations in Columbia, SC to ensure the dataset robustness and generalizability. As shown in **Figure 4**, video recordings for sites in **Figures 3(a)** and **3(b)** were captured using webcams, while data for sites in **Figures 3(c)**, **3(d)**, and **3(e)** were collected using a mini drone. The dataset comprises 25 video clips, each containing 32 frames. Notably, this dataset was exclusively reserved for model testing, ensuring an unbiased assessment of the model performance in real-world settings.

(a). 409 Main ST, Columbia, SC, 29201 (b). 718 Devine ST, Columbia, SC, 29201 (c). 230 Huger ST, Columbia, SC, 29201 (d). 216 Tryon ST, Columbia, SC, 29201 (e). 949 Rosewood Dr, Columbia, SC, 29201

Figure 3: Data collecting sites of Railroad Crossing Dataset (RCD)

Figure 4: Data collecting equipment for Railroad Crossing Dataset

4.2 Network training processes

The training process for the YOLO-RCNN model consists of three stages. In the first stage, the YOLO-FG component is trained for foreground detection and segmentation. This is followed by the second and third stages, which focus on training the RCNN module for foreground classification and tracking.

In the first training stage, the YOLO-FG for foreground detection is trained using the CDnet 2014 dataset. In the second training stage, the RCNN training process follows the methodology presented in GTR. During this stage, the RCNN is pretrained exclusively in detection mode, utilizing a combined dataset of LVISv1 and COCO. In the final training stage, the track head participates in fine-tuning. The combined dataset of LVISv1 and COCO continues to be used throughout this stage to ensure robust feature learning.

Finally, the trained YOLO-RCNN model is validated using the RCD dataset to ensure its real-world applicability.

4.3 Evaluation Metrics

For RCD datasets, mAP is used to evaluate the detection quality of YOLO-RCNN. Given predicted bounding box P and ground-truth bounding box G, calculate their Intersection over Union (IoU):

$$\boldsymbol{IoU}(\boldsymbol{P},\boldsymbol{G}) = \frac{\boldsymbol{Area}(\boldsymbol{P}\cap\boldsymbol{G})}{\boldsymbol{Area}(\boldsymbol{P}\cup\boldsymbol{G})} \quad (1)$$

A detection is considered a match if $\boldsymbol{IoU}$ greater than a threshold τ. Then, True Positives

(TP) indicates the number of matched detection, and False Positives (FP) counts the predictions that failed to match any ground-truth boxes. After knowing the TP and FP, Precision (P) and Recall (R) of the detection results can be calculated as:

$$\boldsymbol{P} = \frac{\boldsymbol{TP}}{\boldsymbol{TP+FP}} \tag{2}$$

$$\boldsymbol{R} = \frac{\boldsymbol{TP}}{\boldsymbol{TP+FN}} \tag{3}$$

Finally, mAP is computed by averaging the precision across recall levels of different threshold τ:

$$\boldsymbol{mAP} = \frac{\boldsymbol{1}}{\boldsymbol{C}} \sum_{\boldsymbol{C=1}}^{\boldsymbol{C}} \boldsymbol{AP}(\boldsymbol{c}) \tag{4}$$

$$\boldsymbol{AP}(\boldsymbol{c}) = \int_{\boldsymbol{0}}^{\boldsymbol{1}} \boldsymbol{P}(\boldsymbol{R}) \boldsymbol{dR} \tag{5}$$

where $\boldsymbol{C}$ is the is the number of classes. $\boldsymbol{AP}(\boldsymbol{c})$ is the average precision for class $\boldsymbol{c}$. $\boldsymbol{P}(\boldsymbol{R})$ is the precision at recall $\boldsymbol{r}$.

Additionally, Segmentation mAP (Seg mAP) is employed to evaluate its segmentation performance. The key difference between Seg mAP and mAP lies in the calculation of IoU: Seg mAP uses segmentation masks, while mAP relies on bounding boxes, as shown in Equation (1).

To assess its tracking performance, tracking mAP@0.5 and HOTA metrics is adopted. tracking mAP@0.5 is based on standard object detection mAP. And their difference is the tracking mAP uses the 3D temporal spatial IoU to mach the predicted trajectory P and the ground-truth trajectory G:

$$\boldsymbol{IoU}(\boldsymbol{P}, \boldsymbol{G}) = \frac{\sum_{\boldsymbol{t} \in \boldsymbol{T_G} \cap \boldsymbol{T_P}}^{\boldsymbol{t}} \boldsymbol{IoU}(\boldsymbol{P_t}, \boldsymbol{G_t})}{\boldsymbol{T_G} \cup \boldsymbol{T_P}} \tag{6}$$

where$\boldsymbol{T_G}$ and $\boldsymbol{T_P}$ are the time range of G and P. $\boldsymbol{IoU}(\boldsymbol{P_t}, \boldsymbol{G_t})$ is the standard 2D IoU between the ground-truth and predicted bounding boxes at frame t. An IoU threshold τ = 0.5 is applied to consider a trajectory as a match.

HOTA is defined as the geometric mean of detection accuracy (DetA) and association accuracy (AssA):

$$\boldsymbol{HOTA} = \sqrt{\mathbf{DetA} \times \mathbf{AssA}} \tag{7}$$

DetA and AssA defined in [12].

4.4 Performance on the Rail Crossing Dataset

To evaluate the efficiency of YOLO-RCNN compared to YOLO-CLIP-DeepSORT, a comprehensive analysis is performed. While both methods utilize YOLO-FG for foreground detection, their approaches to foreground classification and tracking differ. YOLO-RCNN employs the RCNN architecture, whereas YOLO-CLIP-DeepSORT leverages CLIP [8] for classification and DeepSORT [13] for tracking. To further validate YOLO-RCNN in railroad crossing monitoring, it was compared against GTR, a detection-based tracking network. The GTR model retained its original configuration and was trained on the TAO dataset. It is also important to note that GTR does not support segmentation natively, which affects its performance in this application.

Table 1 presents a performance comparison of YOLO-RCNN, YOLO-CLIP-DeepSORT, and GTR on the Railroad Crossing Dataset, with their example detection results illustrated in **Figure 7**. YOLO-RCNN achieves an mAP of 54.86%, a Seg mAP of 48.35%, a Track mAP of 56.31%, and a HOTA score of 63.44%.

Table 1. The comparison of YOLO-RCNN, YOLO-CLIP-DeepSORT and GTR on RCD dataset

Models	mAP(%)	Seg mAP(%)	Track mAP (%)	HOTA (%)
YOLO-RCNN	**54.86**	**48.35**	**56.31**	**63.44**
YOLO-CLIP-DeepSORT	53.12	47.71	53.62	60.30
GTR	45.46		49.86	52.76

When compared to YOLO-CLIP-DeepSORT, YOLO-RCNN demonstrates slight improvements, outperforming it by 1.74% in mAP, 0.64% in Seg mAP, 2.69% in Track mAP, and 3.14% in HOTA. Since both methods rely on the same YOLO-FG model for foreground detection but differ in their classification approaches, their mAP and Seg mAP scores remain relatively close. The moderate differences in tracking metrics, such as Track mAP and HOTA, can be attributed to the fixed-camera setup in this application, which reduces tracking complexity.

(a). 409 Main St, Columbia, SC, 29201

(b). 718 Devine St, Columbia, SC, 29201

(c). 230 Huger St, SC, 29201

(d). 216 Tryon St, Columbia, SC, 29201

(e). 949 Rosewood Dr, Columbia, SC, 29201

Figure 5: Example detection results on the railroad crossing dataset (For each subfigure, from top to bottom: YOLO-RCNN, YOLO-CLIP-DeepSORT and GTR)

Both YOLO-RCNN and YOLO-CLIP-DeepSORT outperform GTR. As an object detection-based method, GTR suffers from false-positive errors caused by detecting numerous background objects, such as road signs, traffic lights, and stationary vehicles. Additionally, despite being trained on 1,203 classes from the LVIS dataset, GTR struggles to detect unusual objects, such as the cloth and cube shown in **Figure 7(d)**. Accordingly, YOLO-RCNN surpasses GTR by 9.40% in mAP, 6.45% in Track mAP, and 10.68% in HOTA, further demonstrating its superior performance in railroad crossing monitoring.

Although YOLO-RCNN and YOLO-CLIP-DeepSORT demonstrate comparable performance on the RCD dataset, YOLO-RCNN achieves significantly higher efficiency, as illustrated in Table 2. As outlined in the Introduction, YOLO-RCNN was designed to overcome the efficiency limitations inherent in "classification and tracking by detection" methods, such as YOLO-CLIP-DeepSORT. Specifically, the latency of using the RCNN architecture for foreground classification and tracking is 25.64 ms, which is 37.53 ms faster than the combined latency of using CLIP and DeepSORT for the same tasks. As a background generation model, SuBSENSE plays a critical role in both YOLO-RCNN and YOLO-CLIP-DeepSORT. However, SuBSENSE introduces significant latency, measured at 158.41 ms, which substantially impacts the inference speed of both systems. Consequently, YOLO-RCNN accumulates a total latency of 196.94 ms, making it 133.83 ms higher than that of the GTR model. To address this limitation, the next section focuses on optimizing and deploying the YOLO-RCNN model.

Table 2. The main components latencies (ms) of YOLO-RCNN, YOLO-CLIP-DeepSORT and GTR on A6000 GPU.

Components	YOLO-RCNN	YOLO-CLIP -DeepSORT	GTR
SuBSENSE	158.41	158.41	
YOLO-FG (n)	12.89	12.89	
RCNN backbone	15.07		41.77
RCNN RPN			10.17
RCNN box head	0.61		3.76
RCNN track head (features extraction)	1.21		1.21
RCNN track head (features association)	8.75		10.57
CLIP		38.49	
DeepSORT		24.68	
Total (except pre- and post-processing)	196.94	234.47	63.11

4.5 Inference pipeline deployment and optimization

This section details tests conducted on both a desktop platform and the Jetson AGX Orin, an edge-computing device. The desktop platform is equipped with an Intel i9-10920X CPU and dual NVIDIA RTX A6000 GPUs, though only one GPU is utilized for inference. The A6000 GPU, with 10,752 CUDA cores and 336 Tensor cores, provides substantial computational power, enabling efficient execution of complex tasks. In contrast, the Jetson AGX Orin features a 12-core ARM CPU, 2,048 CUDA cores, and 64 Tensor cores. Its normal and peak power consumption is 30W and 60W, respectively, making it well-suited for model deployment in resource-constrained environments.

As illustrated in Tables 3 and 4, the YOLO-RCNN inference pipeline was optimized and deployed using C++, TensorRT, and oneTBB. The entire pipeline was converted from Python to C++ to reduce inference time, as C++ provides finer control over hardware and system resources, improving overall performance and efficiency compared to Python.

On desktop systems, the model inference latency was reduced from 38.53 ms to 16.24 ms. On the Jetson AGX Orin, the latency decreased from 105.29 ms to 32.31 ms, demonstrating significant improvements in efficiency and deployment feasibility.

Table 3. Latency (ms) comparison of pipelines within desktop environment

Node	PyTorch + Python	TensorRT + oneTBB + C++
Pre-processing	9.53	17.93
SuBSENSE	158.41	129.22
Model inference	38.53	16.24
Pos-processing	6.86	6.38
Overall	213.33 (4.69 FPS)	**18.25 (54.79 FPS)**

Table 4. Latency (ms) comparison of pipelines within Jetson AGX Orin

Node	PyTorch + Python	TensorRT + oneTBB + C++
Pre-processing	11.93	10.51
SuBSENSE	167.58	130.48
Model inference	105.29	32.31
Pos-processing	9.86	5.34
Overall	294.66 (3.39 FPS)	**34.26 (29.19 FPS)**

5. CONCLUSIONS

This study presents a novel YOLO-RCNN model for railroad crossing monitoring. The proposed approach integrates YOLO-FG, a component designed to detect and segment all objects in a scene, a capability that traditional object detectors lack. Additionally, the model leverages RCNN with the RoIAlign mechanism, effectively addressing the limitations of conventional "classification and tracking by detection" methods while significantly reducing computational overhead. By achieving high-accuracy detection, classification, and tracking with enhanced computational efficiency, YOLO-RCNN is well-suited for real-time applications on resource-constrained edge computing devices.

The results from experiments demonstrated the proposed YOLO-RCNN presenting superior performance in foreground detection and tracking tasks. On the custom RCD dataset, YOLO-RCNN demonstrated strong performance across multiple evaluation metrics, achieving an mAP of 54.86%, a Seg mAP of 48.35%, a Track mAP of 56.31%, and a HOTA score of 63.44%. These results underscore its superiority over YOLO-CLIP-DeepSORT and GTR in classification, segmentation, and tracking tasks.

The efficiency of YOLO-RCNN was further validated through deployment and optimization. Transitioning from Python to C++ and leveraging TensorRT and oneTBB significantly improved the model's inference speed, achieving a remarkable increase from 4.69 FPS to 54.79 FPS on desktop systems and from 3.39 FPS to 29.19 FPS on the Jetson AGX Orin. These optimizations demonstrate the practical feasibility of YOLO-RCNN for real-time monitoring in real-world applications.

The proposed YOLO-RCNN model has been successfully deployed to an edge-computing device and validated its real-time inference capability. The developed model is ready to be integrated into a UAV or other portable platform for field-monitoring.

6. REFERENCE

[1] Y. Tang, Y. Wang, and Y. Qian, “Railroad missing components detection via cascade region-based convolutional neural network with predefined proposal templates,” *Computer-Aided Civil and Infrastructure Engineering*, 2024.

[2] A. Zaman, B. Ren, and X. Liu, “Artificial intelligence-aided automated detection of railroad trespassing,” *Transportation Research Record*, vol. 2673, no. 7, pp. 25–37, 2019.

[3] F. Guo, Y. Wang, and Y. Qian, “Computer vision-based approach for smart traffic condition assessment at the railroad grade crossing,” *Advanced Engineering Informatics*, vol. 51, p. 101456, 2022.

[4] K. He, G. Gkioxari, P. Dollár, and R. Girshick, “Mask R-CNN,” in *Proc. IEEE Int. Conf. Comput. Vis.*, 2017, pp. 2961–2969.

[5] K. J. Joseph, S. Khan, F. S. Khan, and V. N. Balasubramanian, “Towards open world object detection,” in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2021, pp. 5830–5840.

[6] P. L. St-Charles, G. A. Bilodeau, and R. Bergevin, “SuBSENSE: A universal change detection method with local adaptive sensitivity,” *IEEE Trans. Image Process.*, vol. 24, no. 1, pp. 359–373, 2014.

[7] N. Wojke, A. Bewley, and D. Paulus, “Simple online and realtime tracking with a deep association metric,” in *Proc. IEEE Int. Conf. Image Process.*, 2017, pp. 3645–3649.

[8] A. Radford et al., “Learning transferable visual models from natural language supervision,” in *Proc. Int. Conf. Mach. Learn.*, 2021, pp. 8748–8763.

[9] G. Jocher, J. Qiu, and A. Chaurasia, “Ultralytics YOLO (Version 8.0.0) [Computer software],” Ultralytics, 2023. [Online]. Available: https://ultralytics.com

[10] D. Bolya et al., “YOLACT: Real-time instance segmentation,” in *Proc. IEEE/CVF Int. Conf. Comput. Vis.*, 2019, pp. 9157–9166.

[11] X. Zhou et al., “Global tracking transformers,” in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2022, pp. 8771–8780.

[12] J. Luiten, et al. "Hota: A higher order metric for evaluating multi-object tracking." *International journal of computer vision* 129 (2021): 548-578.

[13] N. Wojke, A. Bewley, and D. Paulus, “Simple online and realtime tracking with a deep association metric,” in *Proc. IEEE Int. Conf. Image Process.*, 2017, pp. 3645–3649.