Received: 24 April 2025 | Revised: 29 December 2025 | Accepted: 2 April 2026 | Published online: 27 April 2026

RESEARCH ARTICLE

Artificial Intelligence and Applications
2026, Vol. 00(00) 1–10
DOI: 10.47852/bonviewAIA62026000

BON VIEW PUBLISHING

Instance Segmentation for Infrastructure and Obstacle Detection in Automatic Train Operation Systems

Vladimir A. Fedorov[1, 2,*]

[1] *Department of Electronic Engineering, Ural Federal University, Russia*

[2] *Department of Research and Development for Integrated Security Systems, NPO SAUT LLC, Russia*

Abstract: The paper considers the application of instance segmentation methods for solving problems of detecting railway infrastructure objects and obstacles in automatic train operation (ATO) systems. Particular attention is paid to the possibility of using the You Only Look Once version 11 (YOLOv11) deep neural network architecture in the context of increasing automation levels to Grade of Automation 3 and Grade of Automation 4, where reliable operation of perception systems in real time is critical. One of the major contributions in this paper includes developing a specialized dataset on the perception of railways with 20,000 images, each with pixel-level annotation on 46 object classes. This paper has evaluated 25 different YOLOv11 models, which vary in terms of depth and input image resolutions. All configuration models were trained on a specially collected dataset of images obtained from rolling stock video cameras in real operating conditions. Performance evaluation included such metrics as segmentation accuracy, inference speed, and computational complexity. The results demonstrate that YOLOv11 provides a flexible choice of configurations, allowing the model to be adapted to specific technical conditions and requirements. This confirms the feasibility of using YOLOv11 in real ATO systems to improve the reliability of perception, support autonomous navigation tasks, and timely detection of critical objects along the train route.

Keywords: instance segmentation, automatic train operation, automated train control, YOLOv11

1. Introduction

Modern automated train control (ATC) systems [1] are fundamentally changing the operation of railways. They reduce the impact of human error and make transportation safer, more cost-effective, and predictable [2]. This is especially true on busy mainlines and in large urban areas where even small delays are costly. The central aspect of ATC is the Grade of Automation (GoA) [3]. It determines what part of the tasks the system takes on: from simple assistance to the driver to completely autonomous operation. The higher the level of GoA, the greater the requirements for the accuracy of the algorithms, because in real conditions, the train has to take into account dozens of factors: from sudden obstacles to changes in the schedule.

According to the international standard IEC 62267 [4], automation levels are classified into five levels, from GoA0 to GoA4:

1) GoA0 (on-sight train operation). The train is operated by a human driver without any automatic systems. Everything depends on the person in charge.
2) GoA1 (non-automated train operation). Manual driving is supported by automatic safety systems, such as automatic train protection [5], which help prevent collisions by enforcing braking and speed limits.
3) GoA2 (semi-automated train operation). The train runs by itself most of the time—speed and braking are automatic. But someone still needs to open and close doors, start the journey, and respond in an emergency.
4) GoA3 (driverless train operation). There is no need for a human driver to drive the train. Motion is automated completely. But someone from the staff may still be on board to help passengers or respond if something unusual happens.
5) GoA4 (unattended train operation). The train runs entirely automatically. It travels, opens and closes doors, and reacts to emergencies without any staff on board.

As train systems progress toward higher degrees of automation, especially in GoA3 and GoA4 scenarios, human intervention is less and less required or even absent. This sort of shift places tremendous pressure on onboard systems to be extremely reliable, interpretable, and responsive. Particularly for perception and recognition of the environment, these systems must enable the train to "see" and understand its surroundings with the same level of success as a human operator. It must do this in real time and across a wide range of conditions, from variable lighting to weather and operational challenges [6].

***Corresponding author:** Vladimir A. Fedorov, Department of Electronic Engineering, Ural Federal University and Department of Research and Development for Integrated Security Systems, NPO SAUT LLC, Russia. Email: fedorov.vladimir@urfu.ru

Among the critical components of such autonomous systems are computer vision technologies, which must accurately detect and classify elements of railway infrastructure (e.g., tracks, switches, signals), distinguish between safe and hazardous zones, and identify obstacles that could jeopardize safe operation (e.g., people, animals, foreign objects. The ability to detect not only expected elements but also rare and potentially dangerous situations is fundamental for achieving safety and reliability in driverless railway systems.

In standard practice, convolutional neural networks are used for object detection tasks, where an object position in an image is typically represented using a bounding box. But it is not enough for railway object location. It often lacks the accuracy required for deployment in automatic train operation (ATO) systems [7]. In particular, bounding boxes do not offer sufficient information to determine whether an object is within the train clearance envelope and thus poses a potential hazard.

In this paper, we investigate the deep learning model You Only Look Once version 11 (YOLOv11) [8], which combines high performance with accurate object segmentation. We trained and evaluated 25 configurations of YOLOv11, varying the architectural depth (n, s, m, l, x) and input resolution (from 640 × 640 to 1920 × 1920). The objective of this paper is to evaluate the feasibility and effectiveness of deploying YOLOv11 models for railway infrastructure and obstacle segmentation in ATO systems. We focus on the potential role of YOLO11 in enabling higher degrees of automation in real-world railway scenarios.

The rest of this paper is organized as follows: Section 2 presents related work, Section 3 describes the materials and methods, Section 4 discusses the results, and Section 5 concludes the paper.

2. Related Work

In the last decade, nevertheless, there has been phenomenal progress in designing computer vision systems for rail applications, which has been prompted mostly by growing needs for higher levels of automation (GoA3 and GoA4). Initially, techniques used were highly dependent on traditional image processing techniques [9], that is, edge detection, background subtraction, and histogram analysis. Though these techniques functioned wonderfully well under laboratory conditions, they failed to perform in real-world scenarios with reduced lighting, occlusion, and varied weather [10].

The advent of convolutional neural networks marked a major change, ushering in spectacular advances in accuracy and resilience of vision [11]. Perhaps among the most celebrated frameworks to have come along is the You Only Look Once (YOLO) family [12], which has seen widespread use for transportation-related applications because it can strike a balance between high detection precision and real-time inference.

In the context of railway systems, several studies have pointed out the versatility of YOLO models for particular applications. For example, YOLOv3 has been used to detect surface irregularities along railway tracks and identify fractures and deterioration signs [13]. The improvements integrated into YOLOv4 have enhanced the ability to detect locomotive signal lights and people in close proximity to the tracks [14]. Furthermore, YOLOv5 has been applied to signal light detection, with consistent performance in various environmental contexts [15].

In spite of these improvements, most of the applications of the YOLO model continue to be based on bounding-box detection, which can only provide rough object localization. In applications involving greater complexity, in which the precise delineation of object boundaries is critical, this can be inadequate. Accordingly, greater focus has been put on instance segmentation techniques that enable recognition of objects at the pixel level and provide a more nuanced comprehension of the spatial relationships within the scene [16]. This level of detail is particularly required in safety-critical systems, such as automatic train control.

Among the instance segmentation models, Mask Region-based Convolutional Neural Networks (Mask R-CNN) [17] has been of particular interest. Mask R-CNN combines object detection with accurate segmentation at high resolutions and is, therefore, particularly useful in scenarios where there is object occlusion or complicated structural characteristics. The high computational demands of Mask R-CNN [18] can, however, be very problematic, especially in applications where real-time computation is of value or in applications on low-capacity hardware.

In recent developments, the YOLOv8 model [19] has proven to be the better option. By optimizing the YOLO architecture for segmentation tasks, YOLOv8 provides high inference rates with minimal hardware requirements [20]. Although not always equaling Mask R-CNN segmentation accuracy, its ability to provide competitive performance at reduced latency positions it as an acceptable option for deployment in onboard ATO systems, where timely and stable perception is demanded.

Even though YOLOv8 set a strong baseline for instance segmentation in the real-time scenario, the YOLOv11 configuration brings some improvements that are very important to the specific use case of the railway industry. Among the improvements are a better structure for the spatial pyramid pool and the transformer attention mechanism [21]. This is very important to properly delineate the extended rail systems (such as the railways and the overhead wires) and to identify small distant objects against a complex background. Additionally, the YOLOv11 neck and the adaptive upscaling mechanism bring a better computational cost profile than YOLOv8 and a better mask detection accuracy. In the case of an ATO system that has to provide the highest accuracy to the object boundary delineation but still has to perform the task in a real-time manner, YOLOv11 would be a better contender than YOLOv8.

3. Materials and Methods

3.1. Dataset preparation

To aid in the training and testing of instance segmentation models, a dedicated dataset was gathered from videos recorded using front cameras installed in driver cabins of moving trains. The videos were recorded from various locations on the Russian railway network, spanning a diverse set of real-world operational conditions.

In order to guarantee the strength and usability of the dataset in various situations, the content of the video was recorded under an extensive variety of lighting conditions, such as daytime, nighttime, and twilight, and under various weather conditions, such as clear, rain, snow, and fog.

In order to prevent redundancy and have a varied set of visual appearances, still images were systematically sampled from the video at fixed intervals. All images were brought to a uniform Full HD resolution (1920 × 1080 pixels) to standardize the input format for subsequent training and evaluation.

Based on the dataset collected, 20,000 representative frames were sampled and manually tagged for instance segmentation

Figure 1
Samples of dataset images

tasks. Annotation was done in the form of a polygon-based method to properly outline object boundaries at the pixel level. Samples of dataset images are shown in Figure 1.

The dataset includes 46 object classes that were chosen for their relevance to railway infrastructure and safety operations. There are railway tracks, turnouts, signal lights, pedestrians, animals, motor vehicles, railway rolling stock, and other infrastructure components and potential obstacles.

3.2. Model architecture

For instance segmentation, we utilized the YOLOv11 architecture, which is a new version of the YOLO family [22]. YOLOv11 incorporates segmentation capabilities natively into the detection pipeline. Following the strengths of YOLOv8 [23], YOLOv11 introduces a couple of crucial enhancements, such as enhanced spatial pyramid pooling, transformer-based attention modules, and adaptive upsampling methods.

As evident from Figure 2, the YOLOv11 architecture consists of three key parts: the backbone, the neck, and the head. The three play very critical roles: the backbone extracts features from the input image, the neck pools and normalizes the features, and the head produces the final predictions. Each of these plays an important role in the model's performance in object detection and instance segmentation.

The backbone of the YOLOv11 architecture is responsible for extracting features from input images through hierarchical stacking of convolution layers. The spatial image resolution decreases with each pass-through layer, and the feature map depth increases. The structure is designed in such a way that both low-level feature information such as edges and textures and high-level semantic information such as object boundaries and structural information are extracted.

Being an interconnection point between the backbone and the prediction head, the neck is particularly responsible for enriching and concatenating features. Within YOLOv11, the neck integrates a backboned layout drawing inspiration from the Feature Pyramid Network (FPN) [24], through which it has the capability of consolidating knowledge at disparate scales. Through interconnections across layers of distinct resolutions, it retains high-resolution spatial information combined with deeper semantic awareness. This proves particularly useful when detecting objects of varying sizes or those that are partially occluded or in complex scenes.

The head module of YOLOv11 produces the final outputs of the model: object class, bounding-box coordinates, and instance segmentation masks. A departure from traditional detection pipelines, YOLOv11, however, has an innovative segmentation branch [25] that produces fine, pixel-level masks for each detected object. The addition enhances the spatial reasoning and ability of the model in differentiating among individual objects, especially in densely visual or cluttered scenes.

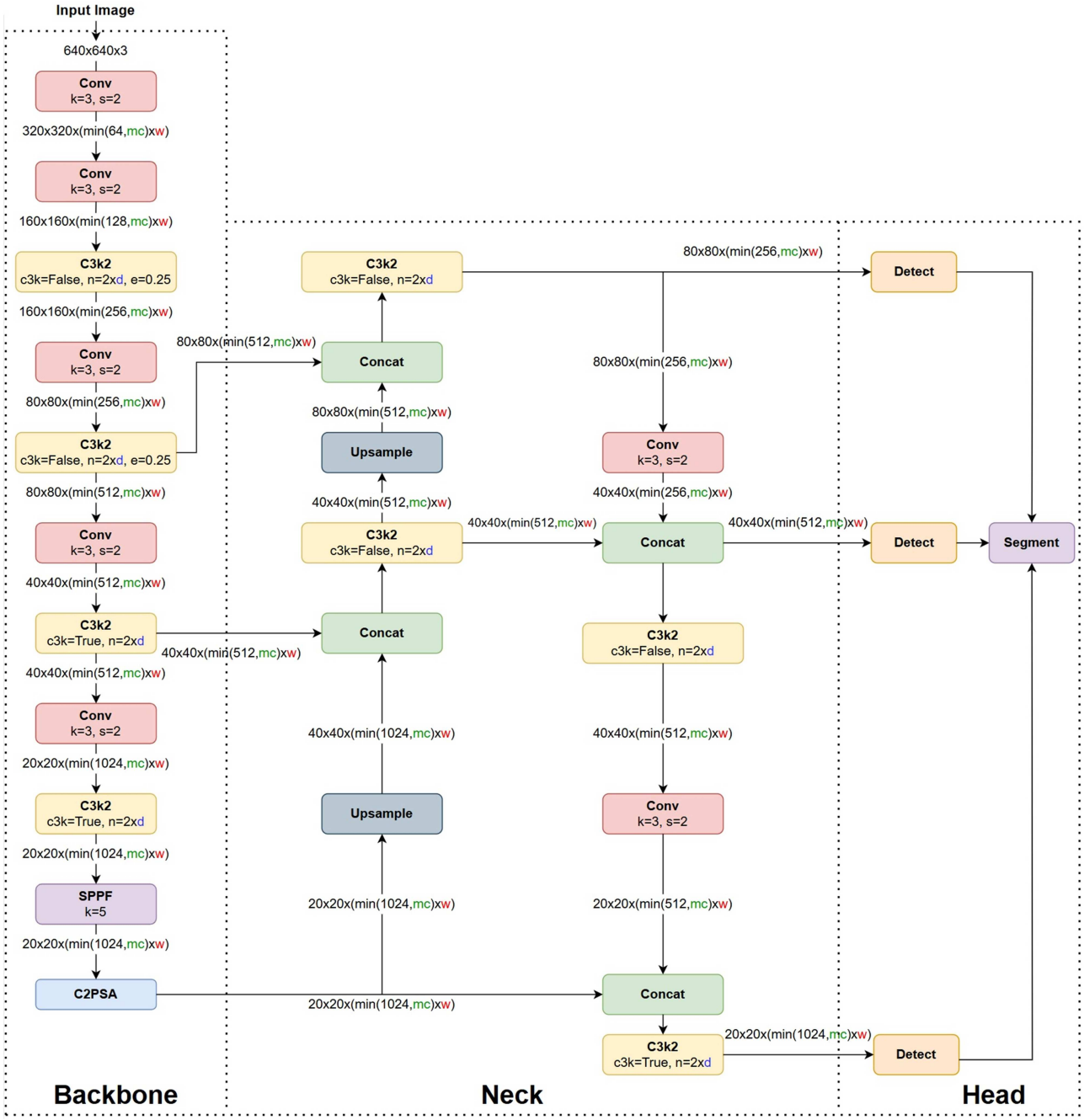
In order to determine the optimal model configuration for ATO tasks [26], we conducted a systematic comparison of 25 different configurations of YOLOv11. Each different alternative was a combination of a network architecture and input resolution, and we were therefore able to see the impact of different configuration choices on performance.

There are five YOLOv11 architecture versions that each try to balance speed, accuracy, and computational needs in a different manner. It is easy to choose the best model for a specific hardware setup or performance requirement:

1) YOLOv11n (nano) is the lightest version, appropriate for deployment on processing-limited devices or where real-time detection is essential.
2) YOLOv11s (tiny) remains tiny but improves detection accuracy and is an acceptable choice when there is some sacrifice in speed.
3) YOLOv11m (small) is a good balance of performance and use of resources and is an acceptable choice for normal usage.
4) YOLOv11l (large) offers higher accuracy because of a more complex model but requires more hardware.
5) YOLOv11x (extra-large) is the most precise one, meant for maximum performance, though it requires plenty of memory and processing capabilities.

Different versions of YOLOv11 differ in terms of complexity, and there are three parameters that control this. Depth multiple (d) modifies the depth of the network by varying the number

Figure 2
Architecture of the YOLO11 model



of layers—raising it makes it stronger in processing complicated patterns. Width multiple (w) controls the number of channels each layer can contain, thereby controlling the amount of information that the model processes at one time. The maximum channels (mc) parameter limits the number of channels the model can use to control memory usage and computation time. Table 1 specifies individual values of these parameters for all versions of the model.

Each of the five variations of YOLOv11 was trained and tested on each of five resolutions as input: 640 × 640, 960 × 960, 1280 × 1280, 1600 × 1600, and 1920 × 1920 pixels. By pairing each model variation with each resolution, we have a total of 25 different configurations on which we can systematically test how both architecture and input size affect performance.

3.3. Training procedure and evaluation metrics

We trained all 25 configurations of the YOLOv11 models on two NVIDIA RTX A5000 GPUs. The dataset was split into three subsets to ensure reliable evaluation: 70% for training, 15% for validation, and 15% for testing.

The training was done according to the YOLOv11 procedure. The optimizer set was stochastic gradient descent with Nesterov momentum. The parameters were an initial learning rate of $lr0$ = 0.01, a final learning rate of lrf = 0.01, a momentum of 0.937, and a weight decay of 0.0005. The batch size was set to 16. The models were trained to a maximum of 400 epochs. During training, they stopped early via the early-stopping rule [27], with

Table 1
YOLOv11 model variants and corresponding scaling parameters

Model variant	*d* (depth multiple)	*w* (width multiple)	mc (max channels)
n (nano)	0.50	0.25	1024
s (small)	0.50	0.50	1024
m (medium)	0.50	1.00	512
l (large)	1.00	1.00	512
x (extra-large)	1.00	1.50	512

a patience of 50 epochs. This was done to avoid overfitting based on the validation mean average precision (mAP) [28].

In addition to making the models more robust, the usual YOLOv11 techniques of mosaic, flip, scale (with 20% change), and photometric transformations (hue, saturation, value) were implemented for the data augmentation. The images were normalized with the ImageNet values of the mean and the standard deviation.

Model performance was tested using mAP. The principal assessment metric was mAP@0.5:0.95 because it evaluates localization and recognition correctly in diverse overlap thresholds [29]. It considers Intersection over Union (IoU) thresholds ranging between 0.5 and 0.95 [30].

We tracked mAP@0.5:0.95 on the validation set throughout training to guide the process and select the best checkpoint for each model variant. These top-performing checkpoints were then used on the held-out test set for reporting final results.

4. Results and Discussion

After the training process, we conducted a thorough assessment of all 25 YOLOv11 model configuration variants on the given test subset of the dataset. Each configuration represented a different model variant and input image resolution pair. To give an objective assessment of model performance, we used several key evaluation measures.

The measure of quality of the segmentations was mAP@0.5:0.95, which averaged over the over-segmented IoU thresholds from 0.5 to 0.95 at step size 0.05. This measure calculates the capability of models to segment objects properly while identifying the degree to which the objects are segmented and over which segmented areas are classified accordingly, and hence, they are a de facto standard for evaluating object detection and segmentation metrics. The larger the mAP values, the more accurate and reliable the segmentation results are.

Other than accuracy, we needed to compare the inference speed of all configurations, especially for real-time systems like automatic train control systems. We did this by comparing the average processing time to process one image on an NVIDIA RTX A5000 GPU. All experiments were executed with the same hardware and software environment to enable equitable comparisons. These are the figures that give us a clear indication of how each of these configurations would perform on challenging timing platforms.

We also measured the computational complexity of each design by estimating the theoretical floating-point operations (GFLOPs). The metric counts the amount of arithmetic computation for a one-pass forward pass through the network. GFLOPs is not a direct function of hardware optimization or memory access patterns but is a reasonable proxy for model complexity and the processing unit load anticipated.

All of the metrics of evaluation for the different configurations are listed in Table 2. This gives an easy-to-view and concise perception so that it becomes easy to compare the different configurations and recognize the best compromise between accuracy, speed, and computational cost.

One should realize that the inference time of the image is not always precisely equal to the theoretical computational complexity in GFLOPs. This is primarily a measure of the extraordinary character of the hardware on which the models are run. Even though GFLOPs gives a rough estimate of the number of operations that the model is performing in one forward pass, it does not consider optimizations like memory locality of access, hardware-level parallelization capability, or software-level optimizations like layer-level fusion and mixed precision.

Table 2
Comparison of YOLOv11 configurations

Model variant	Resolution	Layers	Parameters (M)	GFLOPs	Segmentation time (ms)	mAP 0.5–0.95
YOLOv11n (nano)	640 × 640	355	2.85	10.40	8.2	0.429
	960 × 960			23.41	14.5	0.562
	1280 × 1280			41.62	24.6	0.644
	1600 × 1600			65.03	36.5	0.682
	1920 × 1920			93.64	55.2	0.707
YOLOv11s (small)	640 × 640	355	10.10	35.69	10.7	0.480
	960 × 960			80.30	20.9	0.592
	1280 × 1280			142.76	35.3	0.667
	1600 × 1600			223.06	52.6	0.712
	1920 × 1920			321.20	80.9	0.729

(*Continued*)

Table 2
(*Continued*)

Model variant	Resolution	Layers	Parameters (M)	GFLOPs	Segmentation time (ms)	mAP 0.5–0.95
YOLOv11m (medium)	640 × 640	445	22.39	123.77	16.3	0.530
	960 × 960			278.49	32.1	0.615
	1280 × 1280			495.09	54.8	0.692
	1600 × 1600			773.58	84.0	0.731
	1920 × 1920			1113.96	125.0	0.747
YOLOv11l (large)	640 × 640	667	27.65	142.86	18.2	0.548
	960 × 960			321.43	38.9	0.636
	1280 × 1280			571.43	62.1	0.718
	1600 × 1600			892.87	94.6	0.756
	1920 × 1920			1285.73	137.4	0.772
YOLOv11x (extra-large)	640 × 640	667	62.10	319.97	27.0	0.560
	960 × 960			719.92	53.2	0.650
	1280 × 1280			1279.86	90.9	0.743
	1600 × 1600			1999.79	144.4	0.778
	1920 × 1920			2879.69	213.4	0.795

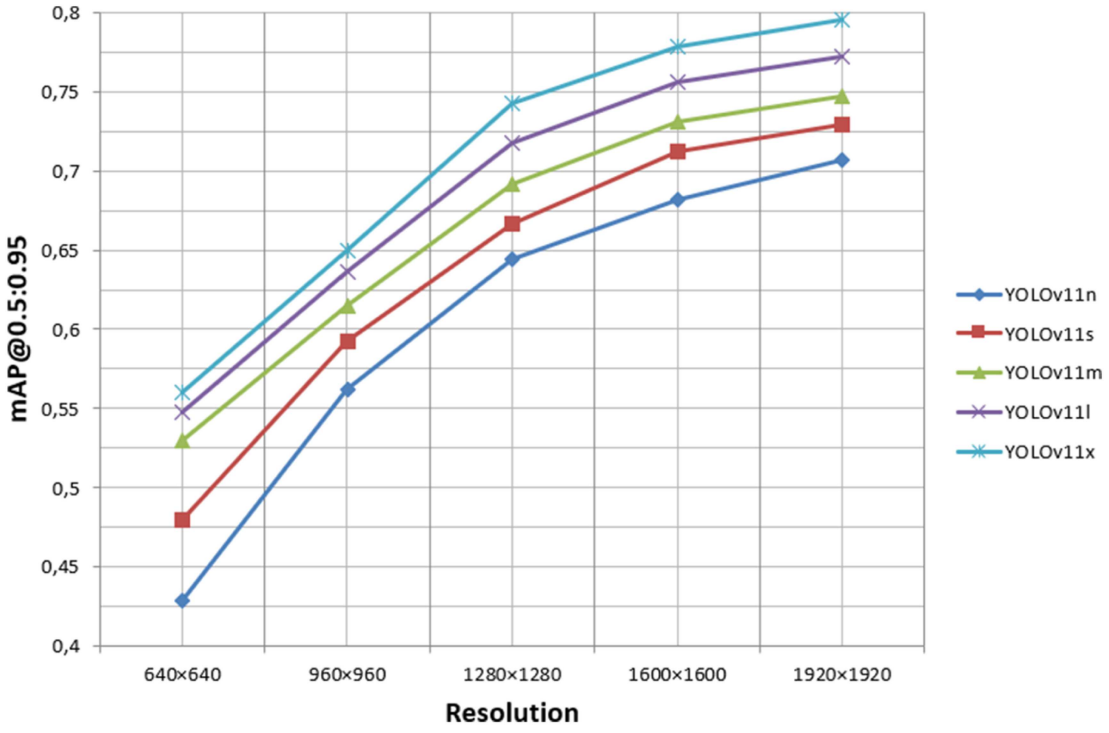
As an example, some higher GFLOPs YOLOv11 configurations still boasted faster inference times compared to more loaded ones because of better GPU core utilization and better parallel processing. Theoretical lower complexity models are not necessarily going to be better in practice if their construction does not optimize memory utilization or if they do not utilize available computational resources to their full capacity.

These results highlight the importance of model testing on the same hardware on which the models are to be run because theory-based benchmarks do not necessarily reflect actual performance.

To enable a clearer comparison of segmentation accuracy across different model configurations, we also produced a plot showing how segmentation accuracy (mAP@0.5:0.95) and input resolution are correlated. This visualization enables comparison of how different resolutions affect the performance of different YOLOv11 configurations in detection accuracy. The result plotted is shown in Figure 3.

The trends visualized in Figure 3 clearly demonstrate a positive correlation between input resolution and segmentation accuracy (mAP) for all model variants, with diminishing returns observed at the highest resolutions, especially for smaller

Figure 3
Impact of changing the input resolution on segmentation accuracy of various YOLOv11 models



models (n, s). This underscores that increasing resolution is an effective but computationally costly way to improve precision, crucial for detecting small infrastructure details.

Also, in order to have better insight into inference rates of different configurations, we graphically displayed how segmentation frame rate (frames per second, FPS) depends on input resolution, which has been tested and verified on an NVIDIA RTX A5000 GPU. It clearly demonstrates how increasing the input resolution impacts every model variant's real-time behavior. The graph can be found in Figure 4.

Figure 4 highlights the critical trade-off with inference speed. The frame rate (FPS) drops exponentially as resolution increases, particularly for the larger architectures (YOLOv11l, YOLOv11x). For instance, while YOLOv11n maintains real-time performance (more than 30 FPS) even at 1920 × 1920, YOLOv11x at the same resolution falls below 5 FPS. This visualization is key for system designers: it provides a direct map for selecting a configuration that meets both the minimum accuracy threshold and the real-time FPS requirement of a specific ATO system.

The comparison of the 25 YOLOv11 model configurations revealed some clear trends. As expected, increasing the input resolution always improved segmentation accuracy in all model configurations. For example, the YOLOv11n (nano) model improved from 0.429 mAP@0.5:0.95 at 640 × 640 to 0.707 mAP@0.5:0.95 at 1920 × 1920, and the YOLOv11x (extra-large) variant achieved the highest score of 0.795 mAP@0.5:0.95 at the highest resolution. But this accuracy gain was accompanied by a significant rise in computational complexity and inference time.

The computational cost, in the form of GFLOPs, and average frame-wise segmentation time both increased drastically with model capacity and input resolution. YOLOv11n, the lightest model, was the most time-efficient during processing, costing as low as 8.2 ms per frame at resolution 640 × 640. But YOLOv11x, with the highest accuracy, was much slower—going up to as high as 213.4 ms per frame at the highest resolution. The range, when measured in terms of GFLOPs, was extremely wide, from a minimum of 10.4 GFLOPs for the smallest configuration to an astounding 2880 GFLOPs for the largest and most computationally intensive model.

In general, the results place in the spotlight the accuracy vs efficiency trade-off: compact models like YOLOv11n and YOLOv11s are adequate to fit into real-time tasks for low-resource hardware, whereas larger models like YOLOv11l and YOLOv11x are to be used whenever maximum achievable accuracy is to be attained and computation capability cannot be made the priority.

All the configurations of the model generated visually correct and consistent segmentation results in different scenes, delineating infrastructure features and obstacles distinctly. An example of the same segmentation is present in Figure 5, where the YOLOv11x model's output is provided with an input resolution of 1920 × 1920.

The model accurately segments distinct instances of critical classes, such as railway tracks, vehicles, people, and traffic signs, with well-defined mask boundaries. This pixel-wise precision is superior to bounding-box detection, as it allows for exact calculation of an object's area and position relative to the train's clearance gauge. However, a closer qualitative examination also reveals common challenges: slight inaccuracies in the mask edges for complex-shaped objects and potential difficulties in segmenting objects that are far away or partially occluded. These observations align with the quantitative mAP scores (less than 0.8) and point to areas for further improvement, such as refining the segmentation head loss function or incorporating temporal information from video sequences.

Figure 4
Impact of input resolution on processing speed (frame rate) of YOLOv11 models

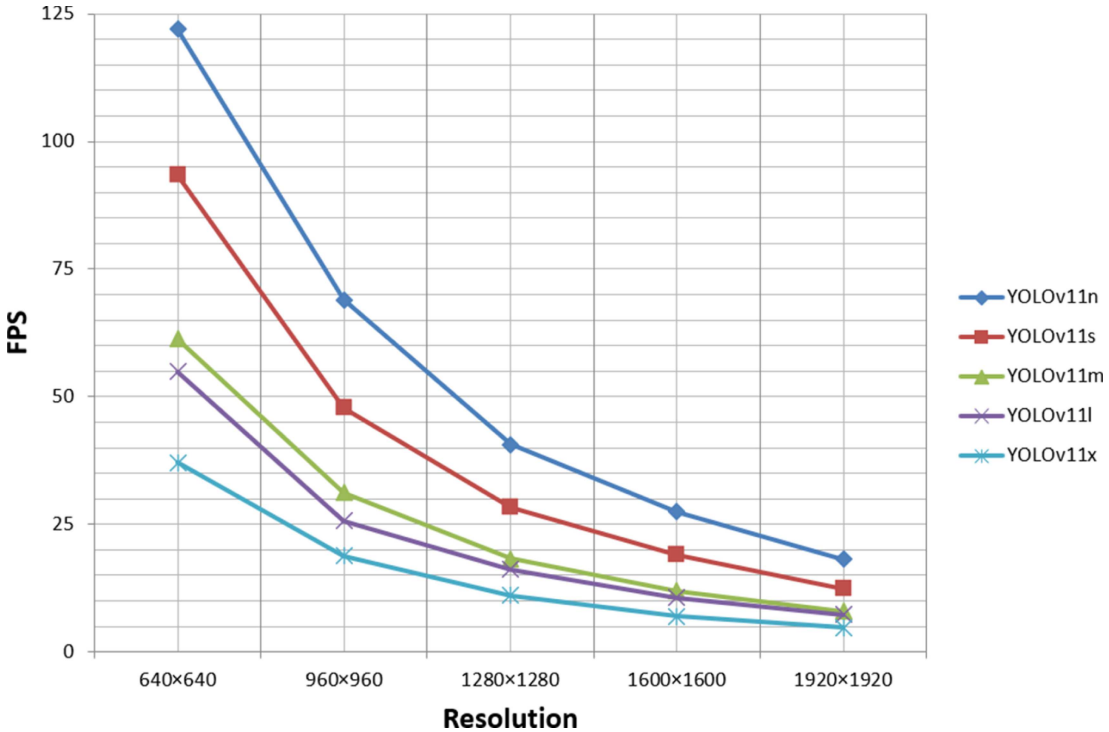
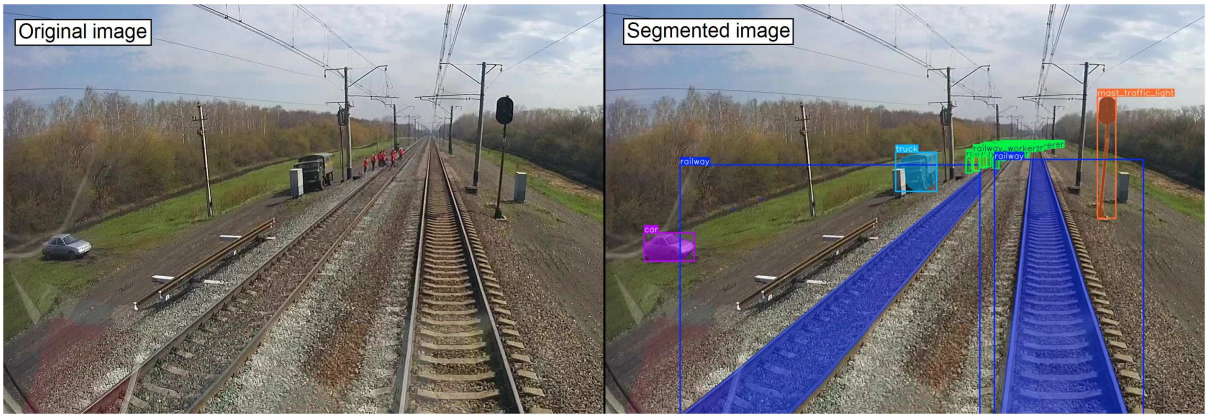


Figure 5
Example of instance segmentation for railway infrastructure and obstacle detection



5. Conclusion and Future Work

This paper shows instance segmentation with YOLOv11 to work well for railway infrastructure and possible hazard detection within ATO systems. We tested 25 model variants varying network depth and input resolution. These results show that, thanks to architectural upgrades in accuracy and quality of segmentation masks with respect to previous versions, YOLOv11 is able to provide a versatile and powerful framework that can be finely tuned to match all the strict and varied requests coming from ATO systems, going from lightweight, real-time perception on limited hardware to accurate, fine-grained scene analysis for crucial decision-making.

The tests also confirm that YOLOv11 offers reliable, precise scene understanding complemented by the necessary speed for real-time autonomous systems. Such features make it a promising contender for ATO systems, where high-quality perception is important for intelligent and safe train operation, especially at higher automation levels (GoA3 and GoA4). By way of accurate object and infrastructure localization, YOLOv11 helps aid improved decision-making, which is vital for features like obstacle avoidance, track monitoring, and dynamic route calculation.

While the results are encouraging, several important limitations of this work highlight clear paths for future development. A primary challenge is hardware dependency. The reported inference speeds for larger configurations, especially YOLOv11x at high resolutions, rely on powerful server-grade GPUs like the NVIDIA RTX A5000. For actual deployment onboard trains, we must adapt these models to far more constrained embedded systems, such as NPU, Jetson, or TPU platforms, which have strict limits on power and physical space. This transition will require a thorough reassessment of the accuracy, latency, and computational cost trade-offs we observed. A key next step will be to build a detailed multi-objective Pareto front that balances accuracy, latency, and power use specifically for these target hardware platforms. Techniques like kernel fusion and INT8 quantization will be crucial in this search for viable onboard configurations.

Another area for improvement concerns our dataset. Although diverse, its 20,000 images are primarily from Russian railways. This geographical focus means the model may not generalize well to other networks with different signage, infrastructure, or operational standards. To build a more robust system, the dataset should be expanded to include examples from a wider variety of regions and, critically, more rare edge-case scenarios like complex obstructions or extreme weather. Developing efficient adaptation protocols will be key to enabling this broader generalization.

Furthermore, while we included various environmental conditions, a deeper, condition-specific analysis is needed. We plan to conduct granular benchmarking to measure performance degradation precisely in severe adverse conditions, such as heavy rain, dense fog, or pitch-dark nights. This analysis will inform the creation of better data augmentation strategies or even conditionally adaptive models that maintain reliability when it matters most.

On the architectural front, there is room to enhance model robustness. Investigating more advanced neck designs, like specific FPN variants, and incorporating multi-scale attention mechanisms could significantly improve handling of partial occlusions and dense, cluttered scenes. Additionally, integrating domain knowledge, such as topological priors from rail graphs, could help constrain predictions and improve the semantic consistency of masks for downstream planning modules.

Finally, the choice of mask decoder itself presents an open research question. Exploring next-generation decoders, including those based on transformers or dynamic kernels, could reshape the accuracy–latency trade-off, particularly at the high resolutions needed for detecting distant objects. Tackling these interconnected challenges, from hardware deployment and data diversity to architectural refinement, is a pressing necessity for advancing toward a certifiable, real-world perception module capable of supporting fully automated GoA4 train operation.

Ethical Statement

This study does not contain any studies with human or animal subjects performed by the author.

Conflicts of Interest

The author declares that he has no conflicts of interest to this work.

Data Availability Statement

Data available on request from the corresponding author upon reasonable request.

Author Contribution Statement

Vladimir A. Fedorov: Conceptualization, Methodology, Software, Validation, Formal analysis, Investigation, Resources, Data curation, Writing – original draft, Writing – review & editing, Visualization, Supervision, Project administration.

References

[1] Yin, J., Tang, T., Yang, L., Xun, J., Huang, Y., & Gao, Z. (2017). Research and development of automatic train operation for railway transportation systems: A survey. *Transportation Research Part C: Emerging Technologies*, *85*, 548–572. https://doi.org/10.1016/j.trc.2017.09.009

[2] Khalil, A., & Mamdouh, M. (2022). Investigating driver/ATC performance impacting railway train operation safety and delays: A field study on Egyptian National Railways. *Engineering Research Journal*, *51*(3), 26–41. https://doi.org/10.21608/ERJSH.2022.245601

[3] Barruffo, L., Caiazzo, B., Petrillo, A., & Santini, S. (2024). A GoA4 control architecture for the autonomous driving of high-speed trains over ETCS: Design and experimental validation. *IEEE Transactions on Intelligent Transportation Systems*, *25*(6), 5096–5111. https://doi.org/10.1109/TITS.2023.3338295

[4] Tonk, A., Chelouati, M., Boussif, A., Beugin, J., & El Koursi, M. (2023). A safety assurance methodology for autonomous trains. *Transportation Research Procedia*, *72*, 3016–3023. https://doi.org/10.1016/j.trpro.2023.11.849

[5] Kumar, P., Kumar, S., Srivastava, R., Singh, D., Mir, W. A., Pal, S., . . . , & Yadav, P. (2023). A review on automatic protection system and risk mitigation in railways. In *2023 5th International Conference on Advances in Computing, Communication Control and Networking*, 1653–1659. https://doi.org/10.1109/ICAC3N60023.2023.10541341

[6] Barruffo, L., Caiazzo, B., Petrillo, A., & Santini, S. (2024). A GoA4 control architecture for the autonomous driving of high-speed trains over ETCS: Design and experimental validation. *IEEE Transactions on Intelligent Transportation Systems*, *25*(6), 5096–5111. https://doi.org/10.1109/TITS.2023.3338295

[7] Tagiew, R., Leinhos, D., von der Haar, H., Klotz, C., Sprute, D., Ziehn, J., . . . , & Klasek, P. (2023). Sensor system for development of perception systems for ATO. *Discover Artificial Intelligence*, *3*(1), 22. https://doi.org/10.1007/s44163-023-00066-4

[8] Khanam, R., & Hussain, M. (2024). Yolov11: An overview of the key architectural enhancements. *arXiv Preprint: 2410.17725*

[9] Solunke, B. R., & Gengaje, S. R. (2023). A review on traditional and deep learning based object detection methods. In *2023 International Conference on Emerging Smart Computing and Informatics*, 1–7. https://doi.org/10.1109/ESCI56872.2023.10099639

[10] Fedorov, V. A. (2024). Recognizing railway infrastructure using CNN and stereoscopic vision. In *2024 International Russian Smart Industry Conference*, 13–18. https://doi.org/10.1109/SmartIndustryCon61328.2024.10516208

[11] Issaoui, H., ElAdel, A., & Zaied, M. (2024). Object detection using convolutional neural networks: A comprehensive review. In *2024 IEEE 27th International Symposium on Real-Time Distributed Computing* (pp. 1–6). https://doi.org/10.1109/ISORC61049.2024.10551342

[12] Jiang, P., Ergu, D., Liu, F., Cai, Y., & Ma, B. (2022). A review of Yolo algorithm developments. *Procedia Computer Science*, *199*, 1066–1073. https://doi.org/10.1016/j.procs.2022.01.135

[13] Yanan, S., Hui, Z., Li, L., & Hang, Z. (2018). Rail surface defect detection method based on YOLOv3 deep learning networks. In *2018 Chinese Automation Congress*, 1563–1568. https://doi.org/10.1109/CAC.2018.8623082

[14] Wang, H., Pei, H., & Zhang, J. (2022). Detection of locomotive signal lights and pedestrians on railway tracks using improved YOLOv4. *IEEE Access*, *10*, 15495–15505. https://doi.org/10.1109/ACCESS.2022.3148182

[15] Liu, W., Wang, Z., Zhou, B., Yang, S., & Gong, Z. (2021). Real-time signal light detection based on Yolov5 for railway. *IOP Conference Series: Earth and Environmental Science*, *769*(4), 042069. https://doi.org/10.1088/1755-1315/769/4/042069

[16] Muhammad, K., Hussain, T., Ullah, H., del Ser, J., Rezaei, M., Kumar, N., . . . , & de Albuquerque, V. H. C. (2022). Vision-based semantic segmentation in scene understanding for autonomous driving: Recent achievements, challenges, and outlooks. *IEEE Transactions on Intelligent Transportation Systems*, *23*(12), 22694–22715. https://doi.org/10.1109/TITS.2022.3207665

[17] He, K., Gkioxari, G., Dollár, P., & Girshick, R. (2017). Mask R-CNN. In *Proceedings of the IEEE International Conference on Computer Vision*, 2980–2988. https://doi.org/10.1109/ICCV.2017.322

[18] Manikanta, M. S., Siva, K., Krishna, M. S., David, N., & Bhupal, M. R. (2025). Deep learning based object detection using Mask RCNN. *IJSAT-International Journal on Science and Technology*, *16*(1). https://doi.org/10.71097/IJSAT.v16.i1.2745

[19] Fedorov, V. A. (2024). Railway infrastructure detection based on YOLOv8 with NPU acceleration. *Doklady Mathematics*, *110*(1), S42–S48. https://doi.org/10.1134/S1064562424601951

[20] Fedorov, V. A. (2023). Railway infrastructure instance segmentation based on convolutional neural networks. In *2023 International Russian Automation Conference*, 443–447. https://doi.org/10.1109/RusAutoCon58002.2023.10272908

[21] Nandal, P., Bohra, N., Mann, P., & Das, N. N. (2025). YOLOv11 with transformer attention for real-time monitoring of ships: A federated learning approach for maritime surveillance. *Results in Engineering*, *27*, 106297. https://doi.org/10.1016/j.rineng.2025.106297

[22] Hussain, M. (2023). YOLO-v1 to YOLO-v8, the rise of YOLO and its complementary nature toward digital manufacturing and industrial defect detection. *Machines*, *11*(7), 677. https://doi.org/10.3390/machines11070677

[23] Chen, C., Wang, F., Yang, M., Qin, Y., & Bai, Y. (2024). An efficient lightweight railway track segmentation network for resource-constrained platforms with TensorRT. *Intelligent Transportation Infrastructure*, *3*, 009. https://doi.org/10.1093/iti/liae009

[24] Xie, J., Pang, Y., Nie, J., Cao, J., & Han, J. (2022). Latent feature pyramid network for object detection. *IEEE Transactions on Multimedia*, *25*, 2153–2163. https://doi.org/10.1109/TMM.2022.3143707

[25] Qiu, Z., Huang, X., Sun, Z., Li, S., & Wang, J. (2025). GS-YOLO-Seg: A lightweight instance segmentation method for low-grade graphite ore sorting based on improved YOLO11-seg. *Sustainability*, *17*(12), 5663. https://doi.org/10.3390/su17125663

[26] Bochmann, P., & Jaekel, B. (2022). Measures and methods for the evaluation of ATO algorithms. *Applied Sciences*, *12*(9), 4570. https://doi.org/10.3390/app12094570

[27] Mahasin, M., & Dewi, I. A. (2022). Comparison of CSPDarkNet53, CSPResNeXt-50, and EfficientNet-B0 backbones on YOLO v4 as object detector. *International Journal of Engineering, Science and Information Technology*, *2*(3), 64–72. https://doi.org/10.52088/ijesty.v2i3.291

[28] Rainio, O., Teuho, J., & Klén, R. (2024). Evaluation metrics and statistical tests for machine learning. *Scientific Reports*, *14*(1), 6086. https://doi.org/10.1038/s41598-024-56706-x

[29] Lin, T.-Y., Maire, M., Belongie, S., Hays, J., Perona, P., Ramanan, D., . . . , & Zitnick, C. L. (2014). Microsoft COCO: Common objects in context. In *Computer Vision – ECCV 2014: 13th European Conference*, 740–755. https://doi.org/10.1007/978-3-319-10602-1_48

[30] Rahman, M. A., & Wang, Y. (2016). Optimizing intersection-over-union in deep neural networks for image segmentation. In *Advances in Visual Computing: 12th International Symposium* (pp. 234–244). https://doi.org/10.1007/978-3-319-50835-1_22

How to Cite: Fedorov, V. A. (2026). Instance Segmentation for Infrastructure and Obstacle Detection in Automatic Train Operation Systems. *Artificial Intelligence and Applications*. https://doi.org/10.47852/bonviewAIA62026000