Comparing Object Detection, Instance Segmentation, and Semantic Segmentation for Automated Vegetation Detection in Railroad Systems

Mingyan Liu*, Van Trung Le†, Hwapyeong Song†, Advay Chandramouli‡, Husnu S. Narman†, Ammar Alzarrad†
* Smith College, iliu32@smith.edu
† Marshall University, {le57, song24, narman, alzarrad}@marshall.edu
‡The University of Texas at Dallas, advay.chandramouli@utdallas.edu

***Abstract*—Vegetation management is a critical component of railway maintenance, directly impacting operational safety, infrastructure longevity, and regulatory compliance. Overgrown vegetation can obscure track visibility, interfere with inspection routines, and degrade track conditions which posing risks to both freight and passenger operations. This paper explores automated vegetation detection within railroad environments using three deep learning models: YOLOv8, U-Net, and DeepLabv3+. The YOLOv8 model was trained using both object detection bounding boxes and instance segmentation masks on a domain-specific dataset comprising around 500 railroad images captured under real-world deployment conditions. In contrast, the semantic segmentation models, U-Net and DeepLabv3+, were trained on a broader dataset of more than 9,800 images representing general vegetation contexts. Comparative analysis reveals that DeepLabv3+ consistently outperforms the other models in accurately identifying vegetation, demonstrating higher precision, recall, and segmentation quality. These findings highlight the effectiveness of semantic segmentation, particularly DeepLabv3+, for detecting irregular, organic features such as vegetation in complex railway settings.**

***Index Terms*—Railroad, Vegetation, Machine Learning**

I. Introduction

There are more than 140,000 miles of railroads in the United States that support the transportation of approximately 1.5 billion tons of goods and 28.6 million passengers annually [1]. As a major economic driver, the railroad industry generated $233.4 billion in 2023 alone [2]. In order to maintain consistent operations, monitoring and regulating railroad defects that could affect the safety and efficiency of railroad transportation is one of the most important maintenance tasks.

One of the most important aspects of defect control is the maintenance of overgrown vegetation along and within the railroad. Not only does vegetation interfere with railroad personnel's track examination routines but it also contributes to track deterioration and slippery rails [3]. According to U.S. regulation [4], vegetation on railroad must be contained so that it does not cause, obstruct visibility, and functioning of the communication lines.

Traditionally, railroad inspection is conducted through visual assessment by human safety inspectors either on site or through video footage. However, such a process is labor intensive, costly, and prone to human errors. As demonstrated by Nyberg et al. [3], human assessments tend to be volatile and unreliable. These are mainly due to raters having differing evaluation of vegetation extent and condition, thus leading to non-uniform and subjective conclusions.

The growing complexity of modern railroad networks, combined with the advancement of technology, has driven the railroad industry to develop Machine Learning (ML) systems to supplement the inspection and maintenance process to minimize errors and cost. Around 58% of papers published on the usage of Artificial Intelligence (AI) in the railroad industry focuses on the subdomain of Maintenance and Inspection [5]. However, while machine learning techniques for detecting structural defects such as missing bolts, cracks, and track misplacement have been tested and applied, their applications in detecting irregular, organic defects such as vegetation remains less developed. Vegetation poses a unique problem for ML models as their irregular shape, color, density, and size varies based on the season and terrain. This irregularity demands flexible ML models and a large, diverse dataset that is capable of reflecting the variation in vegetation growth.

This paper *aims* to automate vegetation detection within railroad tracks using three deep learning models: YOLOv8 [6], U-Net [7], and DeepLabv3+ [8].

The research *objectives* are as follows.

- Training a YOLOv8 model to identify vegetation within railway tracks: The accuracy of two different versions of the model, one trained on object detection bounding boxes and one trained on segmentation masks, will be compared and evaluated. Finally, binary rail masks produced by YOLOv8's segmentation feature will be overlaid on the original images to determine whether each vegetation patch falls inside or outside of our region of interest (ROI).
- Training U-Net and DeepLabv3+ models to identify vegetation in railroad tracks: To improve generalization, the training dataset includes not only railroad images, but also images from diverse environments containing vegetation (e.g., cityscapes, roads, and earth maps). The resulting binary masks are generated through semantic rather than instance segmentation.

The *key contributions* of this paper are as follows.

- Training and deploying YOLOv8 for object detection and instance segmentation in comparison with semantic segmentation models (U-Net and DeepLabv3+) to increase accuracy and speed of railroad vegetation detection in a domain where irregular, organic defects, such as vegetation, remain underexplored.
- Creating a domain-specific, custom-annotated dataset for vegetation detection on railroad with more around 500 railroad images grounded in real deployment conditions. This specialized dataset will be compared with a dataset of 3,857 images capturing vegetation in different environment not limited solely to the railroad.

The paper is organized as follows. Section II presents a review of related work in the field. Section III outlines the deep learning models employed in this study and describes the data collection and training procedures. Section IV analyzes metrics and results produced by trained deep learning models. Section Section IV contains the conclusion and possible considerations for future work.

II. Literature Review

A. Automating Vegetation Detection in Railroad

Light Detection and Ranging (LiDAR), a remote sensing technology that can map out vegetation height and density, operates by emitting laser pulses towards a target and calculating the precise distance based on the measured two-way travel time of the pulse.

The data is recorded by either the Discrete Return LiDAR System, which measures discrete data points at the peaks of the waveform curve, or the full-waveform LiDAR system, which captures the entire waveform of the returning pulse, enabling it to make further variation among its targets, such as distinguishing tree branches from leaves. The resulting clouds of points are then projected as a 3D spatial map. In the field of vegetation detection, LiDAR is most commonly used to measure and quantify forest canopy [9].

LiDAR provides accurate estimates of vegetation height and density without interference from hue, saturation, brightness, or poor weather conditions. However, this technology is also expensive to operate compared to camera-based solutions and requires significant computational resources to process the large volume of 3D data it generates.

Instead, this study approaches vegetation detection through RGB images collected via a camera. This method is generally more cost-effective, capable of capturing color and texture useful for identifying vegetation types and conditions, and is more intuitive for human operators to interpret without specialized training or hardware.

B. Computer Vision

Min et al. [10] created a real-time computer vision system for surface rail defects detection, which builds upon prior research on evaluating the accuracy of surface defects detection systems with the goal of minimizing computational burden. The model uses a rapid target area location method based on the H (Hue) value of color images, which demonstrates strong adaptability to varying light conditions. The portable prototype achieved a real-time detection speed of 2 m/s, with processing times up to 245.61 ms per picture .

Xu et al. [11] built the AED-YOLO model, which is designed to improve the detection accuracy of small-sized components, such as fastener nuts and bracing wire, by integrating Improved Bidirectional Feature Pyramid Network (Hor-Bi-FPN) and Asymmetrically Effective Decoupled Head (AED-Head). This model achieved a mean Average Precision (mAP) of 93.5%, outperforming YOLOv3 with an improvement of 1.8% and YOLOv5 with an improvement of 2.3%.

Although You Only Look Once (YOLO) has been widely applied in vegetation detection, limited research exists in its application in railroad maintenance. Gautam et al. [12] utilized YOLOv5 on drone-acquired images to detect invasive Siam weed in natural environments, reaching an F-1 score of 0.88. Notably, their study also highlighted that the complexity of YOLO models does not noticeably increase its performance. This finding aligns with the research by Andrew et al. [13]. In his work using YOLO to detect railroad cracks, YOLOv5 and YOLOv9 had higher performance compared to most updated versions such as YOLOv10x and YOLOv10n, with YOLOv5 achieving an F-1 score of 0.92.

Kholiya et al. [14] built a YOLO model for automatic plant detection and counting in agriculture based on a diverse dataset in Roboflow. However, their research did not disclose quantitative performance metrics such as F-1 and mAP scores to reflect the effectiveness of such a method.

Gupta et al. [15] utilized machine vision to create binary masks of vegetation and railroad tracks to identify the location and size of vegetation patches. However, this model was not evaluated quantitatively. Nyberg et al. [3] similarly applied color segmentation in the HSV (Hue, Saturation, and Value) space to calculate plant cover.

C. Existing Limitations

Despite the development of machine vision and machine learning techniques in the detection and maintenance of railroads, there remain limitations when it comes to vegetation classification and identification. At present, the majority of research in railroad AI focuses on structural defects such as missing bolts, cracks, and faults. These structural defects are relatively uniform in appearance compared to organic defects, such as overgrown vegetation, which vary significantly based on hue, density, and shape.

Furthermore, many of the current machine learning models, including YOLO variants, require training on a large, diverse dataset to optimize its performance. As Gautam et al. [12] found, a model trained with 1,000 images is capable of obtaining more reliable results. However, publicly available vegetation datasets specific to the railroad context are extremely scarce. Many previously established ML models are trained with agricultural or terrain-based images which may fail to transfer their performance under real-world railroad-related limitations such as debris, background clutter, and

varying lighting conditions. Early efforts, such as Gupta et al.'s [15], which involved 35 images, may not reflect the full needs of the machine model. Nyberg et al. [3] stated that future works should expand upon this methodology of taking pictures of track images manually, which this study seeks to remedy through an automated video/image setup so that a large quantity of diverse images may be automatically collected, reducing the data collection burden.

III. Methodology

A. Deep Learning Models

1) YOLOv8: YOLO is a real-time object detection system first introduced in 2015 by Redmon et al. [16].Before the introduction of YOLO, the most commonly used object detection models were two-stage detectors based on convolutional neural networks (CNN), such as R-CNN and Fast R-CNN. These detectors perform detection in two stages. First, generating regions proposals using the Region Proposal Network (RPN). Second, passing each proposed region to additional CNN layers to perform classification and bounding box regression. While this method is able to produce results with high accuracy, its complexity requires a large amount of computational resources and therefore struggles to meet the demanded processing speed for real-time deployments. On the other hand, single-stage detectors, such as YOLO, combine both steps, streamlining the object detection process by identifying class probabilities and bounding box coordinates in a single pass over the inputted image.

YOLOv8 was selected for this research due to several key considerations.

- Real-Time Detection Needs: The research on detecting vegetation within railroads relies on analyzing frame-based data where inference speed is critical. As a tested and well-documented model, YOLO remains the fastest and most reliable single-stage pipeline available.
- Proven accuracy within railroad defect detection: Prior research and applications of YOLO on structural railroad defects, such as cracks [13], has demonstrated high precision and recall. By building upon previously established conclusions, this research seeks to compare whether the model's performance remains effective in irregular defects such as vegetation.
- Comparing Object Detection vs. Segmentation: An additional factor motivating the selection of YOLO is its ability to perform both object detection and segmentation. YOLO's ability to support both functions within one framework enables direct comparison of these approaches on the same dataset, ensuring that the methodology captures the strengths of each and determines the most effective method for vegetation detection.

2) U-Net and DeepLabv3+: The original U-Net is a convolutional neural network designed for biomedical image segmentation, featuring a symmetric U-shaped architecture with an encoder that captures context and a decoder that restores spatial resolution, connected by skip connections to preserve fine details. Its key innovation is the skip connections, which solved the loss of spatial information during downsampling, enabling precise boundary segmentation.

DeepLabv3+ is a deep learning model for semantic segmentation that focuses on understanding both the overall context of an image and the fine details of object boundaries. It uses atrous (dilated) convolutions to capture features at multiple scales without reducing image resolution, and an Atrous Spatial Pyramid Pooling (ASPP) module to combine information from different receptive fields. To improve accuracy along edges, it includes a decoder module that refines the segmentation map, making boundaries sharper and more precise. Additionally, it applies depthwise separable convolutions to reduce computational cost while maintaining high performance.

The primary motivation for selecting U-Net and DeepLabv3+ is to compare how different deep learning models designed for semantic segmentation perform on the same task alongside an instance segmentation model such as YOLOv8. Furthermore, these models have different modern and complex architectures. U-Net employs a symmetric encoder-decoder structure with skip connections. DeepLabv3+ leverages atrous convolutions and multi-scale context aggregation through ASPP and a decoder module. YOLOv8 for segmentation extends the YOLO architecture by predicting both object bounding boxes and pixel-level masks in a single forward pass. By comparing these architectures, the more efficient one for vegetation detection can be identified for further implementation.

B. Data Preparation

1) Railroad Vegetation Dataset: The hardware equipment is attached to the vehicle at the site location and operates along the rail at various speeds to collect data. For this dataset, the model was trained on images collected at 5, 10, and 15 mph. Data were recorded using a Dell Rugged laptop with a built-in GPS.

The images used to train this model were captured by an overhead Intel RealSense D435 camera at 640×480 resolution (30 fps). The railway videos were recorded through the custom GUI software and saved as video files.

The collected video footage was converted into PNG frames. Using Roboflow, 200 images were manually annotated with object detection bounding boxes and segmentation masks in two categories: vegetation and railway. These annotations were then used to train a Roboflow auto-annotation model to streamline the labeling process. Around 500 images from field footage were labeled twice using two different methods: object detection bounding boxes and segmentation masks.

SAM was explored as an instance segmentation tool to generate vegetation masks and convert them into bounding boxes for automated labeling. However, SAM's segmentation masks struggled to distinguish vegetation from ballast, limiting its effectiveness.

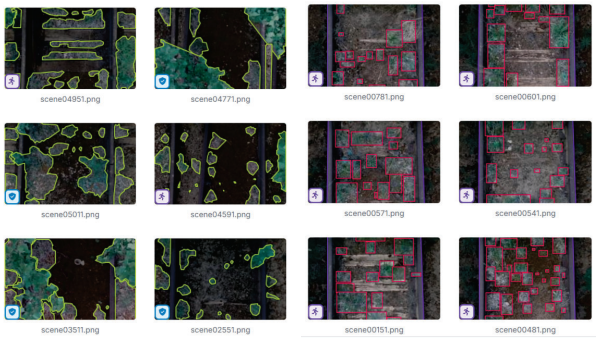

Fig. 1. Example of segmentation dataset (left) and object detection dataset (right).

2) General Vegetation Dataset: Besides the original dataset, five additional datasets forked from Roboflow are used: cvr [17], Cityscapes-external-v1 Computer Vision Dataset [18], vegetation Computer Vision Dataset [19], vegetation Computer Vision Model [20], and Interval-Tree-Mapillary_v4 Computer Vision Model [21]. These datasets include images of vegetation from various environments, not just railroads. The underlying motivation is that vegetation shares common visual characteristics across environments, generally appearing green and often having unclear boundaries. Therefore, when preparing the training dataset, it is not necessary for the vegetation to be specific to railroads. Images from various environment can be used. This approach can help the model generalize better to diverse vegetation scenarios.

Image augmentation was applied to the combined dataset, including:

- Resizing: 512×512 pixels
- Rotation: 90° clockwise, 90° counter-clockwise, and upside down
- Cropping: 0% minimum zoom, and 20% maximum zoom
- Noise: Up to 0.22% of pixels

The final dataset contains 9,865 images, with 9,012 images (91%) used for training and 853 images (9%) for validation. For the testing set, real captured images were used. Figure 2 illustrates some example images along with their corresponding masks.

C. Training

1) YOLOv8: To restrict vegetation detection to the railway region of interest, a YOLOv11 model was trained on 500 images containing railway bounding boxes.

The first YOLOv11 training attempt used 343 training images (69%), 40 validation images (8%), and 117 testing images (23%), which produced inconsistent results. In the second attempt, the dataset was split into 343 training images (69%) and 157 validation images (31%), with testing applied on live footage. This configuration yielded more accurate detections, as shown in Figure 3.

The resulting railway segmentation masks are exported as binary images for subsequent masking as shown in Figure 4.

Fig. 2. Examples of original images (top row) and their corresponding labeled images (bottom row).

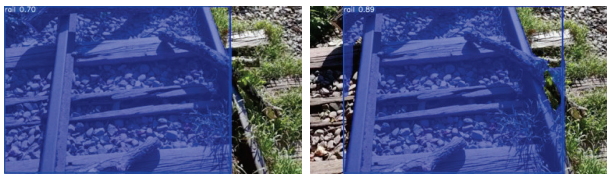

Fig. 3. YOLOv11 first attempt at training (left) vs YOLOv11 second attempt at training with more validation images (right).

Applying this binary mask as part of the post-processing step would be able to filter out predictions falling outside of the ROI, reducing false negatives.

Fig. 4. Binary Mask of Railway

Two variants of YOLOv8 were trained. The first one is based on the object detection bounding boxes dataset, while the second one relies on the segmentation masks dataset. The object detection variant (YOLOv8n) and the segmentation variant of the model (YOLOv8n-seg) were used with pre-trained weights initialized from the COCO dataset. Each of the models was trained for 500 epochs with a batch size of 32 and an image resolution of 640×640. Adamax optimizer was used with an initial learning rate of 0.01, cosine learning rate scheduling disabled, and a warmup of around 2.66 epochs. Default YOLOv8 loss functions, which combine bounding box regression, classification, and mask loss, were used. Early stopping was applied with a patience of 100 epochs. For the object detection model, early stopping occurred at epoch 149. For the segmentation model, the AdamW optimizer with the same parameters, and early stopping occurred at epoch 89.

Augmentations, such as mosaic and horizontal flip, were enabled by default in YOLOv8, while custom augmentations (hue, saturation, blur, and noise) were tested but removed due to a decrease in the model's performance.

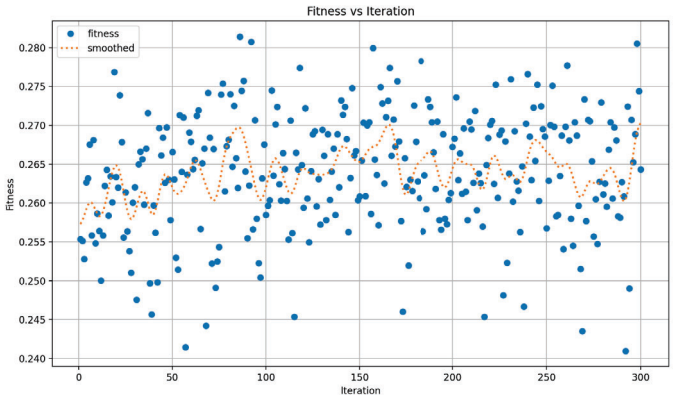

Fig. 5. Fitness vs Iteration graph as a result of hyperparameters tuning

An attempt to tune the models' hyperparameters was unsuccessful and proved to decrease the model's performance. As demonstrated in Figure 5, the fitness values fluctuate significantly in the graph and there is no evident upward trend. This signifies that tuning the hyperparameters has led to little model improvement, or model is insensitive to hyperparameters tuning.

2) U-Net: To train U-Net, BCEWithLogitsLoss was used because it combines a sigmoid activation with binary cross-entropy loss, which is well-suited for pixel-wise binary segmentation tasks. Adam optimizer was employed with a learning rate of 1×10^{-5} and a weight decay of 1×10^{-8}. A learning rate scheduler (`ReduceLROnPlateau`) was applied to reduce the learning rate by a factor of 0.5 if the validation loss did not improve for 5 epochs. Early stopping, monitoring the vegetation class IoU, was also used with a patience of 10 epochs and a minimum delta of 0.001 to prevent overfitting. The model was scheduled to train for 100 epochs, but training stopped at epoch 43 due to early stopping, as the vegetation class IoU did not improve. Unlike YOLOv8, the training of U-Net utilized a dataset containing vegetation from various environments.

3) DeepLabv3+: The same training parameters, including the loss function, optimizer, learning rate scheduler, early stopping, and number of epochs, that were used for U-Net were also applied to train DeepLabv3+. Unlike U-Net, DeepLabv3+ was not stopped early by the early stopping criterion and was trained using general vegetation dataset. Furthermore, the DeepLabv3+ model uses an EfficientNetV2_S_ImageNet backbone, meaning that the backbone is an EfficientNetV2_S model pre-trained on the ImageNet dataset. With this backbone, the model does not have to learn low-level feature representations from scratch, allowing it to focus on learning high-level features specific to vegetation segmentation.

IV. Results and Discussion

A. YOLOv8

TABLE I
YOLOv8 Best Score Comparison for Object Detection and Segmentation

Metric	Object Detection	Segmentation
F1	0.69	0.72
Precision	1.00	1.00
Recall	0.88	0.86
mAP@0.5	0.68	0.73

Evaluation metrics for object detection and segmentation are summarized in Table I. Although the F-1 score does not differ significantly between the models, the segmentation model achieved its optimal F1-score of 0.72 at a higher confidence threshold than the detection model. While this difference does not inherently signify superior performance, it suggests that the segmentation model's predictions tended to have higher confidence. Notably, the precision values for both models reached a perfect score of 1.000, reflecting a high level of accuracy in positive predictions. The object detection model achieved a recall of 0.88, slightly higher than the segmentation model's 0.86. This indicates that the object detection model is slightly more sensitive in identifying all vegetation patches, though this may also produce more false positives. Furthermore, the YOLOv8 segmentation model achieved a higher mAP@0.5 value of 0.73 compared to the object detection model value of 0.68, indicating an improved overall accuracy in detecting vegetation.

Qualitative evaluation of model outputs as demonstrated in Figure 6 showed that the segmentation model more accurately masked shapes, while the object detection model tended to have bounding boxes overlap and fail to capture some instances of vegetation.

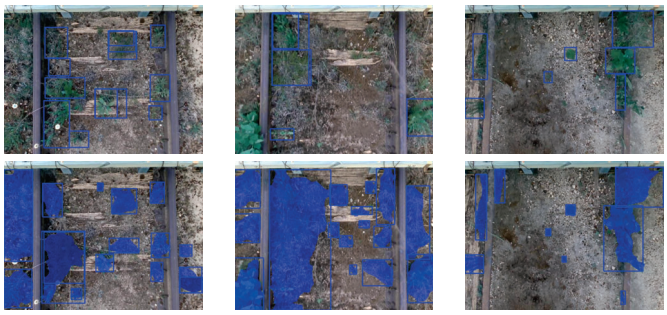

Fig. 6. Examples of YOLO's result bounding boxes (top row) and result segmentation masks (bottom row).

Overall, these results indicate that the YOLOv8 segmentation model achieves higher accuracy at a higher confidence threshold, with fewer false positives, though potentially more false negatives. This makes it generally more reliable than the object detection model. This suggests that segmentation works better for an irregular, organic defect such as vegetation compared to object detection, which has previously proved to show stronger performance on structural defects like missing

bolts and cracks. However, the YOLOv8 segmentation model still produced less than desirable F-1 score and accuracy. This suggests that YOLO's extended instance segmentation feature may not be advanced enough to accurately detect vegetation defects. Nevertheless, having established that segmentation outperforms object detection in terms of vegetation detection, we can move on to experiments with other semantic segmentation models such as U-Net and DeepLabv3+.

B. U-Net

The U-Net model stopped early at epoch 43, as the vegetation IoU did not improve for 10 consecutive epochs. Table II presents the training results of U-Net for vegetation class. The model achieved a validation loss of 0.1059, indicating that its predictions closely match the ground truth masks. The IoU of 0.8096 demonstrates a strong overlap between predicted and true vegetation regions, reflecting good segmentation performance. A high precision of 0.9144 indicates relatively few false positive predictions, while a recall of 0.8760 shows that most actual vegetation pixels are correctly detected. Overall, these results suggest that U-Net performs well in accurately identifying vegetation, effectively balancing false positives and false negatives. Figure 7 illustrates U-Net's validation result masks.

TABLE II
VALIDATION METRICS FOR U-NET

Metric	Value
Validation Loss	0.1059
Validation IoU	0.8096
Validation Precision	0.9144
Validation Recall	0.8760
Validation F1	0.8948

Fig. 7. Examples of U-Net's validation images (top row) and result masks(bottom row).

Although the validation results look promising, the testing results are less accurate, as shown in Figure 8. U-Net sometimes misclassified ballast and tree logs as vegetation. This shows that the model, while performing well on the validation set, has limitations in generalizing to real-world images with more complex backgrounds.

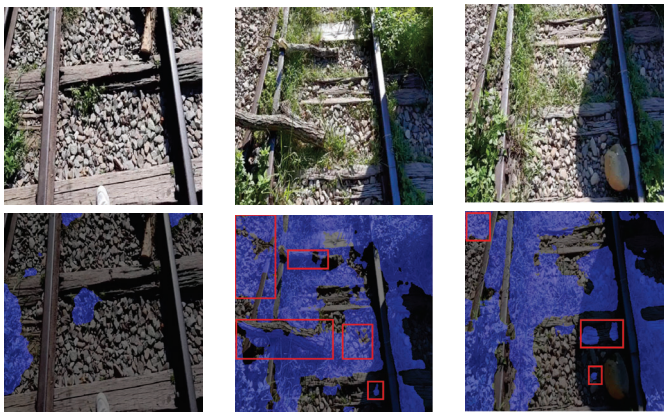

Fig. 8. Examples of U-Net's test images (top row) and result masks(bottom row).

C. DeepLabv3+

Table III shows the results of the DeepLabv3+ validation for the vegetation class. The DeepLabv3+ model achieved a validation loss of 0.0534, indicating that its predictions closely match the ground truth masks. A high IoU of 0.9124 demonstrates excellent overlap between predicted and true vegetation regions, reflecting strong segmentation performance. The model also exhibits high precision (0.9570) and recall (0.9510), indicating it correctly identifies most vegetation pixels while producing few false positives. The F1 score of 0.9540 confirms a strong balance between precision and recall. Figure 9 illustrates the validation result masks of DeepLabv3+. This strong performance persists on unseen testing images, as shown in Figure 10. Unlike U-Net, DeepLabv3+ is able to differentiate vegetation from ballast and tree logs. This improved performance can be attributed to the more complex model structure, with a model size of 346.5 MB, compared to 118.5 MB for U-Net, and the use of a pre-trained backbone.

TABLE III
VALIDATION METRICS FOR DEEPLABV3+

Metric	Value
Validation Loss	0.0534
Validation IoU	0.9124
Validation Precision	0.9570
Validation Recall	0.9510
Validation F1	0.9540

V. CONCLUSION

Vegetation control is vital for railway safety and infrastructure integrity. This study compared three deep learning approaches; YOLOv8, U-Net, and DeepLabv3+, for detecting overgrown vegetation alongside railroad tracks. DeepLabv3+ consistently achieved the highest precision, recall, and segmentation quality, proving most effective in complex rail environments. The YOLOv8 instance segmentation model outperformed its object detection counterpart, achieving a higher F1 score, mAP@0.5, and more stable precision–recall behavior,

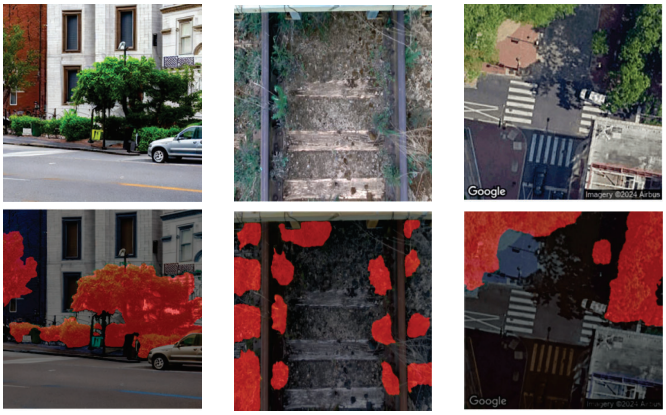

Fig. 9. Examples of DeepLabv3+'s validation images (top row) and result masks(bottom row).

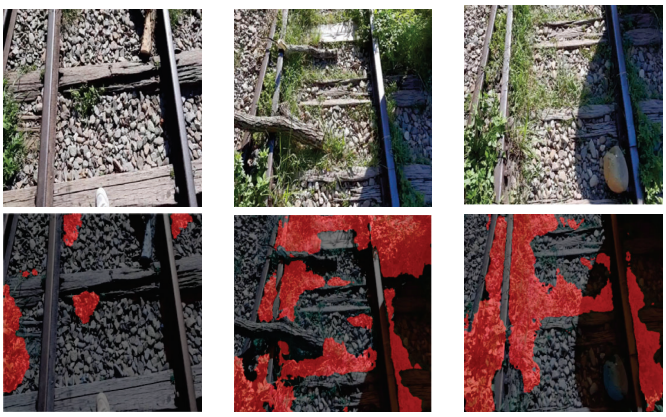

Fig. 10. Examples of DeepLabv3+'s test images (top row) and result masks(bottom row).

which underscores segmentation's advantage for irregular, organic defects. Nevertheless, YOLO's overall performance still falls below desired thresholds. Future efforts should explore the integration of binary railway masks to focus detection on track areas and potentially boost accuracy. To address dataset limitations, additional real-world railroad data will be collected to improve model generalization and performance. Real-world deployment challenges, including variability in lighting, weather, and track environments, will be verified through manual validation to assess model performance under different harsh deployment conditions. Finally, evaluating deeper and more advanced U-Net variants such as attention-augmented or residual-connected architectures, could close the performance gap with DeepLabv3+ on unseen testing images.

Acknowledgment

This research was supported by Marshall University's NSF REU program for Data Analytics. We thank the NSF and the U.S. Army Engineer Research and Development Center for facilitating this work in intelligent transportation systems, as well as program chairs Dr. Haroon Malik and Dr. Yousef Fazea for their guidance and feedback.

References

[1] American Society of Civil Engineers, "2025 report card for america's infrastructure: Rail," 2025.

[2] Association of American Railroads, "Rail Transportation and the U.S. Economy: Fueling Growth, Trade, and Opportunity," 2025.

[3] R. G. Nyberg, "Automating condition monitoring of vegetation on railway trackbeds and embankments," Ph.D. dissertation, Edinburgh Napier University, 2017.

[4] U.S. Code of Federal Regulations, "49 cfr §213.321 - vegetation," 2024.

[5] R. Tang, L. D. Donato, N. Bessinovic, F. Flammini, R. M. Goverde, Z. Lin, R. Liu, T. Tang, V. Vittorini, and Z. Wang, "A literature review of artificial intelligence applications in railway systems," *Transportation Research Part C: Emerging Technologies*, vol. 140, p. 103679, 2022.

[6] G. Jocher, A. Chaurasia, and J. Qiu, "Ultralytics yolov8," 2023. [Online]. Available: https://github.com/ultralytics/ultralytics

[7] O. Ronneberger, P. Fischer, and T. Brox, "U-net: Convolutional networks for biomedical image segmentation," *CoRR*, vol. abs/1505.04597, 2015. [Online]. Available: http://arxiv.org/abs/1505.04597

[8] L.-C. Chen, Y. Zhu, G. Papandreou, F. Schroff, and H. Adam, "Encoder-decoder with atrous separable convolution for semantic image segmentation," 2018. [Online]. Available: https://arxiv.org/abs/1802.02611

[9] P. Lewis, R. Casey, and S. Hancock, "Lidar for vegetation applications," University of Durham, Tech. Rep., 2007.

[10] Y. Min, B. Xiao, J. Dang, B. Yue, and T. Cheng, "Real time detection system for rail surface defects based on machine vision," *EURASIP Journal on Image and Video Processing*, 2018.

[11] S. Xu, Q. Feng, J. Fei, G. Zhao, X. Liu, H. Li, C. Lu, and Q. Yang, "A locating approach for small-sized components of railway catenary based on improved yolo with asymmetrically effective decoupled head," *IEEE Access*, vol. 11, pp. 34 870–34 879, 2023.

[12] D. Gautam, Z. Mawardi, L. Elliott, D. Loewensteiner, T. Whiteside, and S. Brooks, "Detection of invasive species (siam weed) using drone-based imaging and yolo deep learning model," *Remote Sensing*, vol. 17, p. 120, 01 2025.

[13] A. D'Arms, H. Song, H. S. Narman, N. C. Yurtcu, P. Zhu, and A. Alzarrad, "Automated railway crack detection using machine learning: Analysis of deep learning approaches," in *IEEE 15th Annual Information Technology, Electronics and Mobile Communication Conference (IEMCON)*, University of California, Berkeley, CA, Oct. 24–26 2024.

[14] D. Kholiya, A. K. Mishra, N. K. Pandey, and N. Tripathi, "Plant detection and counting using yolo based technique," in *2023 3rd Asian Conference on Innovation in Technology (ASIANCON)*, 2023, pp. 1–5.

[15] N. K. Gupta, M. Dougherty, B. Payvar, R. G. Nyberg, and S. Yella, "Machine vision approach for automating vegetation detection on railway tracks," *Journal of Intelligent Systems*, vol. 22, no. 2, pp. 179–196, 2013.

[16] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," 2016. [Online]. Available: https://arxiv.org/abs/1506.02640

[17] cvr, "cvr dataset," https://universe.roboflow.com/cvr-yqqba/cvr-frvvh , mar 2025, visited on 2025-09-05. [Online]. Available: https://universe.roboflow.com/cvr-yqqba/cvr-frvvh

[18] polesdataset, "Cityscapes-external-v1 dataset," https://universe.roboflow.com/polesdataset/cityscapes-external-v1 , dec 2022, visited on 2025-09-05. [Online]. Available: https://universe.roboflow.com/polesdataset/cityscapes-external-v1

[19] tfg, "vegetation dataset," https://universe.roboflow.com/tfg-o94q4/vegetation-a11kr , aug 2025, visited on 2025-09-05. [Online]. Available: https://universe.roboflow.com/tfg-o94q4/vegetation-a11kr

[20] vegetation, "vegetation dataset," https://universe.roboflow.com/vegetation-ojnpy/vegetation-gynk8 , may 2024, visited on 2025-09-05. [Online]. Available: https://universe.roboflow.com/vegetation-ojnpy/vegetation-gynk8

[21] Tree, "Interval-tree-mapillary-v4 dataset," https://universe.roboflow.com/tree-5s1ky/interval-tree-mapillary-v4 , may 2025, visited on 2025-09-05. [Online]. Available: https://universe.roboflow.com/tree-5s1ky/interval-tree-mapillary-v4