Automated Railway Crack Detection Using Machine Learning: Analysis of Deep Learning Approaches

Andrew d'Arms*, Hwapyeong Song†, Husnu S. Narman‡, Nevzat C. Yurtcu§, Pingping Zhu¶, and Ammar Alzarrad‖
Email: *darms1@marshall.edu,†song24@marshall.edu,‡narman@marshall.edu,§ncanyurtcu@outlook.com, ¶zhup@marshall.edu,‖alzarrad@marshall.edu

***Abstract*—Detecting defects in railway tracks, particularly small cracks or gaps, is traditionally a labor-intensive task. By leveraging machine learning, this process can be automated and accelerated, reducing both time and costs. This paper evaluates several prominent deep learning models for identifying railway cracks and examines the key factors influencing the training process. While our primary focus is on various versions of the You Only Look Once (YOLO) object detection model, we also explore the Residual Networks model. Our findings indicate that YOLOv5 and YOLOv9 achieve high crack detection accuracy, with F1-scores of 0.92 and 0.91, respectively. These results underscore the efficacy of deep learning models in detecting and classifying cracks, thereby potentially lowering labor costs. Additionally, we employ Explainable AI techniques to elucidate the models' decision-making processes in crack detection.**

By automating the detection of railway track defects, we can significantly enhance the efficiency and reliability of railway maintenance. The use of deep learning models, particularly the YOLO series, has shown promising results in accurately identifying even the smallest cracks. This not only speeds up the inspection process but also ensures a higher level of safety by detecting potential issues before they become critical. The integration of Explainable AI techniques further adds value by providing insights into how these models make decisions, which is crucial for gaining trust and improving the models. Overall, our research highlights the potential of advanced machine learning techniques to revolutionize railway maintenance, making it more cost-effective and reliable.

***Index Terms*—Railway, Crack detection, Computer vision, Deep learning, Explainable AI (XAI)**

I. Introduction

Identifying defects in railway tracks is crucial for effective railroad maintenance. One of the primary challenges in detecting small cracks or gaps is the labor-intensive nature of the process, largely due to the extensive length of the tracks. However, recent advancements in machine learning have sparked renewed interest in more efficient methods for defect detection.

A considerable amount of literature has been published on the issue. Gibert et al. researched a machine vision method to detect joint bar cracks, achieving an effectiveness of only 80% accuracy [1]. This study demonstrated that even before the advent of machine learning, there were methods that could be effective for our purposes. Guo et al. later applied machine learning techniques to the same problem, improving the effectiveness to 85% accuracy [2]. However, their work also highlighted the limitations of using convolutional neural networks for this task. While some research has been conducted using You Only Look Once (YOLO) [3], it remains a state-of-the-art object detection model renowned for its speed and accuracy. YOLO achieves this by applying a single neural network to entire images, dividing them into numerous regions, predicting bounding boxes, and weighing them appropriately. Zheng et al. compared various algorithms to determine the optimal method for detecting joint bar cracks [4]. They found that YOLOv3 excelled at detecting small cracks, while RetinaNet was more effective for larger cracks. By combining these models, they developed a deep transfer learning network that proved highly effective for both types of crack detection. In a similar study, Hsieh and Tsai evaluated 68 machine learning-based crack detection methods. They found that FCN and U-net performed most favorably, each achieving scores above 90 on the enhanced Hausdorff distance metric [5]. Moreover, Li et. al. used a YOLOv5s model to detect missing track fasteners with an average mean precision (mAP) of 97.4% and lauded its speed and small memory footprint, in addition to its accuracy [6]. Similarly, Fu et. al. were seeking a less intensive model to detect missing fasteners, and chose MobileNet-YOLOv4 due to its more efficient feature extraction [7]. By using this model, they were able to improve error rates by 80% when compared to YOLOv4 on its own. Min et al. developed a novel approach by designing a device that traverses the tracks, detecting scars on the rails and sleepers with 95% accuracy. Unlike most methods, this detection process is entirely conducted onboard the vehicle [8].

In this paper, our *aim* is not only to identify cracks in the railways but also to locate gaps in the track. This adds an additional layer of complexity and distinguishes our work from the previous studies. Our *objective* is to accurately and efficiently detect and identify cracks and gaps of various sizes in railroad tracks from multiple camera angles using object detection machine learning models. The key *contributions* of this paper are (i) to determine the most suitable machine learning model with optimized parameters for detecting cracks and gaps, (ii) to understand why object detection models succeed or fail in identifying the desired cracks and gaps by using Explainable AI (XAI) techniques, and finally (iii) to observe the effects of different datasets and class categorizations on the behavior of machine learning models. The results indicate that deep learning models can automatically detect cracks and gaps, which can significantly reduce the overall cost of railway maintenance.

The remainder of this paper is organized as follows: Section II discusses the machine learning models. Section III outlines the methodologies and approaches used. Section IV examines the impact of dataset and class categorization on model performance. Section V presents the visual results and their interpretation. Finally, Section VI has the final remarks.

II. Machine Learning Models

Although there are many different machine learning models, in this paper, we explore the effectiveness of You Only Look Once (YOLO) [3] with its versions and Residual Networks (ResNet) [9] models in detecting cracks and gaps on railroads.

A. YOLO and Its Versions

YOLO is a real-time object detection model that identifies and locates objects within an image. It has evolved through various versions, including YOLOv1, YOLOv2, YOLOv3, YOLOv5, YOLOv6, YOLOv7, YOLOv8, YOLOv9, and YOLOv10. Each iteration enhances accuracy and speed by refining the network architecture and incorporating new features to better handle complex scenarios and small objects. The primary differences between these versions lie in their balance of speed and accuracy, with newer versions generally achieving higher performance while maintaining real-time capabilities. Key distinctions among YOLO versions include variations in network architecture, loss functions, anchor box design, and training strategies [10]–[12].

Network Architecture: Each new version frequently integrates updated backbone and neck architectures, such as Darknet, CSPDarknet, or specialized feature aggregation modules. These enhancements usually result in better feature extraction and improved detection performance.

Loss function: Adjusting loss functions can emphasize particular elements of detection, such as bounding box regression or class prediction, thereby influencing the model's attention to various facets of object detection.

Anchor box design: The method of using anchor boxes to predict bounding boxes can differ between versions, affecting the model's ability to manage objects of varying sizes and aspect ratios.

Training strategies: Updated versions might employ sophisticated training methods such as data augmentation and learning rate scheduling to enhance performance even further.

YOLOv1 introduced the concept of single-stage object detection but had limitations in accuracy, especially for small objects. YOLOv2 improved accuracy by incorporating anchor boxes and a batch normalization layer, enhancing both localization and speed. YOLOv3 advanced further with a feature pyramid network (FPN) to handle objects of various sizes, improving detection performance across different scales. YOLOv4 brought in new backbone and neck architectures like CSPDarknet53 and PANet, achieving significant accuracy gains while maintaining speed. YOLOv5 is praised for its ease of use and excellent balance between speed and accuracy, often considered the most user-friendly version. YOLOv6 focused on hardware optimization with a redesigned backbone and neck architecture, delivering high performance on specific hardware platforms. YOLOv7 introduced the "E-ELAN" network architecture, optimized training methods, and improved detection of smaller objects with higher resolution, resulting in increased accuracy. YOLOv8 featured a novel anchor-free split head and state-of-the-art backbone and neck architectures, offering a better balance between accuracy and speed, making it suitable for diverse applications. YOLOv9 aimed to maintain real-time object detection capabilities while achieving higher mean average precision (mAP) compared to previous models, often considered one of the most accurate versions. YOLOv10 emphasized reduced latency with a non-maximum suppression (NMS)-free approach, allowing for faster post-processing while still delivering competitive accuracy, though it may struggle with very small objects due to fewer parameters.

Subcategories for YOLO version: Some YOLO versions are further divided into iterations, from the tiny 't' or nano 'n' variant to the extensive 'e' model or 'x', showcasing enhancements in both accuracy (mAP metrics) and efficiency, with fewer parameters and lower computational requirements (FLOPs).

B. ResNet

ResNet [9] is a widely-used deep learning model in computer vision, particularly for tasks such as image segmentation and object detection. Its notable features include skip connections, residual blocks, and various model variants.

Skip connections: These connections facilitate the smoother flow of gradients throughout the network, enhancing the training process for deeper networks.

Residual blocks: These blocks form the core of ResNet, incorporating the essential skip connection.

Variants: ResNet comes in several variants, such as ResNet18, ResNet50, ResNet101, and ResNet152, each named according to the number of layers in the network.

III. Methodology

Most of the models were trained using the Ultralytics Python package, which was also utilized for validation and prediction. The training was conducted on an Nvidia GeForce RTX 3090, equipped with 24 gigabytes of VRAM, making it an ideal choice for the VRAM-intensive process. Additionally, other tools including Roboflow by Dwyer et al. [13] was employed for dataset compilation and augmentation.

We primarily employ quantitative analysis to identify the most effective model for crack detection. The models' performances are evaluated by using precision, recall, and F1-scores. Precision is the ratio of correctly predicted positives to the total predicted positives, while recall is the ratio of correctly predicted positives to the actual positives, assessing the completeness of positive predictions. The F1-score, which is the harmonic mean of precision and recall, provides a balanced measure of both. In this study, the F1-score is calculated for each class. Additionally, the confusion matrix is utilized to evaluate the accuracy and quality of the model's predictions. Furthermore, XAI has been used in various classes to facilitate decision-making. It is accomplished utilizing Easy Explain, a Python library by Theocharis [14].

The performance of a model is closely tied to the quality of the dataset used. In this study, we conduct experiments using two types of datasets: an independent dataset and a combined dataset. We test various models and perform hyperparameter tuning on these datasets, recording the results for each combination.

IV. Dataset and Categorization Effects on the Performance

Our research has identified several datasets related to railroads with cracks [15]–[21]. To assess the feasibility of crack detection using machine learning models, we initially evaluate the models' performance in identifying the presence of cracks or gaps. Subsequently, we conduct experiments to examine the impact of multi-class classification of cracks and gaps across various datasets.

A. Details on Specific Datasets

The datasets used for this research were all similar in scope, as they all feature images of railway cracks and gaps from various angles and environments. Some images are shared between datasets, but may be augmented in different ways, making them unique in context of the whole, combined dataset.

The dataset used for single classification, being the 'Thesis Group' dataset featured a broad selection of scenarios for our models to work on [15]. These include images with people in the background, cracks from various angles, including top-down, side-view, etc., and both wide and narrow shots. In regard to the defects themselves, gaps of a wide range of sizes are included, as well as both minor and major cracks. Also included are a significant number of images of non-defective tracks, as these are integral to ensuring model accuracy. If only defective track images are included, then the model may be unable to properly classify them, should it see them, resulting in significant false positive rates. The main limitation of this dataset was the singular class, as classifying multiple defects under this class made it difficult for the model to accurately predict new defects, resulting in decreased performance. This was rectified later when it was relabeled for the combined dataset.

The other datasets were similar in scope to the original, but with some minor differences, especially in regards to labeling. The 'Technofly Solution' dataset has a good variety of angles, defects, zooms, and environments, but featured several extraneous classes for other defects outside the scope of this project [16]. These extra defects also meant that there were several images irrelevant to this project's scope, but were still included in the combined dataset to expand the unlabeled category. The dataset by Eunus et. al. featured a good variety of relevant images, but were entirely unlabeled and most of the images were taken from a side-view angle [20]. The dataset by Ranganath had a good variety of angles, defects, and environments, but again were entirely unlabeled [19]. The 'System Thinking Project' dataset featured a decent range of angles, environments, and defects, but had unusable, polygonal labels and was a limited in scope [21]. After combining these datasets, the limitations became much less relevant, especially due to the relabeling process. Together, they form a very broad dataset with many different angles, defects, backgrounds, scenarios, and zooms, all with a consistent labeling scheme, allowing for the increased accuracy previously mentioned.

B. The Effects of the Single Classification on the Different Models

To observe the performance of the models, we use the 'Thesis Group' dataset [15]. This dataset contains nearly 1,000 images, all categorized under a single class by combining gaps and cracks into one category. Through data augmentation, the dataset size increased to over 2,000 images. This simplification facilitates the evaluation of different models' performance. The dataset is divided into 86% for training, 10% for validation, and 4% for testing.

TABLE I
The performance of the selected models on the 'Thesis Group' dataset [15] for a single class.

Model	F1	Precision	Recall
YOLOv10x	0.63	0.73	0.53
YOLOv10n	0.56	0.74	0.51
YOLOv9e	**0.68**	**0.79**	0.60
YOLOv9c	0.63	0.73	**0.61**
YOLOv8	0.59	0.65	0.55
YOLOv6	0.60	0.72	0.54
YOLOv5	0.51	0.61	0.50
YOLOv3	0.62	0.74	0.54
ResNet101	0.59	0.72	0.56

Table I presents the optimized results of YOLOv10x, YOLOv10n, YOLOv9e, YOLOv9c, YOLOv8, YOLOv6, YOLOv5, YOLOv3, and ResNet101 using various optimization techniques and hyperparameter tunning for 'Thesis Group' dataset [15]. For YOLO models besides YOLOv10 and YOLOv9, only the default model size was included to increase readability. According to the obtained results, YOLOv9 outperforms other models with the highest F1 score, Precision, and Recall for the single classification of gaps and cracks. In contrast, YOLOv5 has the lowest F1 score, Precision, and Recall.

C. The Effects of the Multi Classifications on the Different Models

To evaluate the impact of multi-classification on different models, we conducted experiments using three and four classification categories for cracks and gaps. The three classifications are: crack, small gap, and large gap. In this setup, a small gap is defined as any gap between 0-1 inch in the track, while a large gap is any gap greater than 1 inch. For the four classifications, the categories are: crack, small gap, medium gap, and large gap. Here, a small gap is defined as any gap between 0-1 inch, a medium gap is between 1-3 inches, and a large gap is any gap greater than 3 inches.

For this experiment, we created a larger dataset by combining the previously mentioned datasets to ensure we have a sufficient number of labeled data for various classes. However, to observe the effects of multi-class classification, we maintained the same size for the combined dataset in both the three-class and four-class scenarios. Additionally, some augmentations were performed through Roboflow. These augmentations include: horizontal flipping, +/- 10-degree shear horizontal and/or vertical transformations, and +/- 15-degree rotations. Furthermore, the dataset is divided into 70% for training, 20% for validation, and 10% for testing.

C.1 The Effects of the Three Classifications on the Different Models

The same models from Section IV-B were tested again for this expanded dataset to ensure consistency. Table II shows the results for all models. As similar to the single class result, YOLOv9 outperforms YOLOv10 models with the better F1 score, Precision, and Recall. Interestingly, ResNet101 was able to match YOLOv9e in terms of precision.

TABLE II
THE PERFORMANCE OF THE SELECTED MODELS ON THE COMBINED DATASET FOR THREE CLASSES.

Model	F1	Precision	Recall
YOLOv10x	0.71	0.84	0.62
YOLOv10n	0.72	0.72	0.73
YOLOv9e	**0.81**	**0.86**	**0.80**
YOLOv9c	0.79	0.85	0.75
YOLOv8	0.74	0.85	0.67
YOLOv6	0.72	0.80	0.65
YOLOv5	0.72	0.80	0.65
YOLOv3	0.77	0.85	0.69
ResNet101	0.77	**0.86**	0.71

C.2 The Effects of the Four Classifications on the Different Models

Table III presents the optimized results of the four-class detection for the combined dataset. We include the performance metrics for all of the same models as in the previous sections to ensure consistency and possible outliers. While YOLOv9 demonstrates superior performance in Single and Three-Class categorization, YOLOv5 excels in terms of F1 score and Recall, outperforming YOLOv9 by a margin of 0.01. Notably, YOLOv6 and YOLOv9 achieve the highest precision among the models evaluated. These findings highlight the nuanced strengths of each YOLO version, suggesting that the choice of model may depend on the specific requirements of the detection task at hand. For instance, YOLOv5 might be preferred for applications prioritizing overall accuracy and recall, whereas YOLOv6 and YOLOv9 could be more suitable for tasks where precision is paramount.

Due to the YOLOv5 performance for four-class categorization, the F1 Confidence Curve for YOLOv5 is provided in Fig. 1.

TABLE III
THE PERFORMANCE OF THE SELECTED MODELS ON THE COMBINED DATASET FOR FOUR CLASSES.

Models	F1	Precision	Recall
YOLOv10x	0.88	0.90	0.85
YOLOv10n	0.84	0.92	0.79
YOLOv9e	0.91	**0.94**	0.92
YOLOv9c	0.79	0.85	0.75
YOLOv8	0.86	0.87	0.84
YOLOv6	0.88	**0.94**	0.84
YOLOv5	**0.92**	0.92	**0.93**
YOLOv3	0.87	0.89	0.87
ResNet101	0.87	0.90	0.86

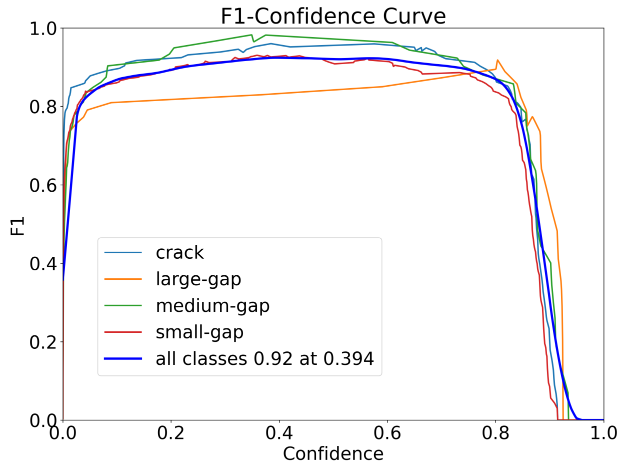

Fig. 1. The F1 confidence curve of the optimized YOLOv5 performance.

D. Discussion on Effects of Dataset and Multi-Class Categorizations

It is well-known that larger and higher-quality datasets significantly enhance object detection performance. Similar to the detection of cracks and gaps in railroads, we have verified this using the 'Thesis Group' dataset [15] and a combined dataset. Notably, our findings indicate that increasing the number of class categorizations within this combined dataset also boosts performance. This improvement is likely due to the model's enhanced ability to distinguish between a greater variety of objects, leading to more accurate predictions.

However, this behavior may not always be anticipated. Typically, one might expect that adding an existing object class would improve the prediction accuracy for that class while potentially reducing the accuracy for similar classes. This is because the model might become more specialized in identifying the newly added class, potentially at the expense of other classes that share similar features. Conversely, introducing a non-existing object class could decrease the prediction accuracy across all classes. This reduction in accuracy might occur because the model could become confused by the presence of an object class that does not exist in the real-world scenarios it is being trained to recognize, leading to overall poorer performance.

Moreover, the complexity of the dataset and the diversity of the object classes play crucial roles in determining the model's performance. A well-balanced dataset with a wide range of object classes can help the model generalize better, thereby improving its robustness and accuracy. On the other hand, an imbalanced dataset with too many similar classes might lead to overfitting, where the model performs well on the training data but poorly on unseen data.

V. Visual Results

In this section, we present various examples of crack and gap detection across different models.

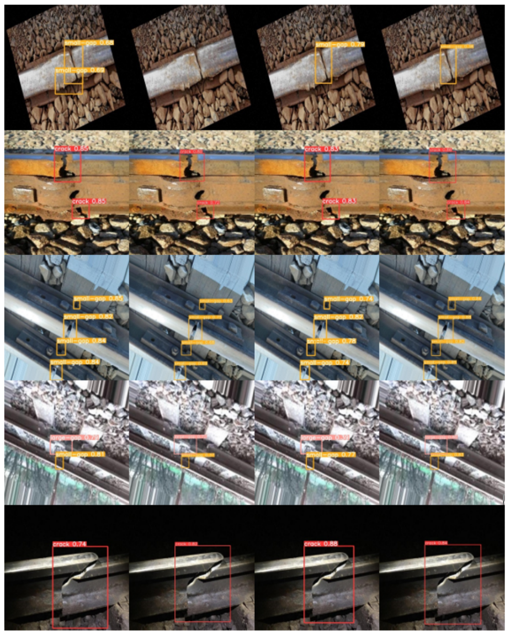

Fig. 2. Crack and gap detection across five images using YOLOv5, YOLOv8, YOLOv9, and YOLOv10, respectively, from left to right.

Fig. 2 illustrates crack and gap detection across five images using YOLOv5, YOLOv8, YOLOv9, and YOLOv10, respectively, from left to right. *First row* shows a top-down view of a small, jagged gap that could be mistaken for a crack. Only YOLOv9 and YOLOv10 successfully identified it, while YOLOv5 misidentified it twice, and YOLOv8 completely missed it. *Second row* presents a side view of a distinct crack. All models identified it as two separate cracks due to its discontinuity. *Third row* features four gaps present simultaneously. All models successfully identified these gaps. *Fourth row* displays a lower-quality image with a large gap in the foreground and a small gap in the background. All models successfully detected both gaps. *Fifth row* shows a crack under low lighting conditions. Despite the poor lighting, all models were able to identify the crack. Due to the similar architecture between YOLO versions, most YOLO models perform similarly.

A. Defect Categories

Fig. 3 illustrates various categories of examples detected using YOLOv9e, the most consistent model in our study. These

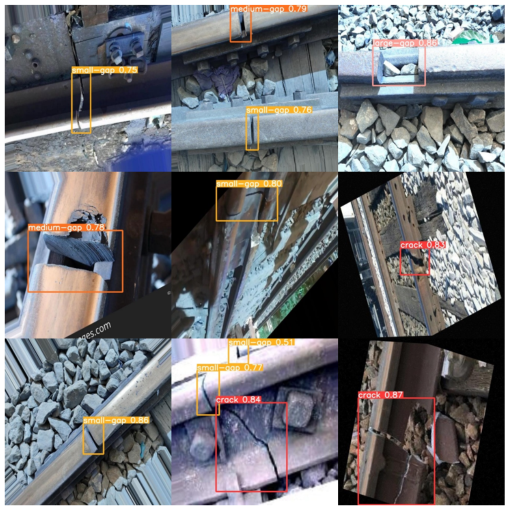

Fig. 3. Illustration of three defect categories detected by YOLOv9e: vertical (row 1), horizontal (row 2), and diagonal (row 3).

broad categories were selected to demonstrate the model's diverse capabilities. *First row* depicts vertical gaps, including instances of multiple defects and gravel on the railroad track. *Second row* hows horizontal gaps and cracks, featuring close-up, upside-down, and sideways shots. *Final row* highlights diagonal defects, including distorted edges due to augmentation, multiple defect types, and an atypical crack. The model successfully identified all these cases with confidence levels of at least 0.75.

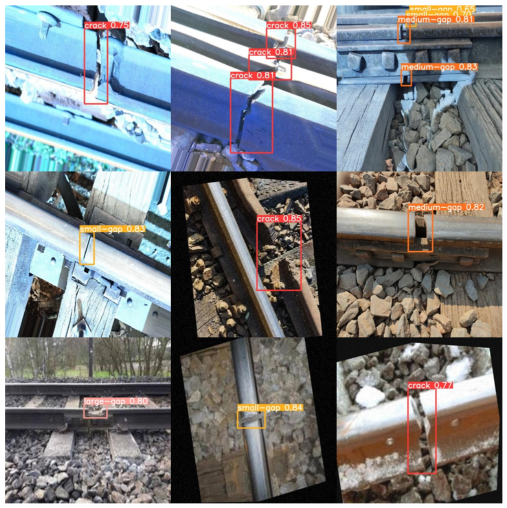

Fig. 4. Illustration of defect categories detected by YOLOv9e, including side-view (row 1), top-view (row 2), and low quality (row 3).

Fig. 4 illustrates various defect categories to demonstrate the model's comprehensive effectiveness. The first row presents side views of defects, such as a track split and an underside gap. The second row shows approximately top-down views, including a crack that has damaged part of the track. The final row features more challenging examples: a large side-view gap, a low-light image of a small gap without a shadow, and a low-quality image of a minimally jagged crack. In all these cases, YOLOv9e detected the cracks and gaps with a confidence level of at least 0.75.

B. Failed Detection

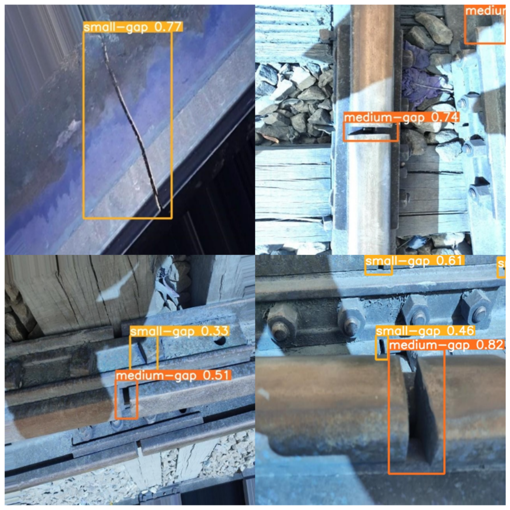

Fig. 5. Illustration of four instances where YOLOv9e failed to correctly detect.

Fig. 5 illustrates four instances where YOLOv9e failed to correctly identify defects. In the top left image, the model mistakenly classifies a crack as a small gap. This error likely arises because the crack's straightness resembles typical gaps. Although the length of the "gap" should have indicated otherwise, the model misclassified it. Nevertheless, the defect was still detected, which is advantageous as reclassifying an error is easier than identifying a missed defect.

The top right photo shows the model incorrectly identifying a medium gap in the top right corner, despite the defect being partially visible and not clearly a gap. This indicates that the model focuses solely on the gap itself, ignoring the surrounding track, which is suboptimal. However, since this was a false positive rather than a false negative, the impact of this error is minimal.

The bottom left photo illustrates YOLOv9e mistakenly identifying a shadow as a small gap. This error likely arises from the model's tendency to associate small gaps with narrow, black rectangles, which the shadow resembles. Correcting this mistake is challenging due to the visual similarity and the model's inability to consider the full context of the photo. However, using segmentation could help, as it would allow the model to focus solely on detecting cracks or gaps in the track, ignoring irrelevant surrounding areas.

Finally, the bottom right photo shows the model inaccurately identifying a small gap on the edge of the image. It is difficult to determine whether the area of interest is a shadow or a gap, but since it is not fully visible, it likely should not have been identified. This mistake is unlikely to significantly impact the model's actual implementation, as the images used in practice

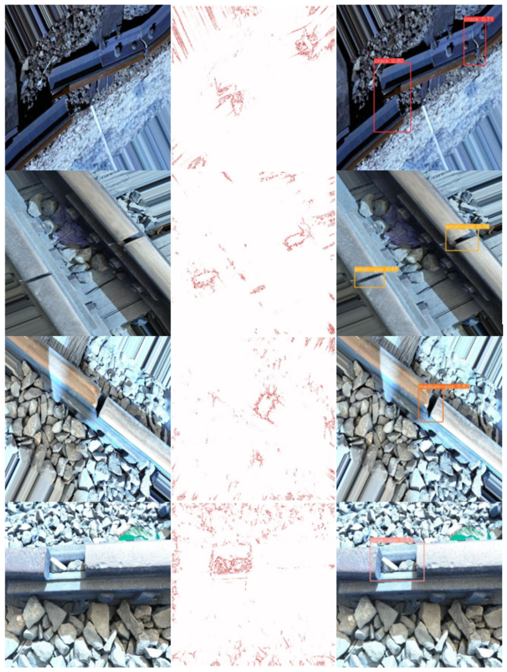

Fig. 6. Explainable AI interpretations of various defects using YOLOv8.

will be more homogeneous and feature less challenging angles, reducing the likelihood of errors.

While these mistakes were relatively minor within the scope of this research, they remain significant considerations. The model was trained on a highly diverse image set, including many challenging situations. Consequently, the results may not accurately reflect its performance in broader applications. At this stage, the best approach to address these issues is to identify and accurately label more edge cases, enabling the model to distinguish minute differences between shadows and gaps. Alternatively, employing a more context-aware model could be beneficial.

C. Explainable AI for YOLO Models in Crack and Gap Detection

Fig. 6 illustrates the application of Explainable AI (XAI) across various classes, specifically supporting YOLOv8. Although the YOLOv8 used in this evaluation is not the top-performing one, it can assist us in understanding how YOLO models work while detecting cracks and gaps. We use the Easy Explain library to provide insights into YOLO's decision-making process.

In the first row, the model is applied to a crack class. The model tends to highlight any sharp edges it detects, with a particular focus on the jagged nature of cracks. Two clusters of red lines are clearly visible, corresponding to the areas where the model identified cracks.

The second row shows the model's performance on a small gap case. Here, the model again highlights sharp edges but outlines the area where the gaps are, rather than highlighting their interiors. These gaps appear almost as eclipses in the image, matching the actual detected gaps. The model consistently identifies the area with the most highlights as the detection.

The third row depicts the model's run on a medium gap case, which looks quite similar to the small gap case. The model forms an eclipse around the gap area, reflecting the frequent misclassification between these two classes.

The final row presents a large gap case, which differs significantly from the others. The model highlights the gravel within the gap and the edges of the rail where the gap occurs. Since large gaps often include gravel, the model logically associates this feature with the class. In this instance, the background also shows highlighted gravel, but the model is not misled as the highlights are not concentrated enough to be considered a detection.

VI. Conclusion

Our research highlights the significant potential of deep learning models in automating the detection of railway track defects, thereby enhancing the efficiency and reliability of railway maintenance. The YOLO series, particularly YOLOv5 and YOLOv9, has demonstrated exceptional performance in identifying even the smallest cracks, achieving high F1-scores of 0.92 and 0.91, respectively. This advancement not only accelerates the inspection process but also ensures a higher level of safety by detecting potential issues before they become critical.

Furthermore, our study illustrates how variations in datasets and the addition of classes can impact the models' performance in detecting cracks and gaps. The incorporation of Explainable AI techniques provides valuable insights into the models' decision-making processes, fostering trust and facilitating further improvements. Overall, our findings underscore the transformative impact of advanced machine learning techniques on railway maintenance, making it more cost-effective and reliable.

In the future, we aim to focus on field testing, which includes creating a new dataset and further tuning our model. Field testing will enable us to develop a more realistic dataset through consistent data collection methods, leading to a more specialized model tailored to our use case and improved performance.

Acknowledgment

We extend our sincere gratitude to the Engineer Research and Development Center for their generous support and funding.

References

[1] X. Gibert, A. Berry, C. Diaz, W. Jordan, B. Nejikovsky, and A. Tajaddini, "A machine vision system for automated joint bar inspection from a moving rail vehicle," in *ASME/IEEE 2007 Joint Rail Conference and Internal Combustion Engine Division Spring Technical Conference*, 03 2007.

[2] L. Zhuang, L. Wang, and Z. Zhang, "Automated vision inspection of rail surface cracks: A double-layer data-driven framework," *Transportation Research Part C Emerging Technologies*, vol. 92, p. 258–277, 07 2018.

[3] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," in *IEEE conference on computer vision and pattern recognition*, 2016, pp. 779–788.

[4] Z. Zheng, H. Qi, L. Zhuang, and Z. Zhang, "Automated rail surface crack analytics using deep data-driven models and transfer learning," *Sustainable Cities and Society*, vol. 70, p. 102898, 03 2021.

[5] Y.-A. Hsieh and Y. J. Tsai, "Machine learning for crack detection: Review and model performance comparison," *Journal of Computing in Civil Engineering*, vol. 34, no. 5, 2020.

[6] X. Li, Q. Wang, X. Yang, K. Wang, and H. Zhang, "Track fastener defect detection model based on improved yolov5s," *Sensors*, vol. 23, no. 14, 2023.

[7] J. Fu, X. Chen, and Z. Lv, "Rail fastener status detection based on mobilenet-yolov4," *Electronics*, vol. 11, no. 22, 2022.

[8] 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*, vol. 2018, no. 1, 2018.

[9] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

[10] M. Hussain, "Yolov1 to v8: Unveiling each variant–a comprehensive review of yolo," *IEEE Access*, vol. 12, pp. 42 816–42 833, 2024.

[11] A. Vijayakumar and S. Vairavasundaram, "Yolo-based object detection models: A review and its applications," *Springer Multimedia Tools and Applications*, pp. 1–40, 2024.

[12] P. Jiang, D. Ergu, F. Liu, Y. Cai, and B. Ma, "A review of yolo algorithm developments," *Elsevier Procedia computer science*, vol. 199, pp. 1066–1073, 2022.

[13] B. Dwyer, J. Nelson, and T. Hansen, "Roboflow (version 1.0)," 2024. [Online]. Available: https://roboflow.com

[14] S. Theocharis, "Easy explain," 2024. [Online]. Available: https://github.com/stavrostheocharis/easy_explain

[15] T. Group, "Railway crack detection dataset," Online, 2024, visited on 2024-09-04. [Online]. Available: https://universe.roboflow.com/thesis-group/railway-crack-detection

[16] T. Solution, "Railwaytrackcrackdetection dataset," Online, 2022, visited on 2024-09-04. [Online]. Available: https://universe.roboflow.com/technofly-solution/railwaytrackcrackdetection

[17] V. Chennai, "Track crack damage detection dataset," Online, 2023, visited on 2024-09-04. [Online]. Available: https://universe.roboflow.com/vit-chennai-japsi/track-crack-damage-detection

[18] S. Hossain, S. I. Eunus, A. Adnan, and A. Ridwan, "Railway track fault detection," 2021. [Online]. Available: https://www.kaggle.com/dsv/1884733

[19] C. Ranganath, "Railway track fault detection dataset," Online, 2023, visited on 2024-09-04. [Online]. Available: https://universe.roboflow.com/chinmay-ranganath-ohlji/railway-track-fault-detection

[20] S. I. Eunus, A. Adnan, S. Hossain, and A. Ridwan, "Railway track fault detection," 2021. [Online]. Available: https://www.kaggle.com/datasets/salmaneunus/railway-track-fault-detection?select=Railway+Track+fault+Detection+Updated

[21] S. T. Project, "Damage detection in tracks dataset," Online, 2023, visited on 2024-09-04. [Online]. Available: https://universe.roboflow.com/system-thinking-project-4rfww/damage-detection-in-tracks