Optimized Transfer Learning Models for Rail Surface Defect Classification with Explainable AI Validation

Murat Alparslan Gungor
Department of Electrical and Electronics Engineering, Faculty of Engineering and Natural Sciences, Hitit University, Corum, Turkiye
alparslangungor@hitit.edu.tr

Kenan Gencol
Department of Electrical and Electronics Engineering, Faculty of Engineering and Natural Sciences, Hitit University, Corum, Turkiye
kenangencol@hitit.edu.tr (corresponding author)

Received: 10 April 2026 | Revised: 28 May 2026 | Accepted: 1 June 2026

 DOI: https://doi.org/10.48084/etasr.19215

ABSTRACT

Rail surface defects such as flaking, spalling, and squat are critical indicators of railway track degradation and require reliable and efficient automated inspection systems to ensure operational safety. This study presents a comparative evaluation of seven state-of-the-art pre-trained Convolutional Neural Network (CNN) architectures for rail surface defect classification, namely Inception-V3, MobileNet-V1, MobileNet-V2, MobileNet-V3, NasNetMobile, ResNet50, and EfficientNet-B0. Transfer learning was employed by freezing the convolutional backbones and optimizing the classifier head using Optuna-based hyperparameter search. The models were trained and evaluated on a benchmark rail surface defect dataset containing balanced samples of the three defect classes after redundancy reduction and class equalization. Performance was assessed using quality metrics, whereas computational efficiency was analyzed using Floating-Point Operations (FLOPs) and parameter complexity. Experimental results show that MobileNet-V2 achieves the highest classification accuracy of 81.6% with 599M FLOPs, whereas MobileNet-V3 achieves competitive accuracy of 78.5% with the lowest computational cost of only 132M FLOPs. Inception-V3 also demonstrates strong performance with 80.8% accuracy but requires substantially higher computational complexity (11.4B FLOPs). Pareto analysis confirmed that both MobileNet variants provide the best efficiency–accuracy trade-off among all evaluated models. To improve interpretability, Gradient-weighted Class Activation Mapping (Grad-CAM) was applied to the Pareto-optimal models. The visualizations revealed that MobileNet-V2 generally focuses more consistently on defect-relevant regions, particularly for flaking defects. The findings highlight the importance of combining performance benchmarking with Explainable Artificial Intelligence (XAI) analysis for safety-critical railway monitoring applications.

Keywords-rail surface defect; Convolutional Neural Network (CNN); classification; Explainable AI (XAI)

I. INTRODUCTION

Railways play a key role in transporting passengers and freight in many commercial and industrial sectors. As railway transportation continues to grow, various studies have focused on improving railway safety and operational reliability, for example, by predicting accident severity at highway–railway level crossings [1] and by designing decentralized railway traffic control systems [2]. In addition to such system-level approaches, railway safety also depends critically on the physical condition of the track. As the primary load-bearing component of the track, the rail is a key part of the railway infrastructure and can exhibit various surface defects, including flaking, spalling, and squat [3]. To address these issues, a variety of methods have been proposed in the literature to detect and classify rail surface defects. Authors in [4] developed a laser-based defect-detection system for railway tracks and train wheels. Authors in [5] proposed a novel high-speed railway inspection robot and an associated fault-detection method to address the difficulty and low accuracy of real-time, online detection of rail defects and damage. Authors in [6] presented an embedded eddy-current system for online rail-defect detection/localization and a complementary Convolutional Neural Network (CNN) method that classifies defects from wavelet-transform features of the eddy-current signals. Authors in [7] achieved real-time identification of

railway track faults using image-processing techniques, including Canny edge detection and the Two-Dimensional Discrete Wavelet Transform (2D-DWT). Authors in [8] employed machine learning to classify defects in railway components. Authors in [9] proposed an improved variant of You Only Look Once (YOLO)v5s—called YOLOv5s-VF—for rail-surface defect detection. Authors in [10] provided a review of computer-vision inspection methods applied to railway defect detection. Authors in [11] proposed a real-time rail-defect-detection network with a MobileNet backbone and YOLO/FPN-style multiscale heads, achieving fast inference and high accuracy. Authors in [12] improved YOLOv7 using MobileNetV3, EIOU loss, and k-means++ clustering for rail surface defect detection. Authors in [13] proposed a real-time rail surface defect detection system using edge computing devices and an improved YOLOv5 model. Authors in [14] introduced a semi-supervised student–teacher model to improve railway defect detection performance. Authors in [15] proposed a Spiking Neural Network (SNN) with time-varying weights for rail squat detection using axle-box acceleration measurements.

In light of the literature, deep learning–based methods—particularly CNNs—have been widely employed for rail-surface defect detection. In this study, we further investigate and optimize several widely used state-of-the-art pre-trained CNNs for rail surface defect classification, and evaluate their performance in terms of accuracy, precision, recall, and F1-score. We also present computational complexity-accuracy Pareto plots of the models to select optimal choices and visualize the spatial attention behaviors of the Pareto-frontier models using Explainable Artificial Intelligence (XAI) to validate our results.

II. MATERIALS AND METHODS

For classification, we use the dataset described in [3] and available from [16]. In recent years, this dataset has become a popular benchmark for rail defect detection and classification studies [17, 18]. The dataset was collected using EKEN H9R cameras mounted on both sides of the railway inspection vehicle. The cameras were capable of recording videos at 120 Frames per Second (FPS), and the required frames were extracted from these recordings. The extracted images were then manually annotated. We use the flaking, spalling, and squat defects from this dataset. The original dataset contained 291 spalling, 2,829 flaking, and 1,844 squat images. However, the class distribution of these image categories is not numerically balanced, which may lead to overfitting during training. In addition, since the images were extracted from videos, there are many groups of highly similar frames. This temporal continuity increases the redundancy in the dataset and may negatively affect the model's ability to generalize. Therefore, for the flaking and squat classes, highly similar images are progressively reduced by half in each iteration until approximately 300 images are retained for each class, while the spalling class already contained approximately 300 images. Then, each defect class is split into training, validation, and test sets in an 8:1:1 ratio. Figure 1 shows the distribution of training, validation, and test images for each defect class, whereas Figure 2 presents representative examples from each category in the dataset.

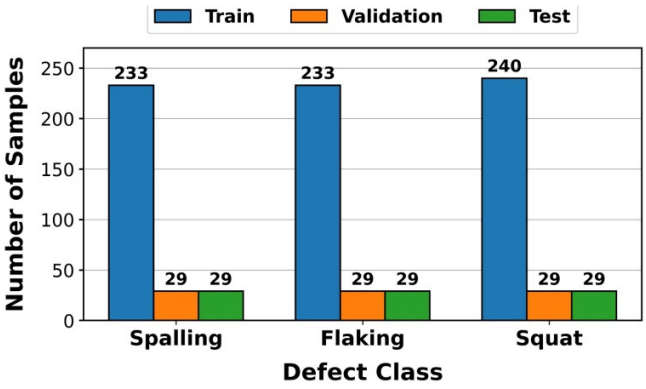


Fig. 1. Distribution of training, validation, and test images for each defect class.

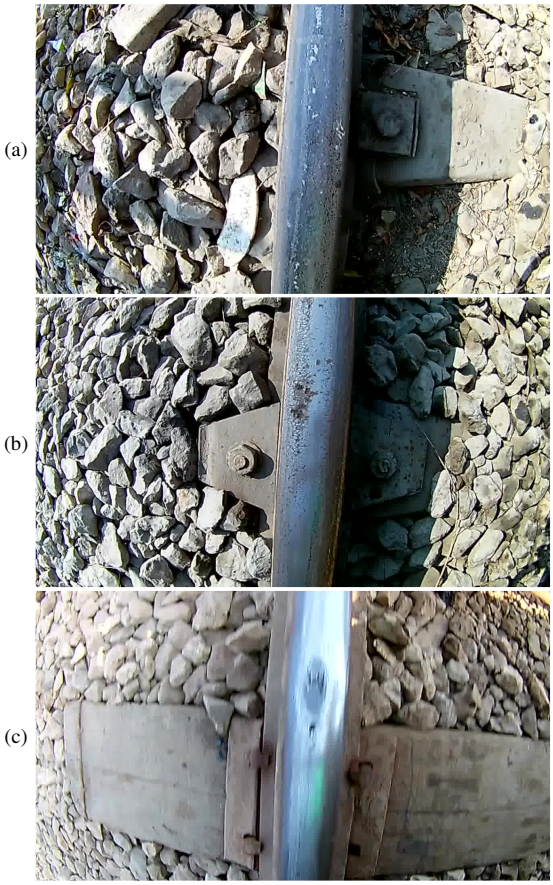


Fig. 2. Representative images from each defect category: (a) flaking, (b) spalling, (c) squat.

In this study, we use pre-trained CNN models for classification, leveraging features learned from large-scale image datasets and adapting them to rail surface defect images instead of training deep networks from scratch. Seven state-of-the-art CNN models are employed: Inception-V3 [19], MobileNet-V1 [20], MobileNet-V2 [21], MobileNet-V3 [22],

NasNetMobile [23], ResNet50 [24], and EfficientNet-B0 [25]. Figure 3 shows the architecture of the classification model used in this study.

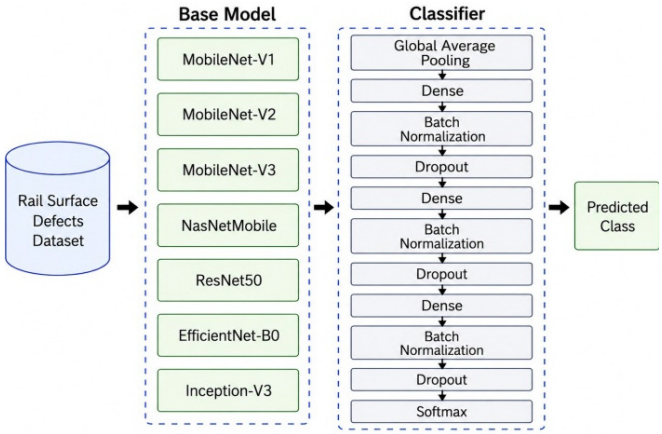


Fig. 3. The architecture of the classification model.

In this study, the base of the pre-trained CNN is kept frozen and utilized as the feature extraction component, while only the classifier is trained. Optuna [26] is employed for hyperparameter optimization using a Tree-structured Parzen Estimator (TPE) sampler, which iteratively refines the search based on prior trial evaluations. By focusing on promising hyperparameter regions instead of exploring the space randomly, this approach provides a more efficient and systematic optimization framework [27]. For each model, the classifier architecture shown in Figure 3 is optimized using Optuna by determining the number of dense layers, the number of neurons in each layer, the dropout rate for each layer, and whether batch normalization is applied. In addition, the learning rate, batch size, and optimizer type are optimized separately for each model. Table I shows the hyperparameter configurations determined by Optuna for each pre-trained CNN model.

In Table I, "Dense units per layer" refers to the number of neurons in each dense layer of the classifier, whereas "Dropout per layer" represents the dropout rate applied to the corresponding layer. Besides the hyperparameter settings shown in Table I, each model is trained for up to 50 epochs with an early stopping strategy to reduce the risk of overfitting. In addition, categorical cross-entropy is used as the loss function and the output layer consists of a three-class softmax classifier.

We evaluate the pre-trained CNN models using standard metrics, namely accuracy, recall, precision, and F1-score. In classification, accuracy quantifies the proportion of correct predictions to the total number of instances, providing a global indicator of performance:

$$\text{Accuracy} = \frac{(TP+TN)}{(TP+TN+FP+FN)} \quad (1)$$

where TP (True Positives) = predicted positive & truly positive; TN (True Negatives) = predicted negative & truly negative; FP (False Positives) = predicted positive but truly negative; FN (False Negatives) = predicted negative but truly positive.

Another metric is the F1-score, which balances precision and recall; precision is the share of positive predictions that are correct, and recall is the share of actual positives that are detected. Precision, recall, and F1-score are defined as follows:

$$\text{Precision} = \frac{TP}{(TP+FP)} \quad (2)$$

$$\text{Recall} = \frac{TP}{(TP+FN)} \quad (3)$$

$$\text{F1} - \text{score} = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} \quad (4)$$

TABLE I. HYPERPARAMETER CONFIGURATIONS OBTAINED USING OPTUNA FOR EACH PRE-TRAINED CNN MODEL

Model	Learning rate	Batch size	Optimizer	Batch normalization	Number of dense layers	Dense units per layer	Dropout per layer
Inception-V3	0.00014	8	Adam	No	1	[256]	[0.4]
MobileNet-V1	0.00048	8	Adam	Yes	1	[128]	[0.4]
MobileNet-V2	0.00057	8	SGD	No	2	[64, 32]	[0.1, 0.0]
MobileNet-V3	0.00089	8	SGD	No	3	[128, 64, 32]	[0.3, 0.0, 0.3]
NasNetMobile	0.00096	8	SGD	No	2	[256, 128]	[0.2, 0.2]
ResNet50	0.00092	8	Adam	Yes	1	[64]	[0.2]
EfficientNet-B0	0.00050	8	Adam	No	1	[128]	[0.1]

A. Explainable Artificial Intelligence Using Gradient-Weighted Class Activation Mapping (Grad-CAM)

To interpret the decision-making behavior of the CNN, several XAI techniques were considered, including gradient-based attribution methods such as Integrated Gradients [28], as well as perturbation-based methods such as LIME [29] and SHAP [30]. Among these alternatives, Gradient-weighted Class Activation Mapping (Grad-CAM) [31] was selected as the primary interpretability technique due to its strong compatibility with CNN architectures and its ability to generate spatially meaningful visual explanations [32].

Grad-CAM offers several practical advantages for CNNs used in visual recognition tasks. First, it preserves the spatial structure of the convolutional feature maps, enabling intuitive visualization of class-discriminative regions. Second, it requires only a single forward and backward pass through the network, making it computationally efficient compared to perturbation-based approaches. Third, it produces explanations that are directly aligned with the internal feature representations learned by the network. Grad-CAM operates directly on the feature maps of convolutional layers and therefore exploits the spatial structure learned by the network [33]. This property makes it particularly suitable for image

classification tasks where the objective is to identify which regions of the input image contribute most strongly to the model's prediction. The resulting activation maps allow qualitative assessment of whether the trained model focuses on relevant regions when predicting the target classes [34].

III. RESULTS AND DISCUSSION

In this study, the performance of the models is assessed using standard performance metrics, and the results are presented in Table II.

TABLE II. CLASS-WISE AND MEAN PERFORMANCE METRICS OF THE PRE-TRAINED CNN MODELS FOR RAIL SURFACE DEFECT CLASSIFICATION

Model	Defect class	Accuracy	Precision	Recall	F1-score
Inception-V3	Flaking	0.8046	0.7	0.7241	0.7119
	Spalling	0.8046	0.6875	0.7586	0.7213
	Squat	0.8161	0.76	0.6552	0.7037
	Mean value	0.8084	0.7158	0.7126	0.7123
MobileNet-V1	Flaking	0.7241	0.5862	0.5862	0.5862
	Spalling	0.7931	0.6897	0.6897	0.6897
	Squat	0.8161	0.7241	0.7241	0.7241
	Mean value	0.7778	0.6667	0.6667	0.6667
MobileNet-V2	Flaking	0.8391	0.8261	0.6552	0.7308
	Spalling	0.8046	0.6765	0.7931	0.7302
	Squat	0.8046	0.7	0.7241	0.7119
	Mean value	0.8161	0.7342	0.7241	0.7243
MobileNet-V3	Flaking	0.7586	0.6539	0.5862	0.6182
	Spalling	0.8046	0.7143	0.6897	0.7018
	Squat	0.7931	0.6667	0.7586	0.7097
	Mean value	0.7854	0.6783	0.6782	0.6765
NasNetMobile	Flaking	0.7471	0.6296	0.5862	0.6071
	Spalling	0.7701	0.6667	0.6207	0.6429
	Squat	0.7701	0.6364	0.7241	0.6774
	Mean value	0.7625	0.6442	0.6437	0.6425
ResNet50	Flaking	0.7241	0.6191	0.4483	0.52
	Spalling	0.7471	0.8889	0.2759	0.4211
	Squat	0.6092	0.4561	0.8966	0.6047
	Mean value	0.6935	0.6547	0.5402	0.5152
EfficientNet-B0	Flaking	0.7816	0.6923	0.6207	0.6546
	Spalling	0.7816	0.6471	0.7586	0.6984
	Squat	0.7701	0.6667	0.6207	0.6429
	Mean value	0.7778	0.6687	0.6667	0.6653

As shown in Figure 1, the three classes contain an equal number of test samples; therefore, the mean value reported in Table II is calculated as the average of the values obtained for these classes. Table II indicates that MobileNet-V2 yields the best performance in terms of accuracy and F1-score, followed by Inception-V3 and MobileNet-V3, respectively, whereas ResNet50 exhibits the poorest performance. A closer examination of Table II shows that MobileNet-V2 yields the best results, particularly for the flaking class. As reflected by the precision value, MobileNet-V2 achieves high precision for the flaking class, indicating that its flaking predictions are generally accurate. However, its lower recall value suggests that a considerable number of actual flaking defects are incorrectly classified. For the spalling class, precision is low whereas recall is high, suggesting that MobileNet-V2 is more prone to misclassifying samples as spalling, although it does not tend to overlook actual spalling defects. For the squat class, the absence of a large difference between precision and recall suggests that the rate of misclassifying actual squat defects as other classes is approximately similar to the rate of classifying non-squat defects as squat.

A. Computational Complexity–Accuracy Analysis

Figure 4 illustrates the relationship between computational complexity in terms of Floating-Point Operations (FLOPs) and classification accuracy for several pre-trained CNN architectures on the dataset. Here, the x-axis represents FLOPs on a logarithmic scale, the y-axis represents accuracy, and the marker size is proportional to the number of model parameters.

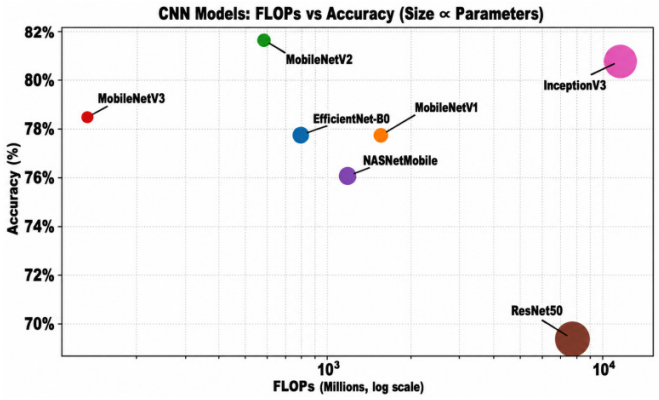


Fig. 4. Computational complexity and classification accuracy for pre-trained CNN architectures on the railway dataset.

The results indicate noticeable differences among architectures in terms of the trade-off between computational cost and predictive performance. MobileNet-V3 exhibits the lowest computational cost with approximately 132M FLOPs while achieving an accuracy of 78.5%. This result demonstrates the high computational efficiency of MobileNet-V3. MobileNet-V2 achieves 81.6% accuracy with approximately 599M FLOPs, providing a highly favorable balance between computational cost and predictive performance. MobileNet-V1 requires about 1.13B FLOPs but attains an accuracy of 77.8%, which is slightly lower than MobileNet-V2. This observation highlights the efficiency improvements introduced in the later MobileNet architectures. EfficientNet-B0 achieves 77.8% accuracy with approximately 787M FLOPs. Although it provides a reasonable trade-off between parameter efficiency and performance, its accuracy is slightly lower than that of MobileNet-V2. NasNetMobile requires approximately 1.14B FLOPs and achieves 76.2% accuracy. Despite having a computational cost comparable to MobileNet-V1, its classification performance is somewhat lower. ResNet50 requires approximately 7.73B FLOPs while achieving 69.3% accuracy, which is among the lowest accuracies observed in this comparison. This result suggests that deeper architectures do not necessarily guarantee improved performance for the evaluated dataset. Inception-V3 exhibits the highest computational cost with approximately 11.4B FLOPs, achieving 80.8% accuracy. While the accuracy is relatively high, the computational requirement is nearly 20 times larger than that of MobileNet-V2.

The evaluated architectures can also be interpreted in terms of Pareto efficiency, where an architecture is considered

Pareto-optimal if no other model achieves higher accuracy with equal or lower computational cost. The results suggest that MobileNet-V2 and MobileNet-V3 lie close to the Pareto frontier, as they provide competitive accuracy while maintaining significantly lower computational complexity compared to larger architectures. In particular, MobileNet-V2 achieves the highest accuracy among the low-FLOPs models, indicating an excellent balance between efficiency and performance. MobileNet-V3, although slightly less accurate, achieves this performance with extremely low computational requirements, making it particularly attractive for resource-constrained environments.

These benchmarks were performed on an AMD Ryzen 7 5825U with Radeon Graphics, and 64 GB of RAM. From a practical perspective, to better understand the actual computational burdens, we also report mean inference times and FPS values of models that lie close to the Pareto frontier in Table III.

TABLE III. INFERENCE TIMES AND FPS VALUES OF TWO NEAR-PARETO-OPTIMAL MODELS

Model	Mean inference time (ms)	FPS
MobileNet-V2	13.520	73.96
MobileNet-V3	8.936	111.90

B. Gradient-Weighted Class Activation Mapping (Grad-CAM) Visualization Pipeline for MobileNet-V2 and MobileNet-V3 Classifiers

To improve the interpretability of the trained MobileNet-V2 and MobileNet-V3 classifiers, a Grad-CAM analysis pipeline was implemented in TensorFlow/Keras. To interpret the decision-making behavior of the trained MobileNet-V2 classifier, Grad-CAM was applied to the test images. Here, all images are resized to the model input size and normalized using the standard MobileNet-V2 preprocessing function. For each input, the last convolutional feature layer is automatically identified, and the gradient of the target class score with respect to this layer is computed using automatic differentiation. Channel-wise importance weights are obtained by global average pooling of the gradients, and these weights are combined with the convolutional feature maps to generate a class activation map. After ReLU filtering and min–max normalization, the activation map is resized to the original image resolution and overlaid on the corresponding RGB image. For each test sample belonging to the selected target class, two Grad-CAM maps were produced: one for the predicted class and one for the ground-truth class. The resulting visualizations were saved as three-panel figures containing the original image, the predicted-class Grad-CAM, and the true-class Grad-CAM, enabling qualitative comparison of the model's attended regions in both correct and incorrect predictions.

For the MobileNet-V3-based classifier, the Grad-CAM implementation required several architectural adaptations due to the nested backbone structure used in the trained model. Unlike the MobileNet-V2 implementation where the convolutional feature maps could be accessed directly from the top-level model, the MobileNet-V3 network stores its convolutional backbone as an internal submodule. Therefore, the Grad-CAM pipeline was modified to explicitly access the backbone network and compute gradients within this nested architecture. To ensure that the selected Grad-CAM layer provides meaningful visual explanations, several candidate convolutional layers within the MobileNetV3Small backbone were empirically evaluated. Based on this empirical analysis, the 'Conv_1' layer was selected as the Grad-CAM target layer for all experiments reported in this study. This choice ensures that the generated activation maps capture high-level class-discriminative information while still preserving sufficient spatial resolution for accurate localization of defect regions.

C. Gradient-Weighted Class Activation Mapping (Grad-CAM) Visualizations

We further analyzed the input images from each class which MobileNet-V2 predicted correctly but MobileNet-V3 did not, or vice versa, using Grad-CAM, thus exploiting spatial attention behaviors of both models on the dataset. Figures 5 through 7 present these visualizations.

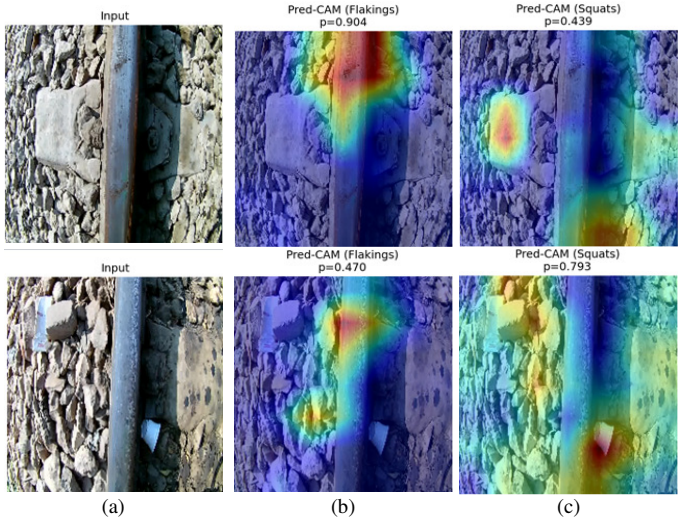
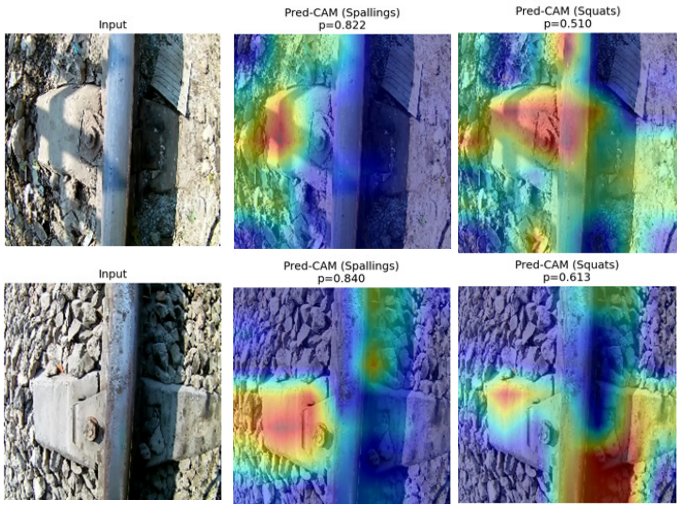


Fig. 5. Mobilenet-v2 and Mobilenet-v3 visualizations for flaking class using Grad-CAM: (a) input (flaking class), (b) MobileNet-V2 prediction, (c) MobileNet-V3 prediction.

The Grad-CAM visualizations provide important insights into the spatial attention mechanisms of both MobileNet-V2 and MobileNet-V3 models for the flaking class, as given in Figure 5. In the first input, MobileNet-V2 correctly classifies the sample by focusing on the true defect region, suggesting that its learned feature representations align well with the discriminative regions of the class.

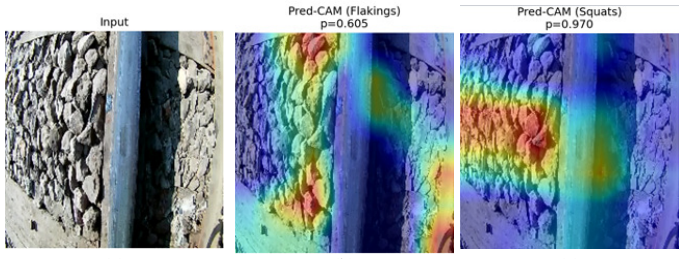
In contrast, MobileNet-V3 misclassifies the same input due to attention being directed toward non-relevant regions, indicating a failure in spatial feature attribution. In the second input, MobileNet-V2 partially attends to the defect region and produces a correct prediction, demonstrating that partial localization is sufficient for correct classification. MobileNet-V3 again focuses on irrelevant regions, resulting in misclassification.



(a) (b) (c)

Fig. 6. Mobilenet-v2 and Mobilenet-v3 visualizations for spalling class using Grad-CAM: (a) input (spalling class), (b) MobileNet-V2 prediction, (c) MobileNet-V3 prediction.

For the inputs in the spalling class, the MobileNet-V2 model produces a correct classification; however, the Grad-CAM visualization reveals that the model focuses on irrelevant regions rather than the actual defect area, as given in Figure 6. This indicates that the prediction is not based on meaningful features but rather on spurious correlations present in the input. In contrast, the MobileNet-V3 model results in a misclassification, and its Grad-CAM map similarly highlights incorrect regions of the image. This suggests that the model fails both in feature localization and decision-making, likely due to insufficient or misleading feature extraction.



(a) (b) (c)

Fig. 7. Mobilenet-v2 and Mobilenet-v3 visualizations for squat class using Grad-CAM: (a) input (squat class), (b) MobileNet-V2 prediction, (c) MobileNet-V3 prediction.

On the other hand, for the input in the squat class, unlike the spalling class, the MobileNet-V3 model produces a correct classification; however, the model focuses on irrelevant regions rather than the actual defect area, as given in Figure 7. In contrast, the MobileNet-V2 model results in a misclassification, and its Grad-CAM map similarly highlights incorrect regions of the image.

Overall, both models demonstrate a lack of consistent attention to defect-relevant regions, indicating that they have not sufficiently learned robust spatial representations for the spalling and squat classes. This highlights the importance of incorporating explainability tools such as Grad-CAM to diagnose model weaknesses beyond standard accuracy metrics.

IV. CONCLUSIONS

This study presents a comparative investigation of seven state-of-the-art pre-trained Convolutional Neural Network (CNN) architectures for rail surface defect classification, focusing on three defect categories: flaking, spalling, and squat. By leveraging transfer learning with frozen convolutional backbones and Optuna-based hyperparameter optimization of the classifier head, the study establishes a systematic framework for evaluating both predictive performance and computational efficiency.

Among the evaluated architectures, MobileNet-V2 achieved the best overall classification performance, yielding the highest mean accuracy and F1-score, while Inception-V3 and MobileNet-V3 followed closely in predictive performance. However, when computational complexity was jointly considered, the Floating-Point Operations (FLOPs)–accuracy Pareto analysis demonstrated that MobileNet-V2 and MobileNet-V3 provide the most favorable trade-off between efficiency and performance, making them particularly suitable for deployment in real-time and resource-constrained railway inspection systems.

The Gradient-weighted Class Activation Mapping (Grad-CAM)-based explainability analysis further provided important qualitative insights into the spatial attention behavior of the Pareto-optimal models. The visualizations showed that MobileNet-V2 generally exhibited more reliable localization of defect-relevant regions, particularly for the flaking class, which is consistent with its superior quantitative performance.

In contrast, both MobileNet-V2 and MobileNet-V3 occasionally relied on irrelevant image regions for spalling and squat predictions, revealing the presence of spurious correlations and highlighting limitations that are not apparent from standard performance metrics alone. These findings emphasize the importance of integrating Explainable Artificial Intelligence (XAI) techniques into railway defect classification pipelines to assess not only how accurately a model predicts, but also why it reaches its decisions.

Consequently, using the dataset described in [3], the results suggest that MobileNet-based architectures offer the most practical solution for intelligent railway inspection applications, where both high inference speed and reliable defect recognition are critical. For future work, incorporating different datasets and object-detection-based frameworks may further improve localization robustness and class discrimination performance in complex real-world railway environments. Also, the visualization results indicate that areas beyond the rail head may influence the classification decisions.

Future studies may also focus on non-rail-head regions in order to evaluate the effect of this information on the model decisions. Finally, to enhance the practical applicability of the proposed approach, future work may also consider evaluating the degree of defects in addition to focusing on defect presence or absence.

DECLARATION OF COMPETING INTERESTS

The authors declare that they have no competing interests.

ACKNOWLEDGMENT

This research received no grant from any funding agency.

DATA AVAILABILITY

The dataset used in this study is available from [16] and is further discussed in [3].

REFERENCES

[1] A. K. Chhotu and S. K. Suman, "Predicting the Severity of Accidents at Highway Railway Level Crossings of the Eastern Zone of Indian Railways using Logistic Regression and Artificial Neural Network Models," *Engineering, Technology & Applied Science Research*, vol. 14, no. 3, pp. 14028–14032, June 2024, https://doi.org/10.48084/etasr.7011.

[2] T. Kara and M. C. Savas, "Design and Simulation of a Decentralized Railway Traffic Control System," *Engineering, Technology & Applied Science Research*, vol. 6, no. 2, pp. 945–951, Apr. 2016, https://doi.org/10.48084/etasr.631.

[3] A. Arain, S. Mehran, M. Z. Shaikh, D. Kumar, B. S. Chowdhry, and T. Hussain, "Railway track surface faults dataset," *Data in Brief*, vol. 52, Feb. 2024, Art. no. 110050, https://doi.org/10.1016/j.dib.2024.110050.

[4] F. Afonso *et al.*, "Surface defect detection systems for railway components," *Procedia Structural Integrity*, vol. 54, pp. 545–552, Jan. 2024, https://doi.org/10.1016/j.prostr.2024.01.117.

[5] Y. Wang, B. Miao, Y. Zhang, Z. Huang, and S. Xu, "A Novel Rail Damage Fault Detection Method for High-Speed Railway," *Sensors*, vol. 25, no. 10, May 2025, Art. no. 3063, https://doi.org/10.3390/s25103063.

[6] T. A. Alvarenga, A. L. Carvalho, L. M. Honorio, A. S. Cerqueira, L. M. A. Filho, and R. A. Nobrega, "Detection and Classification System for Rail Surface Defects Based on Eddy Current," *Sensors*, vol. 21, no. 23, Dec. 2021, Art. no. 7937, https://doi.org/10.3390/s21237937.

[7] A. A. Shah, B. S. Chowdhry, T. D. Memon, I. H. Kalwar, and J. A. Ware, "Real Time Identification of Railway Track Surface Faults using Canny Edge Detector and 2D Discrete Wavelet Transform," *Annals of Emerging Technologies in Computing*, vol. 4, no. 2, pp. 53–60, Apr. 2020, https://doi.org/10.33166/AETiC.2020.02.005.

[8] C. ÖzdemiR and Y. Kaya, "Ray Bileşenlerinde Meydana Gelen Arızaların Görüntü İşleme Teknikleri ile Tespit Edilmesi," *Bilişim Teknolojileri Dergisi*, vol. 14, no. 1, pp. 105–113, Jan. 2021, https://doi.org/10.17671/gazibtd.762853.

[9] M. Wang, K. Li, X. Zhu, and Y. Zhao, "Detection of Surface Defects on Railway Tracks Based on Deep Learning," *IEEE Access*, vol. 10, pp. 126451–126465, 2022, https://doi.org/10.1109/ACCESS.2022.3224594.

[10] A. Kumar and S. P. Harsha, "A systematic literature review of defect detection in railways using machine vision-based inspection methods," *International Journal of Transportation Science and Technology*, vol. 18, pp. 207–226, June 2025, https://doi.org/10.1016/j.ijtst.2024.06.006.

[11] J. H. Feng, H. Yuan, Y. Q. Hu, J. Lin, S. W. Liu, and X. Luo, "Research on deep learning method for rail surface defect detection," *IET Electrical Systems in Transportation*, vol. 10, no. 4, pp. 436–442, Nov. 2020, https://doi.org/10.1049/iet-est.2020.0041.

[12] Y. Zhang, T. Feng, Y. Song, Y. Shi, and G. Cai, "An Improved Target Network Model for Rail Surface Defect Detection," *Applied Sciences*, vol. 14, no. 15, Aug. 2024, Art. no. 6467, https://doi.org/10.3390/app14156467.

[13] W. Yaodong, Y. Hang, G. Baoqing, S. Hongmei, and Y. Zujun, "Research on Real-Time Detection System of Rail Surface Defects Based on Deep Learning," *IEEE Sensors Journal*, vol. 24, no. 13, pp. 21157–21167, July 2024, https://doi.org/10.1109/JSEN.2024.3402730.

[14] R. Ozdemir and M. Koc, "On the enhancement of semi-supervised deep learning-based railway defect detection using pseudo-labels," *Expert Systems with Applications*, vol. 251, Oct. 2024, Art. no. 124105, https://doi.org/10.1016/j.eswa.2024.124105.

[15] W. Phusakulkajorn, J. Hendriks, Z. Li, and A. Núñez, "Spiking neural network with time-varying weights for rail squat detection," *Applied Soft Computing*, vol. 184, Dec. 2025, Art. no. 113689, https://doi.org/10.1016/j.asoc.2025.113689.

[16] Dileep Kumar, "Railway Track Surface Faults Dataset." Mendeley, Jan. 06, 2022, https://doi.org/10.17632/8HXTGYYXRW.2.

[17] S. Jatoi *et al.*, "From Detection to Diagnosis: Elevating Track Fault Identification with Transfer Learning," in *2024 International Conference on Robotics and Automation in Industry*, Rawalpindi, Pakistan, 2024, pp. 1–6, https://doi.org/10.1109/ICRAI62391.2024.10894696.

[18] J. Zhao, A. W.-L. Yeung, M. Ali, S. Lai, and V. T.-Y. Ng, "CBAM-SwinT-BL: Small Rail Surface Defect Detection Method Based on Swin Transformer With Block Level CBAM Enhancement," *IEEE Access*, vol. 12, pp. 181997–182009, 2024, https://doi.org/10.1109/ACCESS.2024.3509986.

[19] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, "Rethinking the Inception Architecture for Computer Vision," in *2016 IEEE Conference on Computer Vision and Pattern Recognition*, Las Vegas, NV, USA, 2016, pp. 2818–2826, https://doi.org/10.1109/CVPR.2016.308.

[20] A. G. Howard *et al.*, "MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications." arXiv, Apr. 17, 2017, https://doi.org/10.48550/arXiv.1704.04861.

[21] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen, "MobileNetV2: Inverted Residuals and Linear Bottlenecks," in *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, Salt Lake City, UT, USA, 2018, pp. 4510–4520, https://doi.org/10.1109/CVPR.2018.00474.

[22] A. Howard *et al.*, "Searching for MobileNetV3," in *2019 IEEE/CVF International Conference on Computer Vision*, Seoul, Korea (South), 2019, pp. 1314–1324, https://doi.org/10.1109/ICCV.2019.00140.

[23] B. Zoph, V. Vasudevan, J. Shlens, and Q. V. Le, "Learning Transferable Architectures for Scalable Image Recognition," in *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, Salt Lake City, UT, USA, 2018, pp. 8697–8710, https://doi.org/10.1109/CVPR.2018.00907.

[24] K. He, X. Zhang, S. Ren, and J. Sun, "Deep Residual Learning for Image Recognition," in *2016 IEEE Conference on Computer Vision and Pattern Recognition*, Las Vegas, NV, USA, 2016, pp. 770–778, https://doi.org/10.1109/CVPR.2016.90.

[25] M. Tan and Q. Le, "EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks," in *Proceedings of the 36th International Conference on Machine Learning*, Long Beach, CA, USA, 2019, pp. 6105–6114.

[26] T. Akiba, S. Sano, T. Yanase, T. Ohta, and M. Koyama, "Optuna: A Next-generation Hyperparameter Optimization Framework," in *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, Anchorage, AK, USA, 2019, pp. 2623–2631, https://doi.org/10.1145/3292500.3330701.

[27] M. Imani, A. Beikmohammadi, and H. R. Arabnia, "Comprehensive Analysis of Random Forest and XGBoost Performance with SMOTE, ADASYN, and GNUS Under Varying Imbalance Levels," *Technologies*, vol. 13, no. 3, Mar. 2025, Art. no. 88, https://doi.org/10.3390/technologies13030088.

[28] M. Sundararajan, A. Taly, and Q. Yan, "Axiomatic attribution for deep networks," in *Proceedings of the 34th International Conference on Machine Learning*, Sydney, Australia, 2017, pp. 3319–3328.

[29] M. T. Ribeiro, S. Singh, and C. Guestrin, "'Why Should I Trust You?': Explaining the Predictions of Any Classifier," in *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, San Francisco, CA, USA, 2016, pp. 1135–1144, https://doi.org/10.1145/2939672.2939778.

[30] S. M. Lundberg and S.-I. Lee, "A Unified Approach to Interpreting Model Predictions," in *31st Conference on Neural Information Processing Systems*, Long Beach, CA, USA, 2017, pp. 4768–4777.

[31] R. R. Selvaraju, M. Cogswell, A. Das, R. Vedantam, D. Parikh, and D. Batra, "Grad-CAM: Visual Explanations from Deep Networks via Gradient-Based Localization," in *2017 IEEE International Conference*

on Computer Vision, Venice, Italy, 2017, pp. 618–626, https://doi.org/10.1109/ICCV.2017.74.

[32] S. Balanageshwara, V. Kumara, M. Badiger, and A. Naik, "Explainable AI for Precise Leaf Disease Diagnosis: A Comparative Study," *Engineering, Technology & Applied Science Research*, vol. 16, no. 2, pp. 33806–33812, Apr. 2026, https://doi.org/10.48084/etasr.16335.

[33] A. Chattopadhay, A. Sarkar, P. Howlader, and V. N. Balasubramanian, "Grad-CAM++: Generalized Gradient-Based Visual Explanations for Deep Convolutional Networks," in *2018 IEEE Winter Conference on Applications of Computer Vision*, Lake Tahoe, NV, USA, 2018, pp. 839–847, https://doi.org/10.1109/WACV.2018.00097.

[34] P. Rajpurkar and M. P. Lungren, "The Current and Future State of AI Interpretation of Medical Images," *Obstetrical & Gynecological Survey*, vol. 78, no. 11, pp. 634–635, Nov. 2023, https://doi.org/10.1097/01.ogx.0000996796.25345.c8.