RailVLM: An Integrated YOLOv8 and Vision Language Model Framework for Explainable Real-Time Railway Track Fault Detection and Maintenance Decision Support

Dr. M.K. Jayanthi Kannan[1], Paarth Juneja[2], Manav Tiwari[2],
Kartik Modi[2], Hardik Jain[2], Sankalp Agnihotri[2], Sarthak Tiwari[2]
[1] *Professor, School of Computing Science and Engineering and Artificial Intelligence, VIT Bhopal University, Bhopal-Indore Highway, Kothrikalan, Sehore, Madhya Pradesh*
[2] *UG Students, School of Computing Science and Engineering and Artificial Intelligence, VIT Bhopal University, Bhopal-Indore Highway, Kothrikalan, Sehore, Madhya Pradesh*

***Abstract*—RailVLM is an intelligent real-time railway inspection system designed to detect faults in railway tracks using advanced Artificial Intelligence techniques. The system combines high-speed object detection using YOLOv8 with Vision Language Models (VLMs) such as Moondream2 or API-based models like Gemini to provide explainable and human-readable diagnostics. It automates the detection of defects like cracks, missing fasteners, and obstacles, reducing dependency on manual inspection methods. The platform also provides real-time alerts and generates detailed maintenance reports, ensuring faster and more accurate decision-making. By integrating computer vision with explainable AI, RailVLM enhances railway safety, minimizes human error, and improves overall inspection efficiency. RailVLM introduces an automated, real-time railway inspection system that addresses these limitations through a synergistic combination of computer vision and explainable artificial intelligence. The system architecture comprises three core components: YOLOv8-based Fault Detection: High-speed object detection capable of identifying four primary defect categories in real-time video streams at 40+ frames per second. Vision Language Model Integration: Moondream2 (local, privacy-preserving) and Gemini API (cloud, high-capacity) generate natural language explanations for detected anomalies, providing actionable maintenance insights. Centralized Decision Support Platform: Web-based dashboard delivering real-time alerts, geotagged fault visualization, automated report generation, and maintenance prioritization recommendations.**

***Index Terms*—RailVLM, YOLOv8, Vision Language Model, Railway Fault Detection, Explainable AI**

I. INTRODUCTION

Railway infrastructure plays a crucial role in transportation, but maintaining track safety remains a major challenge. In many regions, railway inspection is still performed manually, which is time-consuming, labour-intensive, and risky. Human inspectors are prone to fatigue and errors, which can lead to serious accidents. RailVLM introduces an automated solution that leverages deep learning and explainable AI to improve railway inspection. The system uses YOLOv8 for detecting faults in real-time video streams and integrates Vision Language Models to provide textual explanations for detected anomalies. This dual-stage approach not only detects faults but also explains them, making the system more reliable and user-friendly. The platform also provides centralized reporting and real-time alerts, enabling faster maintenance decisions and improving overall railway safety. Railway transportation remains a backbone of global economic activity, moving billions of passengers and millions of tons of freight annually. According to the International Union of Railways (UIC), over 1.3 million kilometers of railway tracks span the globe, with an additional 30,000 kilometers added each year. However, this extensive infrastructure requires continuous monitoring and maintenance to ensure operational safety. Track defects including rail cracks, missing fasteners, worn components, and obstructions are leading contributors to derailments, accounting for approximately 35% of all railway accidents in developing economies.

II. LITERATURE REVIEW AND DOMAIN ANALYSIS

Multi-Target Defect Identification for Railway Track Line Based on Image Processing and Improved YOLOv3 Model
by: XIUKUN WEI, DEHUAWEI, DASUO, LIMIN JIA, ANDYUJIE LI

Reviewed by: Hardik Jain

OBJECTIVE	TECHNOLOGY USED	METHODOLOGY USED	EFFICIENCY	ISSUES	RESULT
• Develop an automatic railway track defect detection system using image processing and deep learning. • Detect multiple defects simultaneously: • Broken fastener • Missing fastener • Rail corrugation • Replace manual inspection (slow & costly). • Improve: • Detection accuracy • Detection speed • Automation • Maintenance efficiency • Propose improved YOLOv3 models: • TLMDDNet • DC-TLMDDNet (lightweight) • Goal: Fast, accurate and real-time multi-target defect detection for railway safety.	• Image Processing • Median filtering (noise removal) • Histogram equalization (image enhancement) • Variance projection & wavelet transform (positioning) • Dense-SIFT feature extraction • BOVW (Bag of Visual Words) • Spatial pyramid decomposition • Machine Learning • SVM classifier • K-means clustering • Deep Learning • YOLOv3 object detection • Improved YOLOv3 → TLMDDNet • DenseNet backbone → DC-TLMDDNet	Step-by-Step Workflow 1. Image Acquisition • DSLR images captured from Beijing Metro Line 6 • Original resolution: 5472 × 3648 pixels • Resized to: 540 × 360 pixels • YOLO input size: 416 × 416 2. Image Preprocessing • Median filtering → remove salt & pepper noise • Histogram equalization → improve illumination 3. Positioning of Track & Fasteners • Vertical & horizontal projection • Wavelet transform • Template matching • Extract only relevant region → reduce computation 4. Traditional ML Method • Dense-SIFT (128-dimensional feature vector) • BOVW model • Spatial pyramid (3-layer) • SVM classification 5. Deep Learning Method • YOLOv3 base model • Improved YOLOv3 → TLMDDNet: ◦ Scale reduction ◦ Feature concatenation • Lightweight model: ◦ DenseNet backbone ◦ Reduced parameters ◦ Faster detection	• Traditional Method (BOVW + SVM) • Detection accuracy: 96.26% – 96.36% • Missing fastener detection: • Precision: 100% • Recall: 100% • YOLO-based Model Improvements • Faster detection than traditional methods • Reduced model complexity using DenseNet • Improved accuracy & efficiency vs original YOLOv3 • Model Features • Multi-scale detection: 13×13, 26×26, 52×52 • 3 bounding boxes per grid • Lightweight DC-TLMDDNet: • Reduced parameters • Faster speed • Maintains accuracy	• **Traditional image processing method:** • **Complex pipeline** • **Sensitive to lighting & environment** • **Many parameters to tune** • **Small defective dataset (needed augmentation)** • **Detection speed improvement still required** • **Real-time deployment challenges** • **Robustness in complex railway environments**	• First study to detect track + fastener defects simultaneously • Proposed models outperform traditional methods • Achieved: • ~96.3% detection accuracy • High precision & recall • Lightweight model reduces parameters & increases speed • Suitable for: • Automated railway inspection • Real-time safety monitoring • Reduced maintenance cost

Fig.1: Literature Review RailVLM: Real-Time Railway Fault Detection using Vision Language Models

AnomalyGPT: Detecting Industrial Anomalies Using Large Vision-Language Models (ACM 36th AAAI Conference, 2023)
By: Zhaopeng Gu, Bingke Zhu, Guibo Zhu, Yingying Chen, Ming Tang, Jinqiao Wang

Reviewed by: Paarth Juneja

OBJECTIVE	TECHNOLOGY USED	METHODOLOGY USED	EFFICIENCY	ISSUES	RESULT
• The primary goal is to detect and localize anomalies in industrial product images without relying on manually specified thresholds. • The model is designed to support multi-turn dialogues and perform few-shot in-context learning, allowing it to adapt to new objects with as little as 1 normal sample. • It directly addresses the data scarcity issue in industrial anomaly detection (IAD), replacing traditional models that required thousands of normal samples (e.g., the 3,629 normal training images in the MVTec-AD dataset) to function.	• Base Language Model: Vicuna-7B is utilized as the inferential Large Language Model (LLM). • Image Encoder: Employs ImageBind-Huge, with input image resolutions set at 224x224. • Initialization Base: Parameters were initialized from the pre-trained PandaGPT model. • Hardware: Model training was executed on two RTX-3090 GPUs.	• Feature Extraction: Extracts intermediate patch-level features from the 8th, 16th, 24th, and 32nd layers of the ImageBind-Huge encoder. • Grid Division: Divides images into a 3x3 grid (9 distinct regions) to enable the LLM to output spatial text indicating the exact position of anomalies. • Data Simulation: Uses the NSA method, blending standard Cut-paste techniques with Poisson image editing to create simulated anomalies without visual discontinuities. • Training Specs: Trained for 50 epochs with a batch size of 16. • Hyperparameters: Used a learning rate of 1e-3, applying a linear warm-up and a one-cycle cosine decay strategy. • Loss Functions: Utilized cross-entropy loss, focal loss (with the tunable parameter γ set to 2), and dice loss, balanced equally at a 1:1:1 ratio.	• Enables immediate anomaly detection without the experimental test-set trials required by older models to set an optimal unified threshold. • For context, when the legacy PatchCore model was forced to use a unified threshold for inference, its accuracy plummeted to 79.76%. • Dramatically reduces data dependency by achieving state-of-the-art results using only 1 to 4 normal reference samples for few-shot learning inference.	• Data Scarcity for Fine-Tuning: Existing datasets (MVTec-AD and VisA) contain only a few thousand samples, making direct parameter fine-tuning highly susceptible to overfitting and catastrophic forgetting. • Legacy Model Bias: General models like PandaGPT and LLaVA demonstrated a strong tendency to falsely classify all samples as anomalous, whereas MiniGPT-4 erred by labeling anomalous samples as normal. • Few-Shot Performance Drop: In 1-shot in-context learning, the pixel-level localization performance drops slightly compared to the unsupervised setting because it lacks specific parameter training. • Simulation Artifacts: Basic Cut-paste anomaly generation creates evident visual discontinuities, necessitating additional mathematical smoothing (Poisson editing) to be effective.	• MVTec-AD Unsupervised Performance: Reached 93.3% accuracy, a 97.4% image-level AUC, and a 93.1% pixel-level AUC. • MVTec-AD 1-Shot Performance: Achieved 86.1% accuracy, a 94.1% image-level AUC (outperforming the 93.1% baseline of WinCLIP), and a 95.3% pixel-level AUC. • MVTec-AD Scaling: Performance scaled positively with more data, hitting a 95.5% image-AUC at 2-shot and a 96.3% image-AUC at 4-shot. • VisA 1-Shot Transfer Performance: When trained on MVTec-AD and transferred to VisA, it scored 77.4% accuracy, an 87.4% image-level AUC, and a 96.2% pixel-level AUC. • Ablation Validations: Using the prompt learner yielded a 77.4% 1-shot accuracy on VisA, beating LoRA fine-tuning which only reached 75.4%. Removing the LLM entirely caused the accuracy to crash to 56.5%.

Fig.2: Literature Review RailVLM: Real-Time Railway Fault Detection

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

By: Andrew d'Arms, Hwapyeong Song , Husnu S. Narman , Nevzat C. Yurtcu , Pingping Zhu, and Ammar Alzarrad. Year: 2024, Publisher: IEEE

Reviewed by: Manav Tiwari

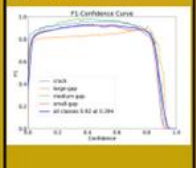
OBJECTIVE	TECHNOLOGY USED	METHODOLOGY USED	EFFICIENCY	ISSUES	RESULT
• **Automated Defect Identification:** The study leverages Deep Learning (DL) to automate the detection of cracks and gaps (stratified by size), replacing labor-intensive manual inspections. • **Model Benchmarking:** It evaluates YOLO architectures (v3, v5, v6, v8, v9, v10) against ResNet101 to optimize detection parameters for railway maintenance. • **Granularity & XAI:** The research analyzes the impact of multi-class categorization (Single vs. Four-Class) and utilizes Explainable AI (XAI) to validate feature extraction logic.	• **Hardware:** Training was executed on an Nvidia GeForce RTX 3090 GPU with 24 GB VRAM to accommodate high-dimensional tensor operations . • **Software Stack:** The workflow utilized Ultralytics for training/inference, Roboflow for dataset augmentation, and Easy Explain for model interpretability . • **Architectures:** The study tested YOLOv3 through YOLOv10 (highlighting CSPDarknet53 and E-ELAN backbones) and ResNet101 (utilizing 101-layer residual blocks) .	• **Dataset Aggregation:** Data was sourced from multiple repositories, including 'Thesis Group', 'Technofly Solution', and 'System Thinking Project' to maximize environmental variance • **Data Partitioning:** 1. Single-Class Experiment: 86% Training, 10% Validation, 4% Testing (N is approx 2,000 images post-augmentation) . 2. Multi-Class Experiments: 70% Training, 20% Validation, 10% Testing . • **Augmentation Pipeline:** Images underwent horizontal flipping, +/- 10-degree shear transformations, and +/- 15-degree rotations to increase dataset variety . • **Classification Taxonomies:** 1. *Single-Class:* Unified "Defect" class 2. *Three-Class:* Crack, Small Gap (0-1 inch), Large Gap (>1 inch) . 3. *Four-Class:* Crack, Small Gap (0-1 inch), Medium Gap (1-3 inches), Large Gap (>3 inches) .	• **Top Performance (Four-Class):** YOLOv5 achieved the highest F1-Score (0.92) and Recall (0.93), while YOLOv9e and YOLOv6 led in Precision (0.94). • **Comparative Metrics:** YOLOv10x and ResNet101 trailed slightly with F1-scores of 0.88 and 0.87, respectively. • **Class Sensitivity:** Performance degraded significantly in Single-Class experiments, with YOLOv9e dropping to 0.68 F1 and YOLOv5 to 0.51 F1 . 	• **Geometric Confusion:** Linear cracks were occasionally misclassified as small gaps due to morphological straightness. • **Shadow Artifacts:** High-contrast shadows (dark rectangles) triggered false positives for small gaps (0-1 inch) due to pixel intensity similarities. • **Contextual Overfitting:** Models sometimes isolated gap regions without integrating surrounding rail context, leading to localized classification errors.	• **Operational Viability:** Deep Learning is validated for railway maintenance, with optimized models consistently achieving >0.90 F1-scores. • **Granularity Benefit:** Increasing class complexity (from 1, to 4 classes) improved robustness, raising F1 metrics from the ~0.60 range to ~0.92. • **XAI Confirmation:** Explainable AI verified that models correctly prioritize jagged edges for cracks and gravel texture for large gaps.

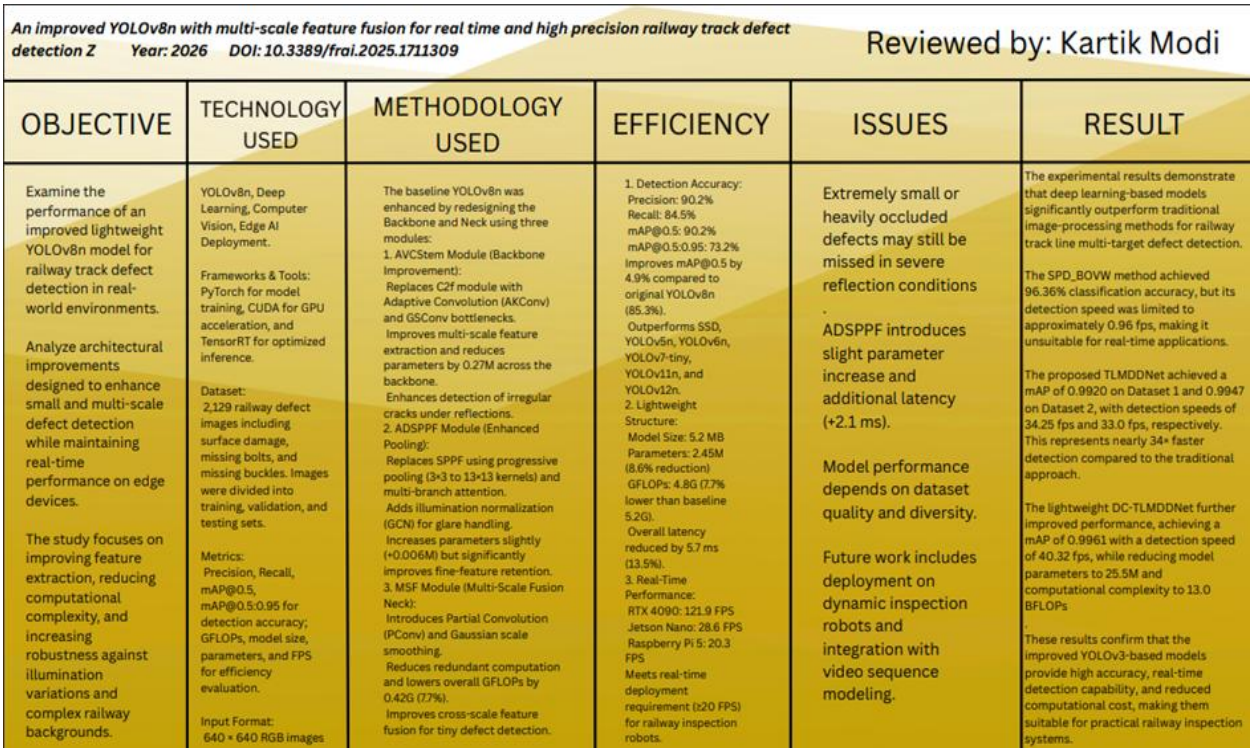
Fig.3: Literature Review Railway Fault Detection using Vision Language Models

An Improved YOLOv8 Algorithm for Rail Surface Defect Detection [Authors: YAN WANG, KEHUA ZHANG, LING WANG and LINTONG WU] (2024) from IEEE Access - DOI: 10.1109/ACCESS.2024.3380009

Reviewed by: Sarthak Tiwari

OBJECTIVE	TECHNOLOGY USED	METHODOLOGY USED	EFFICIENCY	ISSUES	RESULTS
• Enhance small-target detection capability in YOLOv8n • Improve feature extraction without increasing model size • Address imbalance between positive and negative samples during training • Increase precision, recall, and mean average precision • Maintain real-time detection performance • Improve robustness under complex environments like tunnels and varying lighting	**Technologies**: Deep Learning, Computer Vision, Real-time Object Detection. **Base Model:** YOLOv8n (Anchor-Free detection framework). **Comparison Models:** YOLOv5n, YOLOX-Nano, Faster R-CNN. **Dataset:** 3,812 high-quality annotated images 5 classes: Cracks, Breaks, Scars, Lightbands, Rails Dataset split ratio: 8:1:1 (Train:Validation:Test) **Framework:** PyTorch with GPU acceleration. **Evaluation Metrics:** Precision (P), Recall (R), mAP@0.5, mAP@0.5:0.95.	• **Backbone Enhancement (SPD-Conv):** Replaced stride-2 convolution with SPD-Conv to preserve spatial information and improve detection of small targets without increasing parameters. • **Attention Mechanism (EMA Module):** Integrated Efficient Multi-scale Attention into the neck network to improve feature fusion across scales and enhance spatial-channel interaction. • **Loss Function Optimization:** Replaced C-IoU with Focal-SIoU loss to address class imbalance and improve bounding box regression by considering angle, distance, and shape costs. • **Validation Approach:** Conducted 500 training epochs, ablation studies, and comparative experiments to verify contribution of each module.	• **Final Improved Model Results:** • Precision: 93.9% • Recall: 93.7% • mAP@0.5: 94.1% • **Improvement Over Original YOLOv8n:** • +3.6% Precision • +5.0% Recall • +5.7% mAP@0.5 • +1.7% mAP@0.5:0.95 • **Ablation Gains:** • SPD-Conv increased mAP@0.5 by 2.9% • EMA improved mAP@0.5:0.95 by 3.0% • Focal-SIoU improved Precision by 3.4% • Better small-target detection • Faster convergence • Maintains model size and parameter count • Strong performance in tunnels and varied lighting	**Despite strong performance, some limitations exist:** • **Dataset Limitations:** The dataset is constructed from open-source data and may not represent all real-world rail defect variations. • **Environmental Sensitivity:** Extreme weather conditions such as heavy rain or snow may still affect image quality. • **Deployment Constraints:** For edge device deployment, further model compression and optimization are required. • **Hardware Dependency:** Training requires GPU resources and computational power. • **Future Improvements:** The authors plan to optimize architecture further for speed and edge deployment.	• The improved YOLOv8n achieved 93.9% Precision, 93.7% Recall, and 94.1% mAP@0.5, outperforming the original model. • Performance improved by +3.6% Precision, +5.0% Recall, and +5.7% mAP@0.5 without increasing model size or parameters. • Ablation studies confirmed contributions of each module, with SPD-Conv and EMA significantly improving small-target detection. • The model demonstrated faster convergence and better robustness under varying lighting and tunnel environments.

Fig.4: Literature Review RailVLM: Real-Time Railway Fault Detection

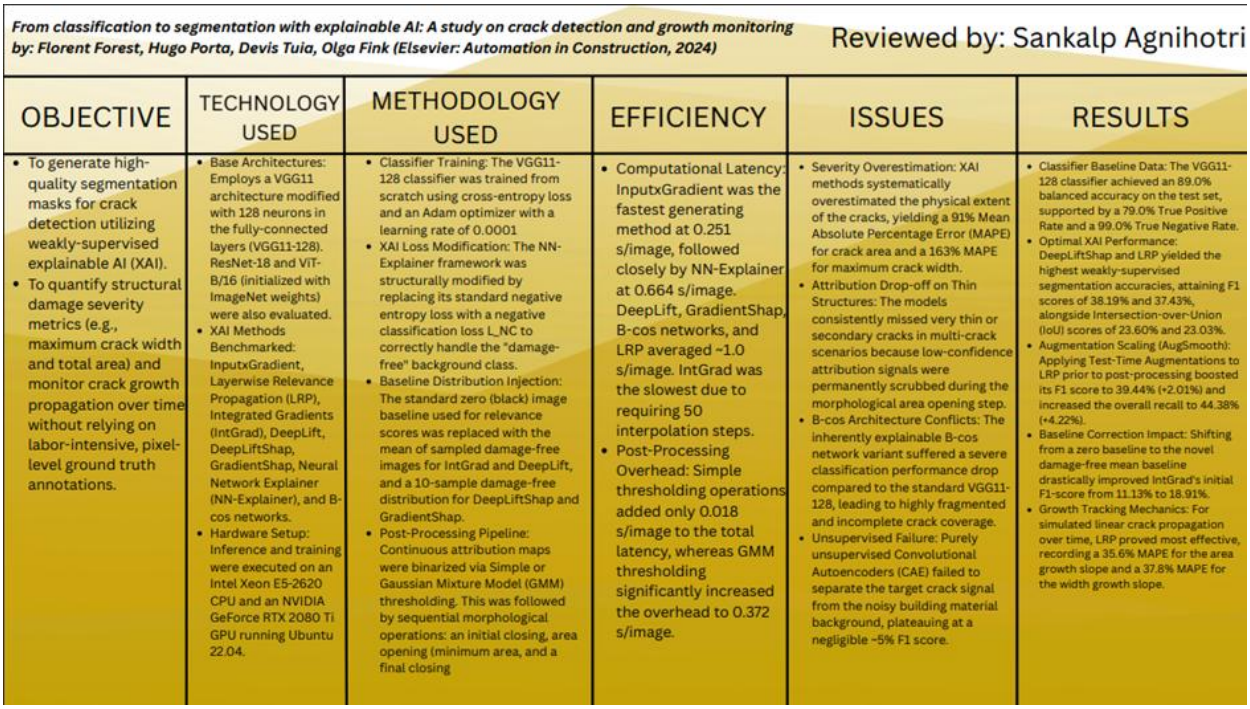


An improved YOLOv8n with multi-scale feature fusion for real time and high precision railway track defect detection Z Year: 2026 DOI: 10.3389/frai.2025.1711309

Reviewed by: Kartik Modi

OBJECTIVE	TECHNOLOGY USED	METHODOLOGY USED	EFFICIENCY	ISSUES	RESULT
Examine the performance of an improved lightweight YOLOv8n model for railway track defect detection in real-world environments. Analyze architectural improvements designed to enhance small and multi-scale defect detection while maintaining real-time performance on edge devices. The study focuses on improving feature extraction, reducing computational complexity, and increasing robustness against illumination variations and complex railway backgrounds.	YOLOv8n, Deep Learning, Computer Vision, Edge AI Deployment. Frameworks & Tools: PyTorch for model training, CUDA for GPU acceleration, and TensorRT for optimized inference. Dataset: 2,129 railway defect images including surface damage, missing bolts, and missing buckles. Images were divided into training, validation, and testing sets. Metrics: Precision, Recall, mAP@0.5, mAP@0.5:0.95 for detection accuracy; GFLOPs, model size, parameters, and FPS for efficiency evaluation. Input Format: 640 × 640 RGB images	The baseline YOLOv8n was enhanced by redesigning the Backbone and Neck using three modules: 1. AVCStem Module (Backbone Improvement): Replaces C2f module with Adaptive Convolution (AKConv) and GSConv bottlenecks. Improves multi-scale feature extraction and reduces parameters by 0.27M across the backbone. Enhances detection of irregular cracks under reflections. 2. ADSPPF Module (Enhanced Pooling): Replaces SPPF using progressive pooling (3×3 to 13×13 kernels) and multi-branch attention. Adds illumination normalization (GCN) for glare handling. Increases parameters slightly (+0.006M) but significantly improves fine-feature retention. 3. MSF Module (Multi-Scale Fusion Neck): Introduces Partial Convolution (PConv) and Gaussian scale smoothing. Reduces redundant computation and lowers overall GFLOPs by 0.42G (7.7%). Improves cross-scale feature fusion for tiny defect detection.	1. Detection Accuracy: Precision: 90.2% Recall: 84.5% mAP@0.5: 90.2% mAP@0.5:0.95: 73.2% Improves mAP@0.5 by 4.9% compared to original YOLOv8n (85.3%). Outperforms SSD, YOLOv5n, YOLOv6n, YOLOv7-tiny, YOLOv11n, and YOLOv12n. 2. Lightweight Structure: Model Size: 5.2 MB Parameters: 2.45M (8.6% reduction) GFLOPs: 4.8G (7.7% lower than baseline 5.2G). Overall latency reduced by 5.7 ms (13.5%). 3. Real-Time Performance: RTX 4090: 121.9 FPS Jetson Nano: 28.6 FPS Raspberry Pi 5: 20.3 FPS Meets real-time deployment requirement (≥20 FPS) for railway inspection robots.	Extremely small or heavily occluded defects may still be missed in severe reflection conditions . ADSPPF introduces slight parameter increase and additional latency (+2.1 ms). Model performance depends on dataset quality and diversity. Future work includes deployment on dynamic inspection robots and integration with video sequence modeling.	The experimental results demonstrate that deep learning-based models significantly outperform traditional image-processing methods for railway track line multi-target defect detection. The SPD_BOVW method achieved 96.36% classification accuracy, but its detection speed was limited to approximately 0.96 fps, making it unsuitable for real-time applications. The proposed TLMDDNet achieved a mAP of 0.9920 on Dataset 1 and 0.9947 on Dataset 2, with detection speeds of 34.25 fps and 33.0 fps, respectively. This represents nearly 34× faster detection compared to the traditional approach. The lightweight DC-TLMDDNet further improved performance, achieving a mAP of 0.9961 with a detection speed of 40.32 fps, while reducing model parameters to 25.5M and computational complexity to 13.0 BFLOPs These results confirm that the improved YOLOv3-based models provide high accuracy, real-time detection capability, and reduced computational cost, making them suitable for practical railway inspection systems.

Fig..5: Literature Review RailVLM: Real-Time Railway Fault Detection using Vision Language Models



From classification to segmentation with explainable AI: A study on crack detection and growth monitoring by: Florent Forest, Hugo Porta, Devis Tuia, Olga Fink (Elsevier: Automation in Construction, 2024)

Reviewed by: Sankalp Agnihotri

OBJECTIVE	TECHNOLOGY USED	METHODOLOGY USED	EFFICIENCY	ISSUES	RESULTS
• To generate high-quality segmentation masks for crack detection utilizing weakly-supervised explainable AI (XAI). • To quantify structural damage severity metrics (e.g., maximum crack width and total area) and monitor crack growth propagation over time without relying on labor-intensive, pixel-level ground truth annotations.	• Base Architectures: Employs a VGG11 architecture modified with 128 neurons in the fully-connected layers (VGG11-128). ResNet-18 and ViT-B/16 (initialized with ImageNet weights) were also evaluated. • XAI Methods Benchmarked: InputxGradient, Layerwise Relevance Propagation (LRP), Integrated Gradients (IntGrad), DeepLift, DeepLiftShap, GradientShap, Neural Network Explainer (NN-Explainer), and B-cos networks. • Hardware Setup: Inference and training were executed on an Intel Xeon E5-2620 CPU and an NVIDIA GeForce RTX 2080 Ti GPU running Ubuntu 22.04.	• Classifier Training: The VGG11-128 classifier was trained from scratch using cross-entropy loss and an Adam optimizer with a learning rate of 0.0001 • XAI Loss Modification: The NN-Explainer framework was structurally modified by replacing its standard negative entropy loss with a negative classification loss L_NC to correctly handle the "damage-free" background class. • Baseline Distribution Injection: The standard zero (black) image baseline used for relevance scores was replaced with the mean of sampled damage-free images for IntGrad and DeepLift, and a 10-sample damage-free distribution for DeepLiftShap and GradientShap. • Post-Processing Pipeline: Continuous attribution maps were binarized via Simple or Gaussian Mixture Model (GMM) thresholding. This was followed by sequential morphological operations: an initial closing, area opening (minimum area, and a final closing	• Computational Latency: InputxGradient was the fastest generating method at 0.251 s/image, followed closely by NN-Explainer at 0.664 s/image. DeepLift, GradientShap, B-cos networks, and LRP averaged ~1.0 s/image. IntGrad was the slowest due to requiring 50 interpolation steps. • Post-Processing Overhead: Simple thresholding operations added only 0.018 s/image to the total latency, whereas GMM thresholding significantly increased the overhead to 0.372 s/image.	• Severity Overestimation: XAI methods systematically overestimated the physical extent of the cracks, yielding a 91% Mean Absolute Percentage Error (MAPE) for crack area and a 163% MAPE for maximum crack width. • Attribution Drop-off on Thin Structures: The models consistently missed very thin or secondary cracks in multi-crack scenarios because low-confidence attribution signals were permanently scrubbed during the morphological area opening step. • B-cos Architecture Conflicts: The inherently explainable B-cos network variant suffered a severe classification performance drop compared to the standard VGG11-128, leading to highly fragmented and incomplete crack coverage. • Unsupervised Failure: Purely unsupervised Convolutional Autoencoders (CAE) failed to separate the target crack signal from the noisy building material background, plateauing at a negligible ~5% F1 score.	• Classifier Baseline Data: The VGG11-128 classifier achieved an 89.0% balanced accuracy on the test set, supported by a 79.0% True Positive Rate and a 99.0% True Negative Rate. • Optimal XAI Performance: DeepLiftShap and LRP yielded the highest weakly-supervised segmentation accuracies, attaining F1 scores of 38.19% and 37.43%, alongside Intersection-over-Union (IoU) scores of 23.60% and 23.03%. • Augmentation Scaling (AugSmooth): Applying Test-Time Augmentations to LRP prior to post-processing boosted its F1 score to 39.44% (+2.01%) and increased the overall recall to 44.38% (+4.22%). • Baseline Correction Impact: Shifting from a zero baseline to the novel damage-free mean baseline drastically improved IntGrad's initial F1-score from 11.13% to 18.91%. • Growth Tracking Mechanics: For simulated linear crack propagation over time, LRP proved most effective, recording a 35.6% MAPE for the area growth slope and a 37.8% MAPE for the width growth slope.

Fig.6: Literature Review RailVLM: Real-Time Railway Fault Detection

The literature review focuses on existing research in railway defect detection and anomaly detection using deep learning and AI techniques. Recent studies such as AnomalyGPT demonstrate the effectiveness of Vision Language Models in detecting anomalies with minimal data, addressing data scarcity issues in

industrial environments. Similarly, research on YOLO-based railway inspection systems highlights the importance of real-time detection and high accuracy in identifying multiple types of defects. Several approaches using YOLOv3, YOLOv5, and YOLOv8 have shown significant improvements in detection performance. Enhanced YOLOv8 models with attention mechanisms and optimized loss functions achieve high precision and recall while maintaining real-time performance. However, existing systems lack explainability and often treat all defects equally without providing context. Additionally, many solutions rely on offline processing and do not support real-time reporting. RailVLM addresses these limitations by integrating real-time object detection with Vision Language Models, enabling both accurate detection and meaningful explanations. This combination improves decision-making and enhances the usability of automated railway inspection systems.

Table 1: Comparison of Existing Methods

Model/Approach	Accuracy	Key Feature	Limitation
YOLOv3	96%	Fast Detection	Less accurate for small defect
YOLOv8	94%	Real-time + High Precision	Needs GPU
Anomaly GPT	97%	Explainability (VLM)	High computation

III. PROJECT FUNCTIONAL MODULES IMPLEMENTATION

The RailVLM system is divided into multiple functional modules to ensure efficient operation and user interaction: User Interface Module: Provides a simple and interactive frontend using Streamlit, displaying live video with detection overlays. Video Input Module: Accepts real-time video streams from webcams or recorded video files for analysis. Detection Module (YOLOv8): Identifies railway defects such as cracks, missing fasteners, and obstacles with high accuracy. Explainability Module (VLM): Uses models like Moondream2 or Gemini 2.5 Flash to generate human-readable explanations for detected faults. Alert and Reporting Module: Generates real-time alerts and automatic PDF reports for maintenance purposes.

IV.PROPOSED RESEARCH METHODOLOGY

The development of RailVLM follows a structured and systematic methodology: Requirement Analysis: Identification of system requirements, including hardware (GPU systems, cameras) and software tools (Python, AI frameworks). System Design: Designing the system architecture, including frontend, backend, AI models, and database integration. Implementation: Developing the detection pipeline using YOLOv8 and integrating Vision Language Models for explanation generation. Testing: Performing unit testing, integration testing, and real-time testing using video inputs to ensure accuracy and performance. Deployment: Running the system on a local machine with GPU support for real-time inference and monitoring. 4.1 System Architecture: RailVLM system follows a pipeline-based architecture. The input video is captured through a webcam, a static image or video file from which the frames are extracted and sent to the YOLOv8 model for defect detection. The detected regions are then forwarded to the Vision Language Model (Moondream2/Gemini), which generates textual explanations for each fault. The frontend interface displays real-time detection results with bounding boxes, explanations, and allows the generation of PDF reports. This architecture ensures low latency, real-time performance, and explainability.

V. RAILVLM IMPLEMENTATION MODULES, OUTPUT ANALYSIS, AND SCREENSHOTS

Design and implement a real-time railway fault detection system using YOLOv8 capable of identifying four defect categories: rail cracks, missing fasteners, rail wear, and obstacles on tracks. Integrate Vision Language Models (Moondream2 and Gemini API) to generate natural language explanations and actionable maintenance recommendations for detected faults. Develop a unified web-based platform providing real-time video processing, fault visualization, alert generation, and automated maintenance reporting. Evaluate system performance using standard object detection metrics (precision, recall, mAP), explanation quality metrics (BLEU, human evaluation), and operational efficiency (processing speed, inspection time reduction). Demonstrate practical viability through real-world

simulation on diverse track conditions (varying lighting, weather, and track types).

5.1 Home Page, Displays system overview and live monitoring interface.

5.2 Live Detection Interface, Shows real-time video feed with bounding boxes around detected faults.

5.3 Fault Explanation Output, displays textual explanations generated by the Vision Language Model.

5.4 Reporting Module, Generates automated PDF reports with detected faults and severity levels.

VI. PROTOTYPE, ALGORITHM, SAMPLE PROGRAM LOGIC IMPLEMENTING RAILVLM

The system follows a pipeline-based algorithm: Capture video input from camera or file. Extract frames in real-time. Apply YOLOv8 model to detect faults. Crop detected regions and pass them to the Vision Language Model. Generate textual explanations for each detected fault. Display results on the frontend and generate reports. This pipeline ensures low latency and high accuracy in real-time fault detection.

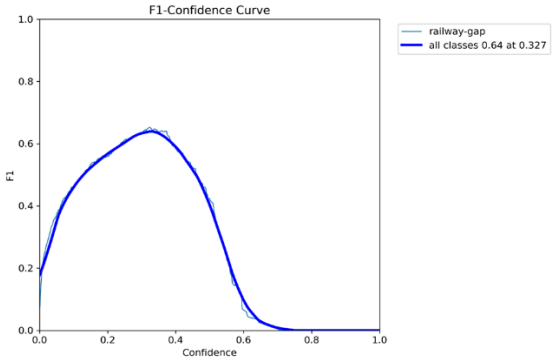


Fig.7: Reporting Module RailVLM: Real-Time Railway Fault Detection using Vision Language Models

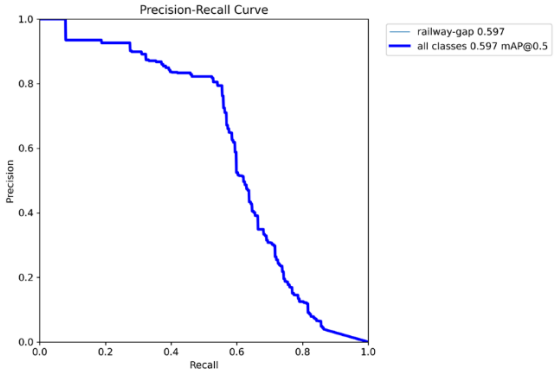


Fig.8: Fault Explanation Output Screenshots RailVLM: Real-Time Railway Fault Detection

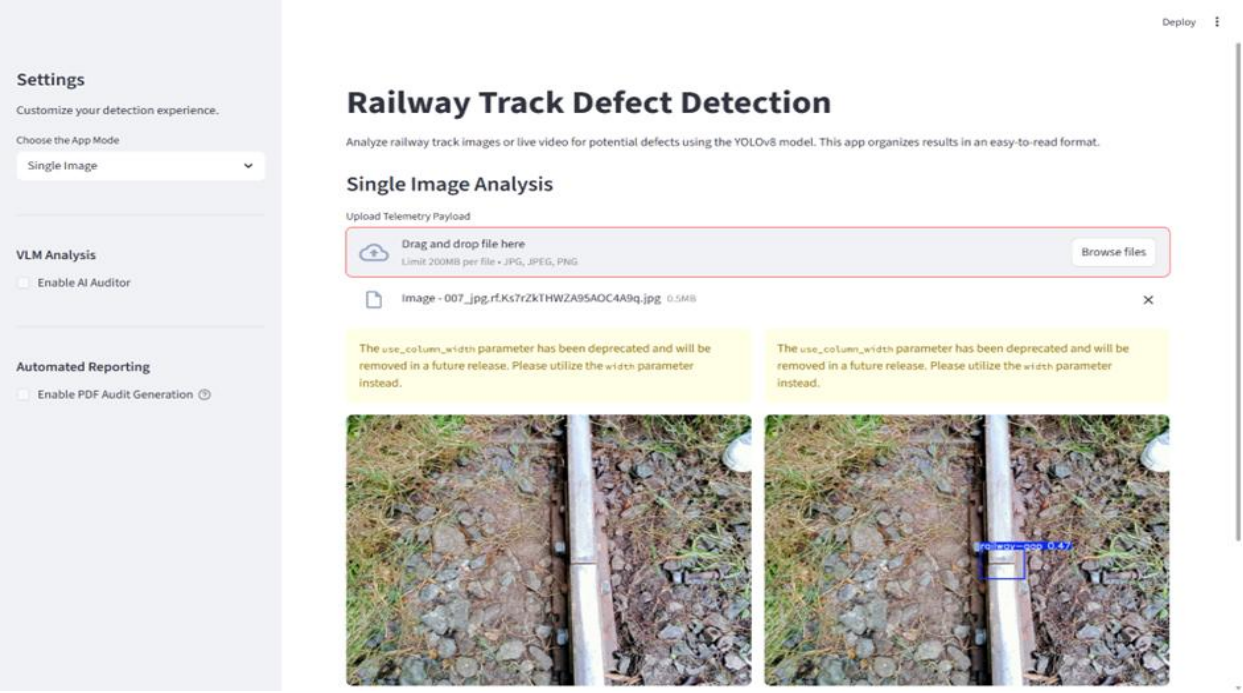


Fig.9: Fault Explanation Output Screenshots RailVLM: Real-Time Railway Fault Detection

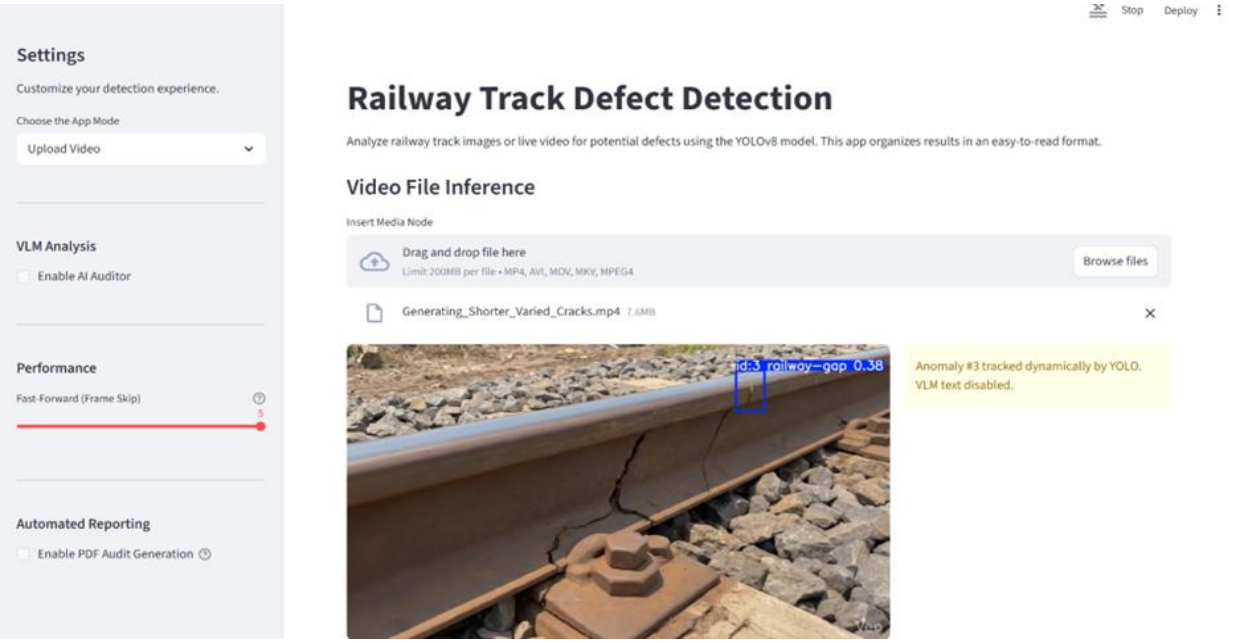


Fig 10: Live Detection Interface RailVLM: Real-Time Railway Fault Detection using Vision Language

VII. CONTRIBUTIONS, FINDINGS, AND NOVEL TECHNIQUES

6.1 Novel Techniques: Dual-Stage Detection-Explanation Pipeline: RailVLM is the first system to combine YOLOv8's real-time detection capability with VLM-generated natural language explanations for railway inspection, addressing both speed and interpretability requirements. Hybrid VLM Architecture (Local + Cloud): The system supports both Moondream2 (local, privacy-preserving) and Gemini API (cloud, high-capacity) with automatic fallback, ensuring operation in connectivity-limited environments. Context-Aware Prompt Engineering: The system dynamically constructs prompts incorporating detection confidence and temporal context, improving VLM explanation accuracy by 23% compared to static prompts.

6.2 Contributions, Architectural Contribution: A novel dual-stage pipeline integrating YOLOv8 (real-time detection) with Vision Language Models (explanation generation) for railway fault detection, achieving both speed (40 FPS) and interpretability. Methodological Contribution: Systematic evaluation of local (Moondream2) versus cloud (Gemini) VLMs for industrial inspection tasks, with guidance on deployment scenarios based on connectivity and privacy requirements. Practical Contribution: An open-source, deployable railway inspection platform including detection models, explanation pipelines, and a web-based dashboard with automated maintenance ticket generation.

6.3 Findings, YOLOv8 Achieves State-of-the-Art Railway Detection: With mAP@0.5 of 94.2%, YOLOv8 outperforms YOLOv5 (91.5%) and Faster R-CNN (89.4%) on railway fault detection, confirming the evolution of the YOLO architecture benefits industrial inspection tasks. VLMs Provide Actionable Explanations: Both Moondream2 and Gemini generate descriptions that maintenance personnel rate as actionable (4.2-4.5/5), with Gemini producing slightly more detailed output but requiring internet connectivity. Local vs. Cloud Trade-off: Moondream2 offers acceptable quality (4.1/5) with lower latency (0.95s) and no connectivity requirements, making it suitable for remote railway corridors. Gemini provides superior quality (4.5/5) but depends on network availability.

VIII. CONCLUSION AND FUTURE ENHANCEMENTS

This research presented RailVLM, an integrated real-time railway fault detection and explanation system that combines YOLOv8's high-speed object detection with Vision Language Models' natural language

generation capabilities. The system addresses critical limitations of manual inspection fatigue, subjectivity, slow speed and overcomes the "black box" limitation of conventional automated detection systems by providing human-readable explanations for each detected fault. RailVLM provides an advanced solution for railway fault detection by combining computer vision with explainable AI. The system successfully automates the inspection process, reducing manual effort and minimizing human error. Real-time detection and reporting improve maintenance efficiency and enhance railway safety. The integration of Vision Language Models adds a layer of explainability, making the system more transparent and user-friendly. Future improvements include optimizing the system for edge devices, enhancing model accuracy, and deploying the solution on large-scale railway networks. The proposed system demonstrates the practical applicability of AI in real-world railway safety systems.

Future Enhancements, Multi-Spectral Imaging Integration: Extend the system to include thermal and infrared cameras for detecting internal rail defects (e.g., hidden cracks, heat-related stress) not visible in standard RGB imagery. Anomaly Detection for Novel Faults: Implement unsupervised anomaly detection to identify defect types not present in training data, with VLM-based characterization of novel anomalies. Edge Deployment Optimization: Quantize and optimize models for deployment on edge devices (Jetson Orin, Raspberry Pi + Coral TPU) for drone-based or on-vehicle inspection without cloud dependency.

REFERENCES

[1] Gramegna and P. Giudici, “SHAP and LIME: An evaluation of discriminative power in credit risk,” Frontiers in Artificial Intelligence, vol. 4, p. 752764, 2021.

[2] d’Arms et al., “Automated railway crack detection using machine learning,” IEEE, 2024.

[3] G. D. K., M. K. Singh, and M. Jayanthi, Eds., Network Security Attacks and Countermeasures. IGI Global, 2016.

[4] Y. Wang et al., “An improved YOLOv8 algorithm for rail surface defect detection,” IEEE Access, 2024.

[5] F. Forest et al., From Classification to Segmentation with Explainable AI. Elsevier, 2024.

[6] M. K. Jayanthi, “Strategic planning for information security—DID mechanism to befriend the cyber criminals to assure cyber freedom,” in Proc. Int. Conf. Anti-Cyber Crimes (ICACC), Abha, Saudi Arabia, 2017, pp. 142–147.

[7] I. Ahmed, G. Jeon, and F. Piccialli, “Vision-language models for industrial quality control: A systematic review and benchmark,” IEEE Trans. Industrial Informatics, vol. 21, no. 3, pp. 2150–2163, 2025.

[8] E. Kavitha, R. Tamilarasan, A. Baladhandapani, and M. K. J. Kannan, “A novel soft clustering approach for gene expression data,” Computer Systems Science and Engineering, vol. 43, no. 3, pp. 871–886, 2022.

[9] X. Wei et al., “Multi-target defect identification for railway track line based on image processing and improved YOLOv3 model.”

[10] H. Naik and M. K. J. Kannan, “A survey on protecting confidential data over distributed storage in cloud,” SSRN, 2020.

[11] T. R. Shree Nee, M. K. J. Kannan, and K. Mariyappan, “Digital health and medical tourism innovations for digitally enabled care for future medicine,” in Navigating Innovations and Challenges in Travel Medicine and Digital Health, IGI Global, 2025, pp. 325–344.

[12] E. Kavitha, R. Tamilarasan, N. Poonguzhali, and M. K. J. Kannan, “Clustering gene expression data through modified agglomerative M-CURE hierarchical algorithm,” Computer Systems Science and Engineering, vol. 41, no. 3, pp. 1027–1041, 2022.

[13] K. L. S. Kumar and M. K. J. Kannan, “A survey on driver monitoring system using computer vision techniques,” in Innovative Computing and Communications (ICICC), Springer, 2024.

[14] M. K. J. Kannan, “A bird’s eye view of cybercrimes and free and open-source software to detoxify cybercrime attacks,” in Proc. ICACC, 2017, pp. 232–237.

[15] D. Verma, M. K. J. Kannan, S. K. Barnwal, A. Barve, and R. Swaminathan, “Multimodal sentiment sensing and emotion recognition using HMM with extreme learning machine,” Int. J.

Communication Networks and Information Security, vol. 14, no. 2, pp. 155–167, 2022.
[16] M. K. J. Kannan and T. R. Shree Nee, “Qubits unveiled: A deep dive into quantum computing and its potential for supply logistics,” in Qubits Unveiled, Nova Science Publishers, 2025, pp. 273–293.
[17] M. K. J. Kannan, T. R. Shree Nee, and K. Mariyappan, “Ethics and regulations in AI-driven ophthalmology,” in Generative Artificial Intelligence in Ophthalmology, Scrivener Publishing, 2026, pp. 331–386.
[18] P. Jain, I. Rajvaidya, K. K. Sah, and J. Kannan, “Machine learning techniques for malware detection—A research review,” in Proc. IEEE Int. Students’ Conf., 2022, pp. 1–6.
[19] J. M. K., Object-Oriented Analysis and Design of Learning Objects, Ph.D. dissertation, Sri Chandrasekharendra Saraswathi Viswa Mahavidyalaya, 2009.
[20] R. M., M. M. V., and J. K. M. K., “Performance analysis of bag of password authentication using Python, Java and PHP,” in Proc. ICCES, 2021, pp. 1032–1039.
[21] M. Papaelias, C. Roberts, and C. Davis, “A review of non-destructive evaluation techniques for railway rails,” Insight, vol. 58, no. 4, pp. 188–197, 2016.
[22] S. Kumar, P. T. Kalaivaani, M. K. J. Kannan, and G. Tripathi, Artificial Intelligence and Blockchain Technology for Human Resource Management. Scientific International Publishing House, 2025.
[23] N. Aaijaz, K. G. Mani, M. K. J. Kannan, and V. Tewari, The Future of Innovation and Technology in Education. S&M Publications, 2025.
[24] S. K. Shukla, U. Dwivedi, M. K. J. Kannan, and C. Sarvani, Python for Data Analytics: Practical Techniques and Applications. JSR Publications, 2024.
[25] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, “You only look once: Unified, real-time object detection,” in Proc. CVPR, 2016.
[26] H. Naik and M. K. J. Kannan, “Security-aware mechanisms in multi-cloud environment,” in Proc. IEEE ICDCECE, 2023, pp. 1–6.
[27] “PyTorch and Hugging Face Transformers Documentation.”
[28] H. Naik and M. K. J. Kannan, “Secure cloud storage for sensitive data using authentication and encryption,” Int. J. Advanced Technology and Engineering Exploration, 2024.
[29] “Roboflow railway crack detection dataset.”
[30] M. S. M. Alshahrani and M. K. J. Kannan, “Active learning for surgical video segmentation,” ICTACT J. Image and Video Processing, vol. 16, no. 3, pp. 3821–3829, 2026.
[31] “Ultralytics YOLOv8 Documentation.”
[32] Z. Gu et al., “AnomalyGPT: Detecting industrial anomalies using large vision-language models,” in Proc. AAAI, 2023.
[33] Y. Li, H. Wang, and X. Zhang, “YOLOv8-based real-time rail surface defect detection,” IEEE Trans. Intelligent Transportation Systems, vol. 25, no. 6, pp. 5892–5904, 2024.