Universiteit Leiden

Master Computer Science

Leveraging Autonomous Train Operation Cameras for Rail Track Monitoring

Name:	Myriana Miltiadous
Student ID:	3699463
Date:	02/12/2025
Specialisation:	Artificial Intelligence
1st supervisor:	Dr. Niki van Stein
2nd supervisor:	Dr. Elena Raponi

Master's Thesis in Computer Science

Leiden Institute of Advanced Computer Science (LIACS)
Leiden University
Niels Bohrweg 1
2333 CA Leiden
The Netherlands

Abstract

Railway infrastructure monitoring has traditionally relied on costly, specialized inspection vehicles. As Automatic Train Operation (ATO) grows across Europe, forward-facing cameras are becoming common on trains. This thesis investigates whether these cameras can reliably detect rail track defects despite uncontrolled operational conditions. We use the OSDaR23 dataset, released by the German Center for Rail Traffic Research (DZSF) for ATO object detection studies, which contains annotated video frames from such cameras but lacks infrastructure defect labels. We contribute: (1) a benchmark dataset of 1,610 binary-labeled track patches with pixel-level masks, extracted from OSDaR23 frames and organized into 71 track spatial sequences, (2) a brightness-based filtering method using adaptive clustering to isolate rail surfaces from such data, (3) a stratified data partitioning strategy preventing spatiotemporal leakage while balancing defect distribution and sequence length.

Through comparison of preprocessing strategies, supervised (ResNet-18, DeiT-Small) and unsupervised (PatchCore) paradigms, and sequential refinement methods, we identify an optimized interpretable system for inspecting data: DeiT-Small with blurred background and CLAHE preprocessing, ensemble averaging, and confidence-weighted isolated removal for post-processing. Gradient-based explainability analysis shows the model concentrates on rail surface irregularities instead of background artifacts. The optimized system achieves an AUPRC of 0.993 on held-out test data, demonstrating that dual-purpose ATO cameras represent a feasible complementary technology for cost-effective early-warning infrastructure monitoring. However, performance variability across cross-validation folds suggests that deployment of the proposed pipeline requires validation across more operational conditions, larger data collection periods, and expert-verified labels.

Contents

Chapter 1

Introduction

1.1 Background and Motivation

Railway infrastructure is critical for transport systems, since it supports the movement of passengers and goods [27, 73]. Specifically, tracks represent the main load-bearing elements of the system and their condition is therefore important to prevent safety incidents and damage to the reputation of operators [42, 16].

Traditional rail track inspection relies on visual scans by human inspectors and sensor-equipped vehicles [20, 18]. However, high costs and logistical demands limit most track sections to annual inspections, creating maintenance gaps that can lead to expensive emergency repairs [20]. Although expansion of railway infrastructure in many networks is limited, existing systems face pressure from increased usage, stricter safety requirements and the rise of high-speed lines [66]. These factors impose more demanding inspection routines on rail track segments than current methods can deliver.

In parallel, rail vehicles increasingly carry cameras and sensors for Automatic Train Operation (ATO), an industry-wide initiative supported by EU-funded research programs [43, 56, 17]. Such ATO systems rely on forward-facing cameras to detect and classify trackside objects and inform real-time driving decisions [61]. Their aim is to enable different levels of train automation, from driver-assisted operation (Grade of Automation 1) to fully autonomous trains (GoA4). As this deployment expands, standardized cameras are expected to become common on rail vehicles in the coming years, providing an opportunity to expand ATO system to include infrastructure monitoring.

This effort, if successful, could result in more frequent and affordable inspections, allowing for earlier detection of defects and thus improving preventive and predictive maintenance capabilities [53, 6]. Additionally, the economic reasons for ATO investments may be stronger, making a better case for their widespread use.

1.2 Research Framework

1.2.1 Problem Context

Despite advances in deep learning for rail defect detection, a critical translation gap persists between research prototypes, often developed on ideal image sets and on-board camera deployments in operational settings [22, 29, 48, 52, 45]. This thesis explores four key aspects of the problem:

(1) Variable viewing angles, lighting, and background elements complicate track surface detection [44, 31, 39].

(2) The prevalent imbalance, with far more 'normal' than 'defective' images, causes traditional supervised models often exhibiting poor recall on minority classes [29].

(3) Many existing methods treat each frame independently and do not exploit the sequential nature of railway imagery. The frames taken as the vehicle moves show slightly different images of same parts of the track. These images include information that can help make detection more reliable [59].

(4) Many state-of-the-art deep learning models are still unclear in their decision-making processes. This limits their adoption by maintenance planners, who require explainable predictions to justify interventions [24].

1.2.2 Research Question and Objectives

This thesis addresses the following research question:

Can forward-facing ATO cameras reliably detect rail track defects, despite the uncontrolled operational conditions?

The experimental work tackles this question through four specific objectives: identifying preprocessing strategies for operational challenges, comparing supervised and unsupervised learning methods under class imbalance, evaluating whether sequential information improves model decisions and performing explainability analysis to show evidence that predictions target real defects.

The hypothesis is that the global context modeling of the transformer paradigm will perform better than CNN, and that sequential refinement methods will improve performance. Additionally, explainability analysis is expected to show that predictions are based on real surface irregularities, rather than false correlations.

1.3 Contributions

In this thesis, we present an end-to-end system that progresses from raw video frames to interpretable defect predictions. To evaluate this, we contribute a novel dataset of 1,610 rail track patches with binary labels and pixel-level defect masks. Patches are extracted from video frames of vehicle-mounted cameras and they form spatial sequences of specific track segments, enabling sequential analysis. To support this process, we propose a novel, multi-stage data preparation method that is directly generalizable to other datasets of this kind. This procedure includes a unique polyline-based patch extraction process and introduces an adaptive brightness-based filtering technique that uses clustering on pixel intensities to isolate the track surface from the background. We also develop a stratified, group-preserving data partitioning procedure that avoids spatiotemporal leakage by keeping all frames from the same track sequence within a single fold while balancing sequence length and defect frequency. Finally, we expand the conventional single-frame classification paradigm by incorporating sequence-aware post-processing methods that, while used in existing literature, have not yet been applied to spatially sequential rail track data.

1.4 Thesis Structure

The remainder of this thesis is organized as follows. Chapter 2 covers the literature connected to this work, Chapter 3 presents the dataset, patch extraction process, and our brightness-based filtering method for background removal. Chapter 4 outlines the experimental methodology, including model architectures and our stratified data partitioning strategy. Chapter 5 details the experimental design and configurations. Chapter 6 reports the results of the conducted experiments and Chapter 7 interprets the results, points out limitations and makes suggestions for future research. Finally, Chapter 8 concludes the thesis.

Chapter 2

Literature Review

This chapter reviews related research on computer vision methods for defect detection and visual inspection, focusing on their application to railway imagery. It covers relevant datasets, preprocessing techniques, deep learning architectures, interpretability methods and sequential modeling approaches, highlighting the research gaps addressed in this thesis.

2.1 Railway Vision Datasets

Existing computer-vision datasets in the railway domain reflect two distinct objectives: autonomous navigation and infrastructure defect detection. This separation shapes both the imagery capture protocols and annotation taxonomies.

Navigation-focused datasets prioritize ego-perspective imagery from operational viewpoints. RailSem19 pioneered public semantic segmentation datasets for rail environments with 8,500 images across 38 countries, annotating track elements, signals, platforms, and buffer stops [72]. Building on this foundation, RailSet introduced 6,600 manually annotated images of normal railway scenes captured from the same ego-perspective using YouTube videos [71]. The dataset includes 1,100 additional images with synthetically generated holes and track discontinuities created by pasting pothole textures onto railway backgrounds using GAN-based augmentation. However, these synthetic obstacles represent collision hazards rather than true rail-surface degradation. More importantly, the image quality and distance from the track surface in these datasets prevent visibility of genuine infrastructure defects.

On the other hand, defect-focused datasets employ close-range inspection protocols. Rail-5k exemplifies this approach, providing over 5,000 high-resolution images (0.03mm/pixel) from specialized inspection systems on high-speed railway and subway infrastructure across China [68]. From these, 1,100 images are annotated across 13 fine-grained defect types, including rail surface defects, wheel-rail contact bands, cracks, spalling, and corrugation. While this close-range, high-resolution perspective enables detailed defect analysis, it cannot validate whether such defects remain detectable from forward-facing cameras.

The Open Sensor Data for Rail 2023 (OSDaR23) addresses the limitations of the aforementioned datasets by providing operational ATO imagery with sufficient quality for defects to be visible [61]. The dataset contains 45 video-frame sequences with 204,000 annotations across 20 classes focused on collision-relevant objects. Although its taxonomy prioritizes immediate operational safety by detecting objects requiring real-time driving responses, its documentation explicitly acknowledges the absence of infrastructure anomaly labels.

This thesis adapts OSDaR23 to address the gap of publicly available ATO camera datasets annotated for rail defect detection. By creating track defect annotations for this operational imagery, we can investigate whether forward-facing ATO cameras reliably detect track defects under uncontrolled conditions. In the following, we therefore turn to the critical task of ensuring data quality in such complicated data through preprocessing.

2.2 Preprocessing and Data Quality

Isolating track surfaces from surrounding infrastructure is critical for reducing false positives in operational imagery. Learning-based methods, utilizing semantic segmentation or rail-region extraction networks, have been employed to identify rail pixels [28, 39]. Beyond the railway domain, strategies such as constant-value replacement or context-preserving blurring of non-relevant regions have been successfully applied in industrial visual inspection and segmentation tasks [4, 34].

Contrast-Limited Adaptive Histogram Equalization (CLAHE) addresses a complementary challenge: enhancing defect visibility. Applied successfully in industrial metal surface inspection, CLAHE improves contrast while limiting noise [40, 51, 47]. This technique is particularly relevant for operational railway imagery where shadows and specular reflections can obscure subtle surface defects [63].

Color space selection represents another preprocessing consideration. Grayscale conversion has been utilized in metal surface inspection to eliminate chromatic noise and illumination-dependent color variations, allowing models to focus on textural patterns characteristic of many defects [19]. This approach provides computational advantages and can facilitate learning of illumination-invariant features, directly related to the varying environmental conditions of operational railway imagery.

Apart from the challenges addressed by these preprocessing techniques, motion blur from vehicle speed presents an additional identified challenge in ATO data [61]. Deblurring methods exist to restore image sharpness in high-speed scenarios, but the imagery examined in this work exhibits minimal motion blur. Consequently, specialized motion restoration methods are not covered here.

2.3 Deep Learning Architectures

While preprocessing techniques improve input data quality, the model architecture plays a complementary role in determining the effectiveness of defect recognition. The literature has explored various deep learning architectures for visual inspection and defect detection. Convolutional neural networks (CNNs) have dominated applications in manufacturing and infrastructure monitoring due to their efficient local feature extraction through hierarchical convolutional layers. ResNet architectures, particularly ResNet-18, have become standard baselines in industrial inspection tasks due to their residual connections, which enable computationally efficient deep network training [25, 32].

More recently, Vision Transformers have emerged as strong alternatives to CNNs in visual inspection tasks. Studies on manufacturing defect benchmarks (MVTec AD) demonstrate that transformers outperform CNNs when defects involve subtle texture deviations, as self-attention mechanisms capture long-range spatial dependencies beyond the limited receptive fields of convolutional filters [41]. For railway infrastructure specifically, RailTrack-DaViT adapts dual attention mechanisms for track defect detection, achieving improved performance over CNN baselines [46]. Additionally, RailFormer demonstrates transformer effectiveness for component-level fastener inspection [23]. These results suggest that global context modeling can benefit rail surface defect detection. Data-efficient Image Transformer (DeiT) introduced improved training strategies, including enhanced data augmentation and optimized training recipes, enabling competitive performance with reduced data requirements compared to standard ViT [62]. The DeiT-S variant provides a representative transformer architecture suitable for evaluation on limited railway defect datasets.

The supervised architectures discussed above rely on labeled defect examples for training, which is the most common approach in railway defect detection. However, the class imbalance in inspection data has motivated research on unsupervised anomaly detection methods that do not require defect labels. PatchCore exemplifies memory-based approaches, using pre-trained CNN backbones to extract features from nominal training samples into a memory bank, then detecting anomalies by measuring feature distance to nearest neighbors [54]. This approach achieved state-of-the-art results on the MVTec AD manufacturing benchmark and has been successfully adapted to railway contexts in Rail-PatchCore, which demonstrates competitive performance using only normal training samples [67]. The memory bank in these methods aims to thoroughly represent the range of nominal changes present in normal data.

2.4 Interpretability Methods in Computer Vision

Complementing architectural advances, research has explored interpretability methods to enhance the transparency of automated systems. Explainable AI (XAI) methods aim to provide visual explanations for model predictions, addressing the ‘black box’ problem of deep neural networks, making them relevant for the railway domain, yet underexplored [46, 28, 32]. Existing XAI approaches can be categorized as architecture-specific or model-agnostic.

Architecture-specific methods leverage internal model structures. For instance, transformer models can use attention weight visualization as a proxy for which image regions the model may be focusing on during classification [23, 15]. Conversely, CNNs employ Class Activation Mapping (CAM) variants to localize discriminative regions by projecting feature map activations back to input space [22]. These methods provide interpretability by design, but require adaptation when changing model architectures.

Gradient-based attribution techniques offer a model-agnostic alternative applicable across architectures without structural modifications. This family of methods generates saliency maps by analyzing how predictions respond to input variations through gradient computation. Simonyan et al. [57] introduced the foundational approach of computing gradients of the prediction with respect to input pixels, identifying regions where small perturbations most affect the output. Sundararajan et al. [60] extended this concept with Integrated Gradients, which addresses gradient saturation by accumulating gradients along an interpolation path from a baseline (typically a black image) to the actual input. Smilkov et al. [58] proposed SmoothGrad as a refinement strategy that reduces visual noise by averaging saliency maps computed over multiple noisy versions of the input image, improving interpretability for human inspection.

Despite their widespread adoption, Adebayo et al. [1] demonstrated that gradient-based methods exhibit limitations: they can be sensitive to implementation details, may highlight imperceptible input features and do not necessarily represent causal relationships between inputs and predictions. Their work emphasizes the importance of applying multiple complementary attribution techniques and validating explanations against domain knowledge. The concurrent use of multiple methods, such as Vanilla Gradients, Integrated Gradients and SmoothGrad, enables identification of consistently highlighted regions (indicating robust feature importance) versus method-specific artifacts (suggesting spurious correlations).

2.5 Sequential Modeling

Rail track imagery captured in consecutive frames exhibits sequential relationships as the vehicle moves forward, which remains largely unexploited in existing railway defect detection research, where frames are typically processed independently. Although consecutive frames represent spatial rather than temporal continuity, they share key characteristics with time-series data: genuine defects usually persist across multiple adjacent frames, while false positives appear as isolated predictions. These characteristics enable the application of temporal modeling techniques to spatially ordered railway data. Leveraging this for reducing false classifications offers practical benefits, as false alarms trigger unnecessary costly inspections and service disruptions, while undetected defects pose serious safety risks [35]. Two main approaches have been applied to temporal data: post-processing methods that refine independent frame predictions, and recurrent architectures that learn sequential dependencies directly.

Post-processing approaches have employed filtering operations like Gaussian kernels, moving averages and median filters to frame-level probabilities for reducing transient artifacts [10]. Research has also explored majority-voting techniques that convert frame-level binary predictions into sequence-level decisions, confirming defects only if they appeared in a threshold proportion of neighboring frames [69].

Recurrent architecture research has investigated learning temporal dependencies directly from sequence data. Studies have utilized Long Short-Term Memory (LSTM) networks that maintain hidden states encoding frame history, enabling predictions to integrate information from previous time steps [26]. This approach has been extended through bidirectional LSTMs that process sequences in both temporal directions [49].

Despite the effectiveness of these approaches in temporal domains, their performance on spatially sequential railway track data remains unvalidated, offering an area for this thesis to explore.

Chapter 3

Dataset and Data Preparation

This chapter describes the features of the chosen source dataset and the novel data preparation methodology used to construct the final sequence-aware defect benchmark dataset.

3.1 OSDaR23 Dataset description

The OSDaR23 dataset [1], introduced in Section 2.1, serves as the source data for this thesis. It is a multi-sensor dataset developed through a collaboration between the German Center for Rail Traffic Research (DZSF), DB Netz AG and FusionSystems GmbH [61].

Data acquisition for OSDaR23 was carried out using a track maintenance vehicle equipped with synchronized (10 Hz) and calibrated sensors mounted on a custom-built structure at the front. These sensors include forward-facing RGB cameras of both high and low resolution, an infrared (IR) camera, multiple LiDAR units, a 2D radar, a GPS receiver and an inertial measurement unit (IMU). As a result, the dataset contains both video frames and rich metadata, including timestamps, sensor identifiers, GPS coordinates, image resolutions and motion-related acceleration measurements.

The specific taxonomy of the dataset includes: (i) infrastructure elements (tracks, signals, signal poles, signal bridges, catenary poles, switches, buffer stops, transitions, dragshoes), (ii) mobile obstacles (people, crowds, road vehicles, trains, wagons, bicycles, motorcycles, animals) and (iii) hazard indicators (flames, smoke). Each object is assigned a unique object ID, allowing it to be consistently tracked across multiple frames. In the case of rail tracks, an additional track ID is provided, which is shared by each connected pair of right and left rails, indicating that they belong to the same track segment.

To manage the annotation data, the RailLabel library [2] was developed and made publicly available together with the dataset. RailLabel is an open source Python library specifically designed for managing and visualizing railway-specific annotations according to a refined subschema of the ASAM OpenLABEL standard. This is an open, widely applicable format developed by the Association for Standardization of Automation and Measuring Systems

[1] https://data.fid-move.de/dataset/osdar23

[2] https://github.com/DSD-DBS/raillabel

(ASAM) [2]. RailLabel specifies and adapts components that are left undefined in the general OpenLABEL standard, tailoring them to the specific requirements of railway applications.

Annotations of the OSDaR23 dataset are provided in JSON format following the RailLabel schema, which supports a variety of geometric representations, including bounding boxes, polygons, polylines and 3D cuboids. Therefore, RailLabel library's importance for this work is twofold: first, it provides a standardized schema for railway object annotations and second, it provides tools for parsing complex JSON annotation files, which are utlised for our data extraction process.

The 45 annotated subsequences of the dataset were collected in Hamburg, Germany, in September 2021 during daylight hours over a seven-day period. They cover 21 stations, with several subsequences recorded at different locations within the same station. The subsequences include both dynamic and static scenes and consist of either 10 or 100 frames corresponding to approximately 1 or 10 seconds of recording time, respectively.

This work uses exclusively the high-resolution RGB camera stream, namely, the forward-facing center camera among the six RGB sensors (Teledyne GenieNano 5GigE C4040, 12 MP resolution) as described in the dataset specification. Additionally, the polyline annotations delineating left and right rail tracks are employed. To enable sequence modeling, we retain only subsequences in which the vehicle is in motion. After this filtering step, 29 of the 45 available subsequences are selected. Each of these 29 subsequences covers a distinct geographical area of the German railway network. Among those, 6 contain 100 frames and 23 contain 10 frames. To define the exact units of analysis for patch extraction, we introduce the term track-specific sub-subsequences, or simply sequences, to refer to the individual rail tracks. The 29 selected geographical subsequences result in a total of 71 track-specific sub-subsequences (10 of 100 frames and 61 of 10 frames). Since one patch is extracted per frame per track (details below), this yields 1,610 frame-level patches in total $(10 \times 100 + 61 \times 10)$.

3.2 Data Preparation and Annotation

The preprocessing pipeline described in this section is ours, designed for this thesis. It aims to prepare the dataset for both defect detection and sequential modeling by extracting ordered image patches along each rail track. Consecutive frames from the same video subsequence form short chronological sequences that preserve the motion continuity of the train along the track. Importantly, this methodology is designed to be generally applicable for any imagery captured by vehicle-mounted cameras. Figure 3.1 provides a high-level overview of the complete pipeline, which consists of two main stages: (i) initial patch extraction with masking (top row) and (ii) refined background filtering mask (bottom rows).

The initial step in this preprocessing pipeline involves incorporating all information from the JSON annotation files into a single structured dataframe. This dataframe consolidates all relevant metadata and sensor attributes. Specifically, it contains the following columns: longitude, latitude, station name, label type, label ID, object ID, timestamp, sensor type, image file path, poly2d coordinates (representing 2D track outlines), track ID and track side. Additional fields include object occlusion level (categorized as 0–25%, 25–50%, 50–75%, 75–99% and 100%), image height and width (resolution) and inertial measurements such

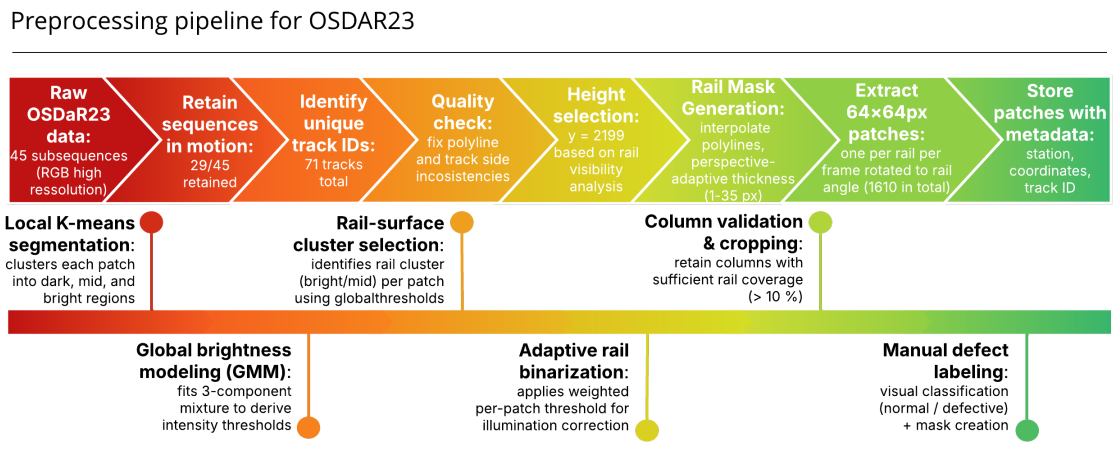

Preprocessing pipeline for OSDAR23
Raw OSDaR23 data: 45 subsequences (RGB high ressolution)
Retain sequences in motion: 29/45 retained
Identify unique track IDs: 71 tracks total
Quality check: fix polyline and track side incosistencies
Height selection: y = 2199 based on rail visibility analysis
Rail Mask Generation: interpolate polylines, perspective-adaptive thickness (1-35 px)
Extract 64×64px patches: one per rail per frame rotated to rail angle (1610 in total)
Store patches with metadata: station, coordinates, track ID
Local K-means segmentation: clusters each patch into dark, mid, and bright regions
Rail-surface cluster selection: identifies rail cluster (bright/mid) per patch using globalthresholds
Column validation & cropping: retain columns with sufficient rail coverage (> 10 %)
Global brightness modeling (GMM): fits 3-component mixture to derive intensity thresholds
Adaptive rail binarization: applies weighted per-patch threshold for illumination correction
Manual defect labeling: visual classification (normal / defective) + mask creation

Figure 3.1: Overview of the novel preprocessing pipeline developed in this thesis for the OSDaR23 dataset, illustrating dataset filtering, patch extraction with initial masking and the subsequent brightness-based background removal stages.

as pitch rate, roll rate, yaw rate, lateral acceleration, longitudinal acceleration and vertical acceleration.

Using this unified dataframe makes the data easier to manage and inspect thoroughly. This inspection reveals that although the poly2d coordinates generally mark the outer edges of rail tracks, six entries deviate from this convention. Additionally, labels of nine frames incorrectly indicate the track side and are manually corrected. These inconsistencies are accounted for during preprocessing to ensure proper implementation, particularly in the construction of image masks used to isolate rail tracks by removing background elements. Detailed information about these inconsistencies is provided in Appendix A.

3.2.1 Rail Mask Construction and Patch Extraction

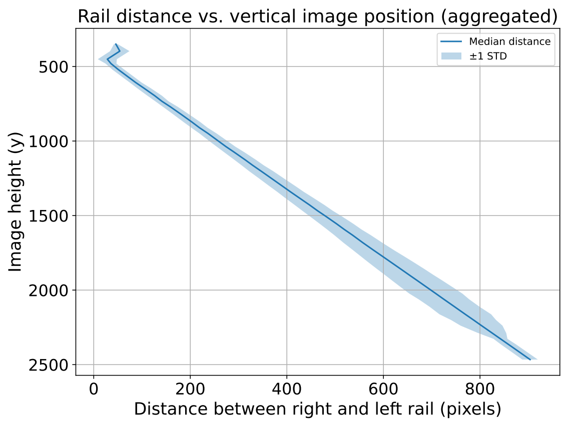
Rail tracks appear wider and more distinct in the lower portion of each image (closer to the camera) and narrower toward the top (further from the camera). The first step is therefore to determine an optimal vertical position for patch extraction. This should be a height that consistently captures a clear and stable section of the rail tracks across all frames.

Since the dataset provides rail polylines but not explicit track width measurements, we analyze the horizontal distance between left and right rails as a proxy for apparent track width. This approach relies on the parallel nature of railway tracks, where the distance between them directly reflects the perceived width of the track structure at any given height.

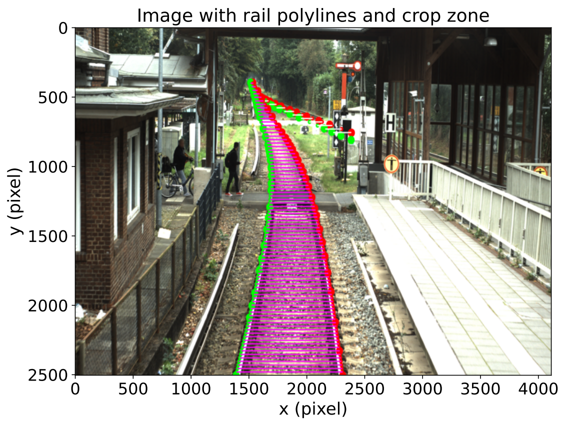
To implement this, the x-coordinates of the left and right rail polylines are interpolated at a common set of vertical (y-axis) positions for each image. Each polyline, representing either the left or right rail, is sorted by its y-coordinates and linearly interpolated at evenly spaced intervals, allowing consistent distance measurement between rails at different heights [9].

The analysis is performed per image by matching left and right rail annotations based on their shared track ID. For each matched pair, the horizontal (x-axis) distance between the rails is calculated at each valid overlapping y-position, representing the apparent track distance in pixel units across varying image heights. An example of this procedure is illustrated on the right side of Figure 3.2.

By aggregating these distance measurements across all annotated images, the median and standard deviation of the rail distances are computed for 50 discrete vertical bins, as shown on the left side of Figure 3.2. This visualization illustrates how the perceived distance between rails changes with vertical position, correlating with physical distance from the camera. The results reveal an approximately linear relationship between image height and apparent rail spacing, with deviations near image boundaries. Since our measurements track the apparent width between rails and the nominal track gauge for standard-gauge railways remains constant at approximately 1,435 mm [8], the observed pixel-spacing variations and boundary deviations are attributed to perspective distortion, vehicle motion and track elevated construction rather than changes in the physical gauge.



(a) Median pixel distance between left and right rail tracks as a function of image height (y-axis), aggregated across all annotated images in the annotated OSDaR23 dataset. The shaded area indicates one standard deviation around the median.



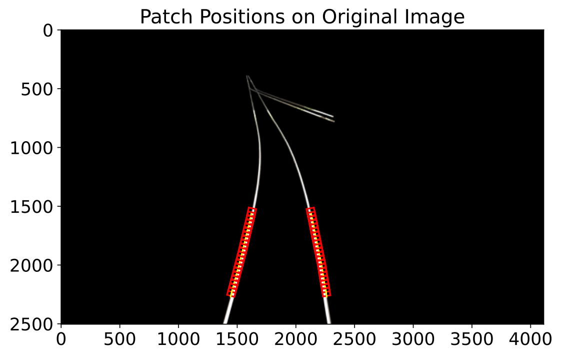
(b) Example image showing the interpolated rail polylines. Original `poly2d` points are shown as large dots, interpolated points as small dots and the horizontal white lines illustrate where rail distances are measured at fixed vertical intervals.

Figure 3.2: Left: Statistical analysis of rail distances by image height. Right: Visualization of polyline interpolation and distance measurement.

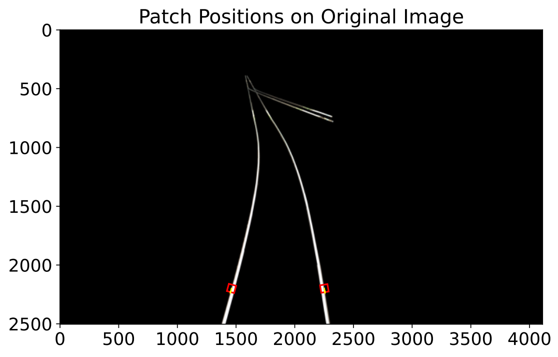
The analysis reveals that regions closer to the camera provide a more stable and clearly defined view of the rails with minimal perspective distortion. Based on these observations, we select a representative height of $y = 2199$ for all subsequent patch extractions. This position lies within the region where the rails are well defined, although neighboring heights would have provided similarly good visibility. Accordingly, in each frame, a patch is extracted from every rail track at the fixed height $y = 2199$, positioned at the corresponding x-coordinate of each track.

After the vertical position selection, binary masks are generated to isolate rail tracks from the surrounding background. Each rail's polyline is interpolated using the same procedure applied for the distance analysis (Figure 3.2), then drawn as a line with adaptive thickness that varies by vertical position. Line thickness is determined using a linear scaling function that increases from 1 pixel at the top of the image to 35 pixels at the bottom. These values are determined through manual analysis of rail widths in 21 representative images, one from each recorded station. This position-dependent thickness ensures that drawn rail segments reflect the true variation in apparent rail width due to perspective. To prevent jagged or anomalously shaped edges in the final mask, the thickness values along each polyline are smoothed using a Gaussian filter. This ensures that the drawn rail segments reflect the true, continuous variation in apparent rail width due to perspective, without artificial discontinuities. The resulting binary masks are applied to the original images to retain only rail pixels, forming the basis for patch extraction.

One square patch of size 64×64 pixels is extracted per rail track per frame, centered at the vertical position $y = 2199$. Each patch is rotated to align with the local rail orientation, determined by the angle between neighboring interpolated points along the polyline. While this work uses one patch per track per frame, the procedure can be extended to extract multiple patches at different vertical positions if denser spatial sampling is required (Figure 3.3 illustrates both configurations).



(a) Example configuration with multiple vertically spaced patches extracted along each rail track (not used in this work).



(b) Configuration used in this work: one patch per rail track and frame, extracted at a fixed vertical position ($y = 2199$).

Figure 3.3: Illustration of possible patch extraction configurations from masked rail images. The left image demonstrates that the same procedure can produce multiple patches at different heights if denser vertical sampling is desired, while the right image shows the single-patch configuration adopted for the sequential dataset. Yellow lines indicate patch orientation along the rail.

All masked patches are saved as separate image files, with filenames encoding their spatial coordinates, orientation angle and track ID. This naming convention is designed to facilitate straightforward identification of the patch's origin within the source image. Visual inspection plots are also generated to verify the accuracy of the interpolation, rotation and extraction process, confirming that the patches accurately represent the intended rail segments. An example of this verification process is presented in Figure 3.4, which demonstrates how a patch's encoded filename can be used to locate and re-extract it from its original position within the source image.

Most of the background is removed by this initial masking step, but some residual non-rail elements are still present in certain sequences. To further purify the patches and ensure that each sample contains only rail surface information, an additional brightness-based filtering stage is applied, as described below.

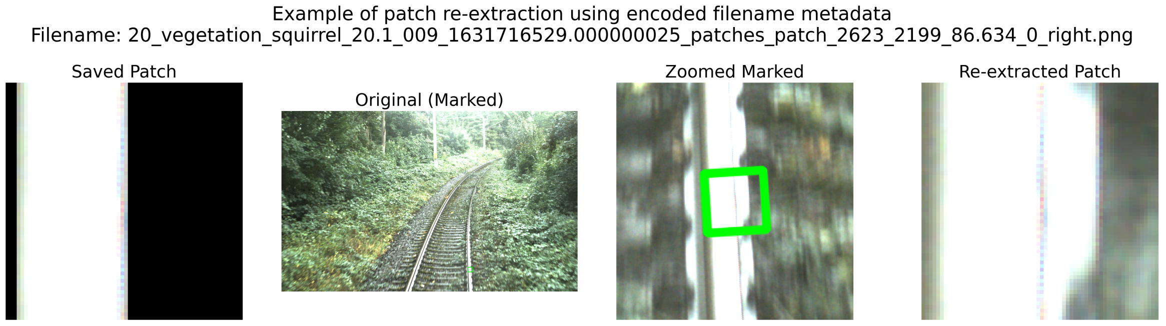

Example of patch re-extraction using encoded filename metadata
Filename: 20_vegetation_squirrel_20.1_009_1631716529.000000025_patches_patch_2623_2199_86.634_0_right.png
Saved Patch
Original (Marked)
Zoomed Marked
Re-extracted Patch

Figure 3.4: Illustration of patch re-extraction from an original image using the encoded filename that specifies station, frame, coordinates and rotation. The filename encodes the following: station (`20_vegetation_squirrel_20.1`), frame (`009_1631716529.000000025`), extraction coordinates ($x = 2623$, $y = 2199$), rotation angle (86.634°), track ID (0) and side (right). The black background in the left image results from applying the binary mask to isolate the rail region.

3.2.2 Refined Background Filtering of Extracted Patches

Visual inspection (e.g., the leftmost photo of Figure 3.6) showed that the rail surface itself usually appears noticeably brighter than its surroundings. In most cases, it shows up either as a shiny, high-intensity stripe or, in dimmer lighting, as a mid-tone region that is still lighter than the background. Since this brightness contrast is more consistent than color or texture, intensity is used as the main feature for an additional filtering step. Unsupervised clustering methods are employed for this, due to their demonstrated effectiveness in pattern recognition [14].

The process begins by analyzing the grayscale distribution of each patch. The grayscale pixel values are grouped into three brightness categories, dark, mid and bright, using K-means clustering with three clusters [36, 38]. This method is chosen because, except from not being dependent on prior labels, it also adapts to the specific brightness distribution of each patch. Thus, it automatically identifies the dominant brightness levels within each patch by grouping pixel intensities based on their similarity.

Each cluster's center value, or mean intensity, is noted, producing a triplet of brightness centres for every patch. These cluster centers are collected across all training patches and aggregated to construct a global distribution of brightness patterns of the dataset.

To model this distribution, a Gaussian Mixture Model (GMM) with three components is fitted [7]. GMM is relevant for this task because it can capture overlapping, multi-modal brightness distributions that arise from varying lighting conditions, rail surface materials and weathering states across the dataset [50]. Unlike simple histogram-based thresholding, which assumes distinct separation between brightness modes, GMM explicitly models the probabilistic mixture of brightness categories and provides threshold values at the intersection points between adjacent Gaussian components [70]. Each component of the GMM corresponds to one of the dominant brightness modes: dark, mid-tone, or bright. These components are represented by Gaussian curves, also known as normal distributions, which describe how values are symmetrically distributed around a central mean, with probability density decreasing as values deviate from the mean. The fitted model thus produces three such curves, capturing the tonal trends in the data.

The points where adjacent Gaussian curves intersect indicate brightness levels at which the likelihood of belonging to either of two neighboring modes is equal. These intersection points, or 'valleys', provide meaningful global thresholds for separating brightness categories. As illustrated in Figure 3.5, the first valley, at approximately grayscale value 61, marks the boundary between dark and mid-tones, while the second valley, near 240, separates mid-tones from bright regions.

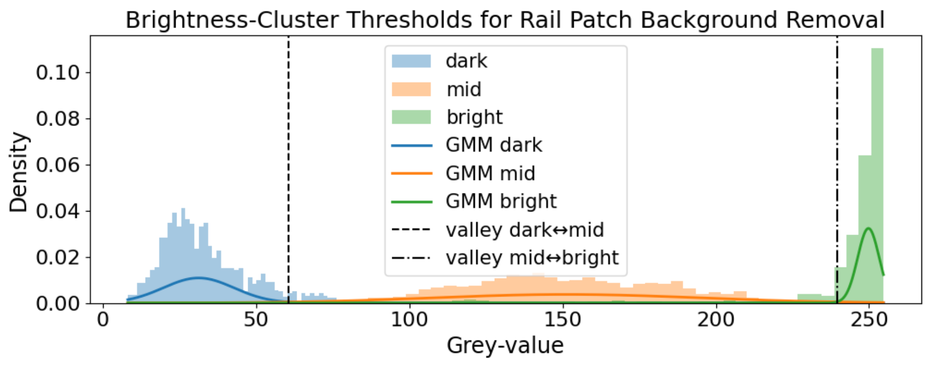


Figure 3.5: Brightness distribution of grayscale pixel clusters across training patches. Cluster centers from K-means (3 clusters per patch) are aggregated and modeled using a Gaussian Mixture Model (GMM) with three components. The dashed and dash-dotted lines indicate the two global thresholds (valleys) separating dark, mid-tone and bright modes.

These global thresholds are used to identify which brightness cluster most likely represents the rail surface in each patch. The selection logic operates as follows: if the bright cluster's centroid exceeds the upper threshold (240), it is selected as the rail surface. If not, but the mid-tone cluster's centroid exceeds the lower threshold (61), the mid-tone cluster is chosen. If neither condition holds, the cluster with more pixels is selected. This logic combines brightness and pixel dominance to reliably identify the rail surface under diverse lighting conditions.

A patch-specific threshold is then computed as a weighted average between the brightness center of the selected rail cluster and that of the next darker cluster, weighted by the relative number of pixels in each cluster. This ensures the threshold adapts to both the brightness gap and the relative dominance of each cluster. The grayscale patch is binarized using this adaptive threshold, retaining only pixels brighter than the computed value.

After binarization, the algorithm counts white (rail) pixels in each image column. A column is retained only if its white-pixel count exceeds 10% of the image height, ensuring that only columns capturing a meaningful portion of track are kept while ignoring isolated bright spots from background elements. All accepted columns form a contiguous span between the leftmost and rightmost indices. With a 2-pixel safety margin on each side, this span is copied from the original RGB image while all remaining columns are set to black, producing a narrow vertical strip containing only the rail surface. If no column meets the 10% criterion, the bounding box of the largest connected white region is extracted instead. An example is shown in Figure 3.6.

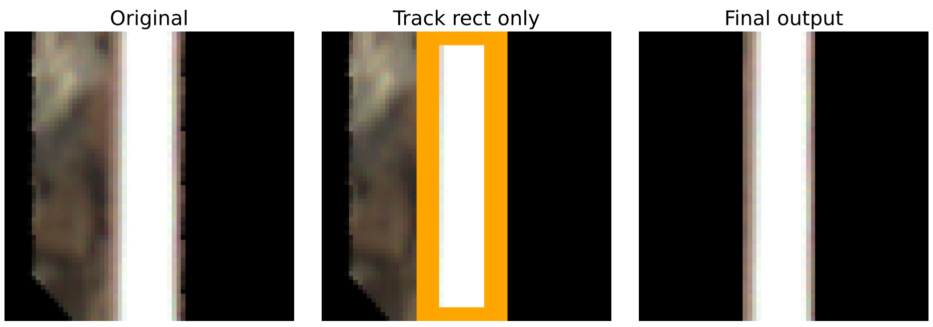


Figure 3.6: Example of the additional preprocessing step for removing background content from a rail patch. From left to right: the original RGB patch, the intermediate binary mask identifying the vertical band with sufficient rail pixels, the rail region highlighted in orange for verification, and the final cropped patch with only the rail surface retained.

Manual Labeling

Each patch is manually assigned a binary label, either `normal` or `defective`, based on visual inspection. Indicators such as abrupt texture discontinuities, localized discoloration and geometric irregularities are used to identify potential defects. For each patch, a visualization similar to Figure 3.4 is examined for determining whether visible markings represent genuine defects or artifacts caused by shadows, reflections, or lighting variations. The manually labeled dataset, which contains $1,610$ image patches in total, consists of 415 (25.8%) defective and 1,195 (74.2%) normal samples.

Chapter 4

Methodology

This chapter describes the methods used in this study. It covers the custom data partitioning strategy (Section 4.1), the candidate approaches for preprocessing, model architectures, sequential refinement and explainability (Sections 4.2–4.5) and formalizes the evaluation metrics and selection criteria (Section 4.6).

4.1 Stratified Data Splitting Strategy

Following the creation of the track patch dataset, the next critical step is to define a fair data splitting strategy. Standard random splitting is unsuitable for this sequential dataset, as it risks data leakage and can create folds with highly imbalanced defect distributions. To address this, a multi-stage stratified allocation algorithm is explicitly implemented for this thesis. The resulting data split must satisfy two key requirements: (i) prevent spatiotemporal leakage by ensuring that all frames from a given track sequence reside in the same partition and (ii) preserve balanced distributions of sequence-level attributes, including sequence length, defect ratio and total patch count, across folds. To meet these constraints, a multi-criteria stratified allocation algorithm is applied at the sequence level. The resulting split is constructed through three sequential stages:

Step 1: Stratification by sequence characteristics. Each sequence is assigned to a stratum based on two categorical attributes: (i) sequence length (short = 10 frames, long = 100 frames) and (ii) defect ratio (none = 0%, low = 1–10%, medium = 11–50%, high >50%). This results in at most $2 \times 4 = 8$ possible strata, noting that not all combinations are necessarily present in the dataset.

Prior to allocation, sequences within each stratum are randomly shuffled using a fixed random seed. This shuffle determines the order in which sequences are considered during greedy assignment: when multiple sequences have identical load characteristics (e.g., two 10-frame sequences with 2 defects each), their assignment order and consequently the content of each fold depend on this initial randomization. Different seeds produce different shuffles, leading to alternative fold configurations that satisfy the same stratification and load-balancing constraints (for reproducibility seed should be fixed).

Step 2: Hamilton-style quota apportionment. Once strata are defined, fold quotas are determined using proportional allocation. For each stratum, a target allocation specifying how many sequences should be assigned to each fold is computed using Hamilton's method [3]. The choice of this method ensures proportional representation of all stratum combinations across folds, since stratification factors interact. For example, without this quota apportionment, all long sequences (100 frames) might be assigned to a single fold, creating imbalance in both patch count and defect distribution.

This approach first allocates sequences proportionally based on the total number in the stratum ($n_{\text{stratum}}/n_{\text{folds}}$), assigning the integer portion to each fold. Remaining fractional sequences are then distributed one-by-one to folds with the largest fractional remainders until all sequences are allocated. This method ensures that: (i) each fold receives either the floor ($\lfloor 71/n_{\text{folds}} \rfloor$) or ceiling ($\lceil 71/n_{\text{folds}} \rceil$) of the average sequence count and (ii) within each stratum, representation is proportional across folds.

Step 3: Greedy load-balancing optimization. After establishing target quotas and shuffling sequences within each stratum, sequences are assigned to folds using a greedy algorithm that minimizes imbalance in two load metrics: total patch count and defective patch count.

Sequences within each stratum are sorted by descending size (patches) and defect count, then iteratively assigned to the fold that currently has the lowest weighted load:

$$\text{load}(f) = w_{\text{total}} \cdot \text{total\_patches}(f) + w_{\text{defective}} \cdot \text{defective\_patches}(f)$$

where $w_{\text{total}} = 0.5$ and $w_{\text{defective}} = 1.0$ are weighting coefficients for the respective criteria.

The dual objective addresses the practical concern that folds should be balanced not only in sequence count, but also in effective training signal. A fold containing many short sequences with few defects provides less training information than a fold with long, defect-rich sequences. The $w_{\text{defective}} = 1.0$ weighting prioritizes minority-class balance, which is critical for model calibration under skewed distributions.

After the initial greedy allocation, fold balance is refined through a local search that evaluates pairwise sequence swaps between folds. The goal is to minimize the variance of the total sequence load across folds:

$$\text{Objective} = \operatorname{Var}\left(\{\text{load}(f_1), \ldots, \text{load}(f_5)\}\right).$$

Swap candidates are assessed greedily and applied only if they reduce this objective. The procedure performs up to two full passes over all fold pairs and terminates early if no further variance-reducing swaps are found.

Upon completion of all the three steps, the result is a set of balanced, sequence-disjoint partitions with comparable defect signal and patch distributions.

4.2 Preprocessing Pipeline Candidates

Having defined how the data is partitioned, the pipeline's focus shifts to optimizing the input data. This section evaluates a set of preprocessing strategies, all built upon the base pipeline from Chapter 3. This base pipeline includes patch extraction, rotation and brightness-based background filtering for mask generation. The evaluation focuses on how variations in background handling, contrast enhancement and color space representation influence defect detection performance.

Before detailing each pipeline, we briefly define the two color representations mentioned throughout this section. RGB (Red, Green, Blue) is the standard image format producing a three-channel color image by mixing primary light intensities, it preserves full color information but does not explicitly separate brightness from chromatic content. In contrast, CIELAB (L*a*b*) is a perceptually uniform color space that separates luminance (L*, representing brightness) from chromatic components (a* for green–red and b* for blue–yellow). This separation allows brightness to be modified independently of color, which is useful for contrast enhancement and color-invariant processing. When a luminance-only representation is used in this work, the L* channel is replicated into three channels to maintain compatibility with pretrained RGB-based neural networks.

4.2.1 Background Handling Strategies

Three background handling strategies are evaluated: no modification (`ORIGINAL`), static background suppression (`BLACK_BG`) and blurred background replacement (`BLUR_BG`).

The `ORIGINAL` pipeline preserves patches without alteration, maintaining full visual context but also including irrelevant background content such as ballast and sleepers.

To suppress background variation, `BLACK_BG` replaces non-rail pixels, identified via the binary rail mask from Chapter 3, with a constant black value. This enforces strong foreground isolation, a practice adopted in industrial anomaly benchmarks such as MVTec AD [4]. While effective at reducing distraction, this approach introduces sharp artificial edges at mask boundaries, which may distort local gradients and introduce non-natural artefacts.

To preserve smoother transitions, `BLUR_BG` replaces the background with a Gaussian-blurred version of the original patch instead of a constant value [34]. Before compositing, the mask is dilated to soften foreground boundaries. The final image retains original rail pixels inside the dilated mask and substitutes blurred pixels elsewhere, reducing background complexity without introducing hard transitions. This improves boundary realism but may remove contextual cues that could otherwise aid defect recognition.

4.2.2 Contrast Enhancement

To strengthen the visibility of subtle surface defects under varying illumination, contrast is enhanced using CLAHE, applied exclusively to the rail region [47].

Unlike global histogram equalization, which applies a single intensity transform and risks over-amplification, CLAHE performs local adaptive equalization with a clipping threshold to limit noise amplification. The enhancement is applied in CIELAB color space, modifying only the L* (luminance) channel within the rail mask, while preserving original chromaticity. The enhanced luminance is then recombined with the untouched a* and b* channels and converted back to RGB.

Two parameters control CLAHE behavior: clip limit, which bounds contrast amplification and tile size, which determines the spatial scale of enhancement.

4.2.3 Color Space Conversion

To assess whether reducing color dependence can improve defect recognition, luminance information is isolated while maintaining compatibility with pretrained CNN backbones. To achieve this, images are transformed from RGB to the CIELAB color space, the luminance channel (L*) is extracted and duplicated across three channels to produce a 3-channel grayscale input.

This representation emphasizes structural rail surface patterns, while reducing sensitivity to color variability. However, this comes at the cost of attenuating color-based defect indicators (e.g., rust or staining), which may be informative for certain fault types.

4.3 Model Architectures

Following input preprocessing, three architectures are evaluated to compare supervised CNN-based, transformer-based and unsupervised anomaly detection paradigms under the constraints of class imbalance and uncontrolled settings of ATO data.

4.3.1 ResNet-18: Convolutional Baseline

ResNet-18 [25] serves as the supervised convolutional baseline. The architecture extracts features using stacked residual blocks, each operating at progressively reduced spatial resolution to encode increasingly abstract semantic patterns. Given an input image $\mathbf{x} \in \mathbb{R}^{3\times H\times W}$ with height H, width W and three color channels, a residual block computes:

$$\mathbf{y} = \mathcal{F}(\mathbf{x}, \{W_i\}) + \mathbf{x}, \tag{4.1}$$

where $\mathcal{F}(\cdot)$ represents a sequence of learnable convolutional operations parameterized by weights $\{W_i\}$. The additive identity shortcut enables direct gradient flow to earlier layers, improving optimization stability and mitigating the degradation effect where deeper networks exhibit higher training error than shallower counterparts due to optimization difficulties [25].

The backbone is initialized with ImageNet-pretrained weights [12], allowing early convolutional filters to retain general low-level edge, texture and shape representations while higher layers adapt to defect-specific patterns through fine-tuning. This transfer learning approach is nearly universal in vision tasks with limited training data [30], based on the hypothesis that features learned from natural images generalize to specialized domains. After the final residual block, spatial features are reduced via global average pooling, producing a compact representation that feeds into a two-neuron fully connected layer for binary classification. The entire network is trained end-to-end using class-weighted cross-entropy loss to counteract dataset imbalance.

4.3.2 DeiT-S: Vision Transformer

DeiT-S [62] represents the transformer-based approach. Unlike convolutional networks that impose locality bias through hierarchical local filtering, vision transformers partition images into patch sequences and model global relationships through self-attention mechanisms.

The architecture operates in three stages. First, an input image is divided into non-overlapping patches of fixed size, with each patch flattened and linearly projected into a fixed-dimensional embedding. A learnable classification token is prepended to the sequence and positional embeddings are added to retain spatial information. Second, the token sequence passes through multiple transformer encoder layers, each applying multi-head self-attention followed by feedforward networks with residual connections and layer normalization. The self-attention mechanism computes pairwise relationships between all tokens:

$$\text{Attention}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = \text{softmax}\left(\frac{\mathbf{Q}\mathbf{K}^\top}{\sqrt{d_k}}\right)\mathbf{V}, \tag{4.2}$$

where $\mathbf{Q}$, $\mathbf{K}$ and $\mathbf{V}$ are learned projections of the input tokens into query, key and value representations and d_k denotes the dimension per attention head. This enables each token to attend to all other tokens simultaneously, capturing spatially distant patterns within a single layer. This represents a key advantage over CNNs, which require multiple stacked layers to achieve comparable receptive fields. Third, after processing through all encoder layers, the classification token's final representation is passed through a linear head to produce the binary defect probability.

The DeiT family includes both standard and distilled variants, where distilled models incorporate knowledge distillation from a larger teacher network during pretraining and include an additional distillation token in the architecture. This work employs the **non-distilled variant** (`deit_small_patch16_224`) to maintain architectural parity with the CNN baseline, ensuring that performance comparisons reflect structural differences in feature extraction rather than training paradigm differences introduced by distillation. Vision transformers typically require larger datasets than CNNs when trained from scratch due to reduced inductive biases [15]. ImageNet pretraining provides transferable low-level features that generalize to rail surface imagery despite domain differences. The model is fine-tuned end-to-end with data augmentation and regularization strategies designed for transformers on small datasets.

4.3.3 PatchCore: Unsupervised Anomaly Detection

PatchCore [54] represents the unsupervised paradigm, offering an alternative to supervised classification by learning representations of normal rail appearance and flagging deviations as anomalous. This non-parametric method stores representations of normal training samples in a memory bank and detects anomalies via distance comparisons. The core intuition is that normal test samples will have similar representations to stored normal samples, while anomalous samples will be distant from all stored normals.

PatchCore operates in three stages. First, a frozen pretrained CNN backbone extracts multi-scale features from intermediate layers, providing both local detail and mid-level semantic information. The backbone remains frozen without gradient updates, as ImageNet pretraining already captures sufficient low and mid-level visual features for anomaly detection. Second, features from all normal training samples undergo coreset subsampling, meaning a greedy selection algorithm identifies a representative subset covering normal pattern diversity, while reducing memory requirements and computational cost. Third, test-time anomaly scoring computes the distance from each query feature to its nearest neighbor in the memory bank. Image-level anomaly scores are typically derived from maximum patch-level scores, as defects manifest as local deviations. Binary classification applies a threshold tuned on validation data to control the precision-recall trade-off.

This approach offers four advantages: (1) training uses only normal samples, eliminating imbalance-related optimization challenges, (2) the frozen backbone and nearest-neighbor scoring avoid overfitting risks from training on small datasets, (3) spatially localized anomaly scores provide interpretable heatmaps indicating which regions appear anomalous and (4) modeling normality may detect defect types absent from training data. However, for low false positive rates, the memory bank must represent the full range of normal variation encountered in operational deployment, including weathered rails, diverse lighting and material differences. In addition, test-time nearest-neighbor search over the memory bank is computationally slower than forward passes through neural networks.

4.4 Sequential Post-Processing Methods

This section describes the post-processing methods evaluated, which exploit the spatial sequential nature of railway inspection data to refine predictions. Ranging from simple smoothing techniques to learned recurrent models, these approaches leverage sequential coherence to improve defect detection performance.

All methods operate on ensemble predictions generated by averaging per-class probabilities across three model instances trained with different random seeds. Ensemble averaging is employed for its benefits in improving predictive performance, while reducing variance from random initialization [13, 33]. Formally, let $p_s^{(t)}$ denote the defect probability for frame t from seed s, the ensemble probability is:

$$\bar{p}_{\mathsf{def}}(t) = \frac{1}{S}\sum_{s=1}^{S} p_s^{(t)}, \tag{4.3}$$

where $S = 3$ is the number of seeds. Binary decisions are obtained via thresholding:

$$\hat{y}(t) = \mathbb{1}\left[\bar{p}_{\mathsf{def}}(t) \geq \tau\right], \tag{4.4}$$

with $\tau \in [0, 1]$ as the decision threshold. While this ensemble baseline enhances robustness [21], it still treats frames independently, ignoring the spatial connections inherent in sequential inspection data. The following methods address this by incorporating information from adjacent frames within each sequence.

Before applying post-processing, frames are grouped into sequences based on their metadata: all frames sharing the same (station, track_id, side) tuple belong to a single sequence.

Within each sequence, frames are sorted by their frame number to establish spatial order.

The evaluated methods fall into three categories based on their mechanism: *spatial smoothing* methods that aggregate predictions over local windows, *binary refinement* methods that operate on discrete decisions to enforce structural constraints and *learned sequential models* that discover spatial patterns from training data. Some of these approaches are additionally combined in two-stage pipelines to leverage complementary strengths.

4.4.1 Spatial Smoothing Methods

The spatial smoothing approaches evaluated in this work aim to enforce local consistency by aggregating predictions across neighboring frames, leveraging the observation that rail defects (e.g. corrugation) typically extend across multiple consecutive frames. Four variants are considered: *simple spatial smoothing*, *confidence-weighted smoothing*, *median filtering* and *multiscale spatial ensemble*.

The simplest approach, *simple spatial smoothing*, applies uniform moving-average smoothing, replacing each frame's probability with the mean of its neighborhood:

$$\tilde{p}_{\mathsf{def}}(t) = \frac{1}{2w+1} \sum_{i=t-w}^{t+w} \bar{p}_{\mathsf{def}}(i), \tag{4.5}$$

where w defines the window radius. While effective for reducing isolated noise, uniform smoothing ignores prediction uncertainty and may over-smooth genuine transitions between defective and non-defective regions.

To address this limitation, *confidence-weighted smoothing* adapts the degree of spatial regularization based on ensemble disagreement. The standard deviation across seeds, $\sigma_{\mathsf{seeds}}(t) = \mathsf{std}(\{p_s^{(t)}\}_{s=1}^{S})$, quantifies prediction uncertainty at each frame. High variance indicates the ensemble is uncertain, suggesting that information from neighboring frames should be trusted more than the potentially unreliable prediction at frame t. Conversely, low variance indicates high confidence in the original prediction, requiring less spatial correction. This intuition is formalized by computing a confidence weight based on ensemble disagreement:

$$w_{\mathsf{conf}}(t) = 1 - \min\left(\frac{\sigma_{\mathsf{seeds}}(t)}{\tau_{\mathsf{conf}}}, 1\right), \tag{4.6}$$

where τ_{conf} is a threshold parameter. When ensemble variance is low ($\sigma_{\mathsf{seeds}}(t) \ll \tau_{\mathsf{conf}}$), the confidence weight approaches 1, indicating the original prediction should be trusted. When variance is high ($\sigma_{\mathsf{seeds}}(t) \geq \tau_{\mathsf{conf}}$), the confidence weight drops to 0, indicating the prediction should be replaced entirely by spatially smoothed values. The final refined probability is then a weighted combination of the original ensemble prediction and a spatially smoothed version:

$$\tilde{p}_{\mathsf{def}}(t) = w_{\mathsf{conf}}(t) \cdot \bar{p}_{\mathsf{def}}(t) + (1 - w_{\mathsf{conf}}(t)) \cdot \mathcal{G}_\sigma[\bar{p}_{\mathsf{def}}](t), \tag{4.7}$$

where $\mathcal{G}_\sigma[\bar{p}_{\mathsf{def}}](t)$ denotes the result of applying a Gaussian filter with standard deviation σ to the sequence of ensemble probabilities. The Gaussian filter computes a weighted average of predictions from neighboring frames, where the weight assigned to each neighbor decreases with its spatial distance from frame t according to a Gaussian (bell-shaped) curve. Specifically, the smoothed probability is:

$$\mathcal{G}_\sigma[\bar{p}_{\mathsf{def}}](t) = \frac{\sum_i \bar{p}_{\mathsf{def}}(i) \cdot \exp\left(-\frac{(i-t)^2}{2\sigma^2}\right)}{\sum_i \exp\left(-\frac{(i-t)^2}{2\sigma^2}\right)}, \tag{4.8}$$

where the summation is performed over frames within the same sequence. The parameter σ controls the effective smoothing window: small values of σ result in narrow filters that preserve sharp transitions between defective and non-defective regions, while large values produce wider filters that emphasize broader spatial trends at the expense of fine-grained detail. This described procedure, ensures that uncertain predictions are smoothed more aggressively, while confident predictions are preserved closer to their original values.

Finally, *Multiscale spatial ensemble* extends single-scale smoothing by aggregating predictions at multiple spatial resolutions. Rather than selecting a single smoothing bandwidth, this method applies Gaussian filters with different standard deviations $\{\sigma_1, \sigma_2, \ldots, \sigma_K\}$ and averages the results:

$$\tilde{p}_{\mathsf{def}}(t) = \frac{1}{K}\sum_{k=1}^{K} \mathcal{G}_{\sigma_k}[\bar{p}_{\mathsf{def}}](t). \tag{4.9}$$

This approach captures both fine-grained transitions (small σ) and broader defect trends (large σ), providing robustness to varying defect cluster lengths.

4.4.2 Binary Refinement Methods

While smoothing methods operate on continuous probabilities, binary refinement methods (*Isolated removal*, *minimum cluster size filtering*, *defect voting*) apply post-hoc corrections to discrete classification decisions, enforcing structural assumptions about defect patterns observed in the training data.

Isolated removal eliminates single positive frames surrounded by negative predictions, since genuine defects rarely occur in isolation [65]. For each predicted defect at position t, the method examines a local context window and removes the prediction if no supporting positives are found:

$$\hat{y}_{\mathsf{refined}}(t) = \begin{cases} 0 & \text{if } \hat{y}(t) = 1 \text{ and } \sum_{i \in \mathcal{N}(t)} \hat{y}(i) = 0, \\ \hat{y}(t) & \text{otherwise,} \end{cases} \tag{4.10}$$

where $\mathcal{N}(t)$ denotes the neighborhood of t (excluding t itself). This rule can increase precision by reducing false positive clusters of length one, though it cannot affect recall since it only removes predictions.

More generally, *minimum cluster size filtering* extends this concept by requiring defect predictions to belong to contiguous runs of a specified minimum length. The method identifies connected components of positive predictions and removes those shorter than a threshold $L_{\min}$:

$$\hat{y}_{\mathsf{refined}}(t) = \begin{cases} 0 & \text{if } \hat{y}(t) = 1 \text{ and } |\mathsf{cluster}(t)| < L_{\min}, \\ \hat{y}(t) & \text{otherwise,} \end{cases} \tag{4.11}$$

where $|\text{cluster}(t)|$ denotes the length of the contiguous positive region containing frame t. This approach directly enforces the observed spatial clustering property of defects but is limited by its inability to recover false negatives.

To overcome this asymmetry, *defect voting* applies majority voting within local windows, allowing both the addition and removal of predictions based on neighborhood consensus. For each frame, the method computes the proportion of predicted defects in its spatial neighborhood and combines this with the ensemble probability through confidence weighting:

$$\tilde{p}_{\text{def}}(t) = w_{\text{conf}}(t) \cdot \bar{p}_{\text{def}}(t) + (1 - w_{\text{conf}}(t)) \cdot \frac{1}{2w+1} \sum_{i=t-w}^{t+w} \hat{y}(i), \tag{4.12}$$

where $w_{\text{conf}}(t)$ is defined as in 4.6. Unlike isolated removal, this mechanism can propagate defect labels to uncertain frames supported by strong local evidence, potentially improving recall in addition to precision.

4.4.3 Hybrid Two-Stage Pipelines

The smoothing and binary refinement methods operate on fundamentally different representations: smoothing methods leverage continuous probability information to improve calibration, while binary refinement methods enforce discrete structural constraints observed in defect patterns. These complementary strengths motivate hybrid pipelines that combine both approaches sequentially, applying probability-based smoothing first to reduce noise and improve confidence estimates, followed by binary refinement to enforce structural guarantees on the final predictions.

The space of possible two-stage combinations is large, but many pairings would be redundant or incompatible. To explore complementary strategies, four pipeline configurations are designed. Each pipeline targets a specific failure mode while maintaining the general principle of applying probability-based refinement before structural constraints:

1. **High-confidence isolated errors**: *Confidence-weighted + isolated removal*, for predictions that are individually confident but spatially inconsistent with neighbors
2. **Scale-dependent patterns**: *Confidence-weighted + multiscale*, for defects appearing at different spatial resolutions requiring adaptive multi-scale analysis
3. **Fragmented detections**: *Multiscale + defect voting*, for genuine defects split into multiple short segments that require consensus-based merging
4. **Sharp boundaries with noise**: *Confidence-weighted median + minimum cluster size*, for scenarios with abrupt transitions between defect/non-defect regions contaminated by short spurious detections

4.4.4 LSTM-Based spatial Refinement

While the previous methods rely on hand-crafted rules or simple aggregation schemes, a more flexible approach employs a learned sequential model to discover optimal spatial patterns directly from training data. Specifically, a bidirectional Long Short-Term Memory (LSTM) network [26, 64] is trained to refine ensemble predictions by modeling dependencies in both forward and backward spatial directions.

The LSTM receives as input a three-dimensional feature vector for each frame t:

$$\mathbf{x}(t) = \begin{bmatrix} \bar{p}_{\mathsf{def}}(t) \\ \bar{p}_{\mathsf{good}}(t) \\ \sigma_{\mathsf{seeds}}(t) \end{bmatrix}, \tag{4.13}$$

where $\bar{p}_{\mathsf{good}}(t) = 1 - \bar{p}_{\mathsf{def}}(t)$ is the complementary probability and $\sigma_{\mathsf{seeds}}(t)$ quantifies ensemble disagreement. The network architecture consists of multiple stacked bidirectional LSTM layers:

$$\mathbf{h}_t = \mathsf{BiLSTM}(\mathbf{x}_t, \mathbf{h}_{t-1}, \mathbf{h}_{t+1}; \theta), \tag{4.14}$$

where θ denotes learnable parameters (cell weights, gates). The hidden representations $\mathbf{h}_t$ capture spatial context from both prior and forward frames. A final fully-connected layer maps hidden states to refined class logits:

$$\mathbf{z}_t = \mathbf{W}\mathbf{h}_t + \mathbf{b}, \tag{4.15}$$

from which calibrated probabilities are obtained via softmax transformation:

$$\tilde{p}_{\mathsf{def}}(t) = \frac{\exp(z_t^{(1)})}{\exp(z_t^{(0)}) + \exp(z_t^{(1)})}. \tag{4.16}$$

The LSTM is the only learned method among all post-processing approaches, all other methods apply deterministic rules or aggregation functions that require no gradient-based training.

The model is trained using cross-entropy loss with class weights inversely proportional to class frequency to address label imbalance:

$$\mathcal{L} = -\sum_{t=1}^{T} w_{y_t} \log \tilde{p}_{y_t}(t), \tag{4.17}$$

where $y_t \in \{0, 1\}$ is the ground-truth label and w_0, w_1 are class weights. To prevent overfitting, dropout is applied between LSTM layers and gradient clipping is used to stabilize training.

Unlike hand-crafted methods, the LSTM can learn complex non-linear relationships between local prediction patterns and ground-truth labels, potentially discovering task-specific spatial features that fixed rules cannot capture. However, this flexibility comes at the cost of increased computational requirements, the need for sufficient training data to avoid overfitting and reduced interpretability compared to rule-based approaches.

4.5 Explainability Methods

This section details the XAI techniques used to interpret model predictions. These techniques are applied to answer critical questions: Does the model focus on genuine rail defects, or is it misled by spurious features like shadows or reflections? What visual patterns cause false positives and false negatives?

To address these questions, three gradient-based attribution methods that generate spatial saliency maps are utilised to highlight pixels most influential to the model's classification decision. These methods are selected for their complementary strengths: Gradient Saliency as a computational baseline, Integrated Gradients for theoretical soundness and SmoothGrad for visual clarity. Their combined use allows for robust validation of the model's reasoning, establishing transparency, which is required for operational deployment.[1].

All three methods operate on a common principle of quantifying pixel importance by analyzing how the model's output changes with respect to input variations. Formally, given an input image $\mathbf{x} \in \mathbb{R}^{3\times H\times W}$ and the model's logit output z_c for class c, each method produces a saliency map $\mathbf{A}(\mathbf{x}) \in \mathbb{R}^{H\times W}$ where higher values indicate greater influence on the prediction.

Gradient Saliency provides a foundational approach by calculating the gradient of the class score with respect to input pixels [57]:

$$\mathbf{A}\text{grad}(\mathbf{x}) = \max c \in R, G, B \left| \frac{\partial z_c}{\partial \mathbf{x}_c} \right|. \tag{4.18}$$

This identifies pixels where infinitesimal changes would most affect the model's output. While computationally efficient (requiring only a single backward pass), this method suffers from the gradient saturation problem: confident predictions yield small, noisy gradients that often highlight edges rather than semantically coherent regions.

Integrated Gradients (IG) resolves the saturation problem by accumulating gradients along a path from a neutral baseline $\mathbf{x}$base to the actual input [60]:

$$\mathbf{A}\text{IG}(\mathbf{x}) = (\mathbf{x} - \mathbf{x}\text{base}) \odot \int \alpha = 0^1 \frac{\partial z_c(\mathbf{x}\alpha)}{\partial \mathbf{x}\alpha} d\alpha, \tag{4.19}$$

where $\mathbf{x}\alpha = \mathbf{x}\text{base} + \alpha(\mathbf{x} - \mathbf{x}\text{base})$ represents interpolated images and $\odot$ denotes element-wise multiplication. The scaling factor $(\mathbf{x} - \mathbf{x}\text{base})$ ensures attributions are proportional to pixel values, while the path integral captures feature importance across all activation levels. This method satisfies the completeness axiom, guaranteeing that attributions sum to the output difference between input and baseline.

SmoothGrad addresses visual noise by averaging saliency maps from multiple noisy input variations [58]:

$$\mathbf{A}\text{smooth}(\mathbf{x}) = \frac{1}{n} \sum i = 1^n \mathbf{A}_{\text{grad}}(\mathbf{x} + \mathbf{n}_i), \quad \mathbf{n}_i \sim \mathcal{N}(0, \sigma^2 \mathbf{I}). \tag{4.20}$$

By introducing Gaussian noise $\mathbf{n}_i$ and averaging the results, SmoothGrad suppresses inconsistent high-frequency artifacts while preserving stable, semantically meaningful features. This produces cleaner visualizations suitable for domain experts, though at increased computational cost proportional to the number of samples n.

For all three methods, a black baseline image (all pixel values set to zero) is used when required by the algorithm (specifically for Integrated Gradients). This choice represents the absence of visual information and provides a neutral starting point for attribution accumulation.

4.6 Evaluation Metrics

All models are evaluated using metrics appropriate for binary classification under severe class imbalance.

4.6.1 Primary Metric: AUPRC

Area Under the Precision-Recall Curve (AUPRC) serves as the primary metric [11]. The precision-recall curve plots precision $P = \frac{\mathsf{TP}}{\mathsf{TP}+\mathsf{FP}}$ against recall $R = \frac{\mathsf{TP}}{\mathsf{TP}+\mathsf{FN}}$ across all classification thresholds $\tau \in [0, 1]$. AUPRC integrates this curve:

$$\mathsf{AUPRC} = \int_0^1 P(R)\, dR. \tag{4.21}$$

Unlike AUROC, which remains high under extreme imbalance due to abundant true negatives, AUPRC focuses exclusively on positive-class performance and drops rapidly when false positives or false negatives increase [55]. For context, AUPRC values should be interpreted relative to the baseline of a random classifier, which achieves AUPRC equal to the positive class prevalence (25.78% for this dataset). Thus, values higher than 0.2578 indicate strong discriminative performance, particularly under severe imbalance where maximizing both precision (minimizing false alarms) and recall (minimizing missed defects) is challenging. The metric is particularly suited to operational deployment scenarios where the cost of investigating false positives (wasted maintenance effort) must be balanced against the risk of missing genuine defects (potential safety incidents).

4.6.2 Secondary Metrics

Complementary metrics provide additional perspectives. **F1 score**, the harmonic mean of precision and recall at a fixed threshold, provides a single-point summary balancing both aspects: $F_1 = \frac{2PR}{P+R}$. **Precision and recall** reported separately enable trade-off analysis, as high precision minimizes false maintenance interventions while sufficient recall ensures acceptable defect coverage. **Confusion matrix elements** (TP, TN, FP, FN) provide raw counts enabling detailed error analysis and computing derived metrics.

Chapter 5

Experiments

Building upon the methodological components defined in Chapter 4, this chapter presents the experimental design for constructing the rail defect detection system. A progressive narrowing strategy is employed, where six sequential experiments each isolate and optimize one system component. The optimal configuration identified at each stage is carried forward to subsequent experiments, leading to the integrated system proposal. This final system defines the specific configurations for data preprocessing, model design and parameters, sequential refinement method, and explainability analysis.

5.1 Experimental Setup

All experiments are conducted in a controlled computational environment to ensure reproducibility. The hardware consists of an NVIDIA GeForce RTX 4070 GPU (8 GB GDDR6 VRAM) and an Intel Core i7-13800H CPU. The software environment includes Python 3.9.21, PyTorch 2.5.1, timm 1.0.20 and scikit-learn 1.6.1. All the code and components necessary to replicate the experiments are publicly available in GitHub through this link.

5.1.1 Pipeline Overview

Table 5.1 outlines the six-stage experimental pipeline.

Table 5.1: Structure of the sequential experimental pipeline.

Stage	Purpose
Exp. 1	Select optimal preprocessing pipeline
Exp. 2	Assess cross-validation robustness and select representative split
Exp. 3	Compare model paradigms and select best-performing approach
Exp. 4	Tune spatial refinement strategies and select optimal
Exp. 5	Tune model hyperparameters and evaluate on held-out test set
Exp. 6	Interpret model decisions using gradient-based attribution

5.1.2 Data Partitioning Strategy

The data partitioning follows a two-stage approach using the stratified splitting procedure presented in Section 4.1:

Outer split (Test set isolation): The full sequence set is partitioned using seed 123 to generate five primary folds, with Fold 1 (named as Set E) permanently reserved as the held-out test set for final evaluation in Experiment 5. Table 5.2 summarizes the resulting splits obtained through the stratified sampling procedure, which ensures balanced distributions across sequence count (13–16), total patches (310–340) and defect ratio (19.41%–32.25%).

Table 5.2: Composition of the outer 5-fold cross-validation split (seed 123) for test set isolation.

Fold	Sequences	Total Patches	Defective Patches	Stations	Defect %
1	13	310	90	12	29.03
2	16	340	66	11	19.41
3	14	320	92	13	28.75
4	15	330	67	14	20.30
5	13	310	100	13	32.25
Total	71	1610	415	–	25.78

Inner splits (Cross-validation): The stratification procedure is re-applied to the remaining 58 sequences using five random seeds (0, 42, 123, 26, 79) to produce five distinct five-fold cross-validation splits. The split generated with seed 123 is used in Experiment 1, while all splits are evaluated in Experiment 2 to select the most representative one to continue.

5.1.3 Common Training Configuration

Unless otherwise specified, all supervised models share the following baseline configuration.

Model architectures: ResNet-18 employs the standard ImageNet-pretrained architecture (11.7M parameters) [25]. **DeiT-Small** uses the `deit_small_patch16_224` variant (22M parameters) with patch size 16×16, embedding dimension 384, 6 attention heads and 12 transformer blocks [62].

Training hyperparameters: Table 5.3 summarizes the training configuration shared across both architectures, with model-specific variations noted where applicable.

Table 5.3: Baseline training configuration shared across supervised models (ResNet-18, DeiT-Small).

Training Parameters	
Optimizer	AdamW ($\beta_1 = 0.9$, $\beta_2 = 0.999$)
Learning rate	1×10^{-3} (ResNet-18), 1.25×10^{-4} (DeiT-Small, scaled as $5 \times 10^{-4} \times 64/256$)
Weight decay	1×10^{-4} (ResNet-18), 5×10^{-2} (DeiT-Small)
Scheduler	Cosine annealing ($T_{\max} = 50$, $\eta_{\min} = 0$) with 5-epoch linear warmup (DeiT-S only)
Loss function	Class-weighted cross-entropy: $w_c = \frac{\max(\text{class\_counts})}{\text{class\_counts}_c}$
Batch size	64
Training epochs	50
Image size	224×224 pixels
Regularization	(ResNet-18), Stochastic depth (rate=0.1) (DeiT-Small)
Data Augmentation	
Geometric	Random horizontal flip (p=0.5)
Photometric	Color jitter: brightness/contrast/saturation ± 0.1, hue ± 0.02 (p=0.3)
Normalization	ImageNet statistics (mean: [0.485, 0.456, 0.406], std: [0.229, 0.224, 0.225])

For **ResNet-18**, training follows common fine-tuning protocols for CNNs [25]: AdamW optimizer [37] with learning rate 1×10^{-3} and weight decay 1×10^{-4}. For **DeiT-S**, we adapt the transformer-specific training recipe [62] with stronger regularization (weight decay 5×10^{-2}, stochastic depth rate 0.1). The base learning rate of 5×10^{-4} is scaled by batch size following standard practice: $5 \times 10^{-4} \times (64/256) = 1.25 \times 10^{-4}$.

Batch size 64 is selected for both models based on available GPU memory (8 GB). Class-weighted cross-entropy addresses the 25.78% defect prevalence imbalance. Only horizontal flips are employed, following standard augmentation practices in computer vision.

Validation protocol: Each configuration is evaluated with 5-fold cross-validation repeated across three random seeds (0, 42, 123), yielding 15 runs per configuration. The configuration with the highest mean AUPRC aggregated over all runs is selected as optimal.

Classification threshold: A fixed probability threshold of 0.5 to distinguish defective from non-defective images provides a consistent baseline for supervised models and supports fair comparisons. For the unsupervised PatchCore approach, a data-dependent threshold is selected on validation data, reflecting its different scoring mechanism.

5.2 Stage-Wise Experimental Implementation

This section presents the six-stage experimental pipeline, where each stage builds upon the previous one by fixing a key design decision.

Experiment 1: Preprocessing Pipeline Selection Six preprocessing pipelines are evaluated to identify the approach that best enhances defect visibility. Table 5.4 summarizes the configurations, which combine strategies define in Section 4.2. Background blur uses Gaussian kernel $\sigma = 5.0$ with 5-pixel mask dilation, CLAHE is applied in CIELAB color space to the L* channel only [47] with clip limit 2.0 and tile size 8×8 pixels. Figure 5.1 illustrates the visual effect of each transformation on a representative rail patch. ResNet-18 serves as the evaluation model (90 total training runs: 6 pipelines $\times$ 5 folds $\times$ 3 seeds).

Table 5.4: Preprocessing pipeline configurations. The L* channel refers to the luminance component of CIELAB color space, replicated to three channels for model compatibility.

Pipeline Name	Background	Contrast	Color Space
`ORIGINAL`	Unmodified	None	RGB
`BLACK_BG`	Constant Black	None	RGB
`BLACK_BG+CLAHE`	Constant Black	CLAHE	RGB
`BLUR_BG`	Gaussian Blur	None	RGB
`BLUR_BG+CLAHE`	Gaussian Blur	CLAHE	RGB
`BLUR_BG+CLAHE+GREY`	Gaussian Blur	CLAHE	Grayscale (L*)

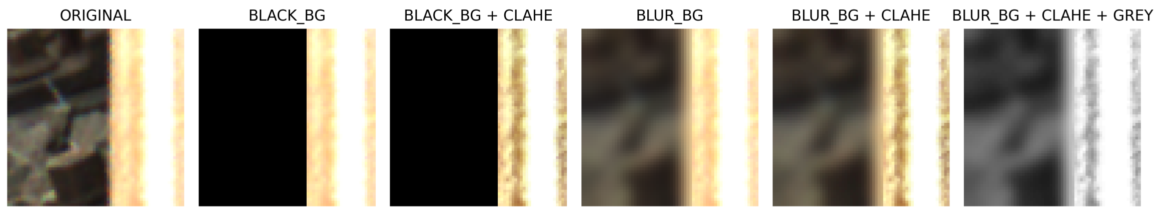


Figure 5.1: Example rail patch processed with each preprocessing pipeline. From left to right: `ORIGINAL`, `BLACK_BG`, `BLACK_BG+CLAHE`, `BLUR_BG`, `BLUR_BG+CLAHE`, `BLUR_BG+CLAHE+GREY`.

Experiment 2: Data Splitting Robustness Assessment The five candidate splits generated using the stratified allocation algorithm with random seeds 0, 26, 42, 79 and 123 are compared. ResNet-18 with the selected preprocessing pipeline from Experiment 1 is trained on each split (75 total training runs: 5 splits $\times$ 5 folds $\times$ 3 seeds). *Selection criterion: Median-performing split (rank 3 of 5 by mean AUPRC) to reduce overfitting risk while avoiding worst-case scenarios.*

Experiment 3: Model Architecture Comparison Three architectural paradigms are evaluated: ResNet-18 (supervised CNN), DeiT-Small (supervised transformer) and PatchCore (unsupervised anomaly detection). ResNet-18 and DeiT-Small follow the configurations established in Section 5.1.

PatchCore [54] uses a frozen ImageNet-pretrained Wide-ResNet50-2 backbone to extract mid-level patch features from layers 2 and 3, each projected to 1024-D embeddings. Images are resized to 256 and center-cropped to 224×224. A memory bank is built from normal training samples only, using greedy k-Center coreset subsampling (10% sampling ratio).

Anomaly scoring uses 1-nearest-neighbor search (using the FAISS library for optimized Euclidean distance calculation), aggregating patch-level scores to image level via max-pooling. A single global threshold is selected on validation data.

Experiment 4: Sequence-aware Post-Processing Optimization Multiple spatial refinement strategies are evaluated using per-seed predictions from Experiment 3. First, baseline ensemble averages probabilities across the three seeds as described in Section 4.4 ($\bar{p}_{\text{def}}(t) = \frac{1}{3}\sum_{s\in\{0,42,123\}} p_s^{(t)}$). This ensemble also computes cross-seed uncertainty: $\sigma_{\text{seeds}}(t) = \text{std}(\{p_0^{(t)}, p_{42}^{(t)}, p_{123}^{(t)}\})$, which quantifies prediction consistency across seeds. All subsequent spatial refinement methods operate on these ensemble probabilities. Sequences are constructed by grouping frames by *station*, *track_ id*, *side* and sorting by frame number, ensuring spatial ordering within each track segment.

Methods evaluated: Twelve refinement methods are evaluated with over 100 unique parameter configurations tested in total. The specific search spaces are detailed in Table 5.5. All random operations (including LSTM inner cross-validation splits, weight initialization and data shuffling) use one fixed random seed (42).

Per-fold optimization protocol: All hyperparameters are tuned *independently per fold* using only the training data from that fold. This design choice reflects real-world deployment scenarios where methods are optimized on available training data and critically ensures each validation set remains truly held out for unbiased performance assessment. The objective here is to prevent artificially optimistic results from cross-fold information leakage. Thus, this procedure provides honest estimates of whether spatial refinement improves performance compared to the baseline ensemble.

The optimization procedure differs by method type. Deterministic methods (all except LSTM) employ exhaustive grid search over predefined parameter ranges (Table 5.5). Each configuration is evaluated on training data using the criterion: maximize precision subject to recall ≥ 0.75, with F1 as fallback. This absolute threshold (0.75) provides a consistent target across all folds during hyperparameter search, representing an acceptable minimum detection rate for operational deployment. The best-performing configuration is selected and applied once to validation data.

The LSTM spatial refinement method requires a distinct procedure due to its learned nature. Unlike deterministic methods, the LSTM needs gradient-based training, creating risk of overfitting during hyperparameter search. Therefore, training data from the current fold are split into train_inner (80%) and val_inner (20%) at the track group level. This means, all sequences from the same *station*, *track*, *side* are kept together to prevent leakage. For each hyperparameter configuration, the model is trained on train_inner with early stopping on val_inner (patience = 30 epochs using validation AUPRC). The configuration achieving best val_inner performance is selected, then a final model is retrained on the full training data from the fold using the selected hyperparameters (100 epochs, monitoring training AUPRC). This final model is evaluated once on held-out validation data. This inner cross-validation ensures fair comparison with deterministic methods while following best practices in deep learning pipelines.

LSTM architecture and features: The LSTM processes three features per frame: ensemble defect probability $\bar{p}_{\text{def}}(t)$, ensemble good probability $\bar{p}_{\text{good}}(t) = 1 - \bar{p}_{\text{def}}(t)$ and cross-seed uncertainty $\sigma_{\text{seeds}}(t)$. The bidirectional architecture captures both forward and backward spatial context. Training employs class-weighted cross-entropy loss to handle class imbalance, gradient clipping (max norm = 1.0) for stability and sequence-aware padding to process variable-length track segments. The model outputs refined binary predictions per frame.

Aggregation and selection: After completing all five folds with independent per-fold tuning, results are aggregated to compute cross-fold mean and standard deviation for each method. Two methods are selected for application in Experiment 5 based on aggregated validation performance across all folds. The primary selection maximizes precision subject to recall $\geq 0.95 \times$ recall$_{\text{ensemble}}$, with F1 score as tiebreaker if no method meets the threshold. This *relative* threshold differs from the absolute 0.75 used during hyperparameter tuning: it adapts to the baseline ensemble's achieved performance and allows a maximum 5% recall degradation, reflecting operational priorities where false alarms incur high inspection costs (prioritizing precision) while maintaining detection performance near baseline levels. The secondary selection chooses the method with highest mean AUPRC across folds, applicable only to probability-based methods, for consistency with Experiments 1–3. If the same method satisfies both criteria, only one is carried forward.

Evaluation protocol: All methods report metrics computed from their single evaluation on held-out validation data. Probability-based methods (outputting refined probabilities rather than hard binary decisions) report AUPRC, precision, recall and F1, binary-only methods report precision, recall and F1.

Table 5.5: Hyperparameter search spaces for spatial refinement methods.

Method	**Parameter**	**Search Space**
Spatial Smoothing	window size	$\{1, 2, 3\}$
Confidence-Weighted Smoothing	σ (Gaussian)	$\{0.5, 1.0, 1.5, 2.0\}$
	confidence threshold	$\{0.05, 0.1, 0.2\}$
Confidence-Weighted Median	kernel size	$\{3, 5, 7, 9\}$
	confidence threshold	$\{0.05, 0.1, 0.2\}$
Multiscale Ensemble	σ sets	$\{[0.5, 1, 1.5, 2], [1, 2, 3], [0.5, 1, 2, 3, 4], [0.5, 1, 2], [1, 2, 4]\}$
Defect Voting	window size	$\{3, 5, 7, 9\}$
	confidence threshold	$\{0.05, 0.1, 0.2\}$
Min Cluster Size	min cluster size	$\{2, 3, 4, 5, 6, 7\}$
Isolated Removal	context size	$\{1, 2, 3, 4, 5\}$
Four Hybrid Methods (listed in 4.4.3)	(component parameters)	(product of component grids)
LSTM	hidden size	$\{32, 64\}$
	num layers	$\{1, 2\}$
	dropout	$\{0.2, 0.3\}$
	learning rate	$\{10^{-3}, 5 \times 10^{-4}\}$

Experiment 5: Final Model Optimization and Test Set Evaluation Random search samples 30 hyperparameter configurations (Table 5.6) using seed 42. Each configuration is assessed via 5-fold cross-validation (150 training runs in total). After selecting

the configuration with the highest mean AUPRC, three final models (seeds 0, 42, 123) are trained on the inner folds 1–5 (i.e., 80% of the data) using the selected preprocessing and model, together with the optimized hyper-parameters, for 50 epochs. Then, ensemble predictions averaged probabilities across seeds. Finally, the two post-processing methods identified in Experiment 4 are applied unchanged to the ensemble output on the held-out test set, using their previously determined parameters, to evaluate generalisation and the effect of spatial refinement.

Table 5.6: Hyperparameter search space for DeiT-Small optimization.

Parameter	**Search Range**
Learning rate	$[1\times10^{-5}, 1\times10^{-3}]$ (log-uniform)
Weight decay	[0.0, 0.1] (uniform)
Warmup epochs	[5, 10] (uniform integer)
Batch size	$\{32, 64\}$ (discrete)
Fixed: 50 epochs, image size 224×224*, drop path 0.1, AdamW optimizer*	

Experiment 6: XAI Analysis Saliency, Integrated Gradients and SmoothGrad are applied to the test-set results from the three trained seeds (with the selected preprocessing and model of Experiment 5). Table 5.7 summarizes the methods and their computational requirements. Analysis focused on 15 representative samples: True Positives (top 5 highest-confidence correct defect predictions), False Positives (top 5 highest-confidence incorrect predictions) and False Negatives (bottom 5 lowest-confidence missed defects). Four-panel layouts are generated at 300 DPI showing (1) original image, (2) saliency heatmap (jet colormap), (3) overlay (α=0.5) and (4) thresholded high-importance regions (threshold=0.6). Supplementary analyses included confidence distribution analysis by prediction category.

Table 5.7: Gradient-based attribution methods for model interpretation.

Method	**Description**	**Cost**
Gradient Saliency	Direct gradient: $\mathbf{A}_{\text{grad}}(\mathbf{x}) = \max_c \left\| \frac{\partial z_c}{\partial \mathbf{x}_c} \right\|$	1 backward pass
Integrated Gradients	Accumulated gradients along a straight-line path from a black baseline (30 steps)	30 forward + backward passes
SmoothGrad	Averaged gradient saliency over 30 noisy samples ($\sigma = 0.15$)	30 backward passes

5.3 Computational Resources

All experiments are conducted on the hardware described in Section 5.1. The total computational cost for this study is approximately **36 GPU-hours**. Training a single DeiT-Small model for 50 epochs on one fold requires approximately **6.3 minutes**. Memory usage peakes at **8 GB** of GPU VRAM and **24 GB** of system RAM during training. The total storage footprint for datasets, model checkpoints and results is approximately **20 GB**.

Table 5.8: Breakdown of computational requirements by experiment.

Experiment	**Approx. GPU-hours**	**Key Contributor**
Exp 1: Preprocessing Selection	7.8	75 runs (3 seeds × 5 pipelines × 5 folds)
Exp 2: Split Robustness	7.8	75 runs (3 seeds × 5 splits × 5 folds)
Exp 3: Model Comparison	3.9	45 runs (3 seeds × 3 models × 5 folds)
Exp 4: Spatial Refinement	0.25	(≈ 300 configurations × 5 folds
Exp 5: Hyperparameter Tuning	16	150 runs (1 seed × 5 folds × 30 configs)
Exp 5: Final Training	0.4	3 runs (3 seeds on full 80% data)
Exp 6: Explainability (XAI)	0.05	CPU-based analysis
Total	**36.2.**	

Chapter 6

Results

This chapter presents the experimental findings from the six-stage pipeline described in Chapters 5. Results are organized sequentially, reflecting the progressive refinement of the rail defect detection system.

6.1 Experiment 1: Preprocessing Pipeline Selection

Experiment 1 aims to identify the preprocessing pipeline that maximizes the ability to distinguish between defective and non-defective rail patches by comparing six alternatives.

Table 6.1 summarizes the mean and standard deviation performance metrics across all five validation folds for each preprocessing pipeline. The performance variance across seeds and folds ranges from 0.109 (BLUR_BG+CLAHE+GREY recall) to 0.212 (ORIGINAL precision), with an average standard deviation of approximately 0.15 across all metrics and pipelines.

The BLUR_BG+CLAHE configuration has the highest mean AUPRC of 0.883 ± 0.122 and outperforms the baseline ORIGINAL pipeline by 0.065 points (7.9% relative improvement). This pipeline also demonstrates the best precision (0.821 ± 0.138) and maintains competitive recall (0.862 ± 0.162), resulting in a balanced F1 score of 0.830 ± 0.123.

Table 6.1: Experiment 1: Performance of preprocessing pipelines (mean $\pm$ standard deviation across 15 runs: 5 folds $\times$ 3 seeds). Best values per metric shown in bold.

Pipeline	AUPRC	F1-score	Recall	Precision
BLUR_BG+CLAHE	$\mathbf{0.883 \pm 0.122}$	0.830 ± 0.123	0.862 ± 0.162	$\mathbf{0.821 \pm 0.138}$
BLUR_BG+CLAHE+GREY	0.873 ± 0.146	$\mathbf{0.842 \pm 0.132}$	$\mathbf{0.885 \pm 0.109}$	0.810 ± 0.158
BLACK_BG+CLAHE	0.840 ± 0.186	0.799 ± 0.150	0.821 ± 0.195	0.796 ± 0.137
BLUR_BG	0.838 ± 0.135	0.790 ± 0.167	0.883 ± 0.141	0.732 ± 0.199
ORIGINAL	0.818 ± 0.192	0.784 ± 0.183	0.845 ± 0.153	0.742 ± 0.212
BLACK_BG	0.772 ± 0.171	0.759 ± 0.138	0.829 ± 0.193	0.727 ± 0.146

Since BLUR_BG+CLAHE is the best performing pipeline we test it also with its grayscale variant (BLUR_BG+CLAHE+GREY). That achieves comparable performance (AUPRC: 0.873 ± 0.146)

and demonstrates the highest recall (0.885 ± 0.109), indicating superior sensitivity to defects. However, precision dropped (0.810 ± 0.158), resulting in a slightly higher false positive rate. The standard deviation of AUPRC for this pipeline (0.146) is higher than the RGB variant (0.122), suggesting greater variance across folds and seeds.

Pipelines using black background replacement (`BLACK_BG` and `BLACK_BG+CLAHE`) shows lower performance compared to their blurred counterparts. The `BLACK_BG` pipeline has the lowest AUPRC (0.772 ± 0.171), while `BLACK_BG+CLAHE` improves to 0.840 ± 0.186 but remains 0.043 points below `BLUR_BG+CLAHE`.

The `BLUR_BG` pipeline without contrast enhancement achieves moderate performance (AUPRC: 0.838 ± 0.135), demonstrating that background handling alone provides improvement over the unprocessed `ORIGINAL` configuration. However, adding CLAHE yields an additional 0.045 AUPRC points, highlighting its ability to improve defect visibility.

Figure 6.1 presents both the mean AUPRC and the full distribution of AUPRC scores for each pipeline, to provide an intuitive visual comparison of performance differences across pipelines. The boxplots further show that `BLUR_BG+CLAHE` not only achieves the highest mean AUPRC, but also maintains relatively stable performance across folds, with most runs in the upper quartile (AUPRC 0.75–1.0). In contrast, pipelines without contrast enhancement (`BLUR_BG`, `ORIGINAL`, `BLACK_BG`) exhibit greater variability and more frequent low-performing runs, reflected in wider distributions and extended lower whiskers.

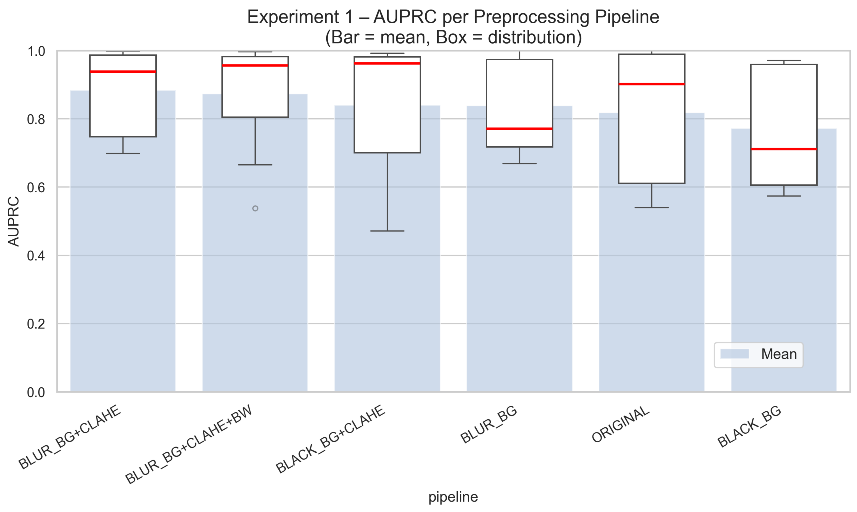


Figure 6.1: AUPRC comparison across preprocessing pipelines. Blue bars show the mean AUPRC averaged over 15 runs (5 folds $\times$ 3 seeds), error bars indicate ± 1 standard deviation. Boxplots overlay the full distribution, with red lines marking the median, box edges representing the interquartile range (25th–75th percentile) and whiskers extending to $1.5\times$ the IQR. Outliers are shown as individual points.

Based on these results, the `BLUR_BG+CLAHE` pipeline is selected for all the following experiments. This decision is motivated by its superior AUPRC, balanced precision-recall trade-off and moderate variance, making it the most reliable choice for subsequent model comparison and optimization, even with the grayscale variant having slightly higher recall.

6.2 Experiment 2: Data Splitting Robustness

Experiment 2 evaluates the stability of cross-validation performance across different fold compositions to ensure that the results are not the product of a single advantageous data partition. The `BLUR_BG+CLAHE` pipeline, selected in Experiment 1, is used.

Table 6.2 presents the aggregated performance across five independently generated 5-fold cross-validation sets (seeds: 0, 26, 42, 79, 123). The spread of mean AUPRC values across seeds ranges from 0.813 to 0.883. The variance within seeds, indicated by standard deviations ranges from 0.122 to 0.192.

Table 6.2: Experiment 2: Sensitivity to cross-validation seed (mean ± SD over 5 folds) using ResNet-18 with the `BLUR_BG+CLAHE` pipeline. Best values per metric are in bold. Seed 79 (shaded) is selected as the median-performing seed for subsequent experiments.

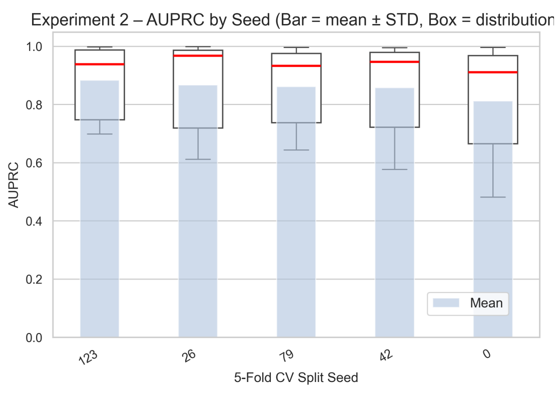
Seed	AUPRC	F1-score	Recall	Precision
123	$\mathbf{0.883 \pm 0.122}$	$\mathbf{0.830 \pm 0.123}$	0.862 ± 0.162	$\mathbf{0.821 \pm 0.138}$
26	0.867 ± 0.145	0.813 ± 0.151	0.850 ± 0.192	0.802 ± 0.145
79	0.861 ± 0.133	0.828 ± 0.129	$\mathbf{0.904 \pm 0.096}$	0.777 ± 0.170
42	0.858 ± 0.158	0.798 ± 0.169	0.835 ± 0.198	0.793 ± 0.174
0	0.813 ± 0.192	0.799 ± 0.163	0.871 ± 0.151	0.763 ± 0.199

Seed 79 is identified as the median performer with an AUPRC of 0.861 ± 0.133, ranking third among the five candidates. Its AUPRC standard deviation of 0.133 positions it in the middle range of observed variance, avoiding both the lowest variance (seed 123: 0.122) and highest variance (seed 0: 0.192), thereby representing a realistic estimate of model stability. This middle-ground performer choice aims to neither overestimate (seed 123) nor underestimate (seed 0) expected model capability, prioritizing generalizability and reproducibility for subsequent experiments.

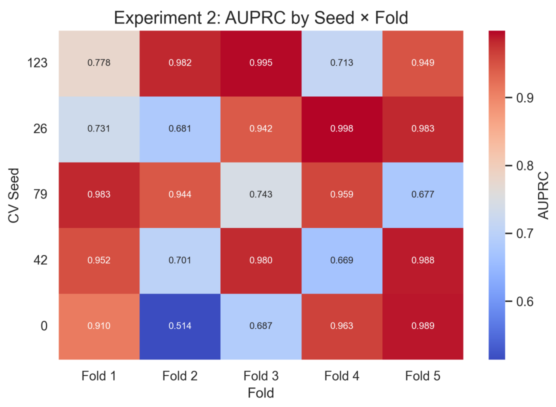
Figure 6.2a illustrates the distribution of AUPRC values across the five seeds. All seeds achieve median AUPRC values above 0.90, with their interquartile ranges concentrated in the upper performance tier. Seed 0 exhibits the lowest median and wider distribution, with its lower whisker extending to approximately 0.51.

Figure 6.2b indicates that the drop in performance is caused by an outlier fold (Fold 2, seed 0, AUPRC = 0.514). This fold is examined further to understand the source of degraded performance. Analysis shows that it contains sequences from stations *ruebenkamp_9.2* and *fire_site_3.4*, where $70 - -100\%$ of patches are incorrectly classified as defective. These stations are completely absent from the training set: the complementary track from sequence 9.2 is reserved in the held-out test set (Set E), while both tracks from station 3.4 appear exclusively in validation fold 2.

Beyond this outlier, Figure 6.2b shows that no single seed universally optimizes all folds, with values ranging from 0.514 (seed 0, fold 2) to 0.998 (seed 26, fold 4). Regarding the coefficient of variation ($CV = \frac{\sigma}{\mu}$) of mean AUPRC across seeds is approximately 2%, while within-seed fold variability is considerably higher (average CV: 6%).



(a) Distribution across seeds



(b) Per-fold performance heatmap

Figure 6.2: Experiment 2: Cross-validation splitting analysis. (a) AUPRC distributions show moderate sensitivity to seed choice, with seed 0 exhibiting lower performance. (b) Heatmap reveals fold-level heterogeneity, with AUPRC ranging from 0.514 to 0.998 across seed-fold combinations.

Table 6.3 summarizes the composition of the selected 5-fold split (seed 79), illustrating the stratification achieved across key dataset characteristics. The selected split demonstrates a reasonable balance across multiple dimensions. Sequence counts range from 11 to 13 per fold, total patches vary from 210 to 290 and station representation is relatively uniform (10–12 stations per fold), ensuring that most folds encounter diverse track environments. However, defect ratios exhibit notable variation across folds (14.48%–43.81%). Fold 2 contains the highest defect concentration (43.81%), while folds 4 and 5 represent more challenging scenarios with defect rates below the overall mean of 25%. This variation, while inevitable given the constraint of maintaining sequence integrity within folds, ensures that model performance is evaluated under both defect-rich and defect-sparse conditions, providing a more realistic assessment of generalization capability.

Table 6.3: Composition of the selected 5-fold cross-validation split (seed 79) used in Experiments 3–5. The stratification algorithm balanced sequence count, patch distribution, defect ratio and station diversity across folds.

Fold	Sequences	Total Patches	Defective Patches	Stations	Defect %
1	11	290	100	10	34.48
2	12	210	92	9	43.81
3	13	220	42	9	19.09
4	11	290	49	9	16.90
5	11	290	42	9	14.48
Total	**58**	**1,300**	**325**	–	**25.00**

The consistent use of seed 79 in Experiments 3, 4 and 5 ensures that model comparisons, post-processing optimizations and tuning of parameters are conducted under fair and equivalent data partitioning conditions.

6.3 Experiment 3: Model Architecture Comparison

Experiment 3 compares the already introduced model architectures to identify the learning paradigm best suited for rail defect detection: ResNet-18 (CNN baseline), DeiT-Small (Vision Transformer) and PatchCore (unsupervised anomaly detection). All models are trained using the `BLUR_BG+CLAHE` preprocessing on the seed 79 cross-validation split selected in Experiment 2.

Table 6.4 presents the performance comparison across all metrics. DeiT-Small achieves the highest AUPRC of 0.937 ± 0.073, outperforming ResNet-18 (0.900 ± 0.120) by 0.037 points (4.1% relative improvement) and significantly exceeding PatchCore (0.608 ± 0.185) by 0.329 points (54.1% relative improvement).

Table 6.4: Experiment 3: Performance comparison of model architectures (mean $\pm$ standard deviation across 15 runs: 5 folds $\times$ 3 seeds). Best values per metric shown in bold. DeiT-Small is selected for subsequent experiments based on superior AUPRC and balanced precision-recall trade-off.

Model	**AUPRC**	**F1-Score**	**Recall**	**Precision**
DeiT-Small	$\mathbf{0.937 \pm 0.073}$	$\mathbf{0.814 \pm 0.121}$	$\mathbf{0.859 \pm 0.189}$	$\mathbf{0.823 \pm 0.161}$
ResNet-18	0.900 ± 0.120	0.773 ± 0.115	0.788 ± 0.192	0.809 ± 0.158
PatchCore	0.608 ± 0.185	0.680 ± 0.117	0.839 ± 0.076	0.591 ± 0.153

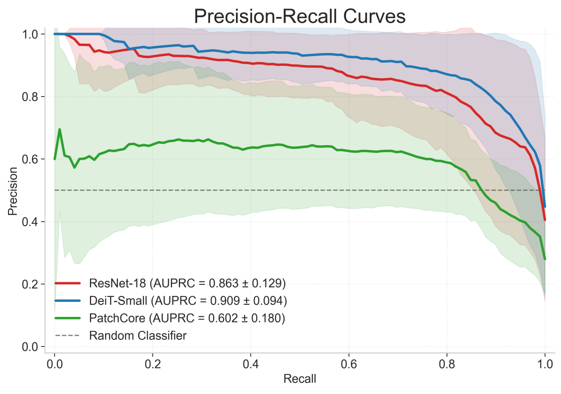
Beyond AUPRC, DeiT-Small demonstrates superior performance across all threshold-dependent metrics. Its F1-score of 0.814 ± 0.121 exceeded ResNet-18 (0.773 ± 0.115) and PatchCore (0.680 ± 0.117), reflecting a better balance between precision and recall. DeiT-Small achieves the highest precision (0.823 ± 0.161) among all models, indicating fewer false alarms. Simultaneously, it maintains the highest recall (0.859 ± 0.189), detecting a larger proportion of genuine defects compared to ResNet-18 (0.788 ± 0.192) while matching PatchCore's recall (0.839 ± 0.076) but with superior precision.

ResNet-18, while achieving respectable AUPRC (0.900 ± 0.120), exhibits notable precision-recall imbalance. Its precision (0.809 ± 0.158) remains competitive with DeiT-Small, but recall (0.788 ± 0.192) lags by 7.1 percentage points. The standard deviation in ResNet-18's AUPRC (0.120) is higher than DeiT-Small's (0.073).

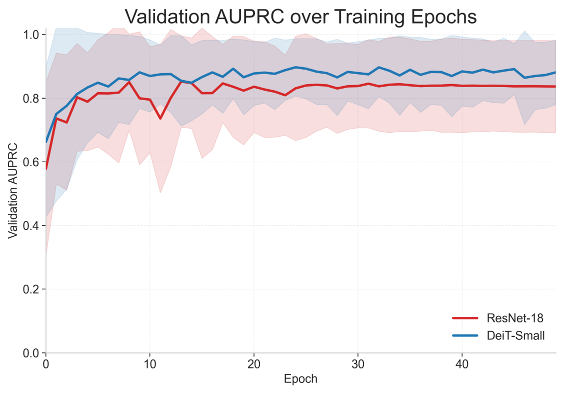
PatchCore's performance falls short of supervised approaches, achieving AUPRC of only 0.608 ± 0.185, the lowest among all models despite exhibiting competitive recall (0.839 ± 0.076). This recall-precision imbalance (recall: 0.839, precision: 0.591) is evident. The larger standard deviation (0.185) further indicates more unstable performance across folds than the other models.

Figure 6.3 provides complementary visualizations of model performance. The precision-recall curves (Figure 6.3a) illustrate the trade-off across all classification thresholds, with shaded regions indicating fold-level variability. DeiT-Small (blue) maintains precision above 0.85 across recall levels up to 0.80. ResNet-18 (red) exhibits a similar curve shape but with consistently lower precision for equivalent recall levels. PatchCore (green) shows a different profile: while achieving high initial recall at low thresholds, precision rapidly degrades, with the curve remaining well below both supervised models across the entire operating range. The horizontal dashed line at precision $= 0.50$ (random classifier baseline) emphasizes PatchCore's limited ability to discriminate in this task.

The training curves (Figure 6.3b) reveal additional insights. DeiT-Small (blue) converges rapidly within the first 10 epochs, reaching a validation AUPRC plateau of approximately 0.90 and maintaining stable performance throughout the 50-epoch training schedule. The narrow shaded region indicates consistent convergence behavior across folds and seeds. ResNet-18 (red) exhibits slower convergence, requiring approximately 15-20 epochs to stabilize at a lower plateau around 0.84. The slightly wider variance area suggests greater sensitivity to initialization and fold composition. Both supervised models demonstrate no signs of overfitting, with validation performance either improving or remaining stable throughout training. This pattern is consistent with the strong regularization provided by limited training data (approximately 1,040 patches per fold after 80-20 train-validation split).



(a) Precision-Recall curves



(b) Validation AUPRC over training

Figure 6.3: Experiment 3: Model performance comparison. (a) Precision-recall curves show DeiT-Small's superior performance across all operating points, with shaded regions indicating ± 1 standard deviation across folds. (b) Training curves demonstrate DeiT-Small's faster convergence and higher plateau compared to ResNet-18, with shaded regions showing variability across 15 runs (5 folds $\times$ 3 seeds).

Based on these results, DeiT-Small is selected as the best-performing architecture for the experiments left: sequential post-processing (Experiment 4) and final evaluation (Experiment 5).

6.4 Experiment 4: Sequential Post-Processing Methods

Experiment 4 evaluates the twelve sequnce-aware post-processing methods (detailed in Section 4.4), that exploit the sequential nature of railway inspection data to refine and potentially improve predictions.

6.4.1 Baseline Ensemble Performance

Before applying sequential post-processing, the baseline ensemble's (Equation 4.9) performance is evaluated and compared against single-seed predictions. Table 6.5 presents this comparison, demonstrating the effectiveness of ensemble averaging.

Table 6.5: Experiment 4: Comparison of single-seed baseline versus ensemble baseline (mean ± standard deviation across 5 folds). The ensemble approach demonstrates consistent improvement across all metrics

Method	**AUPRC**	**F1**	**Recall**	**Precision**
Baseline Single-Seed	0.9107 ± 0.1020	0.8599 ± 0.1014	0.9080 ± 0.1249	0.8299 ± 0.1270
Baseline Ensemble	$\mathbf{0.9246 \pm 0.0924}$	$\mathbf{0.8740 \pm 0.0975}$	$\mathbf{0.9170 \pm 0.1240}$	$\mathbf{0.8474 \pm 0.1241}$

The ensemble baseline achieves superior performance across all metrics compared to single-seed predictions, with AUPRC improving from 0.9107 to 0.9246 ($+1.4$ percentage points), precision increasing from 0.8299 to 0.8474 (+1.8 percentage points) and F1 score rising from 0.8599 to 0.8740 (+1.4 percentage points). Notably, the ensemble approach also reduces variance: standard deviations decreases for AUPRC (0.1020 to 0.0924, -9.4%), precision (0.1270 to 0.1241, -2.3%) and F1 (0.1014 to 0.0975, -3.8%), indicating more stable performance. This improvement validates the ensemble averaging strategy described in Section 4.4 and establishes a baseline for subsequent sequential refinement.

Therefore, all post-processing methods are applied to the ensemble probabilities $\bar{p}_{\text{def}}(t)$ rather than individual seed predictions. This choice offers dual benefits: it reduces computational complexity by avoiding redundant processing across seeds, while ensuring that sequential refinement operations build upon the most stable and noise-reduced probability estimates.

6.4.2 Comparative Performance of Post-Processing Methods

Figure 6.4 visualizes performance changes (Δ AUPRC, Δ Precision, Δ Recall, Δ F1) for all methods relative to the baseline ensemble. Methods are sorted by Δ AUPRC, with green bars indicating improvements and red bars indicating degradations.

The Δ AUPRC panel shows modest improvements, with four methods improving ($+0.0001$ to $+0.0109$) and four degrading (-0.0051 to -0.0706). The best performer is *Multiscale*, which, when considering full numerical precision, marginally outperforms *Confidence-Weighted + Multiscale* by $+0.0001$ AUPRC points. *Defect Voting* shows the largest degradation (-0.0706), where majority voting over-smoothed predictions, hiding short defects while failing to correct high-confidence false positives.

The Δ Precision panel reveals more pronounced improvements, with ten of twelve methods increasing precision. *Confidence-Weighted + Isolated Removal* shows the largest gain ($+0.0232$), demonstrating that removing isolated positive predictions effectively eliminates false alarms. *Isolated Removal* alone gains $+0.0215$ and *Confidence-Weighted Median* gains $+0.0193$. Only two methods degrade precision: *LSTM* (-0.0168) and *Multiscale Vote* (-0.0118). The strong precision improvements validate the assumption that genuine defects rarely occur as isolated single-frame detections.

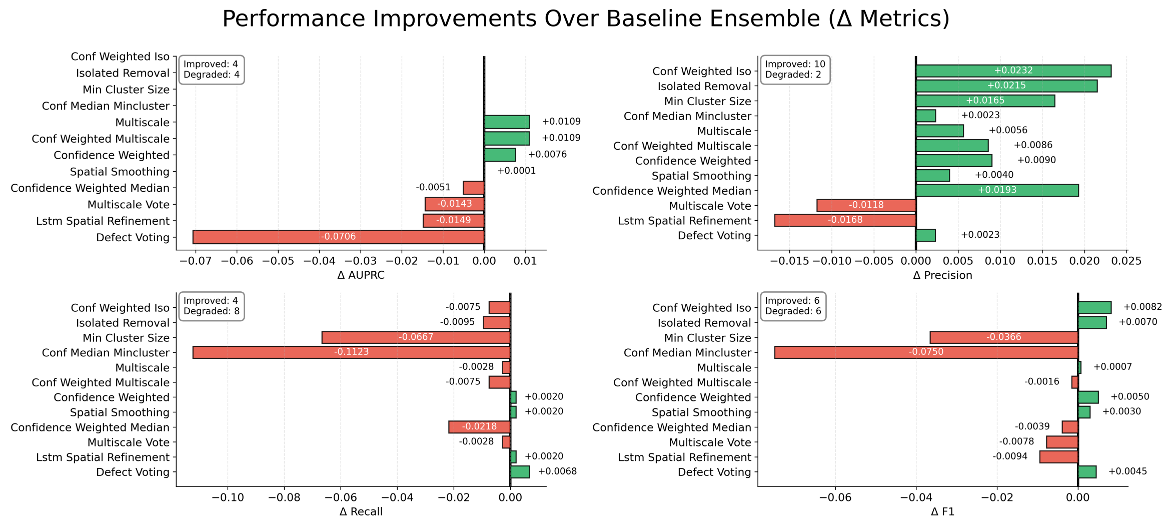


Figure 6.4: Performance improvements over baseline ensemble across all post-processing methods. Each subplot shows the change in a specific metric (Δ = Method - Base Ensemble). Methods are sorted by Δ AUPRC for consistent ordering. Green bars indicate improvement, red bars indicate degradation. Annotations report the number of methods improving versus degrading for each metric. Binary-only methods (*Isolated Removal*, *Conf Weighted Iso*, *Min Cluster Size*, *Conf Median Mincluster*) show nothing for Δ AUPRC as they produce discrete decisions without probability scores.

The Δ Recall panel shows that eight of twelve methods degrade recall, reflecting the asymmetric nature of the specific binary refinement: methods removing predictions can only reduce false positives, not recover false negatives. *Confidence-Weighted Median + Min Cluster* shows the largest drop (-0.1123), followed by *Min Cluster Size* (-0.0667), where enforcing minimum cluster lengths removes short genuine defects. Only *Defect Voting* ($+0.0068$), *Confidence-Weighted* ($+0.0020$), *Spatial Smoothing* ($+0.0020$) and *LSTM* ($+0.0020$) improve recall, as probability-based smoothing can propagate predictions to adjacent uncertain frames. However, these gains are barely noticeable.

The Δ F1 panel shows six methods improving F1. *Confidence-Weighted + Isolated Removal* achieves the largest gain ($+0.0082$), followed by *Isolated Removal* ($+0.0070$) and *Confidence-Weighted* ($+0.0050$). Methods with severe recall degradation shows large F1 losses: *Confidence-Weighted Median + Min Cluster* (-0.0750) and *Min Cluster Size* (-0.0366). The modest improvements reflect the difficulty of simultaneously improving precision and recall when baseline performance is already high ($F1 = 0.8740$).

The bidirectional *LSTM*, designed to learn patterns from data, degrades performance across metrics: AUPRC (-0.0149), precision (-0.0168) and F1 (-0.0094). Limited training data of only 46-47 sequences, $\sim$1,000 patches, per inner training split provides insufficient examples for robust learning, particularly for rare defect configurations. Additionally, the baseline ensemble's high performance (AUPRC > 0.92) leaves limited room for improvement, requiring the LSTM to correct only the most ambiguous cases where patterns are contradictory. Training the *LSTM* requires hyperparameter tuning via inner cross-validation (16 configurations $\times$ 5 folds = 80 runs), consuming clearly more computational resources than deterministic methods without performance gains.

6.4.3 Selected Methods for Final Evaluation

Two methods are selected for Experiment 5 to reflect different priorities: AUPRC-optimal method is suitable for threshold exploration using calibrated probabilities, while the precision-optimal method addresses operational cost-efficiency, where frequent false alerts lead to expensive field inspections [35]. Table 6.6 presents their performance metrics.

Table 6.6: Experiment 4: Performance of the two selected post-processing methods (mean ± standard deviation across 5 folds). Multiscale Spatial Ensemble is selected for the highest AUPRC, Confidence-Weighted + Isolated Removal is selected for the highest precision while maintaining recall ≥ 0.8712 (95% of baseline).

Method	**AUPRC**	**F1**	**Recall**	**Precision**
Multiscale Tuned	$\mathbf{0.9356 \pm 0.0768}$	0.8747 ± 0.1029	$\mathbf{0.9143 \pm 0.1221}$	0.8531 ± 0.1390
Conf Weighted Iso Tuned	-	$\mathbf{0.8822 \pm 0.0963}$	0.9095 ± 0.1308	$\mathbf{0.8706 \pm 0.1215}$

Multiscale Spatial Ensemble ($\sigma = [0.5, 1.0, 2.0]$) is selected as the AUPRC-optimal method, achieving the highest AUPRC (0.9356 ± 0.0768, $+1.2\%$ relative) while maintaining balanced precision (0.8531) and recall (0.9143). The reduced standard deviation (0.0768 vs. baseline 0.0924, -16.9%) indicates more consistent performance across folds.

Confidence-Weighted + Isolated Removal ($\sigma = 0.5$, conf_thr $= 0.05$, context $= 1$) is selected as the precision-optimal method, achieving the highest precision (0.8706 ± 0.1215, $+2.7\%$ relative) and F1 score (0.8822 ± 0.0963, $+0.9\%$ relative). Although this method cannot compute AUPRC due to binary output, it effectively eliminates high-confidence false alarms. The method maintains recall at 0.9095 (-0.8% degradation), satisfying the constraint of ≥ 0.8712 (95% of baseline).

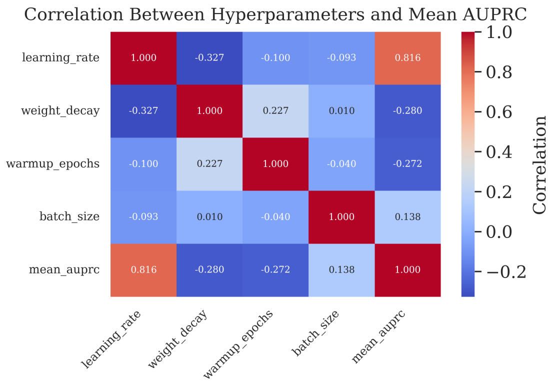
6.5 Experiment 5: Final Model Optimization and Test Set Evaluation

Experiment 5 is the final confirmatory evaluation, consisting of two stages: hyperparameter optimization, followed by training the final model on 80% complete subset and evaluating on the untouched test set (20%, Set E). This design ensures that test set performance provides an unbiased estimate of generalization capability, as Set E is reserved (Section 4.1) only for final evaluation and never included in any prior experimental decisions.

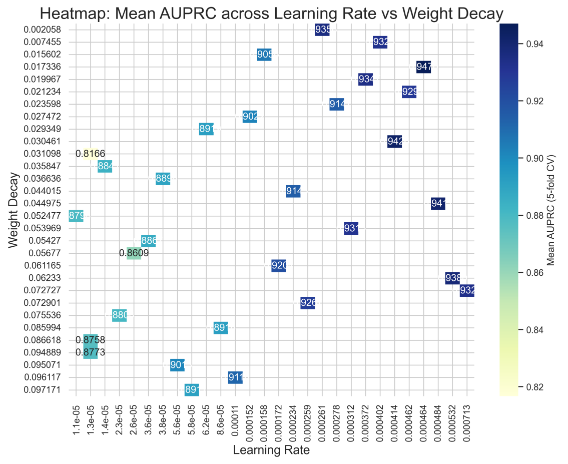
6.5.1 Hyperparameter Optimization

Random search sampled 30 hyperparameter combinations from the search space defined in Table 5.6, with each configuration evaluated via 5-fold cross-validation using three random seeds per fold (15 training runs per configuration, 150 total).

Figure 6.5 presents two complementary views of the hyperparameter landscape. The correlation analysis (Figure 6.5a) shows that the learning rate and weight decay have the strongest relationships with mean AUPRC (correlations: $+0.816$ and -0.280, respectively). Moreover, since they are themselves correlated (-0.327), we present the learning-rate vs. weight-decay heatmap in Figure 6.5b. This demonstrates that optimal performance concentrates in a specific region: moderate learning rates (3×10^{-4} to 7×10^{-4}) combined with low-to-moderate weight decay (0.002 to 0.06). The highest-performing cell (darkest blue, AUPRC ≈ 0.947) occurs at lr $= 4.64 \times 10^{-4}$, wd $= 0.0173$.



(a) Correlation between hyperparameters and mean AUPRC



(b) Mean AUPRC across learning rate and weight decay combinations

Figure 6.5: Experiment 5: Hyperparameter optimization analysis. (a) Correlation heatmap shows learning rate exhibits strong positive correlation with AUPRC ($+0.816$), while weight decay shows moderate negative correlation (-0.280). (b) Heatmap of mean AUPRC across learning rate and weight decay combinations reveals that performance concentrates in the moderate learning rate (3×10^{-4} to 7×10^{-4}) and low-to-moderate weight decay (0.002 to 0.06) region.

Table 6.7 presents the five best hyperparameter combinations ranked by mean AUPRC. The top-ranked configuration has mean AUPRC of 0.9470 ± 0.0655, with learning rate 4.64×10^{-4}, weight decay 0.0173, warmup epochs 5 and batch size 64. Notably, all top configurations use warmup of 5 epochs, suggesting that longer warmup periods do not improve performance for this dataset size.

Table 6.7: Experiment 5: Top 5 hyperparameter combinations ranked by mean AUPRC across 5-fold CV (mean $\pm$ standard deviation over 15 runs: 5 folds $\times$ 3 seeds). The best configuration (rank 1, shaded) is selected for final model training.

Rank	Learning Rate	Weight Decay	Warmup	Batch Size	AUPRC	Precision	Recall	F1
1	4.64×10^{-4}	0.0173	5	64	$\mathbf{0.9470 \pm 0.0655}$	0.8797 ± 0.1335	0.8075 ± 0.2780	0.8084 ± 0.1790
2	4.14×10^{-4}	0.0305	9	64	0.9426 ± 0.0736	0.8420 ± 0.1559	0.8286 ± 0.2401	0.8039 ± 0.1277
3	4.84×10^{-4}	0.0450	6	64	0.9411 ± 0.0733	0.8585 ± 0.1767	0.7599 ± 0.3247	0.7526 ± 0.1984
4	5.32×10^{-4}	0.0623	6	32	0.9380 ± 0.0784	0.8475 ± 0.1913	0.8361 ± 0.2420	0.8044 ± 0.1550
5	2.61×10^{-4}	0.0021	6	64	0.9358 ± 0.0777	0.8331 ± 0.2035	0.7619 ± 0.3299	0.7313 ± 0.1978

Figure 6.6 shows the validation AUPRC progression during training for the selected hyperparameters across all five folds. The training curves demonstrate rapid convergence within

the first 10 epochs, reaching mean validation AUPRC above 0.85, followed by gradual improvement to a plateau around epoch 20 (AUPRC $\approx$ 0.90). Performance remains stable through epoch 50 with no degradation, with the best mean performance occurring at epoch 45 (AUPRC: 0.947), marked by the red star. Individual fold variability (shaded region) remains moderate throughout training, with two folds showing lower performance (gray lines near 0.5 at epoch 0, converging to $\sim$0.80 by epoch 10). Those are folds 3 and 5, shown by the lowest and second-lowest starting curves, respectively, both including challenging cases that explain the lower performance.

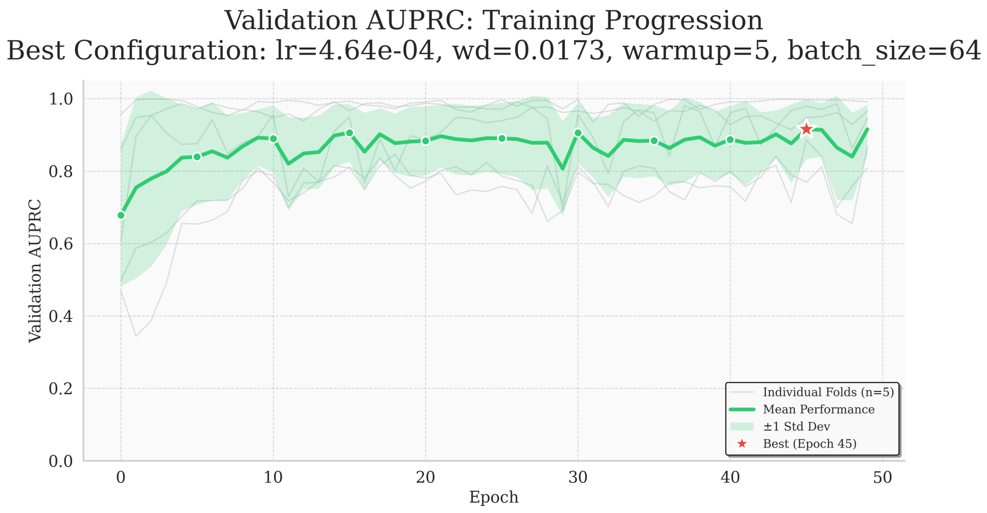
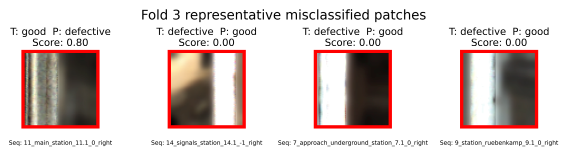
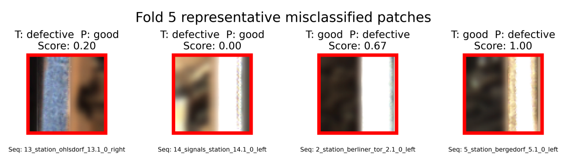


Figure 6.6: Validation AUPRC progression during training with the selected hyperparameters (lr=4.64×10^{-4}, wd=0.0173, warmup=5, batch=64). Thin gray lines show individual fold performance (at epoch 0, the lines correspond (from top to bottom) to folds 1, 4, 2, 5 and 3). Green line shows mean across folds, shaded region indicates ± 1 standard deviation. Red star marks the best epoch (epoch 45, mean AUPRC=0.947). The model converges rapidly within 10 epochs and then maintains stable performance.

Figure 6.7 shows representative examples from these folds, including shadows misclassified as defects (patches 1, 7 and 8, left to right) and subtle defects incorrectly classified as normal (remaining patches).


Fold 3 representative misclassified patches
T: good P: defective
Score: 0.80
Seq: 11_main_station_11.1_0_right
T: defective P: good
Score: 0.00
Seq: 14_signals_station_14.1_-1_right
T: defective P: good
Score: 0.00
Seq: 7_approach_underground_station_7.1_0_right
T: defective P: good
Score: 0.00
Seq: 9_station_ruebenkamp_9.1_0_right

(a) Fold 3


Fold 5 representative misclassified patches
T: defective P: good
Score: 0.20
Seq: 13_station_ohlsdorf_13.1_0_right
T: defective P: good
Score: 0.00
Seq: 14_signals_station_14.1_0_left
T: good P: defective
Score: 0.67
Seq: 2_station_berliner_tor_2.1_0_left
T: good P: defective
Score: 1.00
Seq: 5_station_bergedorf_5.1_0_left

(b) Fold 5

Figure 6.7: Characteristic error cases from lower-performing folds that include shadow artifacts misclassified as defects and subtle genuine defects missed by the model. Below each patch, the annotation indicates its sequence identifier. On top of each patch the true label, predicted label and probaility of being defective are reported. The red border is to highlight the fact that they are misclassified

In addition to these general challenges, the model also faces problems with specific, unexplored infrastructure features. A clear example is shown in Figure 6.8, which shows a series of frames depicting a railroad crossing, i.e., a location where a road crosses a railroad track. These features are rare in the training data, leading to a characteristic failure pattern. The model performs high-confidence classifications of false positives. This suggests that the model has not learned a robust representation for railroad crossings, possibly misinterpreting regular geometric patterns of the crossing surface (e.g., grooves or other material) as potential defects.

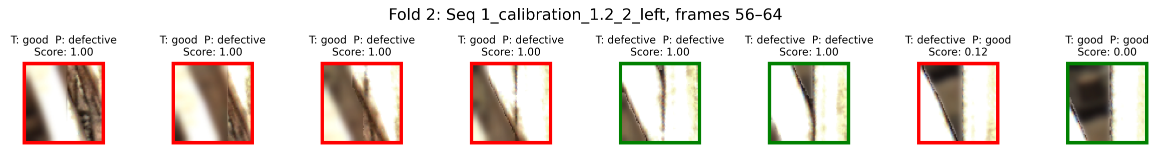


Figure 6.8: Model instability on an underrepresented rail infrastructure element (sequence belongs to fold 2). The sequence shows consecutive frames as the vehicle passes over a rail crossing. On top of each patch the true and predicted labels are noted as well as defective probability. Most model's predictions are wrong producing false positives because of the darker parts in crossings considered as defects. The scores indicate the model's confidence in its (often incorrect) prediction. Only last patch is a true negative correctly classifies probably bcause the crossed trach is barely visible. Green border is for correctly classified patches and red for wrongly classified ones

6.5.2 Final Model Training and Test Set Evaluation

Three final DeiT-Small models are trained on the complete 80% training set (folds 1-5, 1,300 patches) using the optimized hyperparameters. Table 6.8 presents the per-seed performance and ensemble results on the held-out test set (Set E, 310 patches).

Table 6.8: Experiment 5: Performance on the held-out set (Set E, 310 patches) for individual seeds and ensemble. Baseline shows mean ± standard deviation across three seeds, ensemble averages per-class probabilities across seeds before thresholding at 0.5.

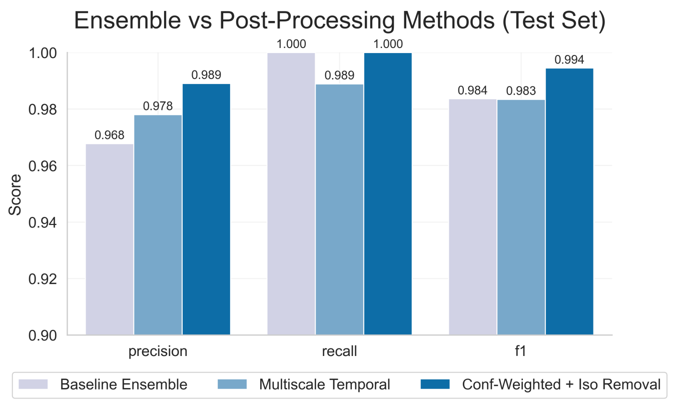
Method	**AUPRC**	**Precision**	**Recall**	**F1**
Baseline (mean ± std)	0.9743 ± 0.0285	0.9430 ± 0.0062	0.9296 ± 0.0995	0.9338 ± 0.0554
Ensemble	**0.9931**	**0.9677**	**1.0000**	**0.9836**

The ensemble achieves exceptional test set performance: AUPRC of 0.9931, precision of 0.9677, perfect recall (1.0000) and F1 score of 0.9836. Compared to the mean of individual seeds, the ensemble improves AUPRC by $+1.9$ percentage points (0.9743 to 0.9931, $+1.9\%$ relative), precision by $+2.5$ percentage points (0.9430 to 0.9677, $+2.6\%$ relative) and F1 by $+5.0$ percentage points (0.9338 to 0.9836, $+5.3\%$ relative). The perfect recall indicates that all 90 defective patches in the test set are correctly identified, with only 3 false positives among 220 non-defective patches (false positive rate: 1.4%). The low standard deviations across seeds, particularly for precision ($\sigma = 0.0062$), indicate highly stable predictions.

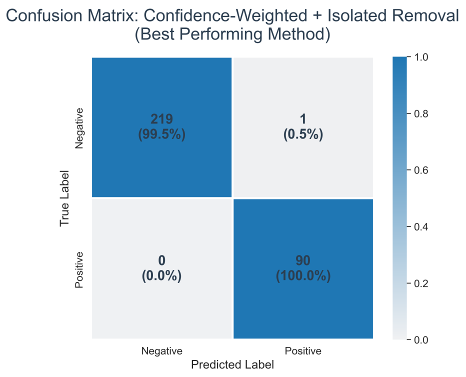
Interestingly, these test set results exceed the cross-validation performance reported during hyperparameter tuning (Table 6.7, rank 1: AUPRC $= 0.9470$, F1 $= 0.8084$).

6.5.3 Post-Processing Method Application

The two methods selected in Experiment 4, *Multiscale Spatial Ensemble* and *Confidence-Weighted + Isolated Removal*, are applied to the test set ensemble predictions using hyperparameters tuned in Experiment 4. Figure 6.9 and Table 6.9 present the performance.



(a) Performance comparison across methods



(b) Optimal confusion matrix

Figure 6.9: Experiment 5: Post-processing method performance on test set. (a) Bar chart comparing baseline ensemble (grey) with Multiscale Spatial (light blue) and Confidence-Weighted + Isolated Removal (dark blue) across precision, recall and F1. (b) Confusion matrix for Confidence-Weighted + Isolated Removal, the best-performing method, showing 90 true positives, 219 true negatives, 1 false positive and 0 false negatives (310 total patches).

Table 6.9: Experiment 5: Test set performance for baseline ensemble and two post-processing methods. Best values per metric shown in bold. Confidence-Weighted + Isolated Removal cannot compute AUPRC as it produces binary predictions.

Method	**AUPRC**	**Precision**	**Recall**	**F1**	**TP**	**TN**	**FP**	**FN**
Baseline Ensemble	0.9931	0.9677	1.0000	0.9836	90	217	3	0
Multiscale Spatial	**0.9942**	**0.9780**	0.9889	0.9834	89	218	2	1
Conf-Weighted + Iso	-	**0.9890**	**1.0000**	**0.9945**	90	219	1	0

The *Confidence-Weighted + Isolated Removal* method achieves the best overall performance, reducing false positives from 3 to 1 while maintaining perfect recall. This represents the final optimized pipeline: DeiT-Small trained with optimized hyperparameters, ensemble averaging across three seeds and Confidence-Weighted + Isolated Removal post-processing.

6.6 Experiment 6: XAI Analysis

Experiment 6 applies gradient-based attribution methods to interpret the final DeiT-Small model's predictions on the held-out test set. The main objectives are to verify that the model decisions are based on physically plausible features of the track surface and not on spurious correlations, to diagnose the root causes of the few remaining errors, and to provide transparent, visual justifications suitable for informing maintenance decisions.

6.6.1 Analysis Setup and Methodology

The analysis focuses on the test set predictions from the final ensemble model (Experiment 5). To ensure a thorough assessment, representative samples from True Positives (TP), True Negatives (TN) and False Positives (FP) are selected. The final model does not have False Negatives (FN) and that is why such cases are not included. This selection strategy targets cases where the model is most confident in correct decisions and most confidently wrong.

Three complementary gradient-based attribution methods are employed concurrently to generate saliency maps. As discussed (Section 4.5), Gradient Saliency serves as a foundational method computing the gradient of the class score with respect to input pixels, Integrated Gradients (IG) addressed gradient saturation by accumulating gradients along a path from a baseline to the input and SmoothGrad reduces visual noise by averaging saliency maps over multiple noisy versions of the input. For each sample, an eight-panel visualization is generated in a 2×4 grid configuration. The top row displays the original image followed by heatmaps from the three attribution methods (Gradient Saliency, Integrated Gradients, SmoothGrad). The bottom row shows the corresponding overlays for each method, created by blending the heatmaps with the original image at 50% opacity, providing both isolated and contextual views of the attribution patterns.

6.6.2 Model Interpretation and Trustworthiness

XAI analysis confirms that the model consistently focuses on semantically important areas of the track surface when making correct predictions. In True Positive cases, the performance maps strongly emphasize the detected defects and surface irregularities. Figure 6.10 shows two representative TP examples where all three performance methods show strong agreement, with regions of high performance centered around the defective areas. This indicates that the model has learned to recognize real defect patterns.

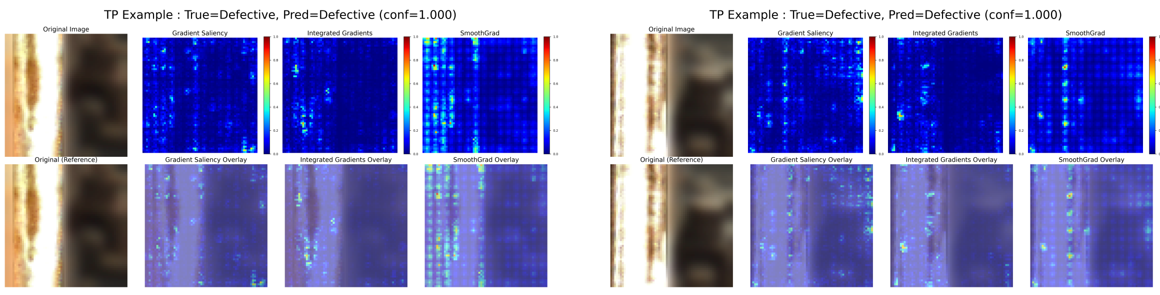


Figure 6.10: XAI analysis for two high-confidence True Positives. Both examples show strong agreement across all three attribution methods, with the model correctly focusing on genuine rail surface defects.

For True Negatives, the model correctly exhibits minimal activation on normal rail surfaces. Figure 6.11 shows a representative TN case where the yield maps exhibit diffuse, low-intensity patterns, indicating correct rejection of uniform, non-defective texture regions.

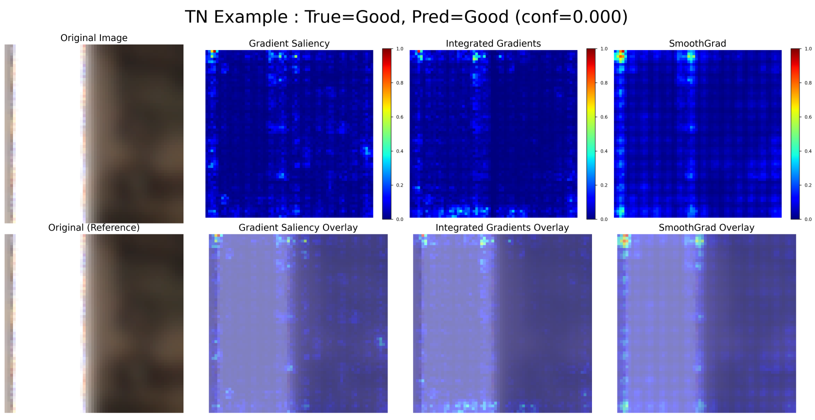

TN Example : True=Good, Pred=Good (conf=0.000)
Original Image
Gradient Saliency
Integrated Gradients
SmoothGrad
Original (Reference)
Gradient Saliency Overlay
Integrated Gradients Overlay
SmoothGrad Overlay

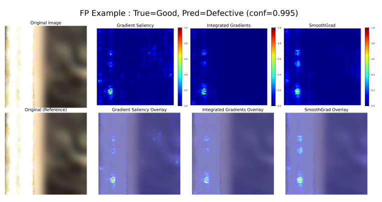
Figure 6.11: True Negative example showing correct classification of normal rail surface. The attribution maps show minimal and dispersed activation, confirming the model correctly identifies the absence of defects.

6.6.3 False Positive Analysis and Diagnosis

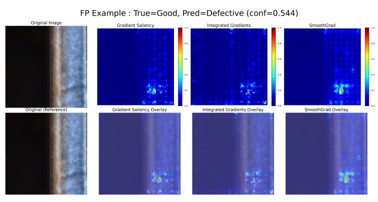
The analysis of False Positives reveals three distinct failure modes, as shown in Figure 6.12. The first case (Figure 6.12a) demonstrates how the contrast enhancement from the selected preprocessing pipeline amplifies shadowed rail regions, creating artificial texture patterns that the model misinterprets as defects with high confidence.

In the second case (Figure 6.12b), the model incorrectly attributes significance to rail segment connections. The absence of similar examples in the training data likely causes the model to classify these structural features as surface irregularities indicative of defects.

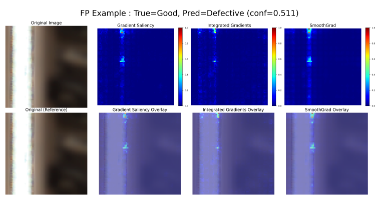
The third failure mode (Figure 6.12c) shows the model focusing on track edges, suggesting vulnerability to sharp transitional boundaries where the model may misinterpret normal geometric features as anomalous.



(a) CLAHE-enhanced Shadow (conf=0.995)



(b) Rail Joint Misclassification (conf=0.544)



(c) Edge Artifact Focus (conf=0.511)

Figure 6.12: Three representative False Positive cases showing different failure modes. (a) CLAHE amplification of shadows creates false defect indications, (b) Structural rail joints are misinterpreted as surface defects, (c) Normal track edges attract spurious attention as potential anomalies.

6.6.4 Confidence and Error Analysis

Confidence distribution analysis is additionally conducted. Figure 6.13 reveals systematic patterns across prediction categories. **True Positives (TP)** cluster tightly near $P(\text{Defective}) \approx 1$, indicating that correctly detected defects are assigned consistently high confidence. **True Negatives (TN)** exhibit the opposite behaviour, with probabilities concentrated near $P(\text{Defective}) = 0$, showing a clear separation between defective and non-defective samples. In contrast, **False Positives (FP)** appear predominantly around the decision threshold, typically in the range 0.5–0.7, suggesting that these errors arise from samples the model finds implicitly weird. Notably, the test set contains **no False Negatives (FN)**, indicating that the classifier tends to over-detect rather than miss defect indications. Overall, these distributions confirm that the model is highly confident in its correct predictions, while the few mistakes occur primarily in borderline-confidence regions.

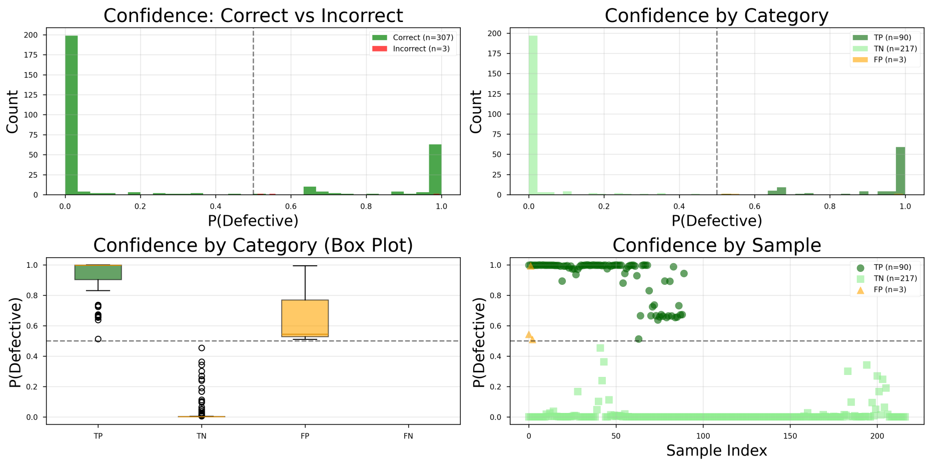


Figure 6.13: Confidence distribution analysis across prediction categories. True Positives (TP) exhibit consistently high confidence with $P(\text{Defective})$ values tightly clustered near 1, while True Negatives (TN) are concentrated near $P(\text{Defective}) = 0$. False Positives (FP) fall predominantly around the decision threshold, typically within the range 0.5–0.7, indicating that these errors arise from borderline or weird samples. No False Negatives (FN) are observed in the evaluation set, suggesting a tendency of the model to over-detect rather than miss defects.

This appropriate uncertainty calibration suggests the model has some awareness of its limitations, with incorrect predictions generally associated with lower confidence scores.

6.6.5 Analysis of sequence-aware Post-Processing Effectiveness

To quantitatively demonstrate the impact of sequence-aware post-processing, we analyze the specific sequences containing misclassified patches. Each sequence is presented individually to enable examination of the spatial patterns and post-processing effects.

The analysis reveals distinct patterns of post-processing effectiveness. In Sequences 7.2 and 13.1 (Figures 6.14 and 6.15), both post-processing methods (multiscale and confidence-weighted + isolated removal) successfully correct the single false positive detections. These isolated errors, occurring at frames with low confidence scores (0.51 and 0.54 respectively), are effectively suppressed because they appear as spatial outliers within otherwise consistently normal sequences.

However, neither method manages to correct the more complex error in Sequence 1.2 (Figure 6.16). Interestingly, multiscale even introduces two new misclassifications in frames of that sequence: one false positive and one false negative. This is likely because the sequence contains genuine defect clusters mixed with normal frames, causing the multi-scale smoothing to occasionally over-generalize spatial patterns.

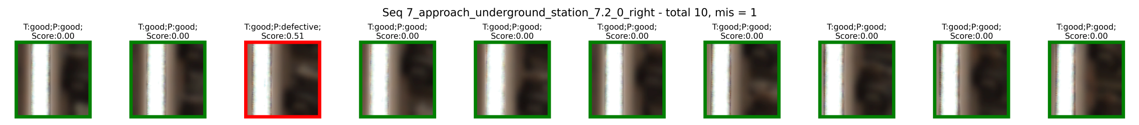

Seq 7_approach_underground_station_7.2_0_right - total 10, mis = 1
T:good;P:good; Score:0.00
T:good;P:good; Score:0.00
T:good;P:defective; Score:0.51
T:good;P:good; Score:0.00
T:good;P:good; Score:0.00
T:good;P:good; Score:0.00
T:good;P:good; Score:0.00
T:good;P:good; Score:0.00
T:good;P:good; Score:0.00
T:good;P:good; Score:0.00

Figure 6.14: Sequence 7.2 (10 frames): An isolated false positive at frame 3 (confidence: 0.51) is successfully corrected by both post-processing methods. The single erroneous detection appears as a spatial outlier within an otherwise consistently normal sequence, making it an ideal candidate for spatial smoothing.

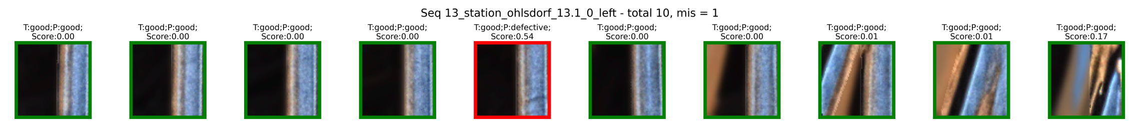

Seq 13_station_ohlsdorf_13.1_0_left - total 10, mis = 1
T:good;P:good; Score:0.00
T:good;P:good; Score:0.00
T:good;P:good; Score:0.00
T:good;P:good; Score:0.00
T:good;P:defective; Score:0.54
T:good;P:good; Score:0.00
T:good;P:good; Score:0.00
T:good;P:good; Score:0.01
T:good;P:good; Score:0.01
T:good;P:good; Score:0.17

Figure 6.15: Sequence 13.1 (10 frames): Similar to Sequence 7.2, this sequence contains a single false positive at frame 5 with low confidence (0.54). Both post-processing methods successfully suppress this isolated error, demonstrating the effectiveness of spatial consistency checks for sporadic misclassifications.

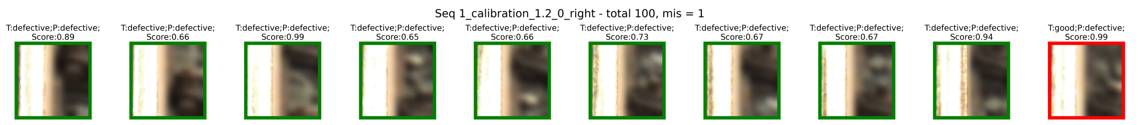

Seq 1_calibration_1.2_0_right - total 100, mis = 1
T:defective;P:defective; Score:0.89
T:defective;P:defective; Score:0.66
T:defective;P:defective; Score:0.99
T:defective;P:defective; Score:0.65
T:defective;P:defective; Score:0.66
T:defective;P:defective; Score:0.73
T:defective;P:defective; Score:0.67
T:defective;P:defective; Score:0.67
T:defective;P:defective; Score:0.94
T:good;P:defective; Score:0.99

Figure 6.16: Sequence 1.2 (100 frames): This longer sequence presents a more challenging scenario with genuine defect clusters (high-confidence correct detections) mixed with normal frames. Neither method corrects those.

Chapter 7

Discussion

This chapter interprets the experimental findings in the context of operational rail infrastructure monitoring, examines their implications and limitations, and identifies directions for future research.

7.1 Interpretation of Key Findings

The proposed final system achieves exceptional performance on held-out test data (AUPRC 0.993, F1 0.995), demonstrating that vision transformers with optimized preprocessing can successfully detect track defects from ATO imagery. However, this strong result should be interpreted in context: the test set (Set E), even if strictly held out, includes illumination and track conditions similar to those seen in training. Performance would likely have been lower if tested on entirely different conditions, as occurred in problematic cross-validation folds. Besides dataset considerations, several design and method insights emerge.

DeiT-Small's consistent outperformance of ResNet-18 confirms that global context benefits rail defect detection. Mechanistically, while convolutional networks like ResNet-18 rely on stacked layers to gradually build receptive fields, transformers' self-attention mechanisms directly model relationships across the entire 64$\times$64 patch in a single layer. For rail surface inspection, where defects often appear as distributed texture anomalies rather than localized features, this global perspective offers an advantage.

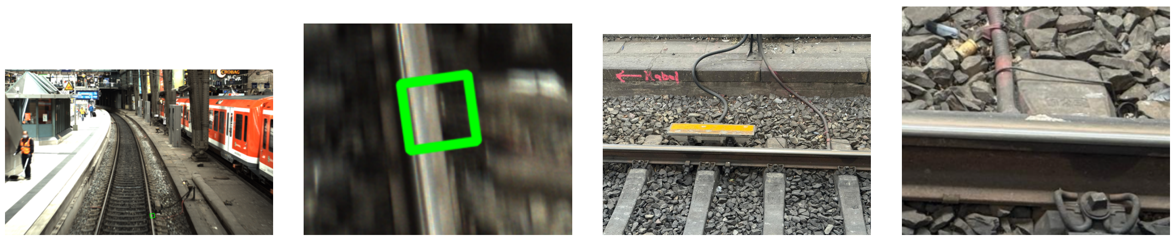
PatchCore's underperformance (AUPRC 0.608) reflects limitations of memory-based anomaly detection under high operational diversity. With ~ 780 normal training patches per fold, the memory bank lacks coverage of normal appearance variations across stations and lighting conditions. This exposes a practical constraint: unsupervised methods require extensive representation of normal variability [54], which may demand larger datasets than supervised alternatives when operational conditions vary largely.

The success of deterministic post-processing methods, primarily as precision-enhancing filters, as Confidence-Weighted + Isolated Removal, comes from their strong, built-in assumptions about the data. Genuine defects are physical phenomena that mostly persist over multiple frames, whereas many false positives are isolated. In contrast, the bidirectional LSTM fails to provide a consistent improvement. This is likely attributable to the limited spatial data (short sequences, few examples of defect onset) and the already high baseline performance, leaving little room for a data-hungry model to learn meaningful patterns. This finding suggests that for sequential refinement with limited data, specific heuristics can be more effective than computationally heavier learned models.

The 7.9 percentage point gain from `BLUR_BG+CLAHE` preprocessing shows that enhancement can actively fix real-world data problems. CLAHE standardizes intensity distributions across varying illumination while amplifying subtle surface irregularities, though XAI analysis reveals it equally enhances shadows, creating false positives that post-processing must filter. The superiority of Gaussian blur over black background replacement suggests that hard, artificial edges may act as distracting features, whereas Gaussian blur provides a form of structured noise reduction that preserves more natural image gradients.

Data Scarcity and the Realities of Model Evaluation The variation in cross-validation performance, especially the significant drop in folds with conditions absent from training data, directly results from our stratified splitting algorithm. This algorithm keeps the sequences intact to prevent spatiotemporal leakage while balancing two main factors: defect distribution and sequence length across folds. When stations in a fold have different characteristics from the ones in the training set, performance decreases. Therefore, our partitioning strategy allows for a fair evaluation of uneven sequential data, a contribution that can be used in any research area with multiple balancing constraints. However, it reveals a vulnerability to unseen operational conditions, not just different stations, but also variations in sunlight, weather and infrastructure situations like switches or tunnels. As a result, cross-validation produces a range of performances instead of a single reliable score. This variability emphasizes that successful deployment requires training data that includes a much wider variety of stations and conditions than captured in our data.

Field Validation of Annotation Quality The practical relevance of the annotation methodology received encouraging, though limited, support from a field validation case. A specific patch from Hamburg main station (file name: `11_main_station_11.1 _282_1631531580.300000026_patches_patch_2244_2199_81.993_0_right`), which had been heuristically labeled as 'defective' based on visible surface markings (Figure 7.1a-b), became the subject of an on-site inspection.



(a) Original 2023 frame-patch marked. (b) 'Defective' extracted patch. (c) 2025 photo of same location. (d) Zoomed view confirming anomaly.

Figure 7.1: Field validation of a heuristic 'defective' label. The surface marking identified in the 2023 dataset (a-b) is confirmed as a persistent physical anomaly through on-site inspection in 2025 (c-d).

An engineer visited the exact location and confirmed the presence of a physical anomaly that corresponded to the markings visible in the 2023 image data (Figure 7.1c-d), reporting it as an indeed real-life anomaly. This finding offers preliminary, qualitative support for the practical relevance of the annotated dataset, suggesting that the labels capture features of interest for infrastructure monitoring, even if their precise classification as critical defects may require further expert analysis.

7.2 Limitations and Future Directions

Dataset Scope and Representativeness The primary limitation of this work is the scale and composition of its dataset. The 1,610 patches, derived from 71 sequences at 21 Hamburg stations in September 2021, represent a single geographic, temporal and operational context. This narrow scope is directly exposed during cross-validation, in Fold 2 of Seed 0 (Section 6.2), where sequences from unseen stations exhibited misclassifications rates of 70–100%. This distribution shift demonstrates that the model's performance on held-out data from known stations may not generalize to the broader rail network.

Furthermore, the dataset's operational conditions do not reflect real-world deployment scenarios. Data was collected from a slow-moving maintenance vehicle, yielding sharp images with minimal motion blur. Deployment on trains at service speeds (80–200 km/h) would introduce more severe blur, potentially degrading performance on subtle defects and requiring strategies like deblurring algorithms or faster shutter speeds. Similarly, collection is limited to daylight hours. This left performance in nighttime and tunnel environments, which are common in continuous rail operations, untested.

Beyond these environmental constraints, the model struggled with underrepresented infrastructure types. Railroad crossings (Figure 6.8) triggered high-confidence false positives due to their characteristic patterns. This issue likely extends to other specialized elements like switches and bridge approaches, where distinctive visual features may be misinterpreted as defects.

The fact that temporal coverage is constrained to a single week presents another limitation, since seasonal variations (e.g., frost, vegetation, accelerated weathering) could impact model performance. Predictive maintenance requires tracking degradation over months or years, making a single snapshot inappropriate for capturing this evolutionary process. Additionally, the analyzed sequences are restricted to short clips of 1–10 seconds, limiting the spatial context available for post-processing methods. Their effectiveness on longer, real-world inspection routes remains an open question.

The manual, heuristic labeling process introduces a final dataset constraint. The binary 'normal'/'defective' scheme oversimplifies gradual rail degradation and involves subjective judgment. However, this limitation should be viewed in light of the proposed operational context. The system is not intended for high-stakes, immediate defect detection but as an early-warning mechanism that identifies initial anomalies which might later develop into defects. Therefore, this represents a use case where binary classification provides a practical first step.

Methodological and Architectural Constraints The methodological pipeline itself introduced specific limitations. The rotation of patches to a canonical vertical orientation simplified the learning problem but created an unanswered dependency on this specific view and its robustness to rotational variance remains unquantified.

The analysis is confined to RGB data, leaving the vehicle's multi-sensor suite unutilized. This suite includes infrared (for subsurface defects), LiDAR (for geometry) and radar (for occlusion penetration). Sensor fusion, an established practice in infrastructure inspection, remains an open opportunity for this application to improve robustness and defect coverage.

The staged experimental design, while enabling computational tractability, represents a final methodological constraint. By fixing preprocessing and model choices at each stage based on greedy optimization, the work sacrificed exhaustive search completeness. A truly comprehensive search evaluating all feasible combinations would explore 6 preprocessing $\times$ 3 architectures $\times$ 12 post-processing methods $\times$ 30+ hyperparameter configurations $\approx$ 65,000 training runs, discovering potential interactions that staged optimization may have missed. While computationally prohibitive for this thesis, future research with greater computational resources should explore broader hyperparameter spaces to identify whether the presented results represent local rather than global optima. Also, the architectural comparison, though informative, is not exhaustive, leaving the performance of video-aware architectures (e.g. TimeSformer [5]) an open question.

Future Research Directions Addressing the aforementioned limitations requires a comprehensive multi-modal and temporal data strategy. Longitudinal data collection over years across diverse geographic regions would establish robust seasonal and material performance baselines, enabling analysis of rail grinding intervals and maintenance scheduling optimization. This expansion should include varied operational conditions, such as high-speed service, nighttime operations and diverse infrastructure types, with labels guided by railway specialists to ensure accuracy. Multi-modal sensor fusion, integrating RGB with LiDAR for lighting-invariant geometry and infrared for thermal signatures, would provide complementary perspectives on defect characteristics. Extended temporal sequences would allow advanced sequence models (e.g., recurrent or attention-based networks) to leverage long-range dependencies for better defect tracking and clustering, both within single inspections and across multiple years of service.

To function effectively as an early-warning mechanism, the system must be deployed across multiple vehicles. By collecting repeated measurements over time, statistical trends can distinguish genuine degradation from sporadic false detections, as consistency across multiple observations provides reliable evidence of deterioration.

Beyond rail head degradation, the methodology could extend to monitoring fasteners, sleepers and catenary infrastructure, creating a comprehensive asset management system. Ultimately, integrating these capabilities into the ATO framework would transform it from a purely operational safety system into a comprehensive monitoring platform, enabling cost-effective, proactive maintenance through continuous degradation tracking across the entire rail network.

Chapter 8

Conclusion

This thesis investigates whether forward-facing cameras installed for ATO can reliably detect rail track defects under uncontrolled operational conditions. The question emerges from a practical opportunity: while specialist inspection vehicles provide detailed assessments but visit track segments infrequently, ATO cameras will traverse networks daily as automation expands across European railways.

To address this question, we contribute a complete methodology for adapting operational camera data to defect detection: a patch extraction and track isolation procedure applicable to vehicle-mounted imagery, a systematic comparison of preprocessing strategies and model paradigms under severe class imbalance, and sequence-aware post-processing methods that exploit spatial continuity in railway data. The final system achieves strong performance on held-out test data (AUPRC 0.993, F1 0.995), with explainability analysis confirming that decisions are grounded in physically plausible surface features. This demonstrates that leveraging ATO cameras for track monitoring is technically feasible.

However, operational readiness requires careful qualification. The dataset's narrow scope cannot validate generalization across the operational diversity of real rail networks. Critical untested conditions include varied geographic regions, seasonal variations, high-speed service, nighttime operations and specialized infrastructure types. Notably, cross-validation revealed performance collapse when encountering unseen conditions, demonstrating vulnerability to distribution shift.

Despite these limitations, the strategic value proposition is compelling. More frequent defect detection enables earlier intervention, potentially preventing failures that pose safety risks and cause service disruptions. Economically, since cameras will be installed for operational safety regardless, infrastructure monitoring through software updates requires minimal additional investment. Even modest detection performance becomes attractive if false positive rates remain low enough to avoid unnecessary inspections, a requirement this work addresses through high precision on test data.

This thesis establishes proof-of-concept demonstrating technical feasibility but does not constitute a deployment-ready solution. Future validation requires multi-site data collection over extended periods with expert annotation protocols. Infrastructure monitoring capabilities can thus develop in parallel with ATO, and this work provides a foundation for that development trajectory.

Bibliography

[1] Julius Adebayo, Justin Gilmer, Michael Muelly, Ian Goodfellow, Moritz Hardt, and Been Kim. Sanity checks for saliency maps. In *Advances in Neural Information Processing Systems (NeurIPS) 31*, pages 9505–9516, 2018. URL https://arxiv.org/abs/1810.03292. arXiv preprint arXiv:1810.03292.

[2] Association for Standardization of Automation and Measuring Systems. ASAM OpenLABEL. https://www.asam.net/standards/detail/openlabel/, 2021.

[3] Michel L. Balinski and H. Peyton Young. *Fair Representation: Meeting the Ideal of One Man, One Vote*. Yale University Press, New Haven, CT, 1982. ISBN 9780300028240.

[4] Paul Bergmann, Michael Fauser, David Sattlegger, and Carsten Steger. Mvtec ad — a comprehensive real-world dataset for unsupervised anomaly detection. In *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 9584–9592, 2019. doi: 10.1109/CVPR.2019.00982.

[5] Gedas Bertasius, Heng Wang, and Lorenzo Torresani. Is space-time attention all you need for video understanding?, 2021. URL https://arxiv.org/abs/2102.05095.

[6] Mario Binder, Vitaliy Mezhuyev, and Martin Tschandl. Predictive maintenance for railway domain: A systematic literature review. *IEEE Engineering Management Review*, 51(2):120–140, 2023. doi: 10.1109/EMR.2023.3262282. URL https://www.researchgate.net/publication/369572389_Predictive_Maintenance_for_Railway_Domain_a_Systematic_Literature_Review/citations.

[7] Christopher M. Bishop. *Pattern Recognition and Machine Learning*. Springer, New York, 2006. ISBN 978-0-387-31073-2.

[8] Britannica Editors. Gauge. Encyclopedia Britannica, April 2018. URL https://www.britannica.com/technology/gauge-railroad-track.

[9] Richard L. Burden and J. Douglas Faires. *Numerical Analysis*. Brooks Cole, 9th edition, 2011.

[10] Rémi Cadène, Thomas Robert, Nicolas Thome, and Matthieu Cord. M2cai workflow challenge: Convolutional neural networks with time smoothing and hidden markov model for video frames classification, 2016. URL https://arxiv.org/abs/1610.05541.

[11] Jesse Davis and Mark Goadrich. The relationship between precision-recall and roc curves. In *Proceedings of the 23rd International Conference on Machine Learning (ICML)*, pages 233–240, Pittsburgh, Pennsylvania, USA, 2006. ACM. doi: 10.1145/1143844.1143874.

[12] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pages 248–255, 2009. doi: 10.1109/CVPR.2009.5206848.

[13] Thomas G. Dietterich. Ensemble methods in machine learning. In *Multiple Classifier Systems*, pages 1–15, Berlin, Heidelberg, 2000. Springer Berlin Heidelberg. ISBN 978-3-540-45014-6.

[14] Tien Phat Dinh, Quang Hoai Le, Thao Nguyen Thach, Byeol Kim, and Yonghan Ahn. Railway track structural health monitoring: Identifying emerging trends and research agendas using bibliometric and topic modeling. *Applied Sciences*, 15(23), 2025. ISSN 2076-3417. doi: 10.3390/app152312462. URL https://www.mdpi.com/2076-3417/15/23/12462.

[15] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale. In *International Conference on Learning Representations (ICLR)*, 2021.

[16] N. Elkhoury et al. Degradation prediction of rail tracks: A review of the existing literature. *The Open Transportation Journal*, 12:88–101, 2018. doi: 10.2174/1874447801812010088.

[17] Europe's Rail Joint Undertaking. Flagship project 2: R2DATO - digital & automated up to autonomous train operations, 2024. URL https://rail-research.europa.eu/rail-projects/fp2-r2dato/. Accessed: 2024-12-02.

[18] A. Falamarzi, M. Moridpour, et al. A review on existing sensors and devices for inspecting railway infrastructure. *Jurnal Kejuruteraan / Journal of Engineering*, 31(2): 1–23, 2019. Available online: https://journalarticle.ukm.my/14293/1/01.pdf.

[19] X. Fang, Q. Luo, J. Ai, Y. Sun, and L. Li. Research progress of automated visual surface defect detection for industrial metal planar materials. *Sensors*, 20(18):5136, 2020. doi: 10.3390/s20185136.

[20] Development Federal Railroad Administration (FRA) – Office of Research and Technology. Reducing hazards for track inspection: A review of track inspection methods. Technical report, U.S. Department of Transportation / FRA, 2021. URL https://railroads.dot.gov/sites/fra.dot.gov/files/2021-05/Reducing%20Hazards%20for%20Track%20Inspection.pdf.

[21] Stanislav Fort, Huiyi Hu, and Balaji Lakshminarayanan. Deep ensembles: A loss landscape perspective, 2020. URL https://arxiv.org/abs/1912.02757.

[22] Gourav and Ruchi Patira. Rail tracks defect detection using deep learning: A review. *International Journal of Creative Research Thoughts (IJCRT)*, 13(4):a444, 2025. ISSN 2320-2882. URL https://www.ijcrt.org/papers/IJCRT2504055.pdf.

[23] Feng Guo, Jian Liu, Yu Qian, and Quanyi Xie. Rail surface defect detection using a transformer-based network. *Journal of Industrial Information Integration*, 38:100584, 2024. ISSN 2452-414X. doi: https://doi.org/10.1016/j.jii.2024.100584. URL https://www.sciencedirect.com/science/article/pii/S2452414X24000281.

[24] Habib Hadj-Mabrouk. Contributions and limitations of ai and machine learning in railway operations, maintenance and safety: A literature review. *Proceedings of the Institution of Mechanical Engineers, Part O: Journal of Risk and Reliability*, pages 1–36, 2025. doi: 10.1177/1748006X251364324. URL https://www.researchgate.net/publication/395467869_Contributions_and_limitations_of_AI_and_machine_learning_in_railway_operations_maintenance_and_safety_A_literature_review.

[25] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778, 2016.

[26] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural Computation*, 9(8):1735–1780, 1997. doi: 10.1162/neco.1997.9.8.1735.

[27] Zhiyi Huang and Xiao Li. A critical review of operations research on the operation and maintenance of railway systems. *Journal of Railway Science and Technology*, 2025. ISSN 3050-8142. doi: https://doi.org/10.1016/j.jrst.2025.08.001. URL https://www.sciencedirect.com/science/article/pii/S3050814225000081.

[28] Kanwal Jahan, Jeethesh Pai Umesh, and Michael Roth. Anomaly detection on the rail lines using semantic segmentation and self-supervised learning. In *2021 IEEE Symposium Series on Computational Intelligence (SSCI)*, pages 1–7, 2021. ISBN 978-1-7281-9048-8. doi: 10.1109/SSCI50451.2021.9659920.

[29] Albert Ji, Wai Lok Woo, Eugene Wai Leong Wong, and Yang Thee Quek. Rail track condition monitoring: a review on deep learning approaches. *Intelligence & Robotics*, 1(2), 2021. ISSN 2770-3541. URL https://www.oaepublish.com/articles/ir.2021.14.

[30] Simon Kornblith, Jonathon Shlens, and Quoc V. Le. Do better imagenet models transfer better? In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2661–2671, 2019.

[31] Ankit 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*, 18(8):207–226, 2024. doi: 10.1016/j.ijtst.2024.06.006.

[32] Ankit 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*, 18:207–226, 2025. ISSN 2046-0430. doi: https://doi.org/10.1016/j.ijtst.2024.06.006. URL https://www.sciencedirect.com/science/article/pii/S2046043024000716.

[33] Balaji Lakshminarayanan, Alexander Pritzel, and Charles Blundell. Simple and scalable predictive uncertainty estimation using deep ensembles, 2017. URL https://arxiv.org/abs/1612.01474.

[34] Hao Li, Changjiang Liu, and Anup Basu. Semantic segmentation based on depth background blur. *Applied Sciences*, 12(3):1051, 2022.

[35] Hongfei Li, Dhaivat Parikh, Qing He, Buyue Qian, Zhiguo Li, Dongping Fang, and Arun Hampapur. Improving rail network velocity: A machine learning approach to predictive maintenance. *Transportation Research Part C: Emerging Technologies*, 45:17–26, 2014. ISSN 0968-090X. doi: https://doi.org/10.1016/j.trc.2014.04.013. URL https://www.sciencedirect.com/science/article/pii/S0968090X14001107. Advances in Computing and Communications and their Impact on Transportation Science and Technologies.

[36] S. P. Lloyd. Least squares quantization in pcm. Technical Report RR-5497, Bell Laboratories, 1957. Published later in IEEE Transactions on Information Theory, 1982.

[37] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization, 2019. URL https://arxiv.org/abs/1711.05101.

[38] J. B. MacQueen. Some methods for classification and analysis of multivariate observations. In L. M. Le Cam and J. Neyman, editors, *Proceedings of the fifth Berkeley symposium on mathematical statistics and probability*, volume 1, pages 281–297, Berkeley, CA, 1967. University of California Press.

[39] Y. Min et al. Self-supervised railway surface defect detection with background variation challenges. *Energies*, 15(10):3592, 2022. doi: 10.3390/en15103592.

[40] Akshansh Mishra. Contrast limited adaptive histogram equalization (clahe) approach for enhancement of the microstructures of friction stir welded joints, 2021. URL https://arxiv.org/abs/2109.00886.

[41] Pankaj Mishra, Riccardo Verk, Daniele Fornasier, Claudio Piciarelli, and Gian Luca Foresti. Vt-adl: A vision transformer network for image anomaly detection and localization. In *2021 IEEE 30th International Symposium on Industrial Electronics (ISIE)*, page 01–06. IEEE, June 2021. doi: 10.1109/isie45552.2021.9576231. URL http://dx.doi.org/10.1109/ISIE45552.2021.9576231.

[42] V. Nezval et al. Identification of factors contributing to broken and buckled rails. *European Transport – Trasporti Europei*, 83:1–17, 2025. doi: 10.1186/s12544-025-00725-w.

[43] International Union of Railways (UIC) and McKinsey & Company. The journey toward ai-enabled railway companies. Technical report, International Union of Railways (UIC), February 2024. URL https://uic.org/com/IMG/pdf/uic_layout_web_05032024.pdf?

[44] Bryan Olivier, Feng Guo, Yu Qian, and David P. Connolly. A review of computer vision for railways. *IEEE Transactions on Intelligent Transportation Systems*, 26(7): 11034–11065, 2025. doi: 10.1109/TITS.2025.3552011.

[45] Haixia Pan, Yanan Li, Hongqiang Wang, and Xiaomeng Tian. Railway obstacle intrusion detection based on convolution neural network multitask learning. *Electronics*, 11(17): 2697, 2022. doi: 10.3390/electronics11172697.

[46] Aniwat Phaphuangwittayakul, Napat Harnpornchai, Fangli Ying, and Jinming Zhang. Railtrack-davit: A vision transformer-based approach for automated railway track defect detection. *Journal of Imaging*, 10(8), 2024. ISSN 2313-433X. doi: 10.3390/jimaging10080192. URL https://www.mdpi.com/2313-433X/10/8/192.

[47] M. Pizer, Stephen P. Amburn, E. D. Austin, J. R. Cromartie, A. Geselowitz, Trey Greer, M. ter Haar Romenij, B. B. Zimmerman, J. and J. Zuiderveld, K.˙Adaptive histogram equalization and its variations. *Computer Vision, Graphics, and Image Processing*, 39(3):355–368, 1987. doi: 10.1016/S0734-189X(87)80186-X.

[48] ProRail. Mijlpaal: Ai vindt scheuren in dwarsliggers. *ProRail Nieuws*, Feb 2025. URL https://www.prorail.nl/nieuws/mijlpaal-ai-detecteert-scheuren-in-dwarsliggers. Online article. Accessed [insert access date].

[49] Ziyuan Pu, Shuo Wang, Chenglong Liu, Zhiyong Cui, and Yinhai Wang. Road surface friction prediction using long short-term memory neural network based on historical data. *arXiv preprint arXiv:1911.02372*, 2019. doi: 10.48550/arXiv.1911.02372. URL https://arxiv.org/abs/1911.02372.

[50] Douglas A Reynolds. Gaussian mixture models. *Encyclopedia of biometrics*, 741: 659–663, 2009.

[51] Ali M. Reza. Realization of the contrast limited adaptive histogram equalization (clahe) for real-time image enhancement. *Journal of VLSI Signal Processing – Systems for Signal, Image, and Video Technology*, 38(1):35–44, 2004. doi: 10.1023/B:VLSI.0000028532.53893.82.

[52] Danijela Ristić-Durrant, Marten Franke, and Kai Michels. A review of vision-based on-board obstacle detection and distance estimation in railways. *Sensors*, 21(10):3452, 2021. doi: 10.3390/s21103452.

[53] Mauricio Rodríguez-Hernández, Adolfo Crespo-Márquez, Antonio Sánchez-Herguedas, and Vicente González-Prida. Digitalization as an enabler in railway maintenance: A review from "the international union of railways asset management framework" perspective. *Infrastructures*, 10(4), 2025. ISSN 2412-3811. doi: 10.3390/infrastructures10040096. URL https://www.mdpi.com/2412-3811/10/4/96.

[54] Karsten Roth, Latha Pemula, Joaquin Zepeda, Bernhard Schölkopf, Thomas Brox, and Peter V. Gehler. Towards total recall in industrial anomaly detection. *CoRR*, abs/2106.08265, 2021. URL https://arxiv.org/abs/2106.08265.

[55] Takaya Saito and Marc Rehmsmeier. The precision-recall plot is more informative than the roc plot when evaluating binary classifiers on imbalanced datasets. *PLOS ONE*, 10(3):e0118432, 2015. doi: 10.1371/journal.pone.0118432.

[56] Shift2Rail Joint Undertaking. Innovation in the spotlight: Towards unattended mainline train operations (ATO GoA 4), September 2019. URL https://rail-research.europa.eu/highlight/innovation-in-the-spotlight-towards-unattended-mainline-train-operations-ato-goa4/. Accessed: 2024-12-02.

[57] Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman. Deep inside convolutional networks: Visualising image classification models and saliency maps, 2014. URL https://arxiv.org/abs/1312.6034.

[58] Daniel Smilkov, Nikhil Thorat, Been Kim, Fernanda B. Viégas, and Martin Wattenberg. Smoothgrad: removing noise by adding noise. *CoRR*, abs/1706.03825, 2017. URL http://arxiv.org/abs/1706.03825.

[59] Maria Di Summa, Maria Elena Griseta, Nicola Mosca, Cosimo Patruno, Massimiliano Nitti, Vito Renò, and Ettore Stella. A review on deep learning techniques for railway infrastructure monitoring. *IEEE Access*, 11:114639–114659, 2023. doi: 10.1109/ACCESS.2023.3309814. URL https://iris.cnr.it/retrieve/b04c8a70-11fa-47de-970f-39047c00d3/A_Review_on_Deep_Learning_Techniques_for_Railway_Infrastructure_Monitoring.pdf.

[60] Mukund Sundararajan, Ankur Taly, and Qiqi Yan. Axiomatic attribution for deep networks. *CoRR*, abs/1703.01365, 2017. URL http://arxiv.org/abs/1703.01365.

[61] Rustam Tagiew, Pavel Klasek, Roman Tilly, Martin Köppel, Patrick Denzler, Philipp Neumaier, Tobias Klockau, Martin Boekhoff, and Karsten Schwalbe. Osdar23: Open sensor data for rail 2023. In *2023 8th International Conference on Robotics and Automation Engineering (ICRAE)*, page 270–276. IEEE, November 2023. doi: 10.1109/icrae59816.2023.10458449. URL http://dx.doi.org/10.1109/ICRAE59816.2023.10458449.

[62] Hugo Touvron, Matthieu Cord, Matthijs Douze, Francisco Massa, Alexandre Sablayrolles, and Hervé Jégou. Training data-efficient image transformers & distillation through attention, 2021. URL https://arxiv.org/abs/2012.12877.

[63] Volodymyr Tverdomed, Zhuk Dmytro, Natalia Kokriatska, and Vaidas Lukoševičius. The detection of railheads: An innovative direct image processing method. *Sustainability*, 16(12), 2024. ISSN 2071-1050. doi: 10.3390/su16125109. URL https://www.mdpi.com/2071-1050/16/12/5109.

[64] Amin Ullah, Jamil Ahmad, Khan Muhammad, Muhammad Sajjad, and Sung Wook Baik. Action recognition in video sequences using deep bi-directional lstm with cnn features. *IEEE Access*, 6:1155–1166, 2018. doi: 10.1109/ACCESS.2017.2778011.

[65] Shuai Wang, Xiaojun Xia, Lanqing Ye, and Binbin Yang. Automatic detection and classification of steel surface defect using deep convolutional neural networks. *Metals*, 11(3), 2021. ISSN 2075-4701. doi: 10.3390/met11030388. URL https://www.mdpi.com/2075-4701/11/3/388.

[66] Y. Wang et al. Review on rail damage detection technologies for high-speed train track operation and maintenance. *Applied Sciences*, 15(14):7725, 2025. doi: 10.3390/app15147725.

[67] Yuanhao Zhang, Zujun Yu, Liqiang Zhu, Baoqing Guo, and Yao Wang. Rail-patchcore: unsupervised learning-based detection of visual anomalies in the railway-turnout environment. *Applied Intelligence*, 55(6):408, 2025. doi: 10.1007/s10489-025-06294-8. URL https://doi.org/10.1007/s10489-025-06294-8.

[68] Zihao Zhang, Shaozuo Yu, Siwei Yang, Yu Zhou, and Bingchen Zhao. Rail-5k: a real-world dataset for rail surface defects detection, 2021. URL https://arxiv.org/abs/2106.14366.

[69] Yue Zhao and Philipp Krähenbühl. Real-time online video detection with temporal smoothing transformers, 2022. URL https://arxiv.org/abs/2209.09236.

[70] Zoran Zivkovic. Improved adaptive gaussian mixture model for background subtraction. *Proceedings of the 17th International Conference on Pattern Recognition*, 2:28–31, 2004.

[71] Arij Zouaoui, Ankur Mahtani, Mohamed Amine Hadded, Hazem Wannous, et al. Railset: A unique dataset for railway anomaly detection. In *2022 IEEE 5th International Conference on Image Processing Applications and Systems (IPAS)*. IEEE, December 2022. doi: 10.1109/IPAS55744.2022.10052883.

[72] Michael Zuther, Markus Eisenbach, Sebastian Houben, and Torsten Bertram. Railsem19: A dataset for semantic rail scene understanding. In *Proceedings of the IEEE Intelligent Vehicles Symposium*, 2020.

[73] Šárka Hošková-Mayerová et al. Rail infrastructure as a part of critical infrastructure. In *Proceedings of the International Conference on Transportation & Traffic Engineering (or similar)*, 2017. URL https://www.researchgate.net/publication/317395331_Rail_infrastructure_as_a_part_of_critical_infrastructure. Online; “One of the critical infrastructure components in most economies across the world is the rail network.”.

Appendix A

Annotation Inconsistencies and Manual Corrections

During the inspection of the unified annotation dataframe of the OSDaR23 dataset, two types of inconsistencies are identified: those related to the *rail-side assignment* and those concerning the *placement of polyline coordinates* with respect to the rail edges. All affected cases are visually verified and are either corrected manually or handled during mask construction to ensure consistent rail geometry representation and reliable preprocessing. Table A.1 provides an overview of the identified issues, which are detailed in the following sections.

Table A.1: Summary of annotation inconsistencies identified and corrected in the OSDaR23 dataset. A total of 15 frames across 4 stations are corrected or adjusted to ensure consistent rail geometry representation.

Inconsistency type	**Affected stations**	**Affected frames**
Incorrect rail-side labeling	2	9
Misplaced polyline edge definition	2	6

Issue 1: Incorrect Rail-Side Labeling

In eight consecutive frames of subsequence `7_approach_underground_station_7.2`, all rail tracks with `trackID = 0` are annotated as `leftRail`. However, visual inspection shows that half of these rails geometrically correspond to the right rail. The affected frames are 675–682 (timestamps 1631705207.600000008–1631705208.300000010).

A similar issue occurs in frame 74 (timestamp 1631176103.500000016) from subsequence `13_station_ohlsdorf_13.1`, where the tracks with `trackID = 0` are both labeled as `leftRail` even though one of them corresponds to `rightRail`.

All affected entries are manually reassigned to their correct side based on their geometric position relative to the track centerline.

Issue 2: Misplaced Polyline Edge Definition

In six frames across three stations, the polyline points defining certain rails are placed along the *inner* edge of the railhead instead of the *outer* edge, which is the convention followed throughout the dataset. Rather than modifying the original annotations, these cases are handled during mask construction, where the mask is generated taking into account that the corresponding polylines are defined along the inner edge rather than the outer one, as in the rest of the data.

Specifically, this issue occurs in four consecutive frames (093–096, timestamps `1631449460.400000020`–`1631449460.700000026`) from station `14_signals_station_14.1` and in two frames (151 and 153, timestamps 1631705118.200000025 and 1631705118.400000015) from station `7_approach_underground_station_7.1`, both with `railSide=rightRail` and `trackID = -1`.

All affected cases are validated visually to confirm consistent rail geometry in the generated masks. Two examples of this issue are shown in Figure A.1.

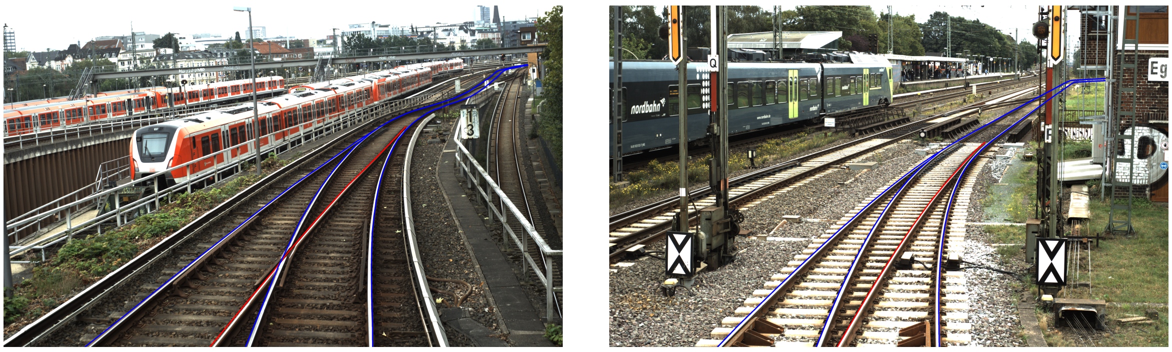


Figure A.1: Examples from the OSDaR23 dataset showing misplaced polyline definitions. Blue polylines mark correctly defined rail points along the outer edges (the dataset convention), while the red polylines mark rails whose points are defined along the inner edge instead. These deviations are accounted for during mask construction to maintain consistent rail geometry. The left image shows frame 151 from station `7_approach_underground_station_7.1` and the right shows frame 093 from station `14_signals_station_14.1`.