

ECARRNet: An Efficient LSTM-Based Ensembled Deep Neural Architecture For Railway Fault Detection

by

Salman Ibne Eunos
22266008

A thesis submitted to the Department of Computer Science and Engineering
in partial fulfillment of the requirements for the degree of
M.Sc. in Computer Science and Engineering

Department of Computer Science and Engineering
BRAC University
July 2025

© 2025. Salman Ibne Eunos
All rights reserved.

Declaration

It is hereby declared that

1. The thesis submitted is my/our own original work while completing degree at BRAC University.
2. The thesis does not contain material previously published or written by a third party, except where this is appropriately cited through full and accurate referencing.
3. The thesis does not contain material which has been accepted, or submitted, for any other degree or diploma at a university or other institution.
4. We have acknowledged all main sources of help.

Student's Full Name & Signature:

Salman Ibne Eunos
22266008
salman.ibne.eunos@g.bracu.ac.bd

Approval

The thesis titled “ECARRNet: An Efficient LSTM-Based Ensembled Deep Neural Network Architecture for Railway Fault Detection”

Submitted by:

Salman Ibne Eunus (22266008)

of Summer 2025 has been accepted as satisfactory in partial fulfillment of the requirement for the degree of M.Sc. in Computer Science and Engineering on July 2, 2025.

Examining Committee:

Supervisor:
(Member)

Dr. Md. Golam Rabiul Alam
Professor
Department of Computer Science and Engineering
BRAC University
rabiul.alam@bracu.ac.bd

Co-Supervisor:
(Member)

Dr. Munima Haque
Associate Professor
Department of Mathematics and Natural Sciences
BRAC University
munima.haque@bracu.ac.bd

Examiner:
(External)

Dr. Ahmed Wasif Reza
Dean
Faculty of Sciences and Engineering
Professor
Department of Computer Science and Engineering
East West University
wasif@ewubd.edu

Examiner:
(Internal)

Dr. Md. Ashraful Alam
Associate Professor
Department of Computer Sciences and Engineering
BRAC University
`ashraful.alam@bracu.ac.bd`

Program Coordinator:
(Member)

Dr. Md Sadek Ferdous
Professor
Department of Computer Science and Engineering
BRAC University
`sadek.ferdous@bracu.ac.bd`

Head of Department:
(Chair)

Sadia Hamid Kazi, PhD.
Associate Professor
Department of Computer Science and Engineering
BRAC University
`skazi@bracu.ac.bd`

Abstract

Accidents due to defective railway lines and derailments are common disasters that are observed frequently in Southeast Asian countries. It is imperative to run a proper diagnosis on the detection of such faults to prevent such accidents. However, manual detection of such faults periodically can be both time-consuming and costly. In this paper, we have proposed a Deep Learning (DL)-based algorithm for automatic fault detection in railway tracks, which we termed an Ensembled Convolutional Autoencoder ResNet-based Recurrent Neural Network (ECARRNet). We compared its output with existing DL techniques in the form of several pre-trained DL models to investigate railway tracks and determine whether they are defective or not, while considering commonly prevalent faults such as defects in rails and fasteners. Moreover, we manually collected the images from different railway tracks situated in Bangladesh and made our dataset. After comparing our proposed model with the existing models, we found that our proposed architecture has produced the highest accuracy among all the previously existing state-of-the-art (SOTA) architectures, with an accuracy of 93.28% on the full dataset. Additionally, we split our dataset into two parts, having two different types of faults, which are fasteners and rails. We ran the models on those two separate datasets, obtaining accuracies of 98.59% and 92.06% on rail and fastener, respectively. Model explainability techniques like Grad-CAM and LIME were used to validate the result of the models, where our proposed model ECARRNet was seen to correctly classify and detect the regions of faulty railways effectively compared to the previously existing transfer learning models.

Keywords: deep convolutional neural networks; RNN; ensemble learning; CNN; residual networks; LSTM; explainable AI; railway; fastener; rail

Dedication (Optional)

This research work is dedicated to my family members who have guided and helped me to come up to this stage in life, and also to the hundreds and millions of people who have faced death worldwide due to accidents of all sorts, including railway and road accidents, and other forms of accidents, and those individuals who are having long-term suffering right now due to accidental damage.

Acknowledgement

I would like to express my heartfelt gratitude to my supervisors, Dr. Md. Golam Rabiul Alam, Professor in the Department of Computer Science and Engineering at BRAC University, and Dr. Munima Haque, Associate Professor in the Biotechnology program, Department of Mathematics and Natural Sciences at BRAC University, for their unwavering support throughout this research endeavor. I'd like to thank my parents and the rest of my family members for their support. I would not be in this position now if it weren't for them. I'd like to thank all the faculty members, staff, students, and other BRAC University stakeholders for their direct and indirect aid during various events of my education, and research.

In addition, I would like to specially mention and thank my mentors and other team members, especially Md. Saiful Alam, senior lecturer at the Department of Computer Science and Engineering, for their immense support and motivation, and for guiding me to conceptualise this research work to make this project a reality. I would also like to mention the contribution of Shahriar Hossain, currently a PhD student at George Mason University, USA, for his endless support and endeavour in conceptualising this research work and for his help in training the comparison model, formal analysis, and data curation, and for his help in writing and in preparing some of the tables and figures. It was not possible to complete this work without his tremendous effort and perseverance. I would also like to thank the efforts of Ashik Adnan, an MS student at Gunma University, Japan, for his efforts in data collection and for writing some of the parts of the literature review along with A.E.M. Ridwan, who is currently a PhD student at Kennesaw State University, USA. This research work involves hours of effort and hundreds of hours of meetings and brainstorming, which also involve Dewan Ziaul Karim, a lecturer in the Department of Computer Science and Engineering, Brac University. His supervision, and efforts, and advice to improve the overall project helped push the research work forward. Last but not least, I would also like to thank Dr. Jia Uddin, professor at Woosong University, South Korea, for reviewing the paper and for providing useful feedback and funding for the research.

Table of Contents

Declaration	i
Approval	ii
Ethics Statement	iv
Abstract	iv
Dedication	v
Acknowledgment	vi
Table of Contents	vii
List of Figures	x
List of Tables	xi
Nomenclature	xii
1 Introduction	1
1.1 Background and Motivation	1
1.2 Problem Statement	1
1.3 Research Objectives	2
1.4 Contributions	3
1.5 Scopes and Limitations	4
1.5.1 Scope of the Research	4
1.5.2 Limitations of the Research	4
1.6 Organization of the Thesis	5
2 Related Work	6
3 Background Studies	10
3.1 Foundations of Deep Neural Networks	10
3.1.1 What Makes a Network “Deep”: Depth vs. Width	11
3.1.2 3.1.2 Mathematical Model of a Neuron	11
3.1.3 Activation Functions: From Sigmoid to G	12
3.2 Network Architecture and Building Blocks	12
3.2.1 Fully-Connected (Dense) Layers	12
3.2.2 Convolutional Layers for Spatial Awareness	12

3.2.3	Recurrent Layers & Temporal Modeling (LSTM, GRU)	13
3.2.4	Emerging Architectures: Attention and Graph Neural Networks	13
3.3	Learning Mechanics: From Forward Pass to Backpropagation	14
3.3.1	Forward Propagation Formalism	14
3.3.2	Loss Functions: MSE, Cross-Entropy & Custom Safety Losses	14
3.3.3	Gradient Computation & Backpropagation Algorithm	14
3.4	Optimization Strategies & Regularization	15
3.4.1	Gradient-Based Optimizers: SGD, Adam, RMSprop	15
3.4.2	Learning-Rate Schedules and Warm-Restarts	15
3.4.3	Overfitting and Dropout, Batch-Norm, Weight Decay	15
3.5	Explainability, Robustness Safety in Critical Systems	16
3.5.1	Why Interpretability Matters for Accident Prevention	16
3.5.2	Saliency Maps, SHAP LIME for Rail/Road Scenarios	16
3.5.3	Adversarial Robustness and Model Assurance	16
3.6	Data Pipelines for Railway Road Safety	17
3.6.1	Data Acquisition: Sensors, Cameras, Track Logs	17
3.6.2	Preprocessing Annotation: Cleaning, Labeling Near-Miss Events	17
3.6.3	Augmentation Techniques for Rare-Event Detection	17
3.7	Deep Learning Framework for Automated Accident Detection	17
3.7.1	System Architecture: Edge vs. Cloud Processing	17
3.7.2	Real-Time Inference: Latency, Throughput, and FAIL-Safe Designs	18
3.7.3	Feedback Loops: Online Learning from New Incidents	18
3.8	Comparison Models Overview	18
3.8.1	Inception-V3	19
3.8.2	Xception	19
3.8.3	InceptionResNetV2	19
4	Methodology	20
4.1	Data Collection	21
4.2	Data Pre-processing	23
4.3	ECARRNet Module	25
4.3.1	Convolutional Autoencoder	26
4.3.2	ResNet-Based Transfer Learning	27
4.3.3	Recurrent Neural Networks	28
4.3.4	Fully Connected Layers	29
5	Performance Evaluation	30
5.1	Performance Metrics	30
5.2	Experimental Results	31
5.2.1	Results Generated on Rail Dataset	31
5.2.2	Results Generated on Fastener Dataset	34
5.2.3	Results Generated on Full Dataset	37
5.2.4	LIME Visualizations to Explain the Output Predictions of ECARRNet	40
5.2.5	Summary of Research Findings	43
5.3	Discussion	44

6 Conclusion	46
6.1 Future Works and Limitations	46
Bibliography	52

List of Figures

3.1	Illustration of the forward pass in one neuron: multiply, accumulate and apply activation.	11
3.2	Pictorial Representation of RNN and LSTM	13
4.1	End-to-End Railway Fault Detection System	21
4.2	Samples of faults on railway tracks	22
4.3	Data Preprocessing Workflow	23
4.4	Example of offline augmentations applied to a single rail image. . . .	23
4.5	Architecture of the Proposed ECARRNet Model	26
5.1	Accuracy curves for each of the models on the rail dataset	32
5.2	Confusion matrices for each of the models on the rail dataset	33
5.3	Grad-CAM visualizations for transfer learning models on the rail dataset	33
5.4	Accuracy curves for each of the models on the fastener dataset. . . .	35
5.5	Confusion matrices for each of the models on the fastener dataset. . .	36
5.6	Grad-CAM visualisations for transfer learning models on the fastener dataset.	36
5.7	Accuracy curves for each of the models on the full dataset.	38
5.8	Confusion matrices for each of the models on the full dataset. . . .	39
5.9	Grad-CAM visualisations for transfer learning models on the full dataset.	39
5.10	LIME visualizations for ECARRNet model on each of the datasets. . .	41
5.11	Accuracy for each of the models on different datasets.	42

List of Tables

2.1	Overview of the Reviewed Sources.	9
4.1	Offline augmentations parameters applied to expand the dataset. . . .	24
4.2	In-code augmentation parameters (ImageDataGenerator)	24
4.3	Model architecture summary	26
5.1	Classification Report for Rail Dataset	34
5.2	Classification Report for Fastener Dataset	37
5.3	Classification Report for Full Dataset	40
5.4	Sum of False Positives & False Negatives from all datasets for each of the models	42
5.5	Research Objectives and Evidence of Fulfillment	45

Nomenclature

The next list describes several symbols & abbreviation that will be later used within the body of the document

GRU Gated Recurrent Units

AI Artificial Intelligence

ANN Artificial Neural Network

CNN Convolutional Neural Network

DL Deep Learning

DNN Deep Neural Network

ELU Exponential Linear Unit

FCN Fully Connected Network

GCN Graph Neural Network

GELU Guassian Error Linear Unit

GNN Graph Neural Network

GRADCAM Gradient-Weighted Class Activation Mapping

LIME Local Interpretable Model-Agnostic Explanations

LSTM Long Short-term Memory

NN Neural Network

RCNN Region-based Convolutional Neural Networks

ROI Region of Interest

Chapter 1

Introduction

1.1 Background and Motivation

Since 1862, the railway has been an integral element of Southeast Asia as a cost-effective, safe, and convenient mode of transportation for all classes of people. Railway tracks have played a crucial part in carrying masses over them in the form of passenger trains and freight trains. But train accidents can inflict huge damage to a nation, as their aftermath brings a much more horrifying scenario compared to any other transportation accident. Therefore, in this thesis, we have used images from abandoned and functional railway tracks and used AI to automate the process of fault detection in railway tracks so that faults can be detected beforehand to reduce accidents and thus to alleviate suffering. We have used transfer learning models and other existing models to create a hybrid ensemble model that can identify faults with reasonable accuracy and surpasses the state-of-the-art model in performance at various instances. In addition, we have also used model explainability to make our state-of-the-art model to increase its transparency and to explain how it is working under the hood to reach such astounding performance.

1.2 Problem Statement

The majority of the accidents occur due to derailments that have gruesomely stacked up many deaths. According to [1], 43% of all subsequent train accidents and 67 percent of the death toll in India are caused by railroad level crossings. The fatality rate from different countries might show different numbers but one thing that remains constant is its high percentage. In [2], a comparative analysis of accidents was performed in Bangladesh during 2008-2014, where among 18771 accidents, (12%) took place on railways. A lot of effort was put into accident prevention and recovery but less work is being conducted on railway accident prevention in Bangladesh [3]. Identifying the faults present on railway tracks and attending to their repairs can prevent accidents. However, the fault detection method is done manually in Bangladesh railways. Manual detection and analysis of faults are hectic, which can also cause a lack of accuracy if people are assigned to cover the huge distances of railway tracks for inspection. This is because they can easily miss some damaged spots which may cause accidents. Additionally, people with good technical knowledge of railway tracks and their subsequent components like fasteners, sleepers, and railway switchers need to inspect the tracks. Here, DL techniques can come in handy as they

can replace any manual detection techniques which are laborious, time-consuming, and inefficient. As a result, this will lead to faster and more rapid crisis response thus leading to less number of accidents and deaths.

DL has effectively proved its capability for image processing for quite a while. One of its huge triumphs was accomplished when transfer learning architectures like AlexNet [4] exhibited their adequacy for the massive scale of image processing tasks. After that, we witnessed the development of a few remarkable Neural Network (NN) models, such as VGG16 [5], as well as exceptional results for image recognition in everyday applications, including medical image analysis [6], [7], [8] for the detection of lung diseases and the investigation of ransom calls, face recognition [9], [10], object detection with imagery from unmanned aerial vehicles [11], [12] and much more.

1.3 Research Objectives

General Objective

This master's thesis's primary goal is to develop, put into practice, and evaluate a novel ensemble-based deep learning framework for the accurate and trustworthy detection of faults in railway tracks. The main focus is on Southeast Asian railway networks, where automated inspection systems are especially crucial due to ageing equipment and erratic maintenance procedures.

Specific Objectives:

- The objective is to propose a scalable and automated fault detection pipeline that can be integrated into real-time railway inspection systems, enhancing operational reliability and railway safety.
- To create and preprocess a domain-specific dataset that includes images of both defective and non-defective railway lines taken from actual locations, where preprocessing includes data augmentation and fixing class imbalance problems.
- To develop a ResNet-based convolutional autoencoder hybrid ensemble architecture that improves classification performance by fusing the benefits of several state-of-the-art deep learning models.
- To assess the proposed model by utilising important performance parameters like accuracy, precision, recall, and F1-score in comparison to some benchmark models, including InceptionV3, InceptionResNetV2, and Xception.
- To demonstrate the effectiveness of the suggested approach in terms of accuracy and generalisability for various fault types (such as faults on rails, fasteners, etc.).
- The proposed model's explainability and decision transparency will be compared to other deep learning models using visualisation approaches like LIME and Grad-CAM.

Therefore, the goal of this project is to bridge the gap between academic deep learning approaches and workable, deployable railway safety systems to eventually help reduce track-related railway accidents.

1.4 Contributions

This thesis brings forward a series of original and novel contributions to the field of deep learning and railway fault detection. The proposed framework introduces several key innovations in the domain of automated railway fault detection, offering improvements across methodology, data, performance, and interpretability. These are the major contributions of this thesis:

- **A novel deep learning architecture (ECARRNet):** ECARRNet is a purpose-built, ensemble deep learning architecture containing a convolutional autoencoder for robust feature abstraction, meaning it can extract meaningful features that are resilient to noise and variations in input data, which reduces data dimensionality and enhances model performance. It also consists of residual networks, or ResNet blocks, and recurrent neural networks, or LSTMs, where the ResNet blocks are used for gradient flow and deep feature extraction, and LSTMs for learning sequential dependencies. This layered design captures spatial and temporal information crucial for detecting defects in rail components across image sequences. Unlike transfer learning models such as InceptionV3, Xception, etc, ECARRNet is custom-engineered specifically for rail inspection, allowing domain-specific optimisation, which gives it its novelty and originality.
- **Use of category-specific dataset:** We have curated and prepared a custom dataset comprising 3 different versions: rail-only images, fastener-only images, and a combined dataset containing both rail and fastener images. Separate and joint training improve generalisability across different fault types. Most of the previous work uses single-category inspection, making this a novel idea. ECARRNet introduces a multi-object defect detection approach within a unified model, advancing comprehensive railway maintenance.
- **End-to-End Fault Detection Pipeline:** This system includes Region of Interest (ROI) cropping and resizing, offline and online augmentation, SMOTE-based oversampling, and repeated stratified K-fold cross-validation. This thus forms a reproducible data preparation and training pipeline tailored for imbalanced and domain-specific data. This complete automation of preprocessing, balancing classes, and evaluation brings a novel approach, making ECARRNet deployable in real-world scenarios with minimal human intervention.
- **Explainable AI Integration:** We have applied GRAD-CAM for visualizing the spatial attention of the CNN-based models and LIME for local interpretability of our proposed model. ECARRNet is one of the first fault detection systems in railway diagnostics to incorporate explainability tools, allowing better debugging and validation.
- **Empirical benchmarking and superior accuracy:** The model was benchmarked against state-of-the-art pre-trained networks. ECARRNet outperformed InceptionV3, Xception, and InceptionResNetV2 in all datasets, achieving accuracies of up to 98.59% on rail-only images and 93.28% on combined datasets. It has also produced competitive results in precision, Recall, and F1 score in all datasets. Results produced by ECARRNet are themselves state-of-the-art in rail and fastener defect detection using deep learning, demonstrating the framework's effectiveness.

- **Contribution to Dataset Availability:**

We have made the datasets fully available online through Kaggle datasets, and the links are shared in ResearchGate and Google Scholar, making the datasets fully open source and readily available. Our dataset is a custom-made dataset collected from railway tracks in Bangladesh, which adds a new geographical and environmental context to defect detection. This brings novelty, as it helps diversify training data and improves the model’s robustness in under-represented regions, which is a key challenge in railway AI systems.

1.5 Scopes and Limitations

1.5.1 Scope of the Research

This research focuses on the development and evaluation of ECARRNet, an ensemble deep learning architecture that combines Convolutional Autoencoder (CAE), ResNet, and LSTM to detect faults in railway tracks. The primary scope of this work includes:

- **Application Domain:** The system is designed specifically for railway track fault detection, including surface defects, cracks, and anomalies, primarily in Southeast Asian railway systems.
- **Input Modalities:** The research uses image-based data collected from track-level perspectives (e.g., camera-mounted trolleys or drones). Other modalities such as vibration sensors, audio data, video, or thermal imagery are excluded.
- **Model Architecture:** Focus is given to a hybrid deep neural network ensemble that incorporates spatial feature extraction (CAE, ResNet) and temporal sequence modelling (LSTM). The research does not explore transformer-based or attention-driven architectures.
- **Explainability:** Tools like Grad-CAM and LIME are employed to enhance model interpretability and support practical deployment in real-world scenarios.
- **Evaluation Metrics:**
The model is assessed based on classification accuracy, precision, recall, F1-score, and false positive/negative rates.

1.5.2 Limitations of the Research

While the ECARRNet framework demonstrates promising results, several limitations are acknowledged:

- **Dataset Constraints:** The dataset used in this research is limited to images collected from specific regions in Southeast Asia and may not generalize to railway systems in drastically different climates, infrastructure types, or geographical conditions.
- **Imbalanced Class Distribution:** Some fault types (e.g., minor surface wear or rare defects) are under-represented, which may affect model generalisation to rare fault cases.

- **Manual Preprocessing Steps:** The data preprocessing stage, including Region of Interest (ROI) cropping and image augmentation, involves some manual heuristics that may introduce bias or variability.
- **No Multimodal Integration:** Although rail faults can be detected via multiple sensor types, this research restricts itself to visual data only, limiting its scope.
- **Computational Resources:** Although designed to be lightweight, training ECARRNet still requires GPU support and offline computation for model development, which may challenge real-time deployment in remote areas. This problem can be solved nowadays using low-cost Single Board Computers (SBCs) readily available in the market, such as the Raspberry Pi 5 and Jetson Nano from Nvidia.
- **Assumptions:** It is assumed that the captured images are of sufficient resolution and that the faults are visible at the surface level. Subsurface or latent defects are outside the scope of this study.

1.6 Organization of the Thesis

The remaining component of the paper is assembled as follows: Chapter 2 presents background data on the algorithm used by earlier researchers working on railway identification of defects, as well as a selection of well-written research publications on our topic of study. Chapter 3 talks about some of the theoretical parts of deep neural networks, how they work, and an overview of the deep learning model used to compare the results of the proposed architecture. Details of data acquisition, preprocessing, breakdown of dataset construction, and the architecture of the proposed model are described in Chapter 4. The performance of the proposed model is compared with SOTA models. It is evaluated in Chapter 5 while conveying the results section, which features visualisations of the observations in terms of the accuracies and losses of each model. At the same time, the conclusion is discussed in Chapter 6.

Summary

This chapter introduced the research context, identified key challenges in railway fault detection, and defined the research objectives and scope. With these foundations in place, the next chapter provides an overview of related work in the field and situates ECARRNet within the current landscape of deep learning-based fault detection approaches.

Chapter 2

Related Work

We have found various ML, DL, and computer vision-based approaches to detect railway track faults autonomously while doing our background study. Authors of [13] proposed a model where they used multiple models and got different results from them. They have proposed DL models in the form of Inception V3, ResNet50, and Faster R-CNN to detect faults in loose ballast, sun kink, track switches, and signals. They ran these models on more than 100 GB of video data and got 96% accuracy for Loose Ballast, 100% accuracy SunKink, 95.6% accuracy for Track Switch, and 99.4% for Signal Colour models. Also, they prepared a track health index using that data. Lin, Y. et al. [14] have suggested a GPS-based method that determines the fault's location and uses a GoPro camera to take photographs. YOLO v3 has been used as an updated model of YOLO in their proposed architecture, which has a base network with darknet-53. Moreover, they have used Feature Prediction Network (FPN) to upgrade the ability of prediction for small objects. They have captured the images of fasteners and emphasized on detection of fasteners. Thus, they have shown the precision to be 89% and the recall being 95%.

In paper [15], the authors have proposed a multiphase DL technique to perform the segmentation of images. First, they detected the track surface defect using visual-based track inspection systems (VTIS) and located the Regions of Interest (ROI). The segmentation usually works by cropping the segmented image to the ROI. They have demanded their accuracy increase from 71% to 90% as shown in Table 2.1. In this study [16], the U-net graph segmentation network and the saliency cues approach of damage location were used to identify high-speed rail damage. Their detection accuracy rate is 99.76%.

Researchers at paper [17] have discovered that combining image processing and feeding that data into a neural network (NN) is more effective and efficient than alternative approaches. They have used the backpropagation method and talked about a hidden layer that will be trained using a dataset created and absolute output, with 1 denoting railway track cracks and 0 denoting no cracks. On the other hand, paper [18] has proposed conventional image processing to detect defects in fasteners and to remove noise from the image which includes techniques like a median filter, binary image conversion, Gaussian noise removal, etc. along with a projection algorithm. The use of Dense-SIFT features was suggested as a strategy for fault detection and identification. Finally, VGG16 was trained to recognize and diagnose fastener defects. Furthermore, Faster RCNN was also applied to increase the detection rate and efficiency.

They have shown an accuracy of 97.14% using the faster VGG16 network. Min et al. [19] used scar and crackle in real-time using machine vision techniques to detect defects. This paper proposed a damage detection model where the image is collected at first using a plane array camera along with LED light to increase the brightness of the picture. Then the target area is localized to extract value from H in Hue, Saturation, and Lightness (HSL). After that, the authors used several image enhancement techniques to get more features from the image such as—image denoising, threshold value, and morphological processing. They also extracted the contour information of the faulty location using directional chain code tracking methods. M. Karakose and colleagues in [20] have devised a technique for detecting railway faults using two separate units. They created IAS to fulfill three tasks: expanding train tracks, enhancing contrast, and automating the defect detection procedure. They transformed the image to grayscale first, then utilized three separate Canny Edge Detection techniques.

In this work [21], the authors have offered an automated video analysis-based rail track inspection approach. They kept track of how many edge pixels each window has. The density of edges in each window is determined by this. The peak value was calculated by charting edge density in each window vs. window numbers. The clip's location is represented by this peak. When the peak value is less than a certain threshold, the window is regarded to contain noise rather than a clip. The frame will be tagged as containing two clips if two peak values are both greater than a defined threshold. As a result, they achieve an average accuracy of 95.3%. In research [22], they have proposed an entirely CNN-based system, trained on ten different classes of materials, then feature-mapped using ten different channels. Their purpose was to concurrently identify the most likely fastener position inside each preset ROI and then categorize those detections into one of three fundamental conditions: background (or absent fastener), broken fastener, or good fastener. They tested our fastener detector on 85 kilometers of continuous trackbed photos to see how accurate it is. Using the Deep CNN MLT 3 Method, they were able to attain 95.02% accuracy.

Alawad, H. et al. [23] present a monitoring technique that makes use of computer vision to automatically and quickly detect hazards in stations by recognizing risky acts, giving real-time support for decision-makers, and minimizing the possible effects of undesirable events. Then they build a sequential model at each layer, which is the simplest form of a layer in the Keras library. With their CNN-0 model, they were able to attain an accuracy of 81.90%. Kamilaris, A. et al. [24] compared strategies for the same data in the same research paper, using the same measure. The prominent CNN architectures including AlexNet, VGG, and Inception-ResNet were used. They also experimented with their designs, some of which combined CNN with other approaches.

Yang, C. et al. [25] have proposed an ML-based railway inspection approaches, which include a feature-based technique and a deep neural network-based technique based on acceleration data. As a result, ResNet and FCN are examined in this work. Three convolution blocks are used to construct FCN, which is then succeeded by a soft-max layer and a global average pooling layer. By establishing a shortcut link between consecutive convolutional layers, ResNet expands neural networks to a very deep structure. The results of the experiments showed that ResNet and fully convolutional networks (FCN) both performed well in joint detection. Yao, H. et al. [26] have suggested a model in which the RNN is seen as having a similar ability to

Convolutional Neural Network to derive features from pictures. CNN and RNN are of equal importance. To better balance the two types of characteristics recovered by the CNN and RNN during the training phase, the model uses the perceptron attention mechanism to weigh the features received by the CNN and RNN, respectively. They used stackable LSTM as the RNN module to extract the time sequence properties from the pixels. The CNN (Inception-ResNet V2) technique was also utilized, and the greatest accuracy was 83%.

Evaluation Metrics used in DL tasks have a critical role in generating the best classifier, according to Alzubaidi, L. et al. [27], they are used in two steps of a typical data categorization procedure: training and testing. They discussed FPGA, which can be used to create CNN overlay engines with over 80% efficiency, eight-bit precision, and over 15 TOPs peak performance. Voxnet's accuracy was 79%, whereas ResNet's was 80%. In the framework of the European Common Agricultural Policy, CamposTaberner, M. et al. [28] want to learn more about an RNN for land use categorization based on Sentinel-2 time series (CAP). The results of the investigation show that the red and near-infrared Sentinel-2 bands provide the most helpful data. The traits acquired from summer acquisitions were the most important in terms of temporal information. The performance precision of the 2-BiLSTM network in every class is over 91.4%.

Shafique, R. et al. [29] provide an acoustic analysis-based autonomous railway track defect detection system. The data collected on Pakistani railway lines using acoustic signals and the application of various classification systems to the gathered data are two major contributions of this work. They achieve the greatest outcomes with RF and DT, which have a 97% accuracy rate. A DL-based fault diagnosis network for RVSFD was built by Ye, Y. et al. [30]. There are three phases to the diagnostic network: The GWN strategy is used to the acceleration signal in the first phase (data preprocessing) to make the diagnostic network resistant to relatively high-frequency influences induced by track imperfections. An EST strategy is presented in the second step (training dataset creation) to increase the diagnostic network's robustness against wheel wear. Finally, a GONEST-1D CNN-based fault diagnosis network of high-speed train suspension systems is created in the third phase (training and recognition). All the states can be completely recognized using the confusion matrix given for the testing samples (100%).

Summary

This chapter examined current developments in deep learning and computer vision for railway fault detection. It found shortcomings such as restricted explainability and low generalisability by contrasting deep learning-based techniques with conventional inspection methods. A new ensembled architecture that overcomes these drawbacks is justified by the review, which is discussed in detail in the following chapter on Background Studies.

Table 2.1: Overview of the Reviewed Sources.

Ref.	Objectives	Datasets	Classifiers	Accuracy
Mittal et al. (2017) [13]	Proposed models to detect faults in Loose Ballast, SunKink, Track Switch, and Signals.	Datasets are made from the 100 GB of video data	Inception V3, ResNet50, and Faster R-CNN	100%
Lin et al. (2019) [14]	GPS determines the fault's location and uses a GoPro camera to take photographs.	Datasets are made from the images extracted from video data	Yolo v3	95%
James et al. (2018) [15]	Proposed a multiphase DL technique to perform segmentation of images.	Datasets are made from the actual rail tracks that are collected by a COTS VTIS.	Visual-based track inspection systems(VTIS), TrackNet	90%
Lu et al. (2020) [16]	To detect high-speed railway rail damage, a combination of the U-net graph segmentation network and the saliency cues approach of damage localization was presented.	Type-I RSDDs dataset	SCueU-Net	99.76%
Wei et al. (2019) [18]	They proposed defect detection and identification methods using Dense-SIFT features.	Dataset is made up of images that are taken from Beijing Metro Line 6.	RCNN, VGG16	97.14%
Min et al. (2018) [19]	Proposes a damage detection model where the Image is collected using a plane array camera and LED lights.	Datasets are made from images of plane array camera	Directional chain code tracking	
Karakose et al. (2017) [20]	A computer-based visual rail condition monitoring is proposed.	Data acquired from a camera placed on top of the train	Image processing	
Singh et al. (2006)[21]	An automated video analysis-based rail-track inspection approach.	Data acquired from a video camera placed in front of the train	Image processing	95.3%
Gibert et al. (2017) [22]	Automated track inspection using computer vision and pattern recognition methods.	Data acquired from a camera placed in front of the moving train	Deep CNN	95.02%
Alawad et al. (2020) [23]	Using computer vision and pattern recognition, perform risk management in railway systems.	Manually collected images from different rail tracks	Keras, ReLU, CNN	81.90%
Kamilaris et al. (2018) [24]	Examines the research questions related to agriculture, models used, data sources, and precision.	PlantVillage, LifeCLEF, MalayaKew and UC Merced	AlexNet, VGG, Inception-ResNet	
Yang et al. (2021) [25]	ML-based inspection using acceleration data via feature and deep learning methods.	Acceleration dataset from sensors on rail inspection car	ResNet, FCN	100%
Alzubaidi et al. (2021)[27]	Survey of DL improvements and applications.	ImageNet, CIFAR-10, CIFAR-100, MNIST	Xillinx, Voxnet, ResNet	80%
Campos-Taberner et al. (2020) [28]	Use of DL for remote sensing data analysis.	Remote sensing	Sentinel-2, BiLSTM	91.4%
Shafique et al. (2021) [29]	Acoustic analysis-based autonomous railway defect detection.	Data from Pakistani railway lines using acoustic signals	SVM, LR, RF, DT	97%
Ye et al. (2021) [30]	DL-based fault diagnosis for RVSFD.	Accelerometer sensor data	GONEST-1D CNN	100%
Eunus et al. (2024) [31]	A Deep Learning (DL)-based algorithm for automatic fault detection in railway tracks, termed an Ensemble Convolutional Autoencoder ResNet-based Recurrent Neural Network (ECARRNet).	Manually collected images from different railway tracks of the Bangladesh railway and prepared 3 different versions of the dataset to perform an ablation study.	Proposed ECARRNet	98.59% (rail), 92.06% (fastener), 93.28 % (combined)

Chapter 3

Background Studies

This chapter begins by establishing the foundation by examining what constitutes a “deep” neural network, breaking down the mathematical representation of a single neuron and the nonlinear activation functions that give contemporary structures their life (Sections 3.1.1–3.1.3). Then, we build up layer by layer, exploring fully connected, convolutional, and recurrent blocks before exploring new designs like graph networks and attention mechanisms, so you can understand both formally and intuitively how each part helps with representation learning and feature extraction (Sections 3.2.1–3.2.4). With the architecture in hand, we turn to the mechanics of learning: formally defining the forward pass, examining loss functions tailored to classification and regression (and even custom safety losses), and deriving the backpropagation algorithm that drives weight updates (Sections 3.3.1–3.3.3). Next, we discuss how to steer that learning process through optimizers (SGD, Adam, RMSprop), schedule and warm-restart learning rates, and regularize models with dropout, batch normalization, and weight decay to guard against overfitting (Sections 3.4.1–3.4.3). Finally, we confront the demands of safety-critical systems by unpacking the importance of model interpretability, exploring saliency maps, SHAP, and LIME for transparent decision-making, and surveying adversarial robustness techniques that ensure reliability in accident-prevention applications (Sections 3.5.1–3.5.3).

3.1 Foundations of Deep Neural Networks

The basic perceptron is expanded upon by deep neural networks (DNNs), which allow the model to learn hierarchical feature representations straight from the data by stacking numerous layers of nonlinear units. Fundamentally, each artificial neuron calculates the weighted sum of its inputs plus a bias term. Then, it applies an activation function, which is often sigmoid or tanh, but more recently, ReLU or its variations, to introduce the nonlinearity required for complex function approximation. Due to the universal approximation theorem, DNNs may approximate almost any continuous mapping by assembling these nonlinear transformations depth-wise. In practice, however, excessive width alone frequently produces less compact and generalisable models than depth. Forward propagation is used in training to calculate outputs and a loss (such as cross-entropy or mean-squared error), while backpropagation is used to acquire gradients for effective end-to-end optimisation using algorithms like Adam or SGD with momentum. Practitioners frequently use regularisation strategies—dropout, batch normalization—and carefully plan learning

rates, balancing robust generalisation with fast convergence, to avoid overfitting and stabilise learning [32] , [33] , [34] .

3.1.1 What Makes a Network “Deep”: Depth vs. Width

An input layer’s ”depth” in a neural network is the number of successive nonlinear transformations (also known as hidden layers) it passes through, while its ”width” is the number of neurons in each layer. Any continuous function on a compact domain can be approximated by a single sufficiently wide hidden layer, according to the universal approximation theorem. However, in reality, this often requires an unreasonably high number of neurons. While depth makes use of compositionality, each layer pulls elements at a different level of abstraction, making it possible to describe complex functions in a compact manner. Extensive depth-separation experiments demonstrate that complex functions can be represented with a polynomial-size deep model, but if limited to a shallow architecture, they would require exponentially many units. Deeper networks empirically discover hierarchical patterns, such as edges, forms, and objects in vision, and frequently generalise better from sparse data, whereas just widening the network can result in redundancy, optimisation challenges, and decreased flexibility [35] ,[36].

3.1.2 Mathematical Model of a Neuron

At its core, an artificial neuron performs two simple steps:

1. Aggregation: Compute a weighted sum of inputs plus a bias term.
2. Activation: Apply a nonlinear function to that sum.

Concretely, given an input vector

$$z = \sum_{i=1}^n w_i x_i + b, \quad y = \sigma(z) \quad (3.1)$$

where σ is an activation function (e.g., sigmoid, ReLU, tanh). These steps have been further illustrated with a simple diagram in Figure 3.1 as shown below:

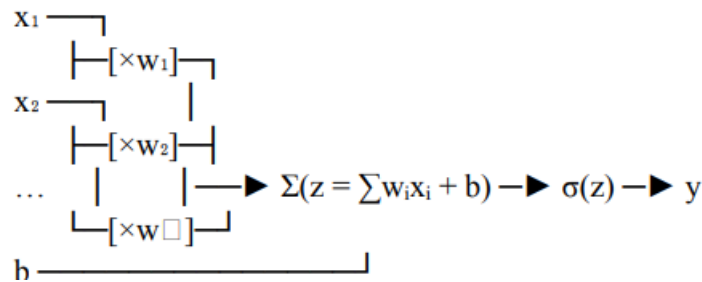


Figure 3.1: Illustration of the forward pass in one neuron: multiply, accumulate and apply activation.

Each neuron thus implements a simple function; by chaining millions of these units, deep networks can approximate highly complex mappings from raw data to predictions.

3.1.3 Activation Functions: From Sigmoid to G

Activation functions introduce the essential nonlinearity that allows deep networks to model complex relationships. Early choices like the sigmoid, where, $\sigma(z) = \frac{1}{1+e^{-z}}$ and $\tanh(z)$ offered smooth, bounded outputs but suffer from saturation—when $|z|$ is large, gradients vanish and slow learning. The Rectified Linear Unit (ReLU), where $\phi(z) = \max(0, z)$, broke this bottleneck by maintaining a constant gradient for positive inputs, accelerating convergence, and inducing sparse representations, though it can “die” when units output zero permanently. To mitigate that, Leaky ReLU, ELU, and SELU introduce small negative slopes or self-normalizing properties, improving gradient flow and stability. The Gaussian Error Linear Unit (GELU), where $\phi(z) = z \cdot \Phi(z)$, with ϕ the standard Gaussian CDF, further refines this idea by weighing activations probabilistically, blending smoothness with non-monotonic curvature—an approach that’s become standard in large-scale transformer models for its superior convergence and performance [37], [38], [39].

3.2 Network Architecture and Building Blocks

Basic modules that iteratively change data representations are stacked and combined to create network designs. Convolutional layers use weight sharing and spatial locality to learn hierarchical patterns, recurrent blocks (LSTM, GRU) incorporate feedback loops to model temporal dependencies, and fully-connected (dense) layers connect all inputs to all neurons, capturing global interactions. More recently, graph neural networks manage irregular, relational data, while attention mechanisms enable models to dynamically focus on prominent characteristics. In order to balance expressivity, efficiency, and domain-specific constraints, practitioners choose, arrange, and adjust these building elements to give models the proper inductive biases.

3.2.1 Fully-Connected (Dense) Layers

To create a new feature vector where each output unit depends on every input, a fully-connected layer multiplies the incoming activation vector by a weight matrix and applies a bias. For tasks requiring feature fusion, like classification heads or embedding transformations, dense layers are perfect because of the network’s ability to learn any linear combinations of features due to its global connectedness. However, because of the potential for overfitting caused by their parameter-intensive nature, practitioners frequently combine them with dimensionality restrictions (bottleneck layers) and regularisation strategies (weight decay, dropout) to balance capacity and generalisation.

3.2.2 Convolutional Layers for Spatial Awareness

Convolutional layers create output “feature maps” that hierarchically recognise edges, textures, and higher-order forms by swiping a set of tiny, trainable kernels over the input feature map to learn spatially local patterns. In contrast to dense layers, weight sharing and local connectivity significantly reduce parameters while automatically recording translation equivariance, which is essential for vision tasks on road and railway imagery. The foundation of contemporary accident-detection pipelines is

made out of stacking many convolutional layers, which produce increasingly abstract representations ranging from simple corners to complicated objects [40], [41].

3.2.3 Recurrent Layers & Temporal Modeling (LSTM, GRU)

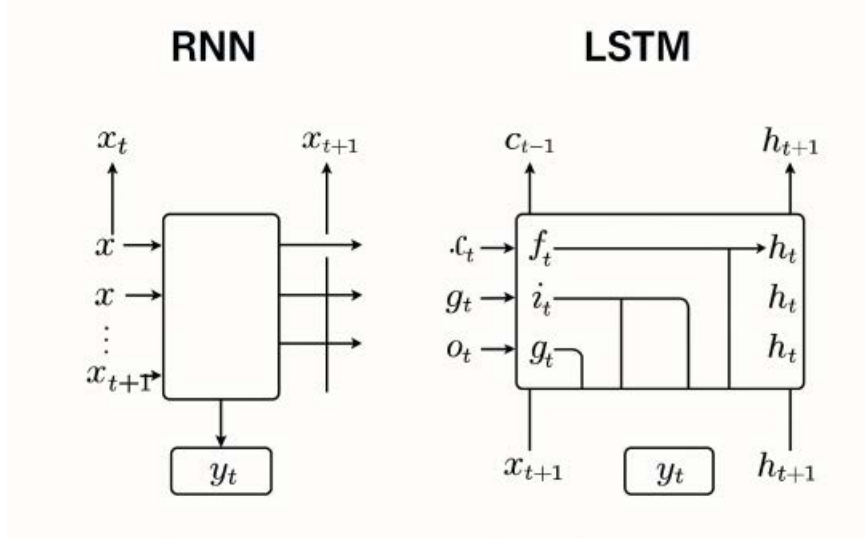


Figure 3.2: Pictorial Representation of RNN and LSTM

As demonstrated in Figure 3.2, recurrent layers introduce a hidden state that updates at each time step, allowing networks to retain and process information in sequences. Vanilla RNNs, however, suffer from vanishing or exploding gradients when modeling long-range dependencies. The Long Short-Term Memory (LSTM) cell overcomes this by incorporating input, forget, and output gates around a dedicated memory cell, which dynamically controls information storage and retrieval. The Gated Recurrent Unit (GRU) further simplifies this design by merging gates into update and reset functions, reducing parameters without sacrificing performance. Both architectures excel at capturing temporal patterns in streams of sensor data or video frames, making them invaluable for early detection of anomalies and accident-prevention systems [42], [43].

3.2.4 Emerging Architectures: Attention and Graph Neural Networks

Recent advances move beyond fixed-receptive-field layers by letting models learn where to look and how to relate arbitrary entities. Self-attention, popularised by the Transformer, computes pairwise affinity scores between all elements in a sequence and uses these to generate context-aware representations in parallel—enabling efficient modelling of long-range dependencies without recurrence. Graph Neural Networks (GNNs) extend this idea to irregular domains: each node iteratively aggregates and transforms features from its neighbors, thereby capturing relational structure in data such as road networks or rail topologies. Graph Convolutional Networks (GCNs) implement this through localized spectral filters on the graph Laplacian, while methods like GraphSAGE perform inductive neighbor sampling and feature pooling to learn embeddings that generalize to unseen nodes. Together, attention

and GNNs furnish powerful, flexible primitives for accident-prevention systems that must fuse spatial, temporal, and relational cues [44], [45], [46].

3.3 Learning Mechanics: From Forward Pass to Backpropagation

In deep neural networks, the very first step of learning starts from a forward pass with the goal to generate predictions. Then the resulting error is backpropagated through each layer to update the parameter of the model iteratively.

3.3.1 Forward Propagation Formalism

Let the input vector be denoted as $\mathbf{a}^0 = \mathbf{x}$. At each layer $l = 1, 2, \dots, L$, if we compute the pre-activation:

$$\mathbf{z}^l = \mathbf{W}^l \mathbf{a}^{l-1} + \mathbf{b}^l \quad (3.2)$$

and then apply an element-wise nonlinearity σ , to get the activation

$$\mathbf{a}^l = \sigma(\mathbf{z}^l) \quad (3.3)$$

By repeating this transformation sequentially from layer $l = 1$ to L , the network maps raw input $\mathbf{x}$ to the final output $\mathbf{a}^L$. This concise sequence of affine mappings and nonlinearities forms the backbone of forward propagation in deep models.

3.3.2 Loss Functions: MSE, Cross-Entropy & Custom Safety Losses

Loss functions guide parameter adjustments during training by quantifying the difference between a model’s outputs and true targets. The mean squared error (MSE), which calculates the average of squared deviations for regression tasks, strongly penalises big errors and promotes accurate value predictions. When the model is confidently incorrect, cross-entropy, which measures the difference between the genuine one-hot distribution and anticipated probability, produces large gradients, accelerating convergence. To ensure that missed hazards cost more during optimisation and that the model learns to prioritise reliable detection over marginal accuracy gains, practitioners frequently create custom safety losses in safety-critical applications like accident prevention that add risk-based penalty terms or weight false negatives more severely [47], [48].

3.3.3 Gradient Computation & Backpropagation Algorithm

Deep networks use backpropagation for gradient calculation, which recursively applies the chain rule to derive derivatives of the loss for each weight and bias. Once the output-layer error, $\delta^L = \frac{\partial \mathcal{L}}{\partial \mathbf{z}^L}$, has been calculated, the approach uses:

$$\delta^l = (\mathbf{W}^{l+1\top} \delta^{l+1}) \odot \sigma'(\mathbf{z}^l) \quad (3.4)$$

where $\sigma'(\mathbf{z}^l)$ the activation derivative propagates this signal backward down each layer.

The gradients of parameters are as follows:

$$\frac{\partial \mathcal{L}}{\partial \mathbf{b}^l} = \delta^l \quad \text{and} \quad \frac{\partial \mathcal{L}}{\partial \mathbf{W}^l} = \delta^l (\mathbf{a}^{l-1})^\top \quad (3.5)$$

Backpropagation makes end-to-end training of deep architectures manageable by computing all gradients in $O(\text{number of parameters})$ time by reusing stored activations and pre-activations from the forward pass.

3.4 Optimization Strategies & Regularization

3.4.1 Gradient-Based Optimizers: SGD, Adam, RMSprop

Gradient-based optimisers update a network’s parameters by following the negative of the loss gradient, but each method handles step sizes and momentum differently. Stochastic gradient descent (SGD) computes updates on small, random batches and can incorporate momentum to smooth out oscillations and speed up on gently sloping surfaces. Optimizers like - RMSprop adjusts each parameter’s learning rate by dividing the gradient by a running average of its recent squared values, effectively normalizing step sizes to the local curvature. Adam unites these ideas by maintaining bias-corrected first and second moment estimates of the gradients, yielding adaptive learning rates and built-in momentum that often lead to faster, more reliable convergence across a variety of deep learning tasks [49], [50], [51].

3.4.2 Learning-Rate Schedules and Warm-Restarts

Adjusting the learning rate over time helps a model converge more smoothly: common schedules include step decay, exponential decay, and cosine annealing, each progressively shrinking the step size to fine-tune around minima. Warm-restarts periodically reset the rate to a higher value—often following a cosine ramp—injecting fresh “momentum” that helps escape shallow valleys and explore new regions of the loss landscape. By blending gradual decay with strategic restarts, networks can achieve faster convergence and improved generalization [52].

3.4.3 Overfitting and Dropout, Batch-Norm, Weight Decay

Overfitting occurs when a network learns spurious patterns in the training data rather than the true underlying structure. Dropout combats this by randomly zeroing a fraction of activations. Each update forces neurons to develop more robust and redundant representations. Batch-normalization standardizes layer inputs on the fly, smoothing the optimization landscape and injecting a slight regularizing effect by adding noise through mini-batch statistics. Weight decay (L2 regularization) penalizes large weights directly in the loss, shrinking model capacity and discouraging overly complex functions. By combining these methods, practitioners can significantly reduce overfitting, speed up training, and improve generalisation [53].

3.5 Explainability, Robustness Safety in Critical Systems

3.5.1 Why Interpretability Matters for Accident Prevention

Although deep neural networks are excellent at identifying minute irregularities in vibration, visual, audio, or thermal sensor data, their opaque decision rules can mask important failure Modes in railway fault detection. Maintenance engineers can confirm that the model’s logic is consistent with recognised physical occurrences by using interpretability to identify which input features—such as certain frequency bands or wheel-rail contact points—drive a problem prediction. Transparency not only promotes trust and facilitates root-cause analysis in safety-critical situations, but it also helps identify biases or erroneous correlations that may cause missed flaws or false alarms. Stakeholders may guarantee that automated alarms result in prompt, focused interventions by incorporating explainable AI techniques (such as saliency maps and SHAP) into railway monitoring systems. This will ultimately lower the probability of accidents and increase operational reliability.

3.5.2 Saliency Maps, SHAP LIME for Rail/Road Scenarios

In railway fault detection, interpretability techniques help translate a DNN’s complex reasoning into actionable insights for engineers. Saliency maps propagate output gradients back to the input, producing heatmaps that pinpoint which time-series segments—such as spike patterns in axle vibration—or spatial zones along the track, most drove a fault score. SHAP uses game-theoretic Shapley values to assign each sensor feature a consistent, additive contribution, making it straightforward to rank detectors (e.g., wheel-rail acoustic channels) by their influence on a given alarm. LIME constructs a simple, local surrogate model by perturbing inputs around an instance and fitting a sparse linear approximation, yielding intuitive “if-then” rules (for example, “if rail temperature rises above threshold in segment i , then fault likelihood $> p$ ”). When applied collectively to multimodal track and road sensor arrays, these methods not only reveal the DNN’s focal points but also ensure that flagged anomalies correspond to genuine mechanical irregularities—thereby accelerating root-cause analysis and reinforcing trust in automated monitoring systems [54].

3.5.3 Adversarial Robustness and Model Assurance

Even small, undetectable changes to sensor inputs can fool a DNN in a safety-critical railway fault detection, causing it to miss a genuine issue or mark a healthy track as defective. By subjecting models to carefully constructed variations or by offering verifiable limitations on output deviations, adversarial robustness techniques—such as adversarial training, input sanitisation, and certified defenses—harden models against these worst-case manipulations. Formal verification and stringent testing procedures that measure a network’s failure modes under both hostile and random noise are also included in model assurance. When combined, these techniques guarantee that automated fault-detection systems uphold safety and dependability assurances even when confronted with deliberate or unintentional data irregularities.

3.6 Data Pipelines for Railway Road Safety

3.6.1 Data Acquisition: Sensors, Cameras, Track Logs

Collecting rich, multimodal data throughout the train line is the first step towards reliable failure detection. Small oscillations and frequency shifts are detected by embedded vibration and sound sensors on axle boxes and rail junctions, which detect fatigue or breakage early. To detect surface flaws, hot spots, or gauge misalignments in real time, thermal and high-resolution optical cameras installed on inspection vehicles or placed wayside offer both visual and infrared imagery. Axle loads, train position, speed, and maintenance history are recorded via digital track logs, which also synchronise these sensor inputs with operating context. Combining these complimentary inputs into a deep neural network gives the system a comprehensive, time-aligned perspective on rail health, which promotes the detection of anomalies with sensitivity and context awareness[55].

3.6.2 Preprocessing Annotation: Cleaning, Labeling Near-Miss Events

In order to eliminate spikes, dropouts, and timestamp misalignments, raw sensor feeds are first cleaned. Interpolation and noise filtering balance multimodal inputs and fill in gaps. A standardised fault-precursor taxonomy is then used by experts to categorise near-miss situations based on synchronised logs, vibration traces, and image/video/thermal clips. The generated annotated dataset is split and balanced to facilitate robust training and assessment of the DNN’s capacity to differentiate between imminent failures and safe operation.

3.6.3 Augmentation Techniques for Rare-Event Detection

Training dependable detectors requires enlarging the minority class because real fault situations are rare in comparison to data from regular operations. Limited anomalous vibration, images or acoustic signals are diversified by time-series manipulations like jittering, scaling, cropping, and time-warping, and new examples are produced by SMOTE-style oversampling, which interpolates between rare-event feature vectors. By discovering the underlying distribution of defect signatures, generative adversarial networks are able to further synthesise realistic fault traces. Contextual realism is added by physics-informed augmentation, which involves inserting simulated crack or wear waveforms onto sound sensor recordings. When combined, these techniques add a variety of high-fidelity anomalies to the training set, allowing deep models to fully capture the range of rare-event patterns [56].

3.7 Deep Learning Framework for Automated Accident Detection

3.7.1 System Architecture: Edge vs. Cloud Processing

When designing an automated accident detection system, real-time responsiveness must be balanced with data management and computing complexity. Edge processing

reduces bandwidth by sending just alarms or compressed features, improves privacy by storing raw data locally, and provides millisecond-level inference by directly embedding lightweight deep models on cameras, sensors, or trackside devices. On the other hand, cloud processing uses massive GPU clusters to run multimodal, deeper networks and aggregate massive logs for continuous learning, but it has higher latency and requires dependable connectivity. Low-latency detection and the full analytical power of remote infrastructures are often married in a hybrid topology, where edge nodes handle immediate anomaly screening and emergency alerts while the cloud conducts in-depth analysis, long-term pattern mining, and periodic model refinement.

3.7.2 Real-Time Inference: Latency, Throughput, and FAIL-Safe Designs

Automated accident detection requires precise timing, which means maintaining high throughput under sensor bursts while maintaining low-millisecond per-frame inference latency—and ensuring safety when things go wrong. Model pruning, quantisation, and operator fusion on edge accelerators are some of the techniques that help reduce execution time, and micro-batching or asynchronous pipelines make sure that incoming data spikes don't overwhelm the system. Simultaneously, watchdog timers and runtime monitors keep track of hardware malfunctions or processing delays, immediately switching to a compact backup model or initiating pre-programmed safe actions (such as alert broadcasts or emergency braking requests) if deadlines are missed or parts fail. In the face of real-world uncertainty, the system maintains responsiveness and reliability by fusing layered fail-safe techniques with ultra-fast inference optimisations [57].

3.7.3 Feedback Loops: Online Learning from New Incidents

When a railway defect detector is deployed in the field, inevitably, new anomaly patterns and near-miss incidents that were not seen during offline training will be discovered. In order to adapt to changing track conditions and failure types, the network continuously improves its decision boundaries by applying lightweight online updates, such as mini-batch fine-tuning or elastic weight consolidation, and streaming labelled event data back into the system. Human-in-the-loop approval, validation hold-out sets, and rehearsal buffers are examples of safeguards that help maintain model integrity and avoid catastrophic forgetting. This closed-loop workflow speeds up the system's capacity to learn from uncommon or new defect signals in real time while also keeping the detector in line with the most recent operational realities [58].

3.8 Comparison Models Overview

We have used 3 different state-of-the-art models for image classification in our research to evaluate our dataset and compare the obtained results with our proposed model. The briefings of the comparison models in terms of their architectures are discussed below.

3.8.1 Inception-V3

A model made by researchers at Google, which is a modified version of the inception architecture [59]. Despite the model consisting of 42 layers, the computational cost is only 2.5 times greater than GoogleNet. In image classification in ILSVRC, it has become the 1st runner-up, which proves its efficacy. The Inception V3 model was trained with the dataset used in this research, and we determined its accuracy in both training and validation.

3.8.2 Xception

Depthwise Separable Convolutions are used in the deep CNN architecture known as “Xception”. In Inception, the original input is compressed using 1×1 convolutions, and each depth space is then given a new set of filters based on the input spaces. Just the opposite occurs with Xception. Before compressing the input space using a 1×1 convolution by applying it across the depth, it first applies the filters to each of the depth maps. We used Xception, as it does not introduce any non-linearity.

3.8.3 InceptionResNetV2

It is a deep convolutional neural network having 164 layers. Input images having dimensions of $[229 \times 229]$ are fed into the model that is passed through a convolutional block with output dimensions of 3×3 having a stride 32 followed by another convolutional layer having the same dimensions and stride. The next layer has a stride of 64 that is fed into a convolutional layer and a maxpool layer. The resulting layer is passed through 2 consecutive layers of convolutional neural networks layers having dimensions of 1×1 with a depth of 64 and 3×3 with a depth of 96 respectively and through 4 layers of the convolutional layer having dimensions 1×1 with a depth of 64, 7×1 with a depth of 64, 1×7 with a depth of 64 and 3×3 with a depth of 96. Both the emerging layers are passed through a filter concat, and eventually, the resulting output layers are passed through a 3×3 convolutional layer and a maxpool layer having strides of 2. Final output layers are passed through a filter concat.

Summary

This chapter provided the necessary theoretical background to understand the components and functioning of ECARRNet, including convolutional layers, BiLSTM, and dense networks. Building on this foundation, the next chapter details the methodology adopted to design, implement, and evaluate the proposed ECARRNet framework.

Chapter 4

Methodology

In this chapter, we demonstrate our end-to-end railway-fault pipeline as shown in Figure 4.1. Raw high-resolution images are first acquired using DSLR cameras (see Section 4.1) and then preprocessed using techniques such as ROI cropping, scaling, offline and online augmentation, creating 3 versions of the dataset, dataset splitting using repeated stratified K-fold shuffle splitting, and class imbalance problems using SMOTE (see Section 4.2). These preprocessed frames are fed into ECARRNet (see Section 4.3), which outputs a binary ‘defective’/‘non-defective’ decision. ECARRNet comprises four core sub-modules:

- **Convolutional Autoencoder:** Denoises inputs and compresses feature maps for more efficient downstream processing.
- **ResNet-based Transfer Learning:** Leverages ImageNet-pretrained weights to extract robust spatial features.
- **Bidirectional LSTM:** Wraps the CNN backbone in a bidirectional LSTM to capture continuity and dependency across adjacent frames.
- **Fully Connected Layers + Sigmoid Head:** Collapses the aggregated features into a single probability in $[0,1]$, suitable for binary classification.

Together, these modules form a robust ensemble that balances denoising, pretrained feature extraction, temporal modeling, and discriminative classification for accurate railway track defect detection.

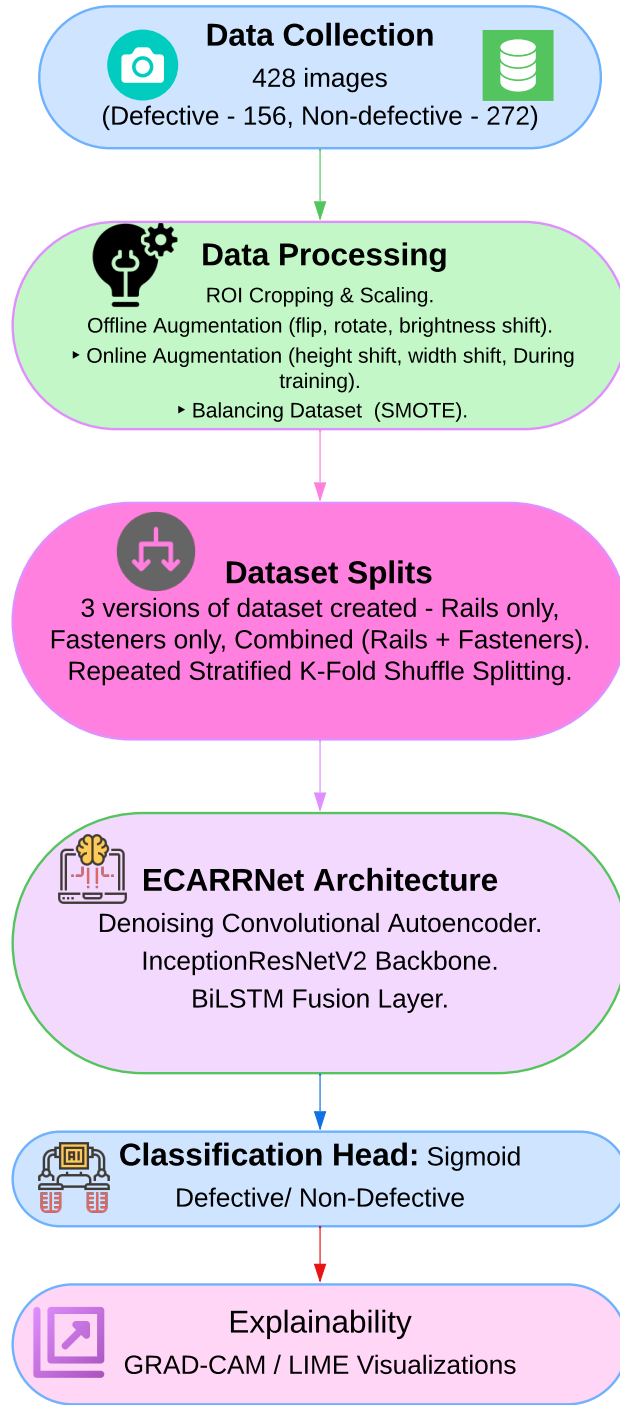


Figure 4.1: End-to-End Railway Fault Detection System

4.1 Data Collection

We have collected our dataset by visiting railway tracks in different locations across Bangladesh [60]. Since the railway tracks were constructed many years earlier, only faults for meter gauge and broad gauge lines were found. We went to the railway tracks at Jessore and Feni and took photographs from different places in the city for

several days. We went to stations and junctions collecting data from old, new, and abandoned rail tracks. We have also considered abandoned tracks because we can not find all kinds of faults in regular tracks. We have mainly found faults in rails and fasteners, which are one of the major components of railway tracks and are usually responsible for causing drastic accidents. To make a statistically significant dataset, we could only use faults of sleepers and fasteners images for training our model. We have collected 428 images in total, containing 156 defective and 272 non-defective images. We tried to take images from different angles for each type of error. Some of the sample images of our initial dataset are shown in Figure 4.2.



Figure 4.2: Samples of faults on railway tracks

4.2 Data Pre-processing

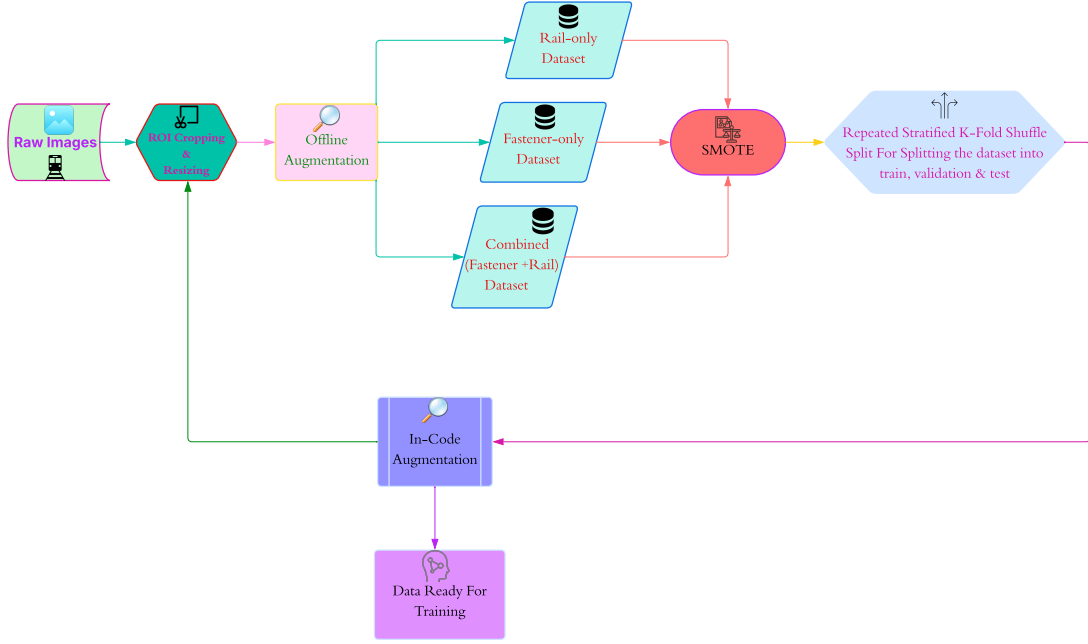


Figure 4.3: Data Preprocessing Workflow

At the very beginning of the pre-processing step, we cropped each image according to the region of interest. Then the images are resized to a final input size of 300×300 . Then, We have done four types of offline augmentation, and thus the number of images increased to four times the initial number, i.e., up to 1712 images. We used rotation, vertical flip, horizontal flip, and brightness shift as parameters in this case as shown in Table 4.1. The complete data preprocessing workflow is shown in Figure 4.3, and an example of offline augmentation is shown in a single image in Figure 4.4.

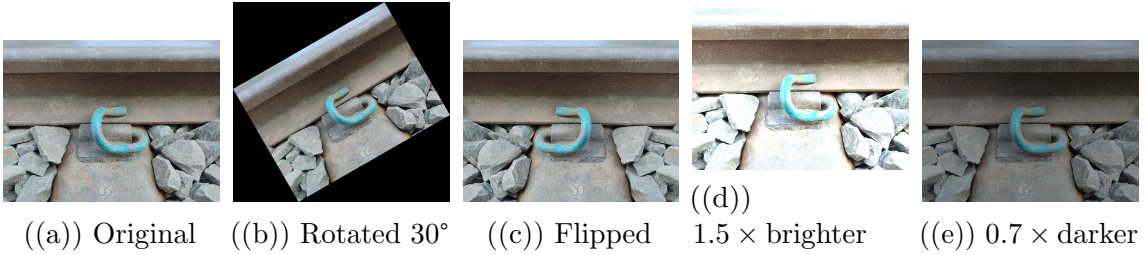


Figure 4.4: Example of offline augmentations applied to a single rail image.

Then we have prepared 3 different versions of the dataset to train and experiment to find which way of arranging the data would work best. In the 1st version, we used only images of rails that contained defective and non-defective classes of images from faults in rails. In the 2nd version, we have defective and non-defective images from fasteners [61] only, and in the last and final version, we have defective and non-defective images from both rails and fasteners combined. This makes our experiment and proposed model more reliable, as it is repeated multiple times in multiple scenarios to produce similar results. On the other hand, we have also faced a class imbalance problem,

as we could not find faulty images in adequate numbers on the railway tracks. As a result, we found more images of railway tracks that are non-defective or with no faults. Therefore, we had to apply an oversampling technique called SMOTE (Synthetic Minority Oversampling Technique), which is a very successful technique for class imbalance problems. SMOTE creates copies of the images for the class that is a minority or has a lesser number of images. SMOTE does this by producing data points on the same lines, which connect a point and one of its KNNs [62]. We would also like to mention that we have used shuffling methods such as repeated stratified K-fold shuffle splits to ensure that both of our classes are in equal quantity in the train and test splits. This ensures that our data is not randomly sampled in train, validation, and test splits. Moreover, we have also used the ImageDataGenerator class from Keras for in-code or online augmentation to multiply the number of images in our dataset. Online augmentation using TensorFlow and Keras increases the size of the dataset during the training process without occupying any physical memory. In this case, we have used rescale, width shift range, height shift range, vertical flip, and channel shift range as parameters, as shown in Table 4.2. This increased the dataset to five times the original size on the fly during training. As a result, this allows us to use up to 8000 images during training. As a result, this makes our experiments more valid by reducing sampling bias and increasing the validity of our experiments.

Transform	Parameter
Rotation	$\pm 15^\circ, \pm 30^\circ$
Horizontal flip	50% chance
Vertical flip	50% chance
Brightness shift	[0.8, 1.2]

Table 4.1: Offline augmentations parameters applied to expand the dataset.

Table 4.2: In-code augmentation parameters (ImageDataGenerator)

Transform	Parameter
Rescale	<code>rescale=1/255</code>
Width shift range	<code>width_shift_range=0.3</code>
Height shift range	<code>height_shift_range=0.2</code>
Vertical flip	<code>vertical_flip=True</code>
Channel shift range	<code>channel_shift_range=15</code>

Justification for Tool Choices

Several tools and techniques were selected throughout this research to address the specific needs of railway fault detection using ECARRNet. The SMOTE (Synthetic Minority Over-sampling Technique) was used to address the class imbalance in the dataset, where certain fault types were under-represented. SMOTE was chosen over simpler resampling methods because it generates synthetic examples rather than

duplicating existing data, which helps avoid overfitting and preserves the diversity of the training set.

In terms of data augmentation, both offline and online techniques were implemented to improve model generalization. Techniques such as rotation, zooming, and horizontal flipping were applied to simulate real-world variability in track imagery. These were chosen because they are computationally inexpensive and widely validated in image classification tasks. Online augmentation using libraries like ImageDataGenerator allowed augmentation during training, ensuring efficiency and variety without increasing the storage burden.

Overall, the selected tools and techniques were chosen based on their alignment with the research objectives, practical effectiveness, and proven success in similar deep learning contexts.

4.3 ECARRNet Module

We propose a new ensembled convolutional autoencoder ResNet-based recurrent neural network (ECARRNet) deep architecture for railway fault detection, comparing our results with conventional models to demonstrate the potential of RNN-based models in image classification. Moreover, there are several major advantages to using a CNN-RNN ensemble model, which also outperforms conventional CNN models. Firstly, we are using convolutional autoencoders to reduce the dimensionality of images and to extract important features from the data. In the next stage, our proposed model contains a ResNet-based transfer learning architecture trained on the ImageNet dataset so that we can use the useful network weights in our model to solve the problem of a lack of enough data, which can reduce our performance. Thirdly, we are also using an LSTM layer, which has memory cells instead of just recurrent units to store and output information with the help of gating mechanisms and thus extend the capability of RNN. Lastly, we are using a few layers of fully connected neural network layers before our final binary classification layer. We are also giving special attention to reducing vanishing or exploding gradient problems, as our proposed architecture is quite deep. We are making use of initialisation parameters such as He kernel initializer to initialise weights, which would play a key role in reducing vanishing or exploding gradients. In addition, batch normalisation layers also contribute to reducing vanishing gradient problems and act as a regularisation parameter. The architecture of the proposed model is described in detail below in Figure 4.5 and in table 4.3 is provided that gives an overview of the model in terms of its layers and their types.

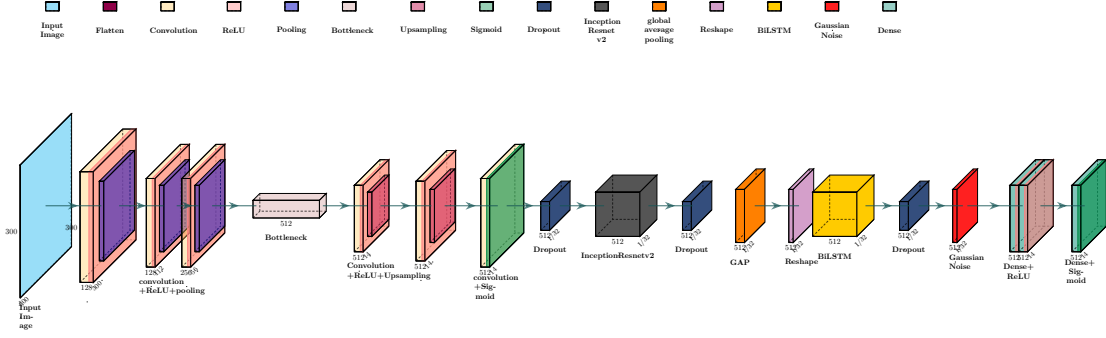


Figure 4.5: Architecture of the Proposed ECARRNet Model

Table 4.3: Model architecture summary

Layer Type	Number of Units/Neurons
Input	-
Conv2D	128
MaxPooling2D	-
Conv2D	64
MaxPooling2D	-
Conv2D	64
MaxPooling2D	-
Conv2D	64
UpSampling2D	-
Conv2D	128
UpSampling2D	-
Conv2D	1
InceptionResNetV2	-
GlobalAveragePooling2D	-
Reshape	-
Bidirectional LSTM	2900
GaussianNoise	-
Dense	100
Dense	100
Dense	1

4.3.1 Convolutional Autoencoder

An autoencoder is an unsupervised neural network model that produces output data with less noise and distortion. It is commonly used in data compression tasks and also reduces the storage space of data, improves the training time, and discards variables that are redundant. On the other hand, the convolutional neural network is also great for extracting useful features from images. So, we are using convolutional layers to improve the performance of a simple auto-encoder. We have also experimented with only a convolutional neural network in the first stage of our model and found that the convolutional auto-encoder gives better performance. Simple Autoencoders mainly have 3 parts: an encoder, a code, and a decoder. At first, the input passes the

encoder to produce the code, where the encoder is a fully connected neural network. Then, the decoder produces an output only using the code, where the decoder is also a fully connected artificial neural network. The objective of this step is to get an output that is identical to the input. The decoder architecture often gives a mirror image of the encoder. In an auto-encoder, the dimensionality of the input and output should always be the same. Convolutional auto-encoder adds specialty to the basic structure of a simple auto-encoder by replacing the fully connected layers in the encoder and decoder portions with convolutional layers. The size of the input layers will also be equal to the size of the output layers, similar to a simple auto-encoder. The decoder network in the convolutional autoencoder also transforms itself into transposed convolutional layers. Along with convolutional layers, max-pooling layers are also included in the encoder and decoder networks. Convolutional autoencoders have many use cases, such as image compression, image denoising, etc [63], [64]. In our proposed architecture, we have experimented with various encoder and decoder architectures to determine the optimal configuration. Our encoder architecture has 3 convolutional layers and 3 max-pooling layers, and the layers are further strengthened with the use of batch normalization and dropout layers. The first convolutional layer has 128 neurons with a 6×6 filter size and ReLU as the activation function. We are also using a special kernel initializer called normalization to reduce the vanishing gradient. All the max-pooling layers consist of a pooling size of 2. In the next two convolutional layers, we are using 2D convolutional layers with 64 neurons, each with a 4×4 filter size. In the decoder portion, the first 2D convolutional layer has 64 neurons, and it also has several 2D Upsampling layers with a filter size of 2×2 to reconstruct the same output dimension as the input. The next convolutional layer has 128 neurons, and the last one has only one neuron with a sigmoid activation function. All the convolutional layers in the decoder have a filter size of 6×6 . In the convolutional auto-encoder part and our overall proposed model, we have used batch normalization layers frequently, and almost in between every layer. This is because, in the beginning, our training was slow as the data set and the model were relatively big and prone to overfitting. After using batch normalisation extensively, our training time improved significantly, and due to the regularisation effect of the batch normalisation, it played a key role in producing state-of-the-art accuracy. According to [65], while training a deep neural network, the distribution of each layer’s input changes during training as the parameters of the previous layers change. As a result, training slows down and needs lower learning rates and much more careful parameter initialization. Thus, it becomes difficult to train models with saturating nonlinearities, which are known as internal covariate shifts. To solve this problem, we can normalize layer inputs for each training mini-batch, as this allows the use of higher learning rates while being less cautious about initialization. Batch normalization is capable of achieving the same accuracy with 14 times fewer training steps and significantly beats the model that did not use batch normalization at all.

4.3.2 ResNet-Based Transfer Learning

In this stage, we are using a ResNet-based transfer learning architecture, which is trained on the imagenet dataset. In transfer learning, we apply the expertise acquired by a model from a task with abundant training data to improve performance on a new task with limited data availability. In our problem, since we did not have so much

initial data, we had to take the help of transfer learning to leverage what they have already learned in one task to enhance the ability to generalize in another context. Therefore, we are using a pre-trained model known as inception-ResNet-v2, which performed better than other pre-trained networks such as inceptionv3, Xception, etc., in our dataset. Inception architecture has already been proven through many research works to give good performance at a low cost comparatively. Moreover, it was found that training with residual inception networks speeds up the training of inception networks largely. Residual connections in resnet consist of convolutional layers along with relu activation functions. In Inception-ResNet, batch normalization is used only on top of the traditional layers, but not on top of the summarizations [66].

4.3.3 Recurrent Neural Networks

This stage explains the unique portion of our proposed model. In this part, we use recurrent neural networks, which are artificial neural networks in which connections between individual units form a directed cycle. RNN can model the dynamical behavior of sequences with arbitrary lengths. In our proposed architecture, along with the convolutional layers in the Inception-ResNet-v2 described above, it thus creates a combined CNN-RNN framework as the data reaches the RNN layer after passing the CNN layers. The CNN-RNN framework for image classification has many benefits [67]. In general, the recurrent neural network is more suitable for modeling sequential data. In the CNN-RNN model, RNN generates sequential predictions for various timesteps using the output of CNN as input. Image data is often considered two-dimensional wave data in which convolution layers act as a filtering process. Convolutional layers can filter simple band information in an image, but often leave behind important features of image information. Therefore, using an RNN layer is important because the CNN-RNN framework can make use of RNN to calculate the dependency and continuity features of the intermediate layer output of the CNN model. It also allows the connection of the characteristics of these middle tiers to the full connection network for classification prediction, which thus leads to a better classification accuracy [68].

However, because the gradient must pass through numerous layers of the RNN, it creates long-term dependency and thus frequently experiences vanishing and exploding gradient issues, making it challenging to model. To solve this problem, we are using Long-Short Term Memory (LSTM) in our proposed model. LSTM consists of memory cells to encrypt knowledge at each instant of time. Three gates—an input gate, a forget gate, and an output gate—control how the memory cell behaves. To be more precise, memory cells of LSTM form a cell state that passes relevant information from earlier time steps to later time steps, thereby reducing the effect of short-term memory. As the cell state carries information throughout the sequence, information is added or removed by the gates. Gates are independent neural networks for deciding which information should be allowed or discarded from the cell state. On the other hand, gates consist of sigmoid activation functions. They give output values between 0 and 1 when passed through them. This principle is used by different gates. Forget gate makes decisions about which information should be kept or discarded. As the information from the previous hidden state crosses the current input, it passes through the sigmoid function. The sigmoid function produces a value between 0 and

1. If the value is closer to 0, it is forgotten, and if it is close to 1, then it has to be remembered by the cell state. Input gates are used to update the cell state. Finally, output gates are used, which decide which hidden state to use. Therefore, these gates assist the input signal to pass through the recurrent hidden states by retaining useful information in the long run and removing unnecessary redundant information. In this way, LSTM is well capable of solving the vanishing and exploding gradient problem and can model long-term temporal dynamics pretty well, which RNN cannot. In our paper, we are using a bidirectional LSTM layer with 2900 neurons, which helps in forming the CNN-RNN architecture, which has several advantages as mentioned above. We have also experimented with another gated version of RNN called GRU, but a bi-directional LSTM with 2900 neurons performed better in our dataset.

4.3.4 Fully Connected Layers

Lastly, our ensemble deep network consists of some fully connected layers as well. After the LSTM layer, we add 2 more dense or fully connected layers consisting of 100 neurons each. Finally, we have some dropout regularizers, along with batch normalization layers before our last layer. The last layer is a dense layer with a sigmoid function for binary classification. Lastly, we want to briefly discuss some essential parts to demonstrate the unique feature of our proposed model, which allows us to produce such good performance. We are using a batch size relatively low due to a lack of computational resources, typically less than 10. Dimensions of images used are typically 300 by 300, but our experimentation shows 500 by 500 and often 600 by 600 images produce the best result. We are using the Adamax optimizer as it can better optimize than Adam. We used binary cross-entropy as our loss function.

Summary

This chapter details the proposed ECARRNet architecture, including data preprocessing (e.g., ROI cropping, online augmentation), model design, and training strategy. It presented a step-by-step breakdown of the experimental pipeline and discussed the datasets used (rails, fasteners, combined). Visual workflows were also provided to aid understanding. With the experimental setup in place, the next chapter presents and analyses the results obtained from extensive evaluations on multiple datasets.

Chapter 5

Performance Evaluation

5.1 Performance Metrics

We are using accuracy as our performance metric, and we are also testing the performance of our model using metrics like classification reports and confusion matrices. Our classification report has four metrics in order to measure the effectiveness of each of the models. The metrics include Support, Precision, F1 score, and recall. Support can be calculated by summing the last rows in the confusion matrix. Support is the number of occurrences of each particular class in the true responses which means that it shows the number of actual occurrences of the class in the dataset.

$$Precision = \frac{TP}{TP + FP} \quad (5.1)$$

$$Recall = \frac{TP}{TP + FN} \quad (5.2)$$

$$F1 = \frac{2 \times Precision \times Recall}{Precision + Recall} = \frac{2 \times TP}{2 \times TP + FP + FN} \quad (5.3)$$

In the above set of equations,

- TP = True Positive
- FP = False Positive
- FN = False Negative
- TN = True Negative

AI models are usually considered black-box models, and they lack transparency. Transparent models can explain why they predict and what they predict. It thus builds trust in the model and makes sure that they are predicting only the desired object in a way that is explainable to researchers. The objective of transparency is to find out cases in which the model has failed substantially. There are several approaches and frameworks available to implement explainable AI, which can produce visual demonstrations to provide a better understanding of what a model predicts on new samples of data and how they do it. In the case of images, patches are produced to indicate the location on which the model is focusing. One of the most popular approaches to providing LIME (Local Interpretable Model-Agnostic Explanations)

is a revolutionary explanation method to faithfully and comprehensibly explain the predictions of any classifier. We are using LIME in our paper to explain the performance of our proposed model, as we could not find any other available solution to explain our custom model. Additionally, we are using Grad-CAM which produces heat-maps on the location of the image where the model is focusing [69]. We are using Grad-CAM to obtain the model explainability of the models, which are used to compare with our proposed architecture. LIME is an algorithm that has the capability to faithfully explain the predictions of any classifier and regressor by localising it with an understandable model [70]. The results produced by both Grad-CAM and LIME are shown in the experimental results section.

$$L^{cGradcam} = ReLU(\sum_k \alpha_k^c A^k) \quad (5.4)$$

Equation 5.4 shows the equation of Grad-CAM, which is a weighted linear combination of feature maps followed by a ReLU function. Where,

- A^k = feature map activation
- α_k^c = neuron significance weight

5.2 Experimental Results

We conducted extensive experiments on our railway dataset to measure ECARRNet’s real-world effectiveness. We benchmarked our model against Xception, InceptionV3, and InceptionResNetV2, measuring overall accuracy, precision, recall, false positive/negative rates, and F1 scores. We utilised explainable AI to make the models more transparent. The following outcomes show how well ECARRNet detects and how prepared it is for deployment in real-time, safety-critical scenarios.

5.2.1 Results Generated on Rail Dataset

Figure 5.1 displays the set of accuracy curves for the implemented algorithms. The curves indicate that our proposed model, ECARRNet, outperforms the other models in terms of peak accuracy and average accuracy.

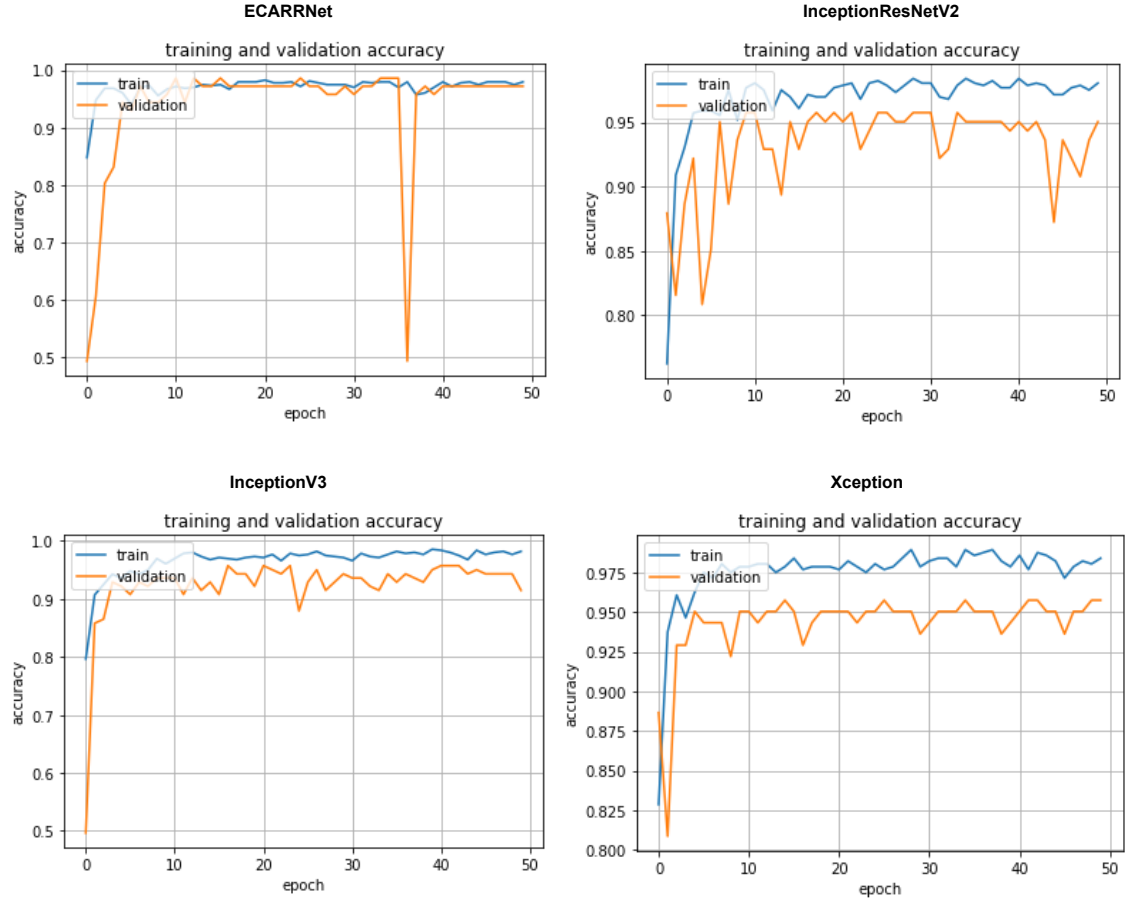


Figure 5.1: Accuracy curves for each of the models on the rail dataset

Figure 5.2 exhibits the confusion matrices for the models on the rail dataset. The results show that ECARRNet produces fewer false positives compared to InceptionV3 and produces similar and comparative results when compared to InceptionResNetV2 and Xception, respectively.

Figure 5.3 portrays the Grad-CAM visualisations for the same image selected randomly from the rail dataset for the compared models. The region of the heatmap generated using the InceptionResNetV2 model classifies the incorrect region in the image where there is no fault as the red region, which represents the highest contribution of the model for classification, is not in the defective portion of the track. InceptionV3 and Xception, on the other hand, produce heatmaps that comparatively fall much closer to the region of faulty rail tracks.

From table 5.1 we can see Xception performs better than the other models when compared on the RAIL dataset. It has the best F1 score (0.96) for both defective and non-defective classes, as well as the maximum precision and recall (0.96). With equal F1 scores of 0.95, InceptionResnetV2 and ECARRNet exhibit very competitive performance, lagging Xception by a small margin. Even though it is still a powerful model, InceptionV3 performs somewhat worse than the others, with the lowest F1 score of 0.91. All things considered, Xception is the top model for this particular dataset, exhibiting the best balance in accurately classifying defective and non-defective cases.

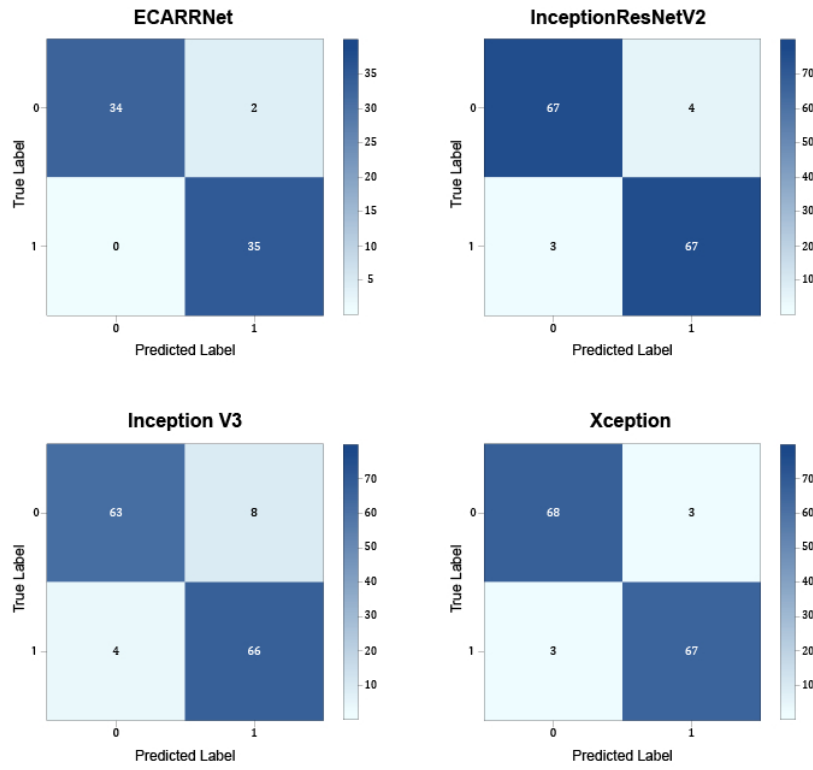


Figure 5.2: Confusion matrices for each of the models on the rail dataset

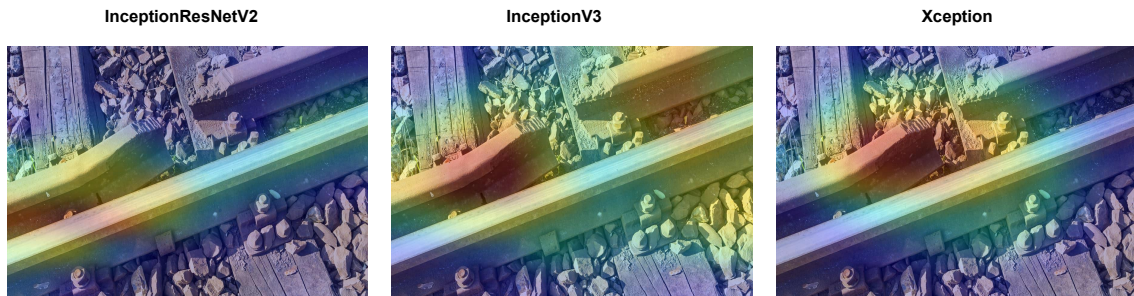


Figure 5.3: Grad-CAM visualizations for transfer learning models on the rail dataset

Table 5.1: Classification Report for Rail Dataset

Models	Metrics	Defective	Non De- fective	Accuracy	Macro Average	Weighted Average
Inception- V3	Precision	0.89	0.94	-	0.92	0.92
	Recall	0.94	0.89	-	0.92	0.91
	F1 Score	0.91	0.92	0.91	0.91	0.91
	Support	67	74	141	141	141
Inception- ResnetV2	Precision	0.94	0.96	-	0.95	0.95
	Recall	0.96	0.94	-	0.95	0.95
	F1 Score	0.95	0.95	0.95	0.95	0.95
	Support	70	71	141	141	141
Xception	Precision	0.96	0.96	-	0.96	0.96
	Recall	0.96	0.96	-	0.96	0.96
	F1 Score	0.96	0.96	0.96	0.96	0.96
	Support	71	70	141	141	141
ECARRNet	Precision	0.94	0.96	-	0.95	0.95
	Recall	0.96	0.94	-	0.95	0.95
	F1 Score	0.95	0.95	0.95	0.95	0.95
	Support	70	71	141	141	141

5.2.2 Results Generated on Fastener Dataset

In Figure 5.4, the accuracy curves of the models based on the fastener dataset are displayed. Once again we see our ECARRNet model outperforming the other transfer learning models. InceptionV3 and Xception produce results that are comparable to ECARRNet, while InceptionResNetV2 performs very poorly on the fastener dataset.

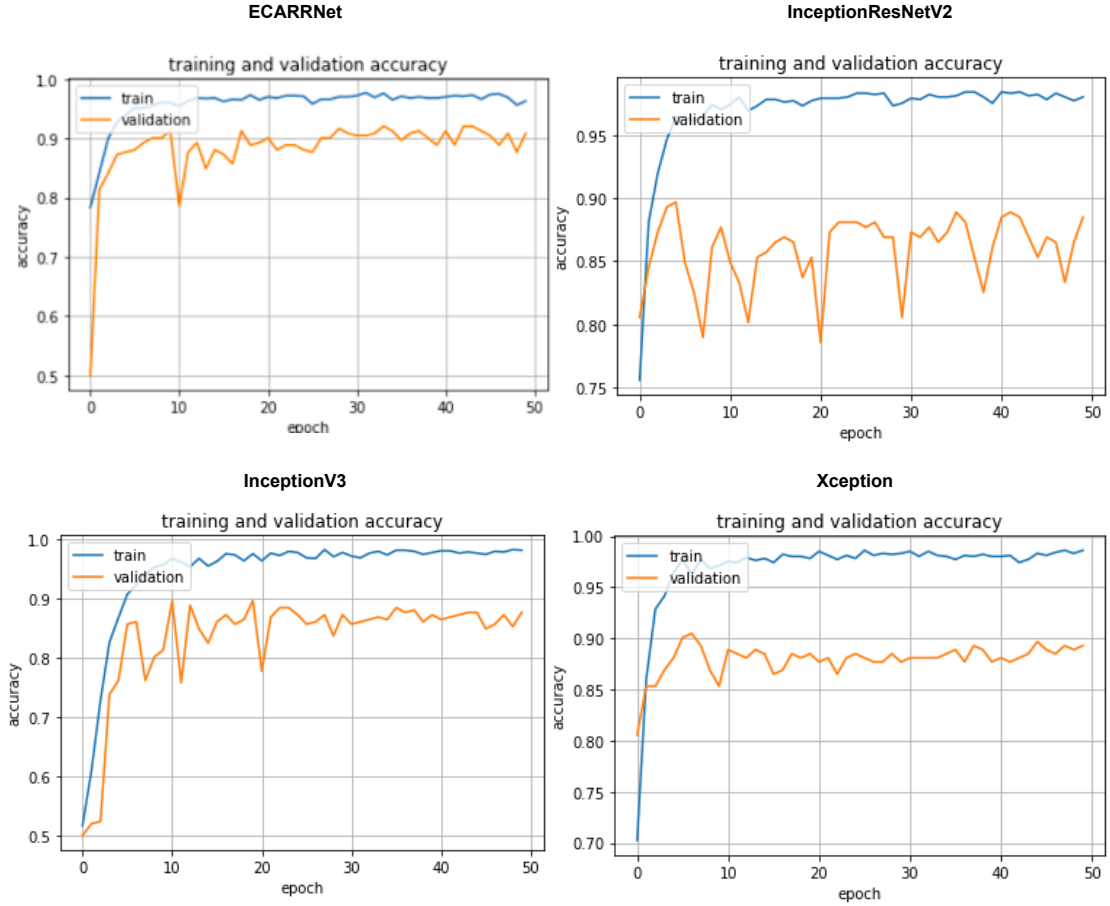


Figure 5.4: Accuracy curves for each of the models on the fastener dataset.

Figure 5.5 exhibits the confusion matrices for the models on the fastener dataset. The results show that ECARRNet produces fewer false positives compared to InceptionV3 and produces similar and comparative results when compared to InceptionResNetV2 and Xception, respectively.

Figure 5.6 portrays the Grad-CAM visualisations for the same image selected randomly from the fastener dataset for the compared models. The region of the heatmap generated using the InceptionResnetV2, InceptionV3, and Xception models produces heatmaps that classify areas somewhat close to the faulty fasteners on all images. However, the red region, which indicates a high contribution to classification, does not specifically focus on the faulty fastener portion which is faulty.

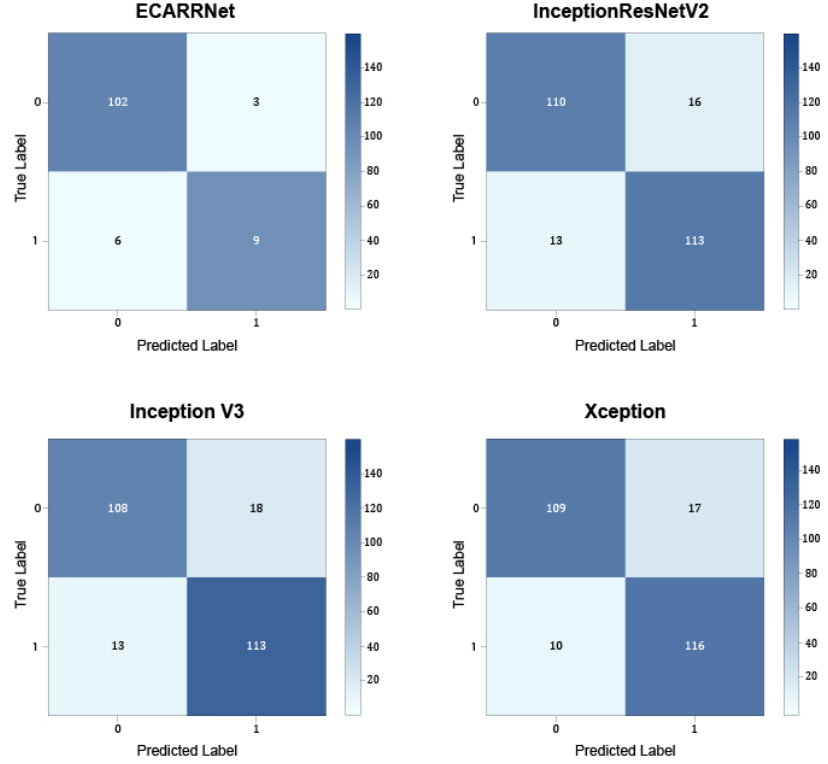


Figure 5.5: Confusion matrices for each of the models on the fastener dataset.

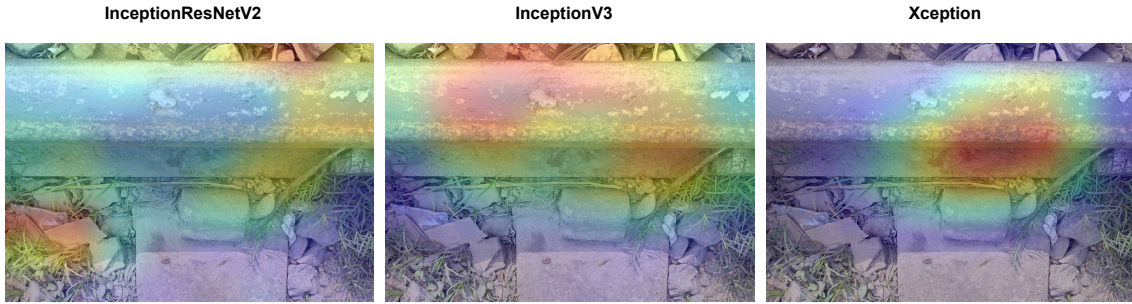


Figure 5.6: Grad-CAM visualisations for transfer learning models on the fastener dataset.

Table 5.2 with the highest F1 ratings of 0.89 for both the defective and non-defective categories, Xception shines out once more and suggests a balanced precision and recall. With an F1 score of 0.87, InceptionV3 comes in second with a good recall and slightly less precision. With an F1 score of 0.87, InceptionResnetV2 and ECARRNet perform identically across all criteria. Overall, on this dataset, Xception performs similarly and consistently to the other models, with the exception of a minor edge in correctly categorizing defective and non-defective items.

Table 5.2: Classification Report for Fastener Dataset

Models	Metrics	Defective	Non De- fective	Accuracy	Macro Average	Weighted Average
Inception- V3	Precision	0.84	0.91	-	0.88	0.88
	Recall	0.91	0.85	-	0.88	0.88
	F1 Score	0.87	0.88	0.88	0.88	0.88
	Support	117	135	252	252	252
Inception- ResnetV2	Precision	0.88	0.87	-	0.87	0.87
	Recall	0.87	0.88	-	0.87	0.87
	F1 Score	0.87	0.87	0.87	0.87	0.87
	Support	128	124	252	252	252
Xception	Precision	0.87	0.92	-	0.89	0.89
	Recall	0.92	0.87	-	0.89	0.89
	F1 Score	0.89	0.90	0.89	0.89	0.89
	Support	119	133	252	252	252
ECARRNet	Precision	0.87	0.87	-	0.87	0.87
	Recall	0.87	0.87	-	0.87	0.87
	F1 Score	0.87	0.87	0.87	0.87	0.87
	Support	126	126	252	252	252

5.2.3 Results Generated on Full Dataset

In Figure 5.7, the accuracy curves of the models based on the full dataset are displayed which includes defective and non-defective images from both fastener and rail. Once again, we see our ECARRNet model outperforming the other transfer learning models. InceptionV3 and Xception produce results that are comparable to ECARRNet, while InceptionResNetV2 performs very poorly on the full dataset.

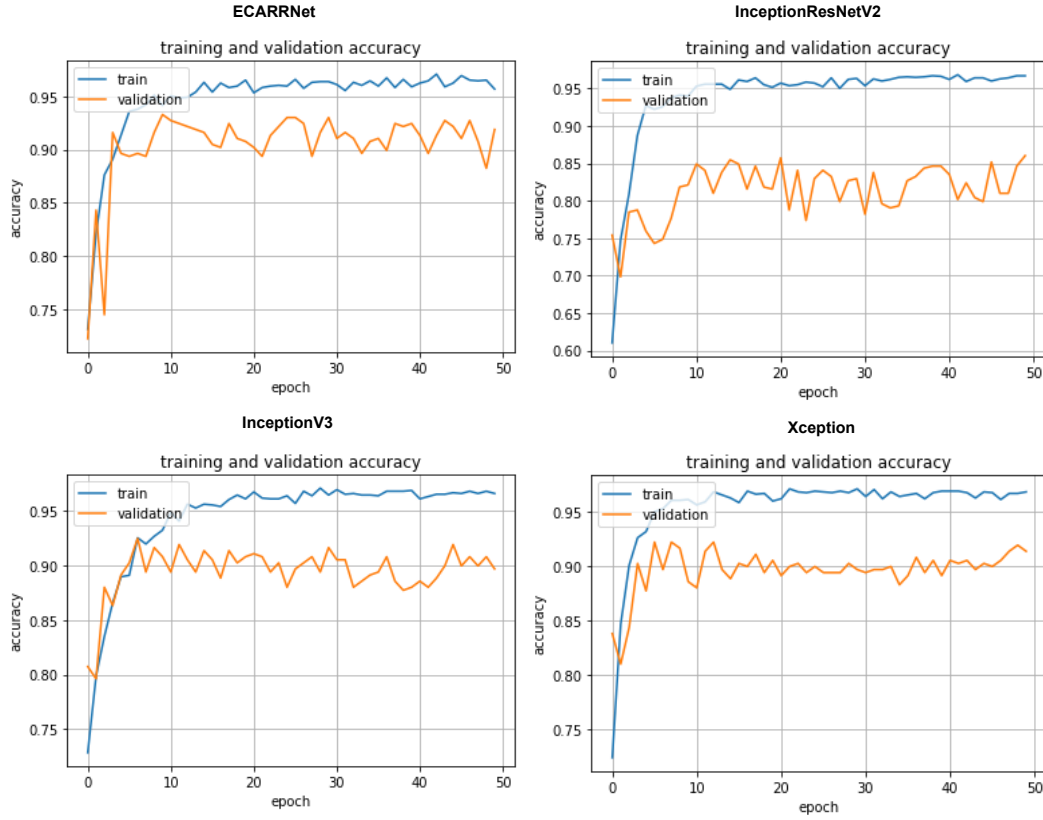


Figure 5.7: Accuracy curves for each of the models on the full dataset.

Figure 5.8 shows the confusion matrices for the models on the full dataset containing defective and non-defective images from both fastener and rail. The results show that ECARRNet produces fewer false positives compared to InceptionV3 and produces similar and comparative results when compared to InceptionResnetV2 and Xception, respectively.

Figure 5.9 shows the Grad-CAM visualizations for the same image selected randomly from the full dataset which contains images of faults from both fastener and rail for the compared models. The region of the heatmap generated using the InceptionResnetV2, InceptionV3, and Xception models produces heatmaps that classify a region that is somewhat close to the faulty fasteners on all images, but the red region, which shows high contribution for classification, does not exactly focus on the fault/gap portion in the rail, which is faulty.

In Table 5.3 with the best recall for non-defective items and the highest precision for defective items throughout the whole dataset, ECARRNet performs exceptionally well, earning an astounding F1 score of 0.92 for both categories. Xception comes in second with an F1 score of 0.91, exhibiting balanced recall and precision in both areas. With an F1 score of 0.90, which is competitive but marginally lower than InceptionV3, the findings are strong. At 0.86 for F1, InceptionResnetV2's score is marginally lower than the others'. The macro and weighted averages agree with the F1 scores, indicating that ECARRNet is the best-performing model in this assessment, followed closely by Xception and InceptionV3, and InceptionResnetV2 being the least successful of the four.

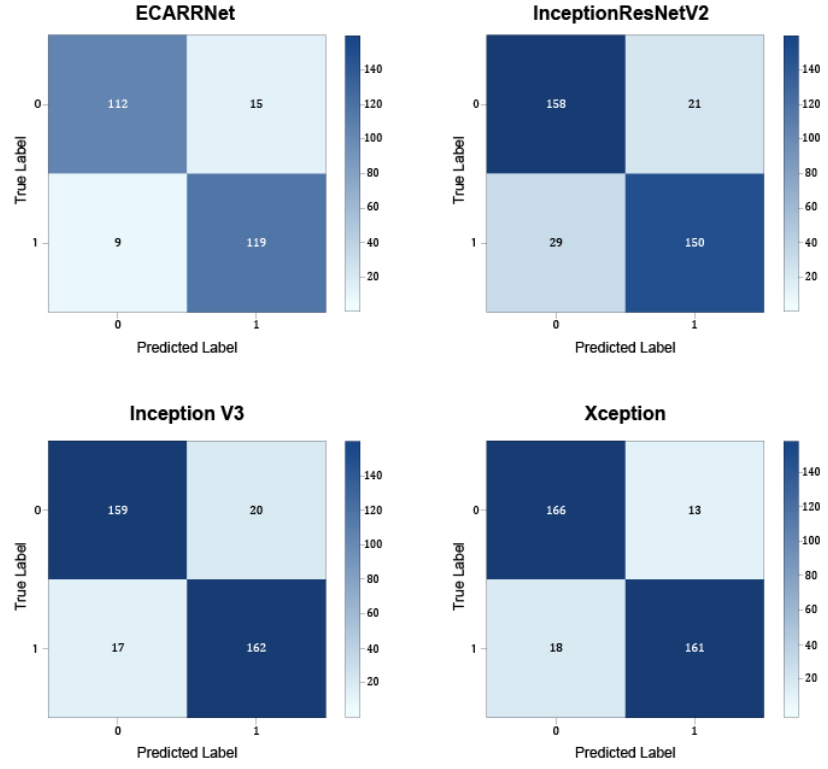


Figure 5.8: Confusion matrices for each of the models on the full dataset.

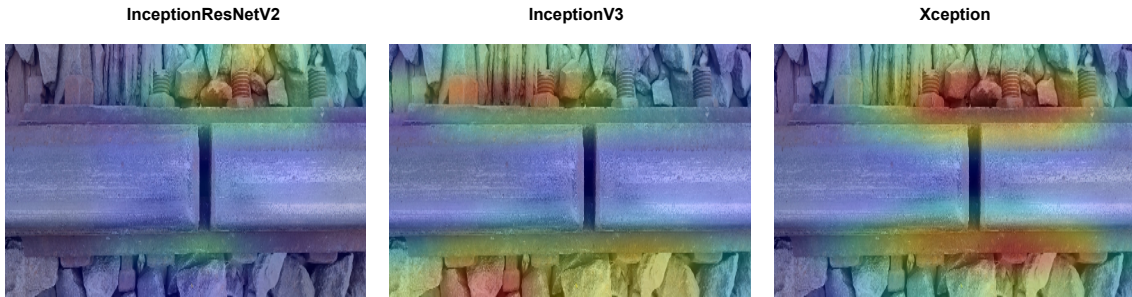


Figure 5.9: Grad-CAM visualisations for transfer learning models on the full dataset.

Table 5.3: Classification Report for Full Dataset

Models	Metrics	Defective	Non De- fective	Accuracy	Macro Average	Weighted Average
Inception- V3	Precision	0.89	0.91	-	0.90	0.90
	Recall	0.90	0.89	-	0.90	0.90
	F1 Score	0.90	0.90	0.90	0.90	0.90
	Support	176	182	358	358	358
Inception- ResnetV2	Precision	0.88	0.84	-	0.86	0.86
	Recall	0.84	0.88	-	0.86	0.86
	F1 Score	0.86	0.86	0.86	0.86	0.86
	Support	187	171	358	358	358
Xception	Precision	0.93	0.90	-	0.91	0.91
	Recall	0.90	0.93	-	0.91	0.91
	F1 Score	0.91	0.91	0.91	0.91	0.91
	Support	184	174	358	358	358
ECARRNet	Precision	0.95	0.89	-	0.92	0.92
	Recall	0.89	0.95	-	0.92	0.92
	F1 Score	0.92	0.92	0.92	0.92	0.92
	Support	189	168	357	357	357

5.2.4 LIME Visualizations to Explain the Output Predictions of ECARRNet

Empirical analysis using Grad-CAM shows that it works exceptionally well with transfer learning models like —InceptionResNetV2, InceptionV3, Xception, etc, but does not work for our proposed ensemble model, which also consists of Bi-directional LSTM along with CNN models. GRADCAM was unable to handle our model, and thus we could not obtain our desired visualization using GRADCAM. On the other hand, LIME is model-agnostic meaning, it can be used with any type of model. For this reason, we are using LIME to explain ECARRNet predictions. Figure 5.10 shows the visualisations for random images selected from the three different versions we have used for our experimentation purpose, i.e., the rail dataset, the combined dataset containing images of faults from both the rail and fastener, and the fastener dataset using LIME visualizations. The generated visualizations, which are wrapped in yellow boundaries are produced using only our proposed model, ECARRNet. This region was produced using LIME, explaining and showing the usefulness and robustness of our proposed model for detecting faults in railway tracks. It shows how accurately ECARRNet pinpoints the location of faults in railway tracks in all 3 versions of our dataset. It accurately wraps the faulty regions in rails and even pinpoints the faulty fasteners with utmost accuracy and very few false results. Therefore, these visualisations prove the supremacy of our ECARRNet model in predicting faults in railway tracks.



Figure 5.10: LIME visualizations for ECARRNet model on each of the datasets.

Figure 5.11 demonstrates the accuracy values in percentage for three different versions of the datasets. For each dataset, it compares the performance of our proposed model, ECARRNet with Xception, InceptionV3, and InceptionResNetV2. These models have also shown great results in image classification use cases. It also shows accuracy values for training, validation and test sets, which clearly distinguishes the performance of ECARRNet from other models. We are considering the test or validation accuracy here for comparison because we have ensured that the test/validation set does not have any labels for each image. We have just provided the images to the models to predict the classes. Therefore, we can see that validation accuracy is the highest for all 3 versions of the dataset for our proposed model, ECARRNet, compared to other SOTA models for predicting faults in railway tracks.

Also, by comparing results from Tables 4.3–5.2, we can also say that ECARRNet has shown expected results in other metrics like —precision, recall, F1 score, and in support as well. This means our proposed model is performing well in predicting and

distinguishing defective and non-defective railway tracks, which once again proves its superiority against other popular CNN-based architectures for railway fault detection. In addition, we can also see the total number of false positives and false negatives obtained from all the datasets for ECARRNet and other SOTA models in Table 5.4. As shown in the table, our proposed model, ECARRNet, has the lowest total false positives and false negatives compared to the other SOTA models. This is very important in real-world scenarios if our model is deployed in a robot or drone for the automatic detection of faults. Since ECARRNet predicts very few false results compared to other models, it will result in identifying more defects in the railway tracks and will thus result in fixing more faulty tracks, thus leading to fewer accidents due to faults in railway rails and fasteners if deployed in railway tracks. Therefore, this result once again proves the robustness and efficiency of our proposed model in predicting faults in railway tracks.

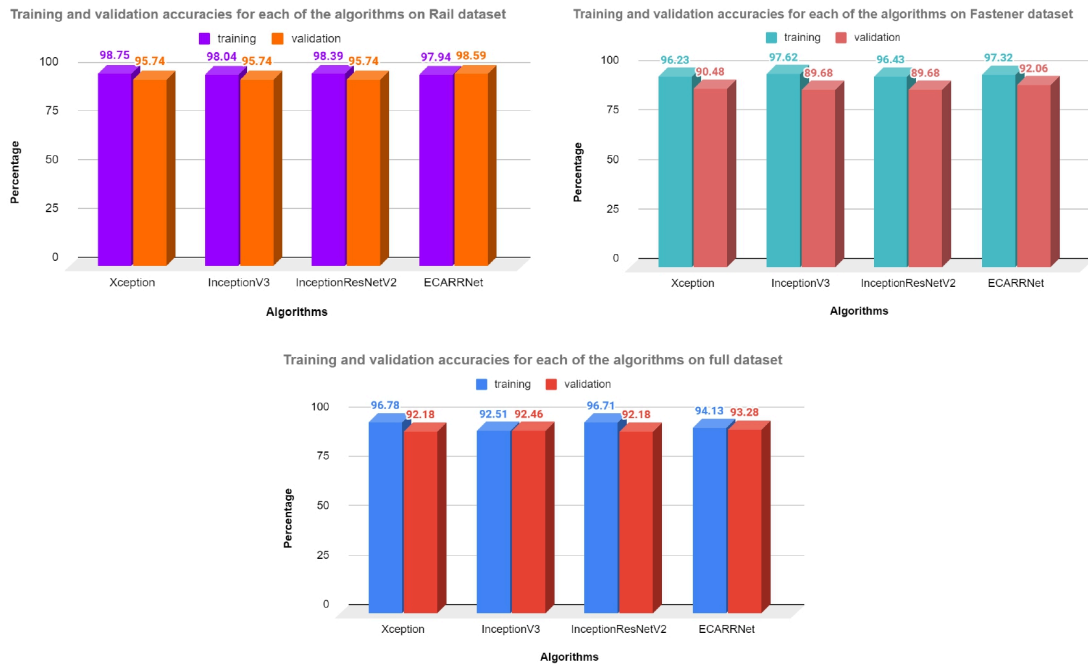


Figure 5.11: Accuracy for each of the models on different datasets.

Table 5.4: Sum of False Positives & False Negatives from all datasets for each of the models

Model	Total False Positives	Total False Negatives
ECARRNet	20	15
Xception	33	31
InceptionResNetV2	41	45
Inception V3	46	34

5.2.5 Summary of Research Findings

In this research, we have performed a series of rigorous experiments, through which the proposed model ECARRNet has demonstrated significant performance in railway fault classification measured through various performance metrics such as accuracy, recall, F1 score, etc. It has proved itself as a robust and interpretable model over existing state-of-the-art approaches.

Key findings from the research are as follows:

- **Superior Detection Performance:** ECARRNet achieves a classification accuracy of 98.59% on the rail dataset, outperforming benchmark models such as InceptionV3 (91.49%), InceptionResNetV2 (95.04%), and Xception(95.74%) on the same dataset. This highlights the efficacy of the ensemble design, which combines convolutional autoencoders, InceptionResNetV2 backbones, and BiLSTM-based temporal modelling.
- **Robustness Across Fault Types:** ECARRNet exhibited consistently high performance across various defect categories, including on the rail and combined (fastener + rail) datasets. Per-class evaluations showed that precision, recall, and F1 scores were at least 92%, affirming ECARRNet’s generalisation capabilities to new and unseen data.
- **Lowest False Positive and False Negative Rates:** ECARRNet minimised both false positive and false negative rates, achieving the lowest misclassification rates among all compared models. For instance, the total false positives for ECARRNet is 20 using all three datasets, whereas false positives for Xception are 33, 41 for InceptionResNetV2, and 46 for InceptionV3. False positives are important in this research, as they show the total misclassified count in the faulty class. Since finding faults in the railway track is one of the primary objectives of this research. Therefore, this reduction in error is very critical for safety-critical systems like railway inspections and surely gives ECARRNet a significant edge compared to other benchmark models.
- **Effective Data Augmentation and Imbalance Handling:** By employing ROI cropping, data augmentation through Keras’ ImageDataGenerator, and SMOTE oversampling techniques, the model was trained on a balanced and diverse dataset of over 8,000 images. This preprocessing strategy mitigated class imbalance and improved learning stability.
- **Real-time Feasibility:** The pipeline integrated ECARRNet into a real-time fault-detection system capable of inference on GPU hardware, meeting latency and scalability requirements for practical deployment.
- **Explainable AI Integration:** Through Grad-CAM and LIME visualizations, the study provided interpretable insights into the model’s decision-making process. ECARRNet exhibited more focused attention on faulty regions compared to baseline models, improving transparency and trust in predictions.

5.3 Discussion

Initially, this project was developed as only a robotics project fitted with sensors like Infrared and SONAR and was displayed in a competition. After that, it came to my mind that the identification of faults with just sensors is not very efficient. Infrared imaging only detects thermal signatures, while sonar provides distance measurements. Neither method communicates details about texture, colour variations, or geometric inconsistencies that are essential for thorough fault assessment. Therefore, after discussion with my team members and mentors, we formed a group to work on a journal paper to detect faults in the railway track. The first challenge was to get image data for defective and non-defective railway tracks. We could not get a suitable and sufficient image on the internet or from any other repositories for the railway track in Bangladesh. Two of our team members came to the rescue and took their DSLR camera and manually collected those images from Satkhira and Feni. Still, the images were fewer in number, and few faults could be addressed in this work. For this reason, I thought of implementing data augmentation to increase the number of images.

It has always been my dream to solve problems that are affecting our country and the world with the help of AI. This is such a dream project for me, and I have spent hundreds of hours coding, reading relevant papers, and brainstorming to propose a unique, robust, and efficient algorithm. Starting with conceptualising the idea to guiding my team members in data collection and curation, and the methodology formulation, I was strongly involved and led the project in each step of this work, making sure that the quality of the work and proper procedure were maintained by every team member. The empirical design, investigation, ablation study, and the programming part were also solely done by me, including writing the abstract, some parts of the introduction and related works, the methodological part, and other important parts like experimental results. In this process, I have stayed in touch with the mentors mentioned above and received consistent help from them. All the codes implemented in this paper by me are available and can be accessed through GitHub. [71].

One of the main reasons ECARRNet performs better than other baseline and state-of-the-art models is because of its hybrid architecture, which combines a bidirectional LSTM module with convolutional layers. Spatial hierarchies and patterns in railway imagery are efficiently captured by the convolutional blocks (such as the encoder-decoder pipeline and InceptionResNetV2), and the BiLSTM module helps describe sequential dependencies between image regions. By helping to distinguish between structural traits that appear identical and those that are defective, this sequential context enhances the model’s precision and recall.

Despite being more intricate and sophisticated than conventional CNN architectures, ECARRNet does not always have deployment needs that are unfeasible. It is now possible to run deep learning models like ECARRNet at the edge thanks to the introduction of inexpensive, contemporary single-board computers like the NVIDIA Jetson Nano or Jetson Xavier NX. Performance and deployability are thus kept in a realistic balance by the model. Domain shift, uneven class distribution, and variations in image circumstances like illumination, motion blur, and occlusion are the main causes of performance variation among datasets. The significance of thorough preprocessing, data augmentation, and potential domain adaptation techniques for

enhancing generalisation in upcoming deployments is highlighted by these factors. In this section, I would also like to demonstrate how we have achieved our research objectives as discussed in the research objectives section. We have worked mainly on six research objectives, and evidence from Table 5.5 elaborates and suggests that each of the research objectives has been fulfilled.

Table 5.5: Research Objectives and Evidence of Fulfillment

Objective	Evidence of Fulfillment
Propose a scalable, automated fault-detection pipeline for real-time inspection.	Chapter 4 introduces the end-to-end pipeline, integrates ECARRNet into a real-time data flow, and demonstrates inference on a GPU, satisfying scalability and latency targets.
Create and preprocess a domain-specific dataset with augmentation and class-imbalance correction.	Section 4.2 details the rails, fasteners, and the combined dataset of rails + fasteners, ROI cropping, Keras <code>ImageDataGenerator</code> use, and SMOTE oversampling, ensuring balanced classes and sufficient variability for training.
Develop a ResNet-based convolutional autoencoder hybrid ensemble.	In Section 4.3, the ECARRNet architecture has been presented: a denoising autoencoder, ResNet-derived InceptionResNetV2 backbone, bidirectional LSTM fusion, and a sigmoid head. Validation curves show ensemble gains over single models.
Assess performance against benchmarks (InceptionV3, InceptionResNetV2, Xception) using accuracy, precision, recall, and F1.	Section 5.2 reports quantitative results comparing ECARRNet against the benchmark models using precision, recall, F1 score, and accuracy comparison shown in Figure 5.11, which shows promising results compared to SOTA models.
Demonstrate effectiveness across fault types (rails, fasteners).	Section 5.2 breaks down per-class metrics: rail-only and fastener-only subsets, and the combined model generalises well, confirming robustness across defect categories.
Compare explainability with visualisations (LIME, Grad-CAM).	Sections 5.2 and 5.2.4 show Grad-CAM heatmaps and LIME explanations for ECARRNet, where ECARRNet highlights more coherent attention on fault regions and improved interpretability.

Summary

This chapter reports the experimental results and critically discusses ECARRNet’s strengths, performance trade-offs, and limitations. To consolidate these insights, the final chapter concludes the thesis and outlines potential directions for future research and deployment.

Chapter 6

Conclusion

The primary objective of this research paper is to differentiate between defective and fully functional railway tracks and to propose an automated solution to detect faults, thereby reducing accidents on the railway track, which result in hundreds of deaths. We have carried out the classification and detection of railway faults using cutting-edge algorithms and our ensemble-based algorithm, ECARRNet. Our proposed model outperformed the other established algorithms, producing better results that were evident in the obtained results.

6.1 Future Works and Limitations

Despite such astounding results, like any other research, this research also has some limitations. In the future, we would like to address some of the limitations. Firstly, we would like to collect more data from different parts of Bangladesh and other parts of the world to classify more categories of faults and improve on our proposed model. More data can be collected by keeping real-world scenarios in mind. Since ECARRNet was only trained on images from Bangladesh, it can be trained with images of railway tracks from different countries that are likely to be structurally different and from different camera angles through the use of on-board, side-mounted, and drone cameras. Secondly, images should be gathered in varied weather condition, such as during rain, fog, snow, etc., and in different lighting conditions, like dawn, dusk, and night. This will reduce the domain shift in the data. Third, using more advanced data augmentation techniques, such as GAN-based augmentations, will create more meaningful data points and thus increase the robustness of the model. Fourth, in safety-critical systems like railway fault detection, both false positives and negatives carry costs. So, we need to define acceptable error rates for rail and fastener faults while deploying the model in the real world. Moreover, rigorous stress testing by introducing noise and blurriness can be carried out to test the stability of the model. Fifth, decision thresholds might need to be calibrated to balance sensitivity and alert fatigue in real-world scenarios. Finally, ECARRNet should be pruned or quantised to reduce network latency and memory footprint, as on-track deployment often lacks powerful servers. To conclude, regulatory safety and security compliance and seamless integration with the existing system will minimise disruption. Continuous monitoring and maintenance of the model should also be ensured after real-world deployment, as models can degrade as track conditions evolve.

Bibliography

- [1] V. Singhal, S. S. Jain, D. Anand, *et al.*, “Artificial intelligence enabled road vehicle-train collision risk assessment framework for unmanned railway level crossings,” *IEEE Access*, vol. 8, pp. 113 790–113 806, 2020. DOI: 10.1109/ACCESS.2020.3002416.
- [2] N. A. Probha and M. S. Hoque, “A study on transport safety perspectives in bangladesh through comparative analysis of roadway, railway and waterway accidents,” ser. APCIM amp; ICTTE 2018, Beijing, China: Association for Computing Machinery, 2018, pp. 81–85, ISBN: 9781450366045. DOI: 10.1145/3321619.3321679. [Online]. Available: <https://doi.org/10.1145/3321619.3321679>.
- [3] M. M. Islam, A. E. M. Ridwan, M. M. Mary, M. F. Siam, S. A. Mumu, and S. Rana, “Design and implementation of a smart bike accident detection system,” in *2020 IEEE Region 10 Symposium (TENSYP)*, 2020, pp. 386–389. DOI: 10.1109/TENSYP50017.2020.9230656.
- [4] A. Krizhevsky, I. Sutskever, and G. E. Hinton, “Imagenet classification with deep convolutional neural networks,” in *Proceedings of the 25th International Conference on Neural Information Processing Systems - Volume 1*, ser. NIPS’12, Lake Tahoe, Nevada: Curran Associates Inc., 2012, pp. 1097–1105.
- [5] K. Simonyan and A. Zisserman, “Very deep convolutional networks for large-scale image recognition,” *arXiv preprint arXiv:1409.1556*, 2014.
- [6] H. Chen, Y. Zhang, M. K. Kalra, *et al.*, “Low-dose ct with a residual encoder-decoder convolutional neural network,” *IEEE Transactions on Medical Imaging*, vol. 36, no. 12, pp. 2524–2535, 2017. DOI: 10.1109/TMI.2017.2715284.
- [7] M. Anthimopoulos, S. Christodoulidis, L. Ebner, A. Christe, and S. Mougiakakou, “Lung pattern classification for interstitial lung diseases using a deep convolutional neural network,” *IEEE Transactions on Medical Imaging*, vol. 35, no. 5, pp. 1207–1216, 2016. DOI: 10.1109/TMI.2016.2535865.
- [8] R. Rahman, M. A. Rahman, S. Hossain, S. Hossain, M. R. Akhond, and M. I. Hossain, “Ransomlistener: Ransom call sound investigation using lstm and cnn architectures,” in *2021 6th International Conference on Inventive Computation Technologies (ICICT)*, 2021, pp. 509–516. DOI: 10.1109/ICICT50816.2021.9358488.
- [9] C. Ding and D. Tao, “Robust face recognition via multimodal deep face representation,” *IEEE Transactions on Multimedia*, vol. 17, no. 11, pp. 2049–2058, 2015. DOI: 10.1109/TMM.2015.2477042.

- [10] Y. Sun, Y. Chen, X. Wang, and X. Tang, "Deep learning face representation by joint identification-verification," in *Advances in Neural Information Processing Systems*, Z. Ghahramani, M. Welling, C. Cortes, N. Lawrence, and K. Weinberger, Eds., vol. 27, Curran Associates, Inc., 2014.
- [11] P. Chamoso, W. Raveane, V. Parra, and A. González, "Uavs applied to the counting and monitoring of animals," in *Ambient Intelligence - Software and Applications*, C. Ramos, P. Novais, C. E. Nihan, and J. M. Corchado Rodríguez, Eds., Cham: Springer International Publishing, 2014, pp. 71–80, ISBN: 978-3-319-07596-9.
- [12] Y. xu, G. Yu, Y. Wang, X. Wu, and Y. Ma, "Car detection from low-altitude uav imagery with the faster r-cnn," *Journal of Advanced Transportation*, vol. 2017, pp. 1–10, Aug. 2017. DOI: 10.1155/2017/2823617.
- [13] S. Mittal and D. Rao, "Vision based railway track monitoring using deep learning," *CoRR*, vol. abs/1711.06423, 2017. arXiv: 1711.06423. [Online]. Available: <http://arxiv.org/abs/1711.06423>.
- [14] Y. W. Lin, C. C. Hsieh, W. H. Huang, S. L. Hsieh, and W. H. Hung, "Railway track fasteners fault detection using deep learning," in *Proceedings of the 2019 IEEE Eurasia Conference on IoT, Communication and Engineering (ECICE)*, [Google Scholar] [CrossRef], Yunlin, Taiwan, Oct. 2019, pp. 187–190.
- [15] A. James, W. Jie, Y. Xulei, *et al.*, "Tracknet - a deep learning based fault detection for railway track inspection," pp. 1–5, 2018. DOI: 10.1109/ICIRT.2018.8641608.
- [16] J. Lu, B. Liang, Q. Lei, *et al.*, "Scueu-net: Efficient damage detection method for railway rail," *IEEE Access*, vol. 8, pp. 125 109–125 120, 2020. DOI: 10.1109/ACCESS.2020.3007603.
- [17] A. Welankiwar, S. Sherekar, A. P. Bhagat, and P. A. Khodke, "Fault detection in railway tracks using artificial neural networks," pp. 1–5, 2018. DOI: 10.1109/RICE.2018.8509083.
- [18] X. Wei, Z. Yang, Y. Liu, D. Wei, L. Jia, and Y. Li, "Railway track fastener defect detection based on image processing and deep learning techniques: A comparative study," *Eng. Appl. Artif. Intell.*, vol. 80, pp. 66–81, 2019.
- [19] Y. Min, B. Xiao, J. Dang, B. Yue, and T. Cheng, "Real time detection system for rail surface defects based on machine vision," *EURASIP Journal on Image and Video Processing*, vol. 2018, no. 3, p. 3, 2018, [Google Scholar] [CrossRef].
- [20] M. Karakose, O. Yaman, M. Baygin, K. Murat, and E. Akin, "A new computer vision based method for rail track detection and fault diagnosis in railways," *Int. J. Mech. Eng. Robot. Res.*, vol. 6, pp. 22–27, 2017, [Google Scholar] [CrossRef].
- [21] M. Singh, S. Singh, J. Jaiswal, and J. Hempshall, "Autonomous rail track inspection using vision based system," pp. 56–59, Nov. 2006. DOI: 10.1109/CIHSPS.2006.313313.
- [22] X. Gibert, V. M. Patel, and R. Chellappa, "Deep multitask learning for railway track inspection," *IEEE Transactions on Intelligent Transportation Systems*, vol. 18, no. 1, pp. 153–164, 2017. DOI: 10.1109/TITS.2016.2568758.

- [23] H. Alawad, S. Kaewunruen, and M. An, “A deep learning approach towards railway safety risk assessment,” *IEEE Access*, vol. 8, pp. 102 811–102 832, 2020. DOI: 10.1109/ACCESS.2020.2997946.
- [24] A. Kamilaris and F. X. Prenafeta-Boldú, “A review of the use of convolutional neural networks in agriculture,” *The Journal of Agricultural Science*, vol. 156, no. 3, pp. 312–322, 2018. DOI: 10.1017/S0021859618000436.
- [25] C. Yang, Y. Sun, C. Ladubec, and Y. Liu, “Developing machine learning-based models for railway inspection,” *Applied Sciences*, vol. 11, no. 1, 2021, ISSN: 2076-3417. [Online]. Available: <https://www.mdpi.com/2076-3417/11/1/13>.
- [26] X. Z. Hongdou Yao Xuejie Zhang and S. Liu, “Parallel structure deep neural network using cnn and rnn with an attention mechanism for breast cancer histology image classification,” *Cancers (Basel)*, vol. 11, 2019.
- [27] L. Alzubaidi, J. Zhang, A. J. Humaidi, *et al.*, “Review of deep learning: Concepts, cnn architectures, challenges, applications, future directions,” *Big Data*, vol. 8, p. 53, 2021, [Google Scholar] [CrossRef] [PubMed].
- [28] M. Campos-Taberner, F. J. García-Haro, B. Martínez, *et al.*, “Understanding deep learning in land use classification based on sentinel-2 time series,” *Scientific Reports*, vol. 10, p. 17 188, 2020, [Google Scholar] [CrossRef].
- [29] R. Shafique, H.-U.-R. Siddiqui, F. Rustam, *et al.*, “A novel approach to railway track faults detection using acoustic analysis,” *Sensors*, vol. 21, no. 18, 2021, ISSN: 1424-8220. [Online]. Available: <https://www.mdpi.com/1424-8220/21/18/6221>.
- [30] Y. Z. Yunguang Ye Ping Huang, “Deep learning-based fault diagnostic network of high-speed train secondary suspension systems for immunity to track irregularities and wheel wear,” *Eng. Science*, 2021.
- [31] S. I. Eunus, S. Hossain, A. E. M. Ridwan, *et al.*, “Ecarnnet: An efficient lstm-based ensembled deep neural network architecture for railway fault detection,” *AI*, vol. 5, no. 2, pp. 482–503, 2024. DOI: 10.3390/ai5020024. [Online]. Available: <https://doi.org/10.3390/ai5020024>.
- [32] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, “Learning representations by back-propagating errors,” *Nature*, vol. 323, pp. 533–536, 1986.
- [33] K. Hornik, M. Stinchcombe, and H. White, “Multilayer feedforward networks are universal approximators,” *Neural Networks*, vol. 2, no. 5, pp. 359–366, 1989.
- [34] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning* (Adaptive Computation and Machine Learning). Cambridge, MA: MIT Press, 2016, ISBN: 9780262035613.
- [35] M. Telgarsky, “Benefits of depth in neural networks,” in *Proceedings of the 29th Annual Conference on Learning Theory (COLT)*, Barcelona, Spain, 2016.
- [36] R. Eldan and O. Shamir, “The power of depth for feedforward neural networks,” in *Proceedings of the 29th Annual Conference on Learning Theory (COLT)*, 2016.

- [37] X. Glorot and Y. Bengio, “Understanding the difficulty of training deep feed-forward neural networks,” in *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics (AISTATS)*, vol. 9, JMLR Workshop and Conference Proceedings, 2010, pp. 249–256.
- [38] D.-A. Clevert, T. Unterthiner, and S. Hochreiter, “Fast and accurate deep network learning by exponential linear units (elus),” *arXiv preprint arXiv:1511.07289*, 2016.
- [39] D. Hendrycks and K. Gimpel, “Gaussian error linear units (gelus),” *arXiv preprint arXiv:1606.08415*, 2016.
- [40] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, “Gradient-based learning applied to document recognition,” *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.
- [41] A. Krizhevsky, I. Sutskever, and G. E. Hinton, “Imagenet classification with deep convolutional neural networks,” in *Advances in Neural Information Processing Systems*, vol. 25, 2012, pp. 1097–1105.
- [42] S. Hochreiter and J. Schmidhuber, “Long short-term memory,” *Neural Computation*, vol. 9, no. 8, pp. 1735–1780, 1997.
- [43] K. Cho, B. van Merriënboer, Ç. Gulcehre, *et al.*, “Learning phrase representations using rnn encoder–decoder for statistical machine translation,” in *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, Association for Computational Linguistics, 2014, pp. 1724–1734.
- [44] A. Vaswani, N. Shazeer, N. Parmar, *et al.*, “Attention is all you need,” in *Advances in Neural Information Processing Systems*, vol. 30, 2017, pp. 5998–6008.
- [45] T. N. Kipf and M. Welling, “Semi-supervised classification with graph convolutional networks,” in *Proceedings of the International Conference on Learning Representations (ICLR)*, 2017.
- [46] W. L. Hamilton, Z. Ying, and J. Leskovec, “Inductive representation learning on large graphs,” in *Advances in Neural Information Processing Systems*, vol. 30, 2017, pp. 1025–1035.
- [47] C. M. Bishop, *Pattern Recognition and Machine Learning*. New York: Springer, 2006, ISBN: 978-0-387-31073-2.
- [48] L. Sun, Z. Jiang, and S.-F. Chang, “Risk-sensitive loss functions for autonomous collision avoidance,” *IEEE Transactions on Intelligent Transportation Systems*, vol. 21, no. 4, pp. 1538–1549, 2020.
- [49] D. P. Kingma and J. Ba, “Adam: A method for stochastic optimization,” in *Proceedings of the International Conference on Learning Representations (ICLR)*, 2015. [Online]. Available: <https://arxiv.org/abs/1412.6980>.
- [50] I. Sutskever, J. Martens, G. E. Dahl, and G. E. Hinton, “On the importance of initialization and momentum in deep learning,” in *Proceedings of the 30th International Conference on Machine Learning (ICML)*, PMLR, vol. 28, 2013, pp. 1139–1147.

- [51] T. Tieleman and G. Hinton, “Lecture 6.5—rmsprop: Divide the gradient by a running average of its recent magnitude,” in *Neural Networks: Tricks of the Trade*, G. Montavon, G. Orr, and K.-R. Müller, Eds., Technical report; slides from Coursera lecture, Springer, 2012. [Online]. Available: https://www.cs.toronto.edu/~tijmen/csc321/slides/lecture_slides_lec6.pdf.
- [52] I. Loshchilov and F. Hutter, “Sgdr: Stochastic gradient descent with warm restarts,” in *Proceedings of the International Conference on Learning Representations (ICLR)*, arXiv preprint arXiv:1608.03983, 2017. [Online]. Available: <https://arxiv.org/abs/1608.03983>.
- [53] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, “Dropout: A simple way to prevent neural networks from overfitting,” *Journal of Machine Learning Research*, vol. 15, no. 56, pp. 1929–1958, 2014.
- [54] J. Chen, Z. Zhang, and X. Li, “Explainable deep learning for railway defect detection,” *IEEE Transactions on Intelligent Transportation Systems*, vol. 22, no. 5, pp. 3278–3289, 2021.
- [55] X. Chen and W. Wang, “Multimodal sensor data fusion for railway track condition monitoring,” *Mechanical Systems and Signal Processing*, vol. 135, p. 106 398, 2020. DOI: 10.1016/j.ymssp.2019.106398.
- [56] I. Goodfellow, J. Pouget-Abadie, M. Mirza, *et al.*, “Generative adversarial nets,” in *Advances in Neural Information Processing Systems*, vol. 27, 2014, pp. 2672–2680.
- [57] S. Han, H. Mao, and W. J. Dally, “Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding,” in *Proceedings of the International Conference on Learning Representations (ICLR)*, 2016. [Online]. Available: <https://arxiv.org/abs/1510.00149>.
- [58] G. I. Parisi, R. Kemker, J. L. Part, C. Kanan, and S. Wermter, “Continual lifelong learning with neural networks: A review,” *Neural Networks*, vol. 113, pp. 54–71, 2019. DOI: 10.1016/j.neunet.2019.01.012.
- [59] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, *Rethinking the inception architecture for computer vision*, 2015. DOI: 10.48550/ARXIV.1512.00567. [Online]. Available: <https://arxiv.org/abs/1512.00567>.
- [60] A. Adnan, S. Hossain, R. Shihab, and S. Ibne Eunus, “Railway track fault detection,” 2021. DOI: 10.34740/KAGGLE/DSV/1884733. [Online]. Available: <https://www.kaggle.com/datasets/salmaneunus/railway-track-fault-detection>.
- [61] A. Adnan, S. Hossain, R. Shihab, and S. Ibne Eunus, “Railway track fault detection — dataset2(fastener),” 2021. DOI: 10.34740/KAGGLE/DSV/2772290. [Online]. Available: <https://www.kaggle.com/dsv/2772290>.
- [62] D. Elreedy and A. F. Atiya, “A comprehensive analysis of synthetic minority oversampling technique (smote) for handling class imbalance,” *Information Sciences*, vol. 505, pp. 32–64, 2019, ISSN: 0020-0255. DOI: <https://doi.org/10.1016/j.ins.2019.07.070>. [Online]. Available: <https://www.sciencedirect.com/science/article/pii/S0020025519306838>.

- [63] M. Chen, X. Shi, Y. Zhang, D. Wu, and M. Guizani, “Deep feature learning for medical image analysis with convolutional autoencoder neural network,” *IEEE Transactions on Big Data*, vol. 7, no. 4, pp. 750–758, 2021. DOI: 10.1109/TBDATA.2017.2717439.
- [64] Y. Zhang, “A better autoencoder for image: Convolutional autoencoder,” 2018.
- [65] S. Ioffe and C. Szegedy, “Batch normalization: Accelerating deep network training by reducing internal covariate shift,” *CoRR*, vol. abs/1502.03167, 2015. arXiv: 1502.03167. [Online]. Available: <http://arxiv.org/abs/1502.03167>.
- [66] C. Szegedy, S. Ioffe, and V. Vanhoucke, “Inception-v4, inception-resnet and the impact of residual connections on learning,” *CoRR*, vol. abs/1602.07261, 2016. arXiv: 1602.07261. [Online]. Available: <http://arxiv.org/abs/1602.07261>.
- [67] Y. Guo, Y. Liu, E. Bakker, Y. Guo, and M. Lew, “Cnn-rnn: A large-scale hierarchical image classification framework,” *Multimedia Tools and Applications*, vol. 77, Apr. 2018. DOI: 10.1007/s11042-017-5443-x.
- [68] Yin, Qiwei, Zhang, Ruixun, and Shao, XiuLi, “Cnn and rnn mixed model for image classification,” *MATEC Web Conf.*, vol. 277, p. 02001, 2019. DOI: 10.1051/mateconf/201927702001. [Online]. Available: <https://doi.org/10.1051/mateconf/201927702001>.
- [69] R. R. Selvaraju, M. Cogswell, A. Das, R. Vedantam, D. Parikh, and D. Batra, “Grad-cam: Visual explanations from deep networks via gradient-based localization,” *International Journal of Computer Vision*, vol. 128, no. 2, pp. 336–359, Oct. 2019, ISSN: 1573-1405. DOI: 10.1007/s11263-019-01228-7.
- [70] M. T. Ribeiro, S. Singh, and C. Guestrin, “why should i trust you?: Explaining the predictions of any classifier, 2016. DOI: 10.48550/ARXIV.1602.04938.
- [71] S. Eunus, *ECARRNET—railway fault detection journal*, <https://github.com/SalmanEunus27/ECARRNET---RAILWAY-FAULT-DETECTION-JOURNAL>, GitHub repository, accessed 2025-07-13, 2024.