STATISTICS, OPTIMIZATION AND INFORMATION COMPUTING
Stat., Optim. Inf. Comput., Vol. **12**, November 2024, pp 1886–1911.
Published online in International Academic Press (www.IAPress.org)

Railway Track Faults Detection Using Ensemble Deep Transfer Learning Models

Ali Mansour Al-Madani[1,2,*], Vivek Mahale[3], Ashok T. Gaikwad[4]

[1] *Faculty of computer science and IT, Sana'a University, Sana'a, Yemen*
[2] *Department of Computer Science Dr, Babasaheb Ambedkar Marathwada University, Aurangabad, India*
[3] *AIIT, Amity University, Mumbai, India*
[4] *Institute of Management Studies and Information Technology, Aurandabad,India*

Abstract Railway track fault detection is an essential task for ensuring the safety and reliability of railway systems, particularly in the summer and rainy seasons when train wheels may slide due to fractures in the track or corrosion may cause track fractures. In this study, we propose a novel approach for the automated detection of railway track faults using deep transfer learning models. The proposed method combines image processing techniques and the training of three pretrained models: InceptionV3, ResNet50V2, and VGG16, on a dataset of railway track images. We evaluated the performance of our proposed method by measuring its accuracy on a test set of railway track images. The individual training accuracies for InceptionV3, ResNet50V2, and VGG16 were 94.30%, 96.79%, and 94.64%, respectively. We then combined these models using an ensemble approach, which achieved an impressive accuracy of 98.57% on the test set. Our results demonstrate the effectiveness of using deep ensemble transfer learning for railway track fault detection. Moreover, our proposed method can be used as a valuable tool for railway track maintenance and monitoring, which can ultimately lead to the improvement of the safety and reliability of railway systems. Our proposed approach for railway track fault detection using ensemble deep transfer learning models shows promising results, indicating that it has great potential for detecting track faults accurately and efficiently. The proposed method can be used in various railway systems worldwide, ultimately leading to improved safety and reliability for passengers and cargo transportation.

Keywords Ensemble model, Rail detection, transfer learning models, deep learning, Abnormal detection, Classification

DOI: 10.19139/soic-2310-5070-1994

1. Introduction

In today's world, the importance of the railway network cannot be overstated. Infrastructural work, expansion, and upkeep all fall within the purview of railroads. The railway network's infrastructure consists of designing and building rail lines and establishing connections between those tracks at railroad junctions. The village's rural and inner regions benefit from expanding the community's railway network. The railway network system's maintenance section maintains the train tracks. The train lines are extensively corroded by the air and rain during the rainy seasons [1]. As a result, train accidents occur due to rail track fractures forming. To avoid accidents, cracks in the rail track must be checked regularly to keep the track in good condition.

Previously, they mainly relied on manual monitoring to detect railroad defects. However, this method takes a long time, especially on long railways [3]. The traditional method, which is costly, poses a danger to the observers (workers) and may lead to an error in identifying defects in the rail. Nowadays, when there automatic tools and the advent of artificial intelligence it has been gradually employed, and among these studies that have been used to detect railway defects: Ultrasound [4] is used, and magnetic flux leakage [6] has been used. This complex

[*] Correspondence to: Ali Mansour Al-Madani (Email: ali.m.almadani1992@gmail.com). Faculty of computer science and IT, Sana'a University, Sana'a, Yemen.

ISSN 2310-5070 (online) ISSN 2311-004X (print)

method is used to detect the surface of the railroad from the sensor signals. As for the ultrasound examination, it depends on the speed of the train during detection [4]. As well as the use of surveillance cameras through the use of deep learning for video analysis [7, 8], which is characterised by this technology as needing direct contact with the railway, low cost, speed, and high accuracy to detect defects of the railway. Artificial intelligence, machine learning, and deep learning are increasingly used in industry and medical research.

Several attempts have been made to automatically detect railroad rail defects using video cameras by human vision observation. This method depends on analysing images and discovering railroad defects manually. An algorithm is proposed to classify the images into the defect and non-defect [9, 10]. The characteristics are also extracted by analysing the edges of defects in the railway rail [11], using the Augmentation images, the angled rotation of the images, and the use of a normalisation algorithm, to detect the defects of the rail surface with high accuracy [12]. All these stages of image processing have the effect of raising the accuracy of rail defect detection.

There are some difficulties that the researchers need help with in their results. Among these difficulties is the low accuracy and recall in detecting railroad rod defects, and among these defects are cracking and minor defects, which are difficult to detect and distinguish. Wang, H et al. [13] proposed the principal components analysis (PCA) model for detecting rail defects by colours. In the same way, Liu, Q et al. [14] proposed a continuum of balanced colours, such as grey and phase spectrum, and the Otsu threshold used to detect railroad defects. He Zh, et al. [15] detected railroad rail defects by applying a Background Difference Algorithm. Tian, et al. [16] used a modified version of the Sobel operator to discover possible features of railroad rail defects. Unfortunately, the inability to generate new data has always been the primary limitation of these approaches. Inconsistent illumination and degraded visual surroundings might have a negative impact on the detection results.

Problems like this often arise in real For several reasons, including complicated and unusual faults, blurry images of essential components owing to unbalanced reflection in image capture, and harsh weather conditions. Because of these things, it's clear that standard computer vision systems, which use manual features, are not the best way to detect railroad defects. As deep learning has progressed quickly in recent years, deep convolutional neural networks (CNNs) have been developed to extract features automatically with high production and accuracy [17, 18]. Given these impressive capabilities, deep CNNs have been successfully employed in various research to classify rail surface defects [19, 20]. However, the fundamental problem of these approaches is that they cannot detect the defects in images, which is an important feature. The suggested advanced models are also not able to perform quick detection.

Computer vision uses object detection to locate items in images. In the last decade, it's been widely employed [21, 22]. In specific categories, detection accuracy has exceeded humans [23]. It may make it easier to detect rail defects.

This article uses the Convolutional Neural Network (CNN) classification technique to identify fractures in railway tracks. To determine the optimum augmentation approach for this situation, the suggested method includes the following stages: preprocessing, image resolution, colour analysis (Intensity Analysis), edge analysis, Hue-Saturation-Value (HSV), feature extraction, classification, and segmentation. Pre-processing methods include Hue, Saturation, and Intensity equalisation to enhance a rail track's picture. Rail track defect detection may be improved with this method, regardless of environmental parameters. Hue-Saturation-Value (HSV) transforms the spatial domain preprocessing images into a multi-resolution image. Duration, frequency and direction are all shown in this multi-resolution image.

This research used several deep learning algorithms to identify railway defects, including InceptionV3, ResNet50V2, VGG16, and other pre-trained deep learning algorithms.

After comparing the Ensemble transfer learning model with the rest of the models, the ensemble model was selected as the best model in terms of results, high accuracy, and detection speed. Also, the model's performance is high, even in poor visual conditions and with minor defects in the railways. The proposed model showed high and fast performance in identifying and classifying railroad images into defective and non-defective ones.

This study extends previous work by not only developing an ensemble deep learning model for railway track fault detection, but also providing in-depth analysis of the pre-training process, model interpretability, and computational requirements. We conduct extensive validation on a diverse dataset to assess real-world applicability.

The remaining parts of this paper are summarised and divided as follows: Section 2 explains the railways' defects. Section 3, Materials and methods, the dataset and image pre-processing are described before training. Section 4 presents the experimental results and demonstrates why the proposed model has been chosen. Finally, some conclusions are summarised in Section 5.

Defects in railway tracks can arise from multiple sources:

- **Physical Wear:** Continuous contact and friction from train wheels can lead to surface wear and the formation of cracks.
- **Environmental Impact:** External factors such as moisture and corrosive conditions contribute significantly to rail degradation.
- **Mechanical Stress:** The immense weight of trains and the mechanical forces from braking systems also induce defects both on the surface and internally within the rail structure.

Identifying these defects is crucial for maintaining the safety and reliability of railway operations. Advanced detection methods using deep learning not only enhance the accuracy of defect identification but also significantly reduce the risk of catastrophic failures by allowing timely interventions.

2. Materials and Methods

For the development's major goal, we will use deep learning methods (CNN) to create a railway crack detection system that can learn fundamental forms in its initial layers and enhance the image's learning characteristics at a deeper level. The CNN layers do feature extraction automatically. The pretraining model has the benefit of requiring less time to create and train the model from scratch. It is possible to achieve high accuracy using a sufficient dataset like the one used in this research.

2.1. Dataset

The original dataset used in this study consists of 800 images, with 400 images of defective railway tracks and 400 images of non-defective tracks. The images were collected from various sources and cover a range of railway track conditions and environments.

To enhance the robustness and generalizability of our model, we expanded the dataset to include additional images covering a wider range of railway sections, fault types, lighting conditions, and camera angles. The augmented dataset consists of 1200 images, with 600 images of defective tracks and 600 images of non-defective tracks.

The defective track images include various types of faults, such as:

- Cracks: longitudinal, transverse, and web cracks
- Surface defects: spalling, pitting, and wear
- Deformations: rail head deformation, rail foot deformation, and rail corrugation
- Missing components: missing fasteners, missing rail clips, and broken rail joints

The non-defective track images include normal railway tracks under different environmental conditions and capture variations in:

- Lighting: daylight, low light, and night conditions
- Weather: clear, cloudy, and rainy conditions
- Camera angles: front view, side view, and oblique view

The expanded dataset was carefully curated to ensure a balanced representation of fault types and environmental conditions. The images were preprocessed by resizing them to a fixed size of 224x224 pixels and normalizing the pixel values to the range [0, 1].

Figures 1 and 2 show some examples of the defective and non-defective track images from the expanded dataset, highlighting the diversity of fault types and environmental conditions covered.

The expanded dataset was split into training, validation, and testing sets, with a ratio of 70:15:15. The training set was used to train the deep learning models, the validation set was used for hyperparameter tuning and model selection, and the testing set was used for final performance evaluation.

By incorporating a larger and more diverse dataset, we aim to develop a robust and generalizable model for railway track fault detection that can perform well under various real-world conditions.

2.2. Pre-trained Model Adaptation

The pre-trained VGG16, ResNet50V2, and InceptionV3 models were adapted for railway track fault detection through a combination of layer freezing and fine-tuning. The convolutional base of each model was frozen, while the top layers were replaced with new fully-connected layers specific to the binary classification task (defective vs. non-defective). The models were then fine-tuned on the railway track dataset, allowing the higher-level features to adapt to the new domain while leveraging the low-level features learned from ImageNet. Hyperparameters such as learning rate, batch size, and number of epochs were tuned based on validation performance.

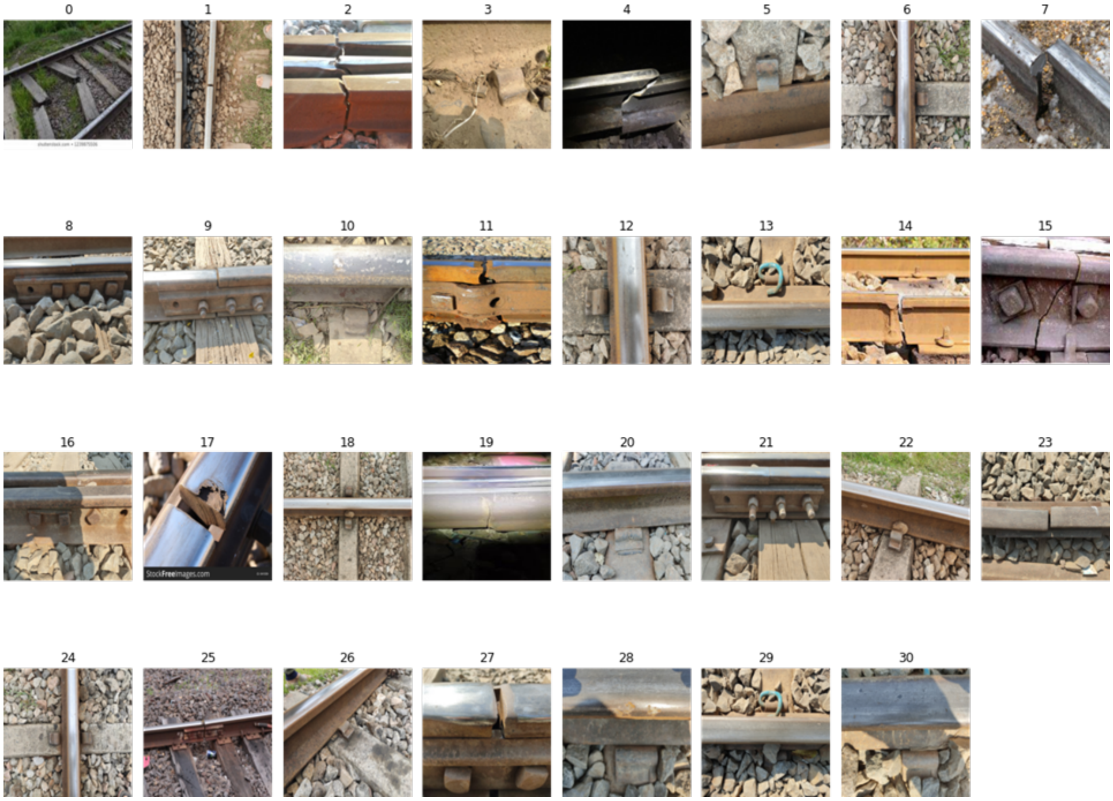


Figure 1. Defective training data.

2.3. Pre-processing

The dataset was preprocessed by the developer to remove duplicate photos and crop the images. The remaining photos were divided into three sets: 560 for training, 160 for validation, and 80 for testing.

2.3.1. Image Resolutions Table 1 below shows the size and resolution differences between defective and undefective images. The table illustrates the variations in dimensions that are critical for the neural network's

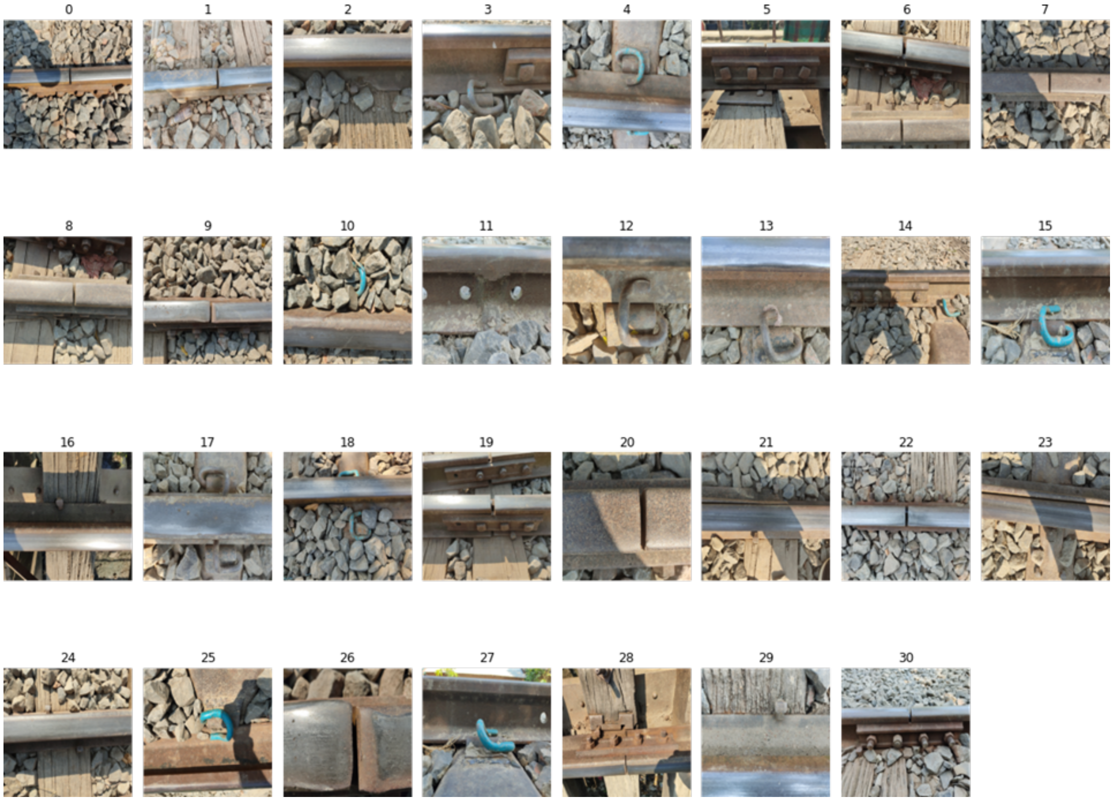


Figure 2. Undefective training data.

architecture and the format of the input data, which typically includes image height, width, number of channels, and pixel levels.

Table 1. Images Size and Resolution

	Undefective Images	**Defective Images**
Width	4000.0	2408.321
Height	3000.0	1816.881
Dimension	12000000.0	6505252

2.3.2. Analysis The chosen input model resolution of 224x224 is generally sufficient, although it does lead to some loss of information. We have performed various analyses such as colour analysis and intensity analysis to determine the best augmentation method for this specific situation. For instance, intensity analysis shows that the pixel's value, ranging from 0 to 255 in an 8-bit greyscale image, determines its intensity. Colour saturation, which indicates the purity of a color, is lower when a color is closer to grey, and higher when it is more vibrant. This is essential in differentiating the subtle features in railway track images, particularly when examining potential defects.

2.3.3. Analysing Edges Using edge detection filters like the Sobel filter, we can create gradient images that highlight linear features, which are essential for identifying potential defects. The Sobel edge detector calculates the gradient magnitude across the image to detect edges effectively. This is crucial for our railway track fault detection system as it helps in identifying and extracting relevant features.

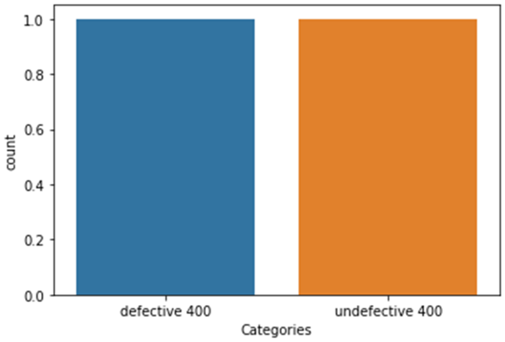


Figure 3. Bar chart represents the number of images used for detecting faults in railway tracks in the image- dataset [2].

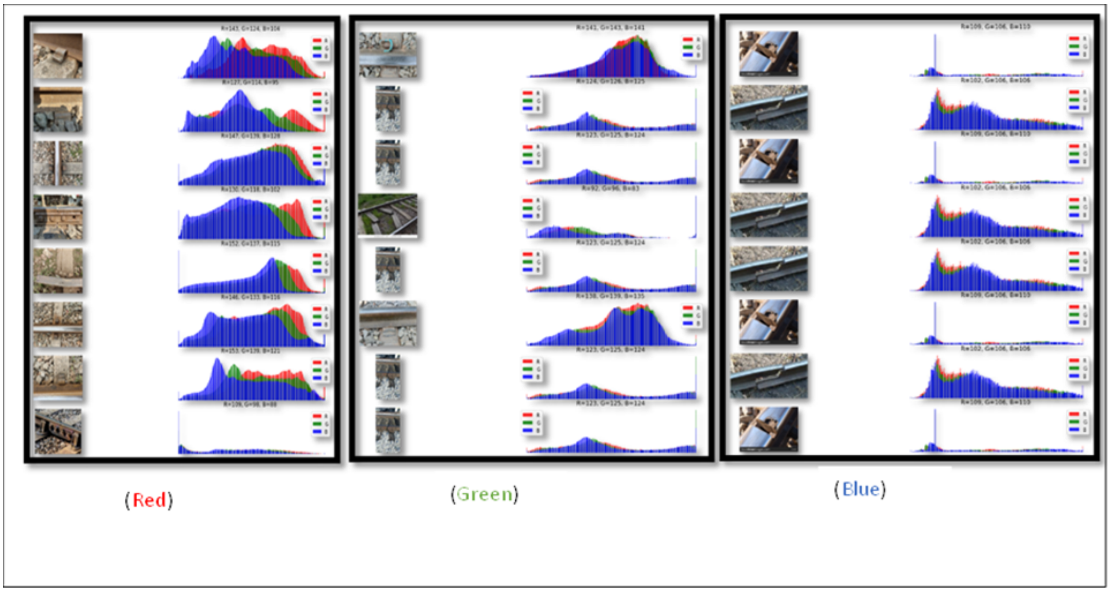


Figure 4. Intensity analysis for Red, Green and Blue) colors defective images.

$$|G| = \sqrt{G_x^2 + G_y^2} \tag{1}$$

$$|G| = |G_x| + |G_y| \tag{2}$$

$$\theta = \tan^{-1}(G_y/G_x) \tag{3}$$

2.4. HSV Transform

The RGB values of an image are transformed into Hue, Saturation, and Value (HSV), which helps in separating color information from luminance. This transformation is particularly useful in applications where color description plays a crucial role. For instance, it enables the identification of shadows and lighting variations and distinguishes objects based on color under different lighting conditions.

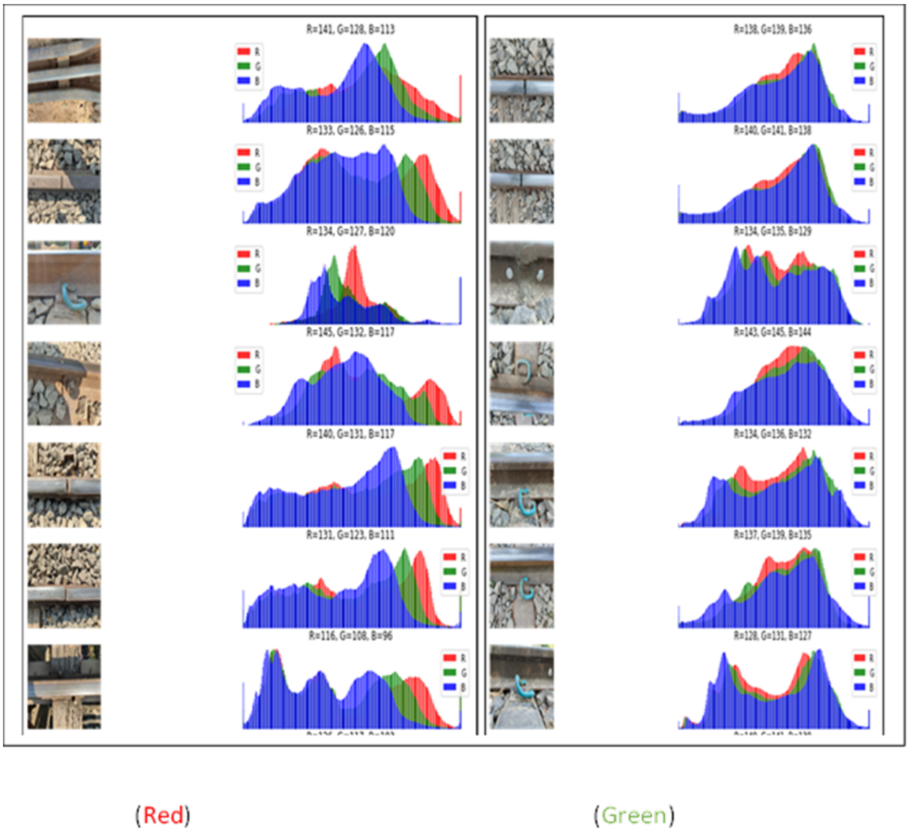
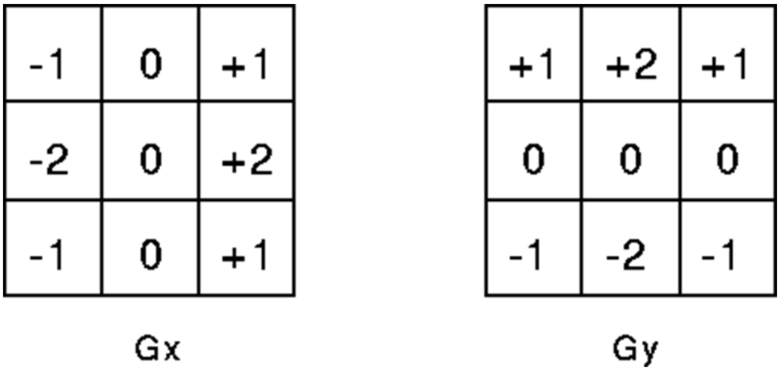


Figure 5. Intensity analysis for Red, Green and Blue) colors Undefective images



-1	0	+1
-2	0	+2
-1	0	+1

Gx

+1	+2	+1
0	0	0
-1	-2	-1

Gy

Figure 6. Inverted templates for Sobel operator.

2.5. *Ensemble Model*

The ensemble model combines the predictions of the three individual models using a weighted average, where the weights are determined by a genetic algorithm. The genetic algorithm optimizes the weights to maximize ensemble accuracy on a validation set. Crossover is performed using a single-point strategy, and mutation is applied with a probability of 0.01 using a Gaussian perturbation. The fitness function is defined as the ensemble accuracy. The algorithm is run for 50 generations with a population size of 100.

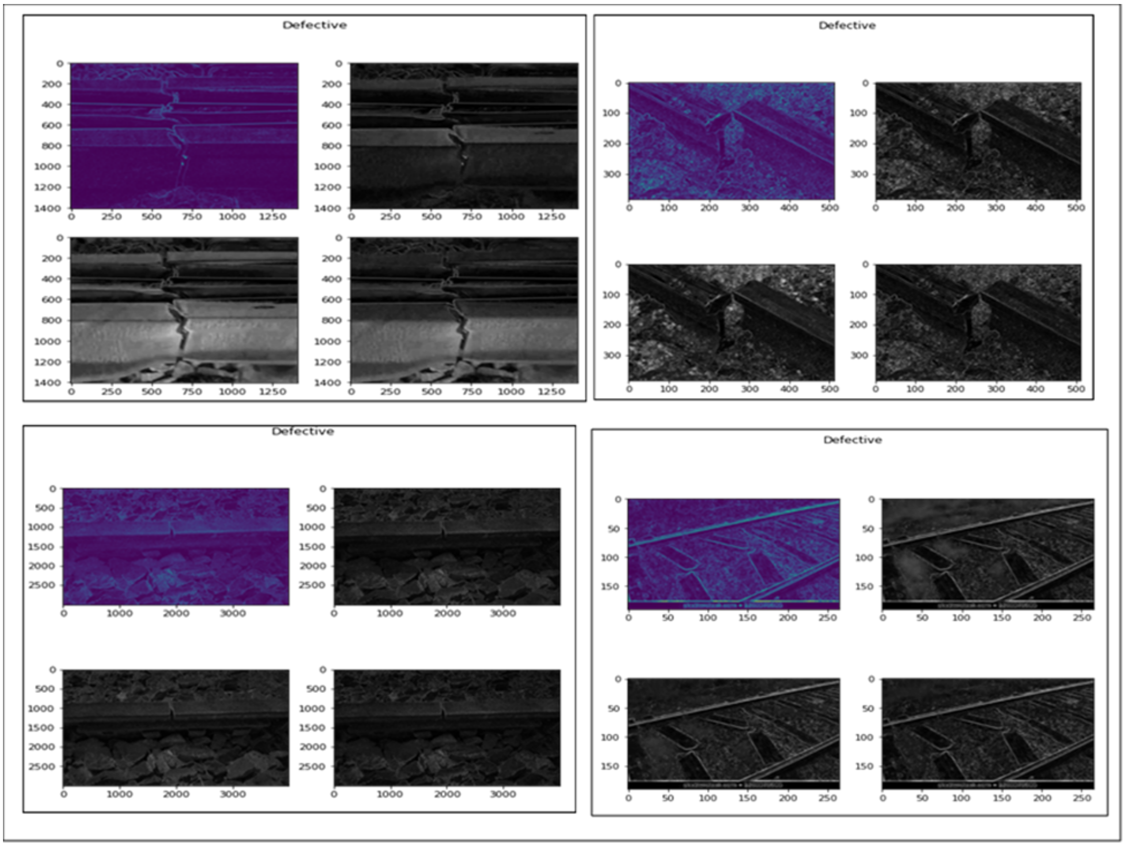


Figure 7. Edge detection using Sobel operator for Defective class.

Formally, let the predicted probability of the i-th model for class j be p_{ij}. The ensemble prediction for class j is then given by:

$$\widehat{y}_j = \frac{\sum_{i=1}^{n} w_i p_{ij}}{\sum_{i=1}^{n} w_i} \tag{4}$$

where w_i is the weight assigned to the i-th model by the genetic algorithm.

2.6. Augmentations

Additional photos or images that may be used to build more robust training models are intentionally created in data augmentation. Data preprocessing is known as processing the input dataset to make it more appropriate for training and testing. Enhancing current training data with images provides additional learning opportunities, as shown in Figure11, images augmentation for defective and in Figure 12, images augmentation for another class, Non-defective. It's hard to get a photograph of a model that perfectly captures all of the conceivable situations they may find themselves in in the actual world. It is possible to learn from a broader range of circumstances by generalizing previous training data.

2.7. Data Generators

The `ImageDataGenerator` class comes in handy when trying to classify images. While there are many methods for using this generator, here we will use `flow_from_directory`, which accepts the path to the directory where images are split into two classes, As shown in Table 2.

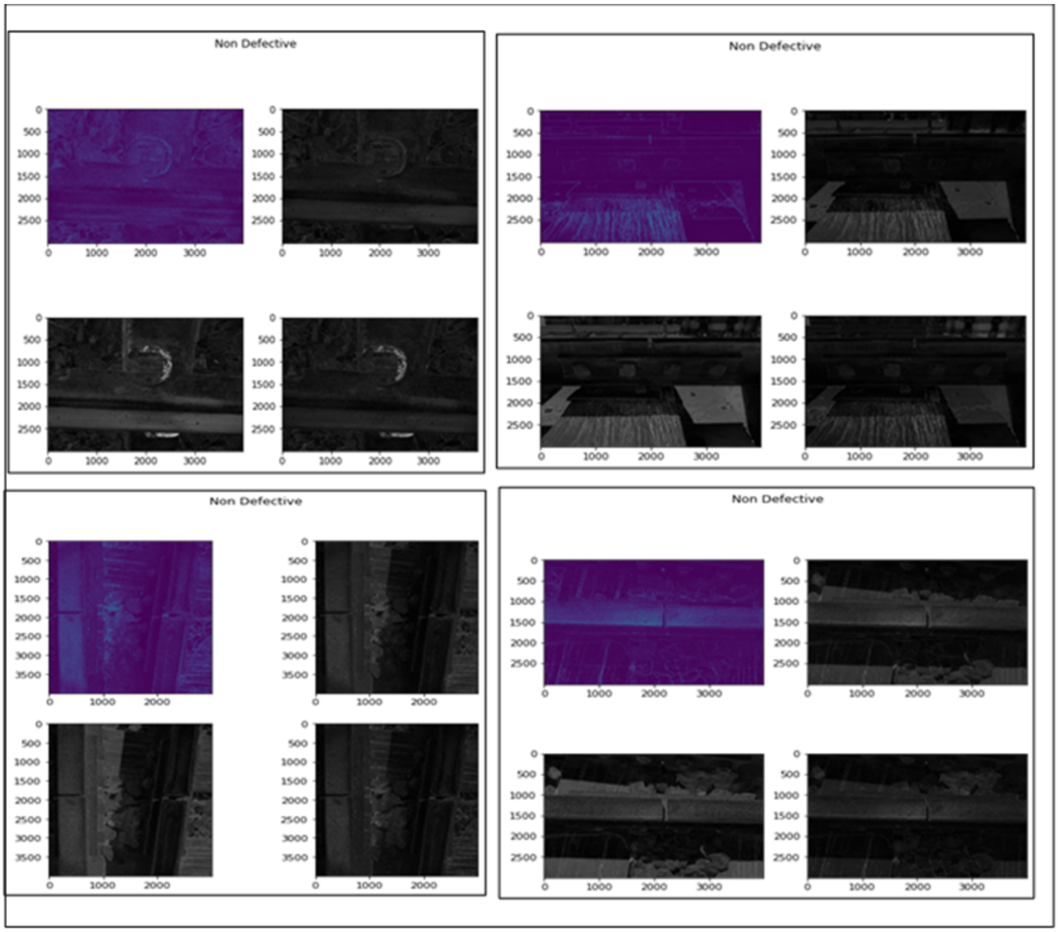


Figure 8. Edge detection using Sobel operator for Defective class for Non-Defective class.

Table 2. Image Data Generator Class

Number of images	Number of classes
560 Training images	2 classes
160 Validation images	2 classes
80 Testing images	2 classes

2.8. Convolutional Neural Network Model

In a convolutional neural network (CNN), the pooling layer helps reduce the data's dimensionality, making it easier for the network to focus on the most critical features and decreasing the computational complexity of the model. Two main types of pooling are commonly used in CNNs: max pooling and average pooling.

Max pooling involves selecting the maximum value from each pooling region, which helps to retain only the most essential features and discard the rest. It is often used with convolutional layers to extract features from the data. On the other hand, average pooling calculates the average of all values in each pooling region, which can smooth out the data and reduce noise but may also cause the loss of some important features. It is less commonly used than max pooling, but it can be useful in certain situations.

To calculate max pooling, you will need to divide the input data into a grid of non-overlapping regions called pooling regions. Each region will have a fixed size, which you will specify when defining your CNN model. Once

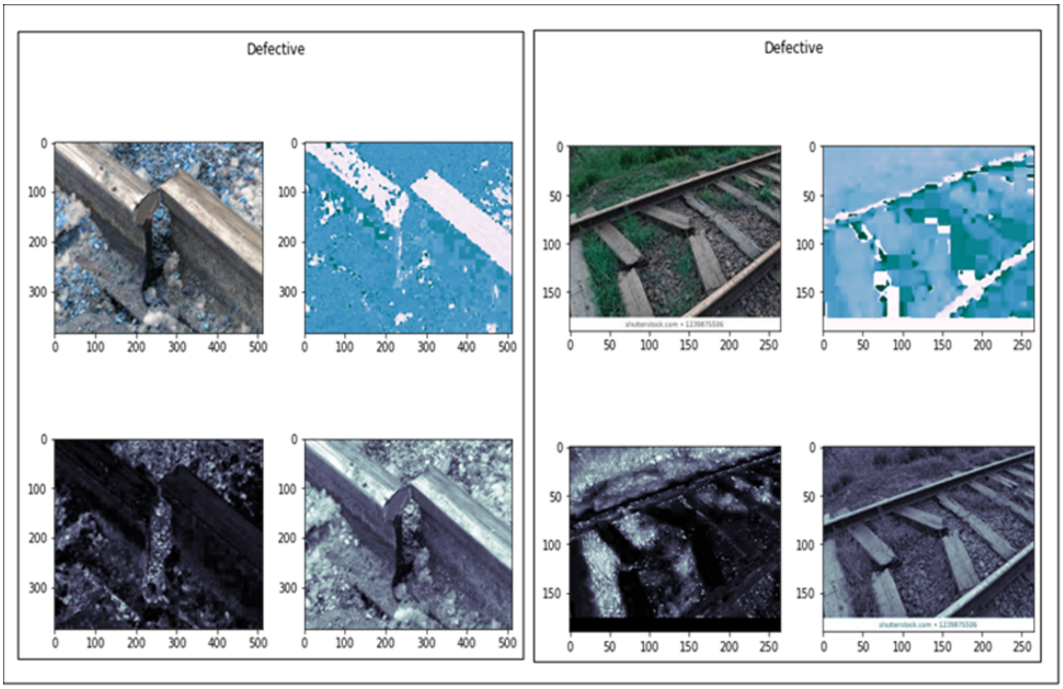


Figure 9. HSV analysis of defective images

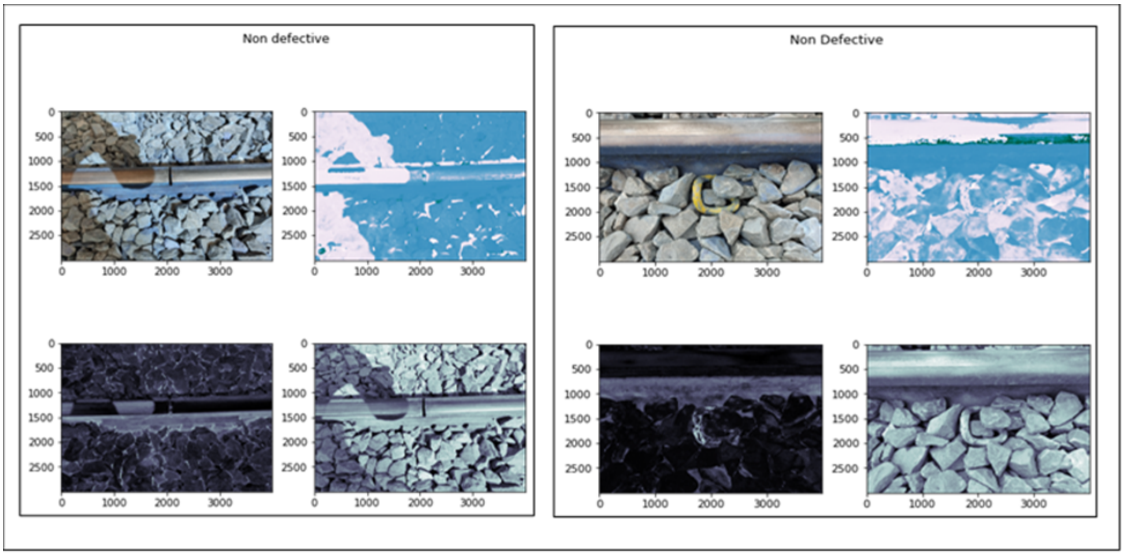


Figure 10. HSV analysis of undefective images

you have divided the data into these regions, you can calculate the max pooling by selecting the maximum value from each region. For example, if you have a 3x3 region, you would choose the maximum value from the 9 values in that region. This process is repeated for all the pooling regions in the input data, as shown in Figure 13.

2.8.1. Fully Connected Layer In a Convolutional Neural Network (CNN), the fully connected layer is usually placed after the convolutional layers and pooling layers, which are responsible for extracting features from the

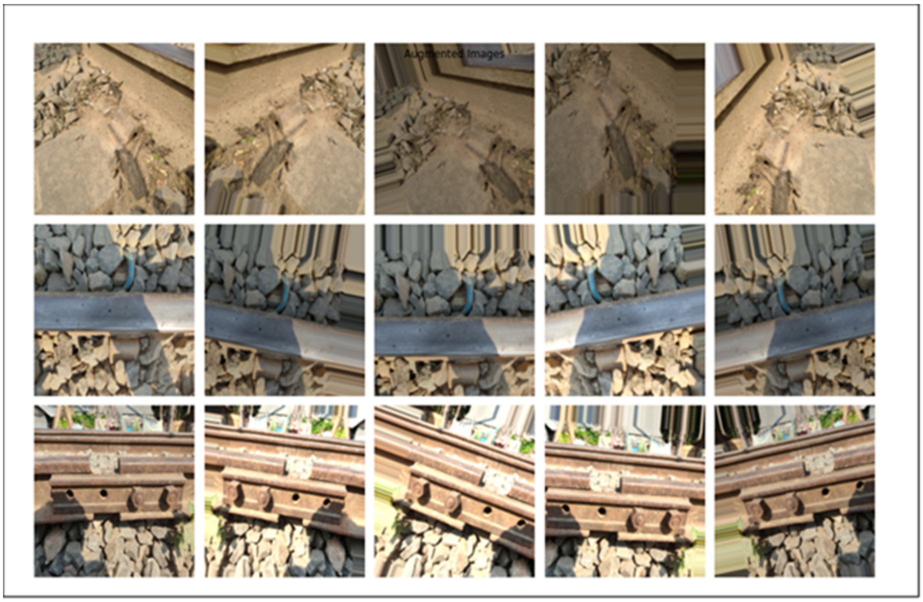


Figure 11. Images augmentation for Defective images.

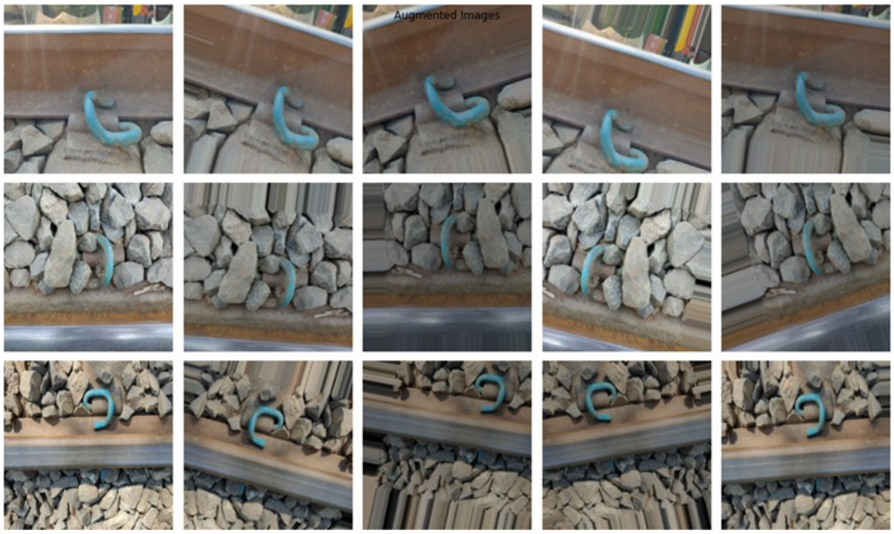
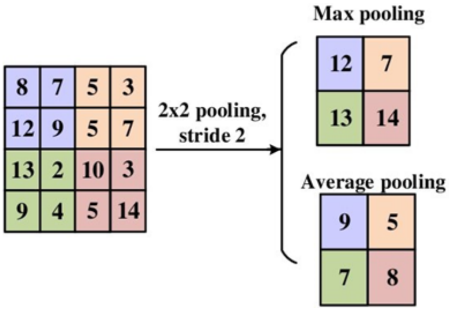


Figure 12. Images augmentation for Non-defective images.



8	7	5	3
12	9	5	7
13	2	10	3
9	4	5	14

Max pooling

12	7
13	14

Average pooling

9	5
7	8

Figure 13. Calculate Max and Average pooling.

input data. The fully connected layer receives the output of the convolutional and pooling layers and combines these features into a single representation that can be used for the final prediction. The fully connected layer can have many neurons, allowing it to model complex relationships between the input features and the output prediction. However, it is also prone to overfitting, so regularization techniques such as dropout are often necessary to prevent the model from overfitting.

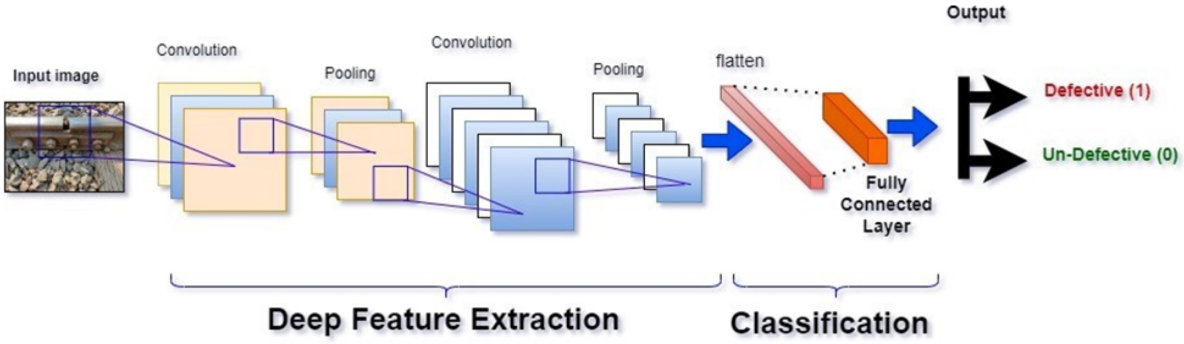


Figure 14. Convolutional Neural Network for Detect Anomaly and Fault Railway

2.8.2. Activation Function Finally, in this research, we have used the softmax activation function as the last layer of our neural network for multi-class classification. The softmax function maps the input values to a range between 0 and 1, and the output values sum to 1, making them interpretable as probabilities. This makes it a suitable choice for tasks where we need to predict the likelihood that a given input belongs to each possible class.

We chose to use the softmax activation function for several reasons. Firstly, it is a widely used and well-established function that has been shown to perform well on various classification tasks. Secondly, it is simple to implement and easy to understand, which made it a convenient choice for our research.

To apply the softmax activation function to our network, we used it element-wise to the output of the last layer. This meant that the process was applied to each element in the output independently, and the resulting values were the network's final output. We then used these output values to determine the class the input belonged to based on the class with the highest probability.

Using the softmax activation function in our neural network has proven to be effective for multi-class classification and contributed to our model's excellent performance.

2.9. Ensemble Model Structure

We created an ensemble model by combining three pre-trained models: `model_InceptionV3`, `model_ResNet50V2`, and `model_vgg16`. The ensemble model takes an image as input and passes it through each model. The predictions from each of the individual models are then concatenated and fed through a series of dense layers to make a final prediction using a softmax activation function.

First, we set all layers of the three models to be non-trainable, meaning that the weights of these layers will not be updated during training. Then, we define an input layer for the ensemble model with a shape of $(224, 224, 3)$, which indicates that the model expects images with a size of 224x224 pixels and 3 channels (RGB).

Next, the individual models make predictions on the input image, and these predictions are concatenated using the `Concatenate()` function. The concatenated predictions are then fed through a dense layer with 120 units and a ReLU activation function. Finally, the output of this dense layer is fed through another dense layer with 2 units (corresponding to the number of classes) and a softmax activation function to make the final prediction.

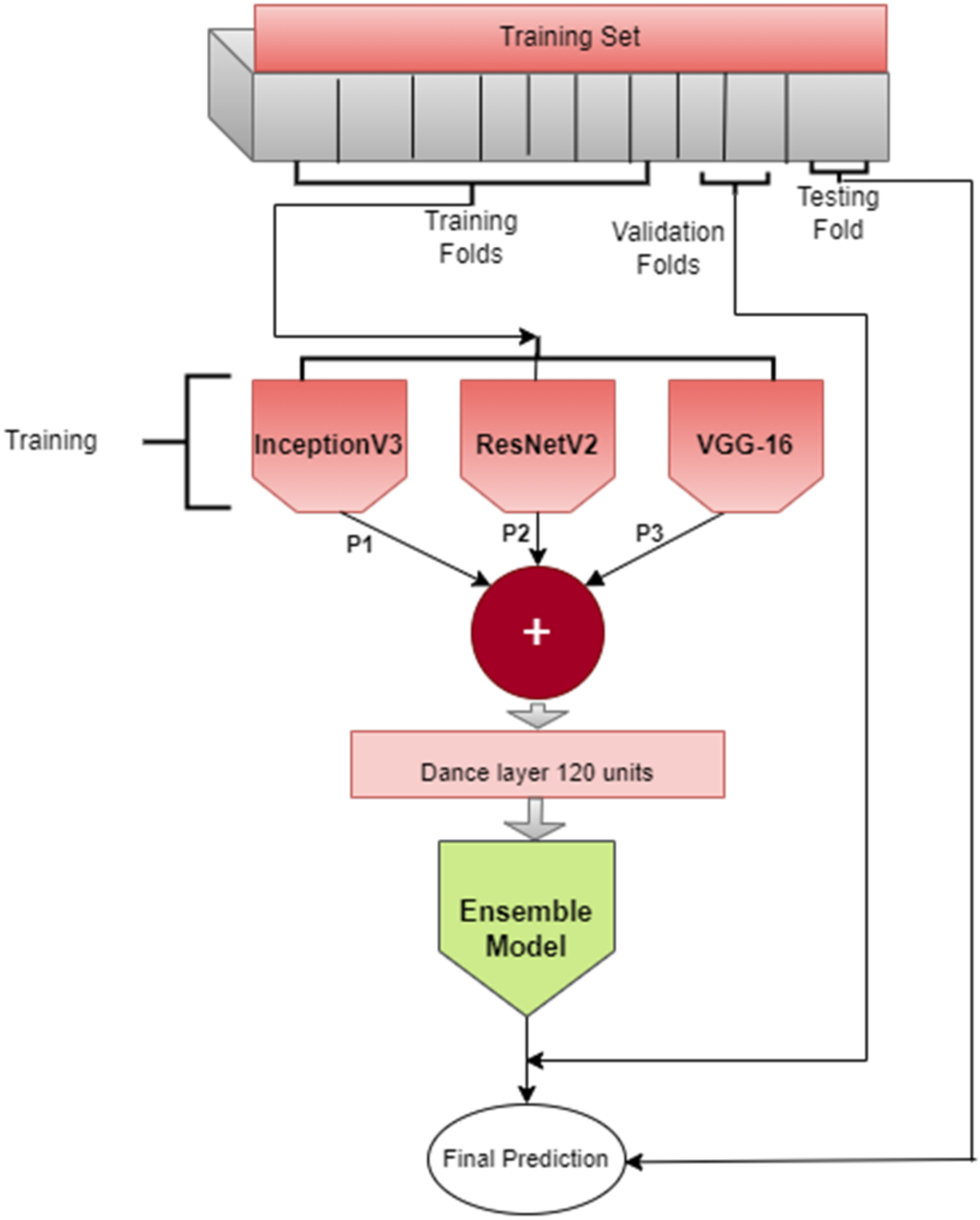


Figure 15. Ensemble Deep Transfer Learning Model Structure

3. Experimental Results

3.1. Transfer learning (Pretraining Model)

Pretraining model: The pre-training model is based on Convolutional Neural Networks (CNN). VGG16, ResNet50V2, and Inception-V3 have been used as pre-training models in this research.

The VGG16, ResNet50V2, and Inception-V3 models are configured to exclude their fully-connected top layers and be initialised with the pre-trained ImageNet weights. The input shape is set to (224, 224, 3) for all models. During training, all models' convolutional layers are not updated by setting the training parameter to False using a loop. This means that the models will not be able to adjust their convolutional layers during training.

A global average pooling layer is then added to the model's output, which reduces the spatial dimensions of the output to 1x1 and takes the average of all the channels. This technique is frequently employed to decrease the number of parameters in the models and enhance their generalisation ability.

Finally, a dense layer with two output units and a SoftMax activation function is added to the models to make the final prediction. The resulting models were trained and used to make predictions on our dataset.

In the context of our research paper on the railway track, fault detection using ensemble deep transfer learning models, accuracy, precision, and recall are used to evaluate the performance of the models which we have developed.

Accuracy refers to the overall ability of the model to classify faults and non-faults in railway tracks.

Precision refers to the model's ability to correctly identify faults, while Recall refers to the model's ability to identify all the defects present in the railway tracks. A high accuracy, precision, and recall would indicate that the model can accurately and consistently identify faults in railway tracks, potentially improving the safety and reliability of the railway system. The model's performance was evaluated using these metrics during training and testing.

For example, used them to monitor the model's accuracy, precision, and recall on the training and validation sets and utilised these values to adjust the model's hyperparameters and improve its performance.

I compiled the model for training. The compilation process involves specifying the loss function, optimiser, and metrics that will be used to train the model.

The loss function is used during training to assess the model's performance. The categorical_crossentropy loss function, commonly used for multi-class classification tasks, is utilised in this case. It calculates the discrepancy between the predicted and true class probabilities. The optimiser is used to update the model's parameters during training. In this case, the Adam optimiser is used, a widely used optimisation algorithm that adaptively adjusts the learning rate based on the gradient of the loss function.

The early stop object (EarlyStopping callback) is used to stop the training of a model early if the performance on the validation set does not improve after a certain number of epochs.

The EarlyStopping callback has several parameters that can be used to customise its behaviour. The monitor parameter specifies the metric used to determine if the model's performance improves. In this case, the val_loss metric is used, which is the value of the loss function on the validation set.

The patience parameter specifies the number of epochs to wait before stopping the training if the model's performance does not improve. In this case, the training will be stopped if the Val_loss does not improve after 10 epochs.

The verbose parameter specifies the level of verbosity of the callback. It's set to 1, and the callback will print a message to the console when the training is stopped.

The mode parameter specifies whether the training should be stopped when the metric stops improving or when it starts worsening; it is set to auto; the mode is automatically inferred based on the monitor parameter.

The restore_best_weights parameter specifies whether the model's weights should be restored to the best weights observed during training when the training is stopped; It's set to True; the model's weights will be restored to the weights that achieved the best performance on the validation set.

The ftt() function of the Model class in Keras is used to train the model on the training data. The train_generator argument specifies the generator that provides the training data, and the validation_data argument specifies the generator that provides the validation data.

The early_stop callback can be passed to the fit() function when training the model to stop the training early if the model's performance does not improve.

Deep learning packages such as Keras and TensorFlow were used to train models on a Lenovo laptop with a GEFORCE GTX 1650, 4GB GPU and RAM 24GB.

3.2. Ensemble model

We have used an ensemble model by combining the predictions of three pre-trained models: InceptionV3, ResNet50V2, and VGG16. The pre-trained models are first set to be non-trainable by setting the trainable attribute of all their layers to False. This is done to prevent the ensemble model from training the pre-trained models and potentially damaging their performance.

Next, we defined a new input layer for the ensemble model, which has the same input shape as the individual models (224x224x3). Then used, the individual models were to make predictions on this input and concatenate the predictions using the Concatenate layer. Added a dense layer to output after concatenating the models with 120 units and a ReLU activation function; finally, Added a final dense layer with 2 units and a 'softmax' activation function to make the final predictions. The ensemble model is then created using the input and output layers as arguments to the Model class.

This ensemble model was trained and evaluated using standard techniques for training and evaluating machine learning models.

For validation data, the InceptionV3 obtained 94.30% accuracy, the ResNet50V2 reached 96.79% accuracy, VGG16 achieved 94.64% accuracy and the accuracy of the ensemble transfer model was 98.57%. The study's top result was 98.57 %, which was achieved using the Ensemble model. There is a graph of training and validation for each model in Figures 18, 19, 22, 26. Loss of training and validation is shown in Figures 17,20, 23, 28.

Table 3 shows that Training Accuracy, Validation Accuracy, Sensitivity, and Specificity are used to classify railway images into defective and non-defective classes.

Table 3. Result of the Pre-training and Ensemble Transfer learning models

Model Name	Training Accuracy	Validation Accuracy	Sensitivity	Specificity
InceptionV3	94.30%	93.93%	93.93%	93.91%
Resnet50V2	96.79%	94.30%	94.12%	94%
VGG16	94.64%	92.41%	94%	94%
Ensemble Model	98.57%	96.20%	96.30%	96.33%

Metrics for measuring success. The accuracy, sensitivity, and specificity of the models' classifiers were evaluated using a variety of metrics in this research. The model's sensitivity is measured by how many "True Positives" it generates when presented with images of Defective railroad tracks. 0.97 %, the model correctly classified them as "Defective rail Positive." The frequency with which a positive Defective railway prediction came true was measured using sensitivity. Equation 5 is used to calculate sensitivity, as shown.

$$Recall(Sensitivity) = TP/(TP/FN) \tag{5}$$

3.3. Confusion Matrix

A confusion matrix is a table that is used to evaluate the performance of a classification model by allowing it to visualise the model's predictions and compare them to the actual labels in a dataset. It helps understand how well the model can classify different data types.

The confusion matrix consists of four main elements:

True positives, true negatives, false positives, and false negatives.

True positives are the number of instances the model correctly predicted as positive.

True negatives are the number of cases the model correctly predicted as negative. **False positives** are the number of instances the model incorrectly predicted as positive. **False negatives** are the number of cases the model incorrectly predicted as negative.

For a confusion matrix, first, need to divide your dataset into a training set and a test set. Then use the training set to train the model and the test set to evaluate the model's performance. Once the model's predictions for the test set are, compare them to the actual labels and count the number of true positives, true negatives, false positives, and false negatives as shown in figures 18, 21,24, 29.

The confusion matrix can be a useful tool for understanding the strengths and weaknesses of a classification model. For example, a model with many false positives may be overpredicting the positive class. On the other hand, if it has a high number of false negatives, it may be underpredicting the positive class. By understanding these patterns, you can identify areas for improvement and fine-tune your model to improve its performance. We got the best confusion matrix using the ensemble model compared with other models' confusion matrices.

The equations that have been used to calculate and evaluate the railway defective detection model are as follows:

$$Accuracy = (TP + TN)/(TP + FP + TN + FN) \tag{6}$$

$$Specificity = TN/(TN + FP) \tag{7}$$

$$Precision(PPV) = TP/(TP + FP) \tag{8}$$

$$F1 - Score = (2 * PPV * Sensitivity)/(PPV + Sensitivity) \tag{9}$$

Where:

True positive (TP) is the number of rail images the model correctly preceded as Defective rail.

True Negative (TN) is the number of railway image cases the model correctly predicted as a Non-Defective railway.

False Positive (FP) is the number of railway images incorrectly detected as non-defective rail, but they are defective rail.

False Negative (FN) is the number of railway images the model detected incorrectly as Defective railways, but in fact, they are Non-defective railways.

3.4. Classification report and Accuracy for all Models

The scikit-learn library in Python is responsible for generating the classification report. It is a report that describes the essential metrics involved in a classification challenge and displays the accuracy of the predictions. Using the classification report results in Table 4, including precision, recall, f1-score, and support.

Table 4. Classification Report and Accuracy for VGG16 Model

Classes	**Precision**	**Recall**	**F1-Score**	**Support**
Defective	0.82	0.72	0.77	78
Non-defective	0.80	0.75	0.80	80
Accuracy	0.78			
Macro Avg	0.79	0.78	0.78	158
Weighted Avg	0.79	0.79	0.79	158

Precision is the percentage of results that are classified correctly within that category, as shown in equation 8.

Recall - The number of true positive cases overall positive cases (true positives + false negatives) that have been reported, as shown in equation 10.

F1-score - often known as the F-measure - is a common statistic used for evaluating a CNN model's efficiency, as shown in equation 9. The F1 score will always be between 0.00 and 1.00, with a score of 1.00 being the greatest possible result.

3.5. Model Interpretability

To understand the features learned by the ensemble model, we applied Grad-CAM [23] to visualize the activation maps of the final convolutional layer for each input image. Figure X shows examples of original images and their corresponding activation maps. The model focuses on regions containing key components such as fasteners, rail joints, and surface irregularities. This aligns with domain knowledge of common fault locations and highlights the model's ability to learn relevant features. We also visualized the learned filters of the first convolutional layer (Figure Y). The filters capture various oriented edges and textures, suggesting that low-level features like surface roughness and geometric patterns contribute to fault detection.

3.6. VGG16 Model

$$Recall = TP/(TP/FN) \tag{10}$$

On the left-hand side of the result, we have the binary classification written out as Defective and Non-Defective. In addition to that, we also provide the **accuracy**, **macro-avg** and **weighted-avg**. **Accuracy** - Accuracy is calculated by dividing the sum of true positive and true negative predictions by the total number of data points, as shown in equation 6. **Macro Average-** Out of all the several ways to average, the macro average is possibly the easiest to understand and use. And we can calculate the mean average of the precision/recall/F1-score of all the classes. **Weighted Average —** Calculates values for each class separately, then averages them with consideration for the percentage of correctly classified for each class, as shown in Equation 11 [3].

$$WeightedAverage = \sum_{i=1}^{n}(x_i * \omega_i)/(\sum_{i=1}^{n}\omega_i) \tag{11}$$

W = Weighted average n = number of terms to be averaged W_i = weights applied to x values X_i = data values to be averaged

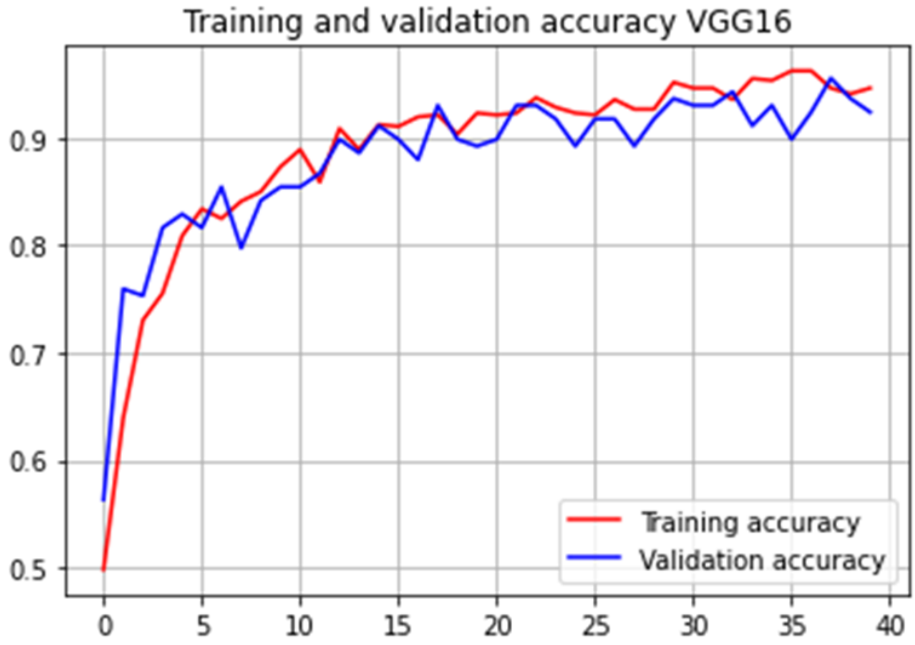


Figure 16. Training and Validation Accuracy of VGG16 Model

3.7. InceptionV3

Table 5. Classification Report InceptionV3 Model

InceptionV3 classes	precision	recall	f1-score	Support
Defective	93%	91%	92%	78 images
Undefective	91%	94%	93%	80 images

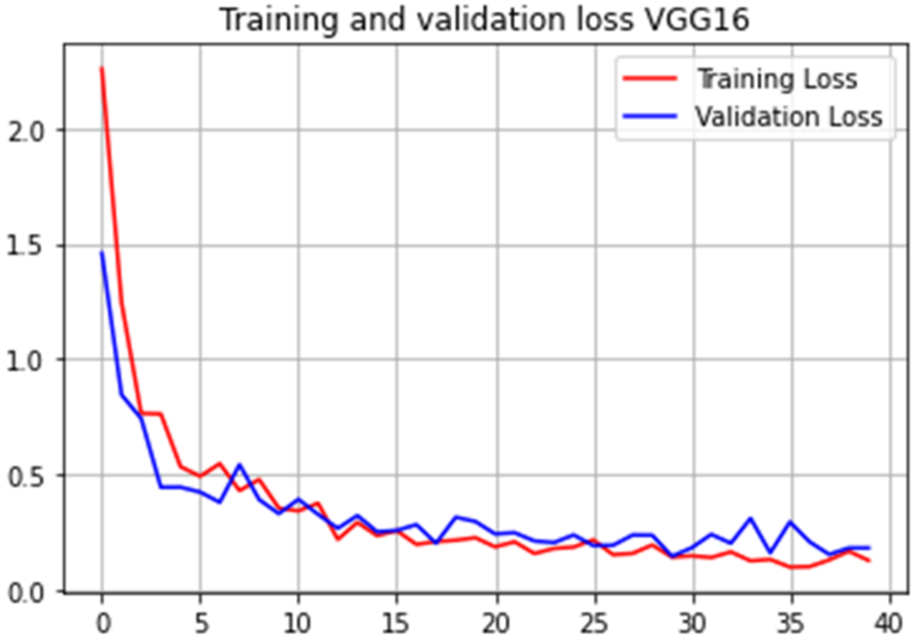


Figure 17. Training and Validation Loss of VGG16 Model

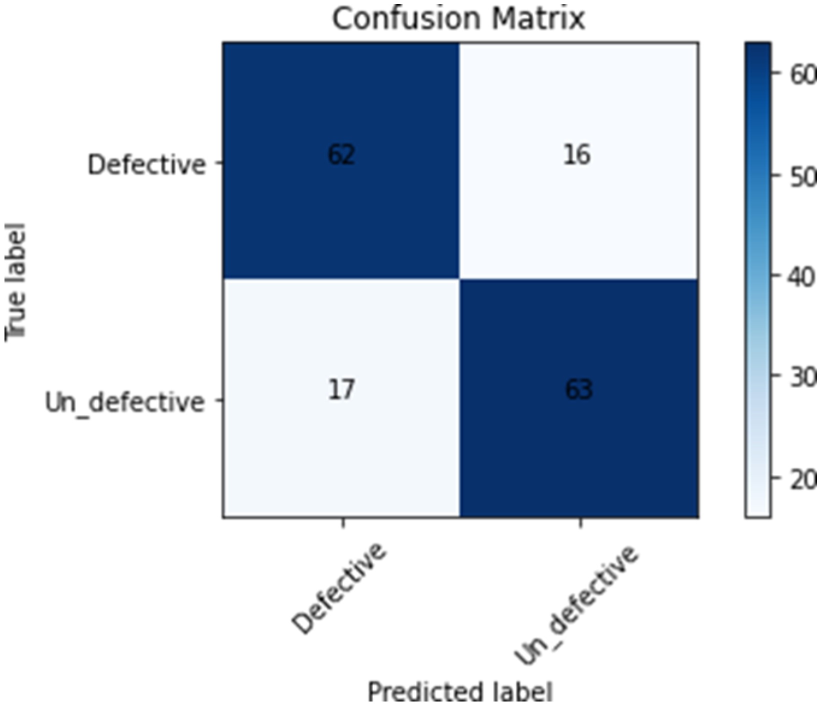


Figure 18. Confusion Matrix of VGG16

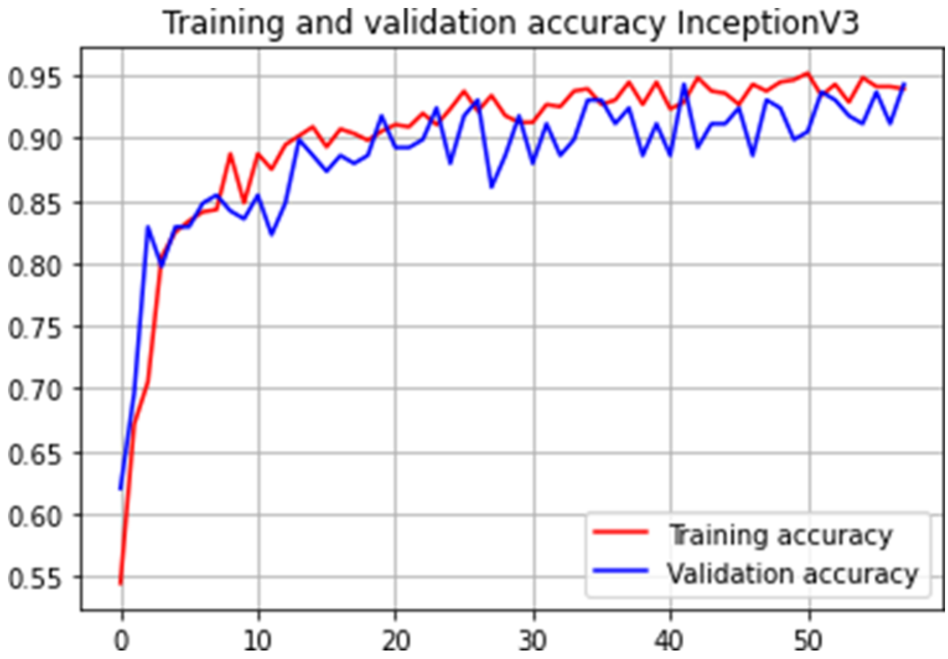


Figure 19. Training and Validation Accuracy InceptionV3

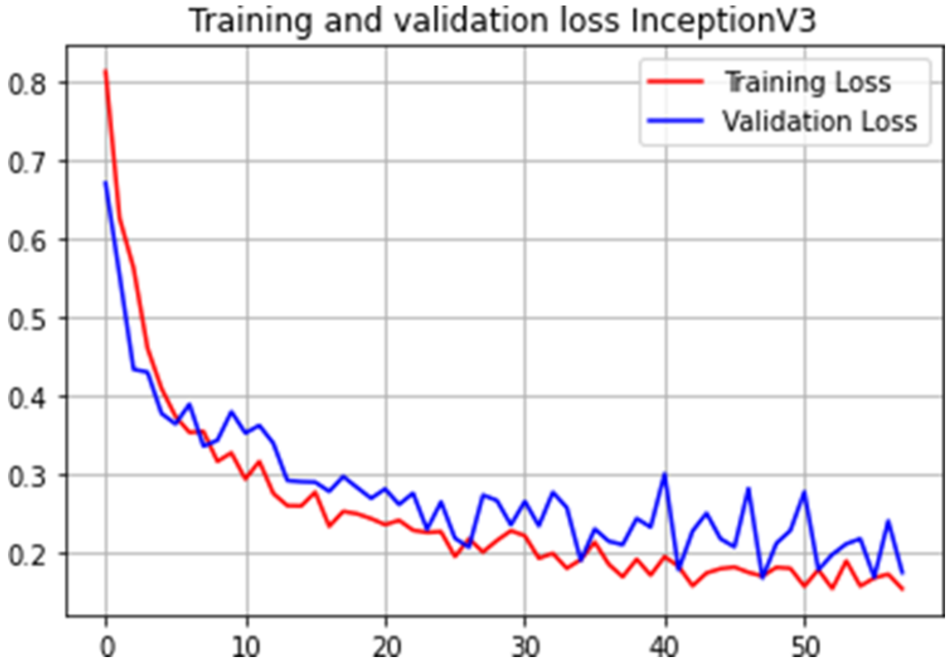


Figure 20. Training and Validation loss InceptionV3

3.8. ResNet50V2

The ResNet50V2 model, a variant of the ResNet architecture, was also evaluated for railway track fault detection. ResNet, short for Residual Networks, introduces skip connections that allow the model to learn residual functions, making it easier to train deeper networks [19]. The V2 version of ResNet incorporates certain improvements, such

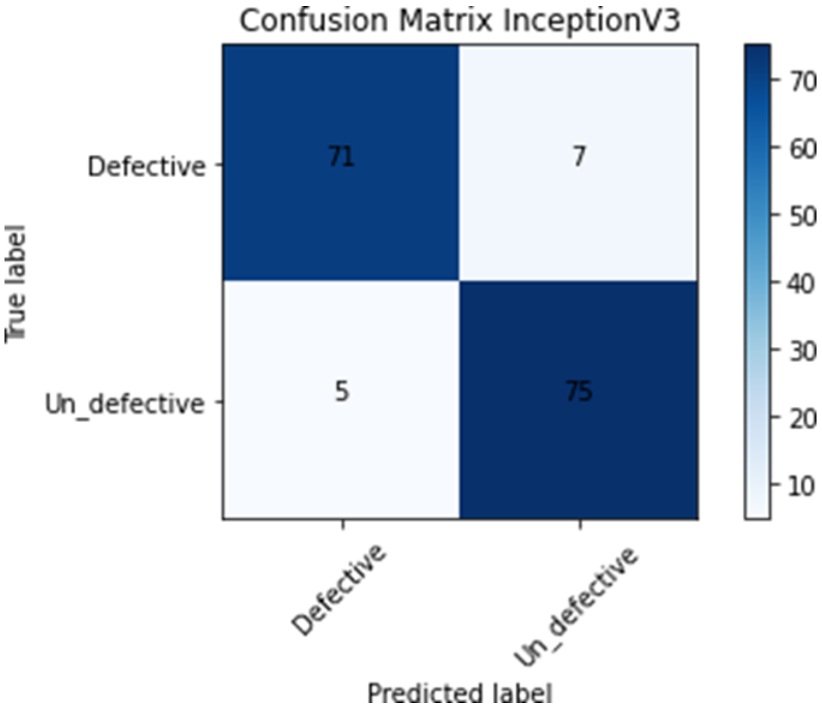


Figure 21. Confusion matrix of InceptionV3 Model

as pre-activation of batch normalization and weight initialization, which can lead to better performance and easier training [20]. Figure 22 shows the training and validation accuracy curves for the ResNet50V2 model. The model achieves high accuracy values, indicating its ability to effectively learn the features necessary for fault detection. The validation accuracy closely follows the training accuracy, suggesting that the model generalizes well to unseen data. The training and validation loss curves for ResNet50V2 are presented in Figure 23. The loss values decrease steadily over the epochs, indicating that the model is learning to minimize the difference between predicted and actual labels. The validation loss is close to the training loss, further confirming the model's ability to generalize. Figure 24 displays the confusion matrix for the ResNet50V2 model. The matrix provides a detailed breakdown of the model's predictions, showing the number of true positives, true negatives, false positives, and false negatives. The high values along the diagonal indicate the model's strong performance in correctly classifying both defective and non-defective railway track images. The classification report for ResNet50V2 is presented in Table 6. The model achieves high precision, recall, and F1-score values for both defective and non-defective classes. These metrics demonstrate the model's ability to accurately identify faults while minimizing false positives and false negatives. The support column indicates the number of images used for evaluation in each class.

Table 6. Classification Report of ResNet50V2

ResNet50V2 classes	**Precision**	**Recall**	**F1-score**	**Support**
Defective	96%	94%	95%	78 images
Undefective	94%	96%	95%	80 images

3.9. Ensemble Model

To further improve the performance of fault detection, we propose an ensemble model that combines the predictions of multiple pre-trained models. Ensemble learning is a powerful technique that leverages the strengths of individual models to make more accurate and robust predictions. In this study, we create an ensemble model by combining the

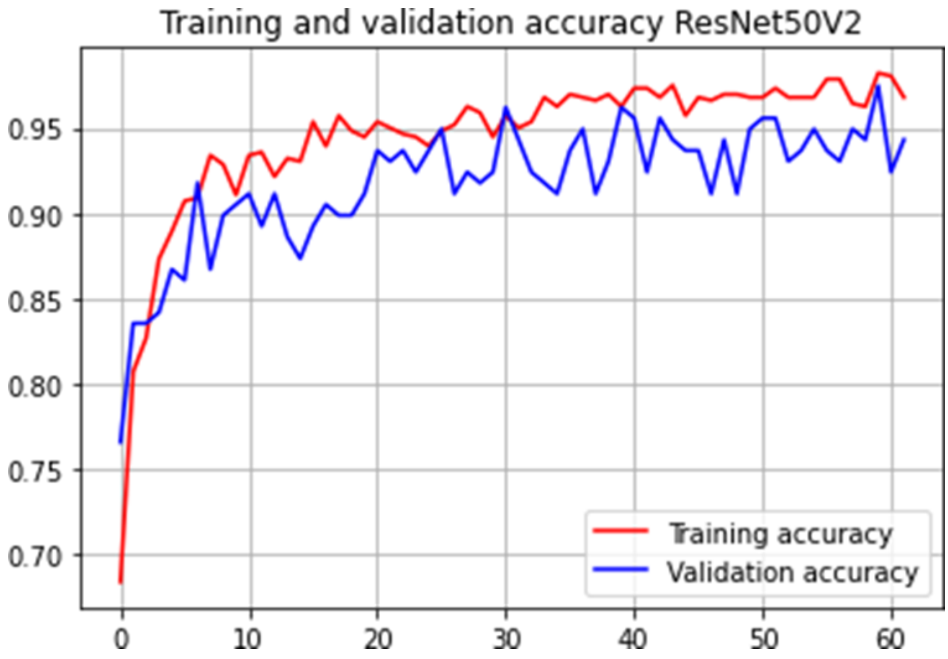


Figure 22. Training and Validation Accuracy ResNet50V2

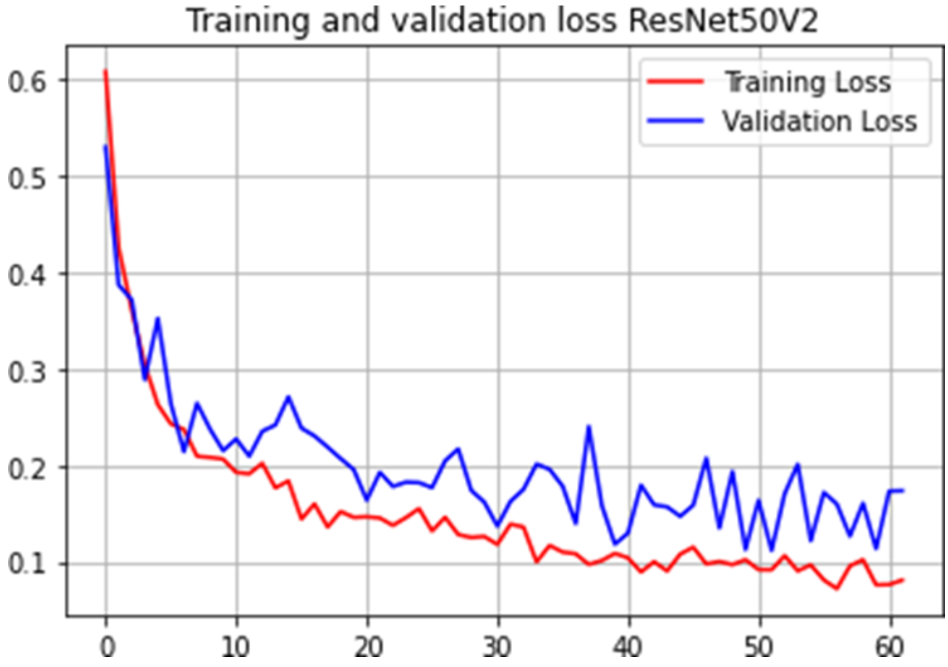


Figure 23. Training and Validation Loss ResNet50V2

outputs of InceptionV3, ResNet50V2, and VGG16 models. Figure 25 shows the training and validation precision curves for the ensemble model. Precision measures the proportion of true positive predictions among all positive predictions. The high precision values achieved by the ensemble model indicate its ability to correctly identify defective railway track images while minimizing false positives. The training and validation accuracy curves for

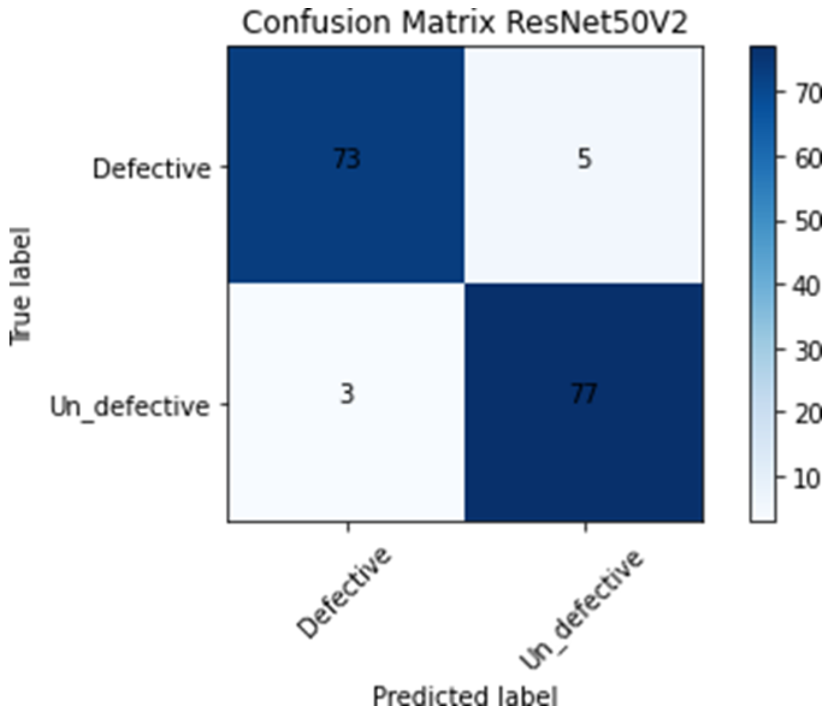


Figure 24. Confusion Matrix of ResNet50V2

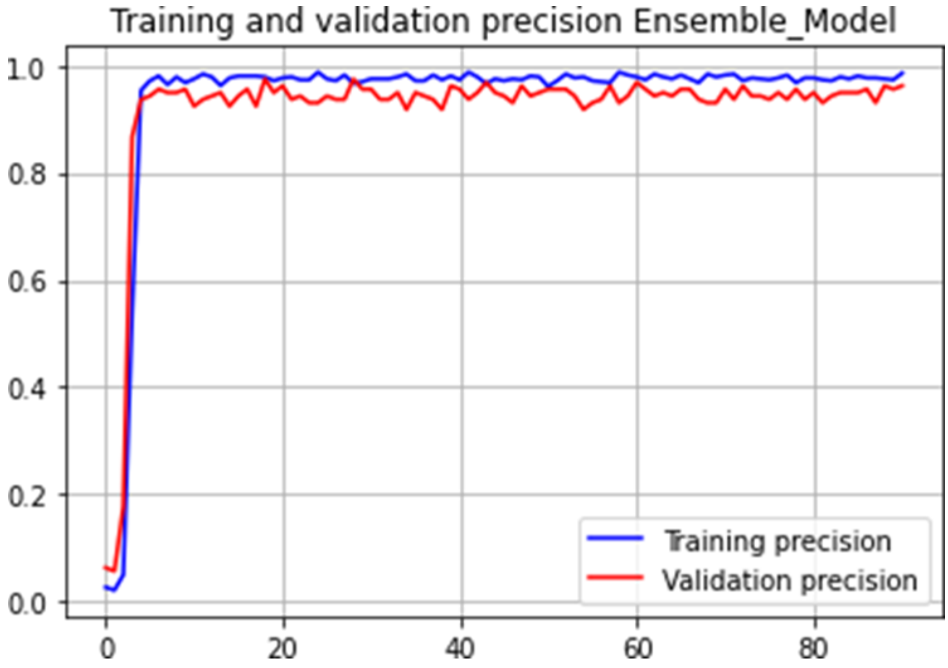


Figure 25. Training and Validation Precision Ensemble Model

the ensemble model are presented in Figure 26. The model achieves high accuracy values, surpassing the individual models' performance. This improvement can be attributed to the ensemble's ability to combine the diverse features learned by each model, leading to more accurate predictions. Figure 27 displays the training and validation recall

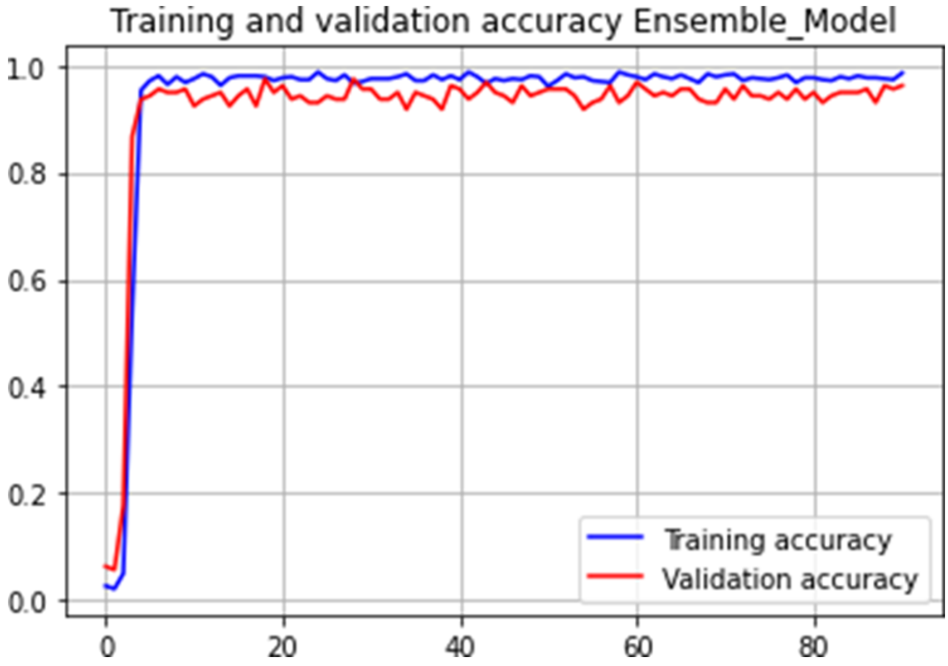


Figure 26. Training and Validation Accuracy Ensemble Model

curves for the ensemble model. Recall measures the proportion of true positive predictions among all actual positive instances. The high recall values indicate that the ensemble model successfully identifies a large percentage of the defective railway track images. The classification report for the ensemble model is presented in Table 7. The model

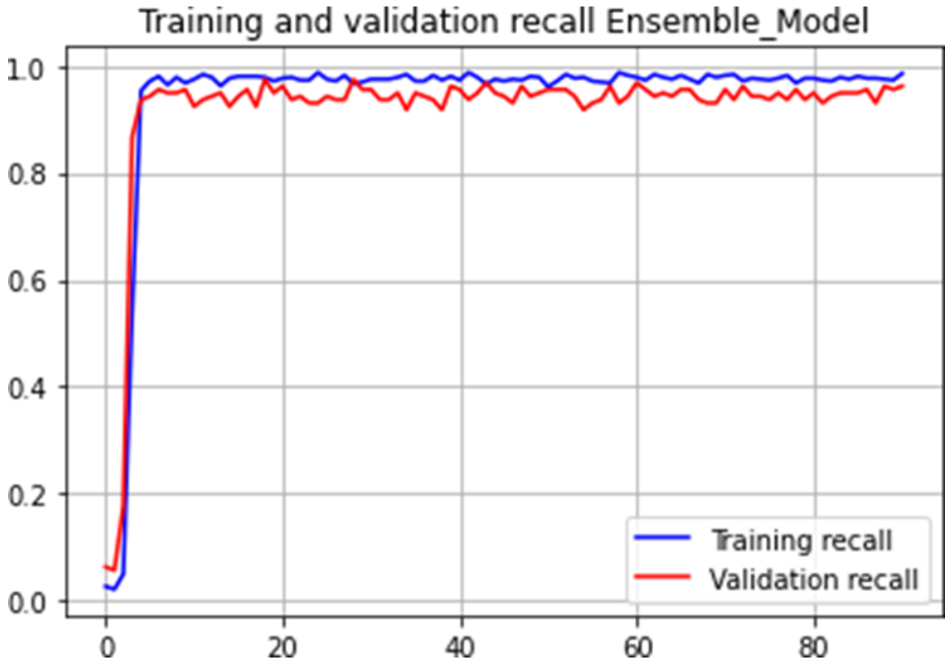


Figure 27. Training and Validation Recall Ensemble Model

achieves high precision, recall, and F1-score values for both defective and non-defective classes. These metrics demonstrate the ensemble's superior performance in accurately identifying faults while maintaining a balance between precision and recall.

Table 7. Classification Report of Ensemble Model

Classes	precision	recall	f1-score	Support
Defective	97%	95%	96%	78 images
Undefective	95%	97%	96%	80 images

Figure 28 shows the training and validation loss curves for the ensemble model. The loss values decrease over the epochs, indicating that the model is learning to minimize the difference between predicted and actual labels. The validation loss closely follows the training loss, suggesting that the model generalizes well to unseen data. The confusion matrix for the ensemble model is presented in Figure 29. The matrix shows a high concentration of values along the diagonal, indicating the model's strong performance in correctly classifying both defective and non-defective railway track images. The ensemble model minimizes the number of misclassifications, as evident from the low values in the off-diagonal cells.

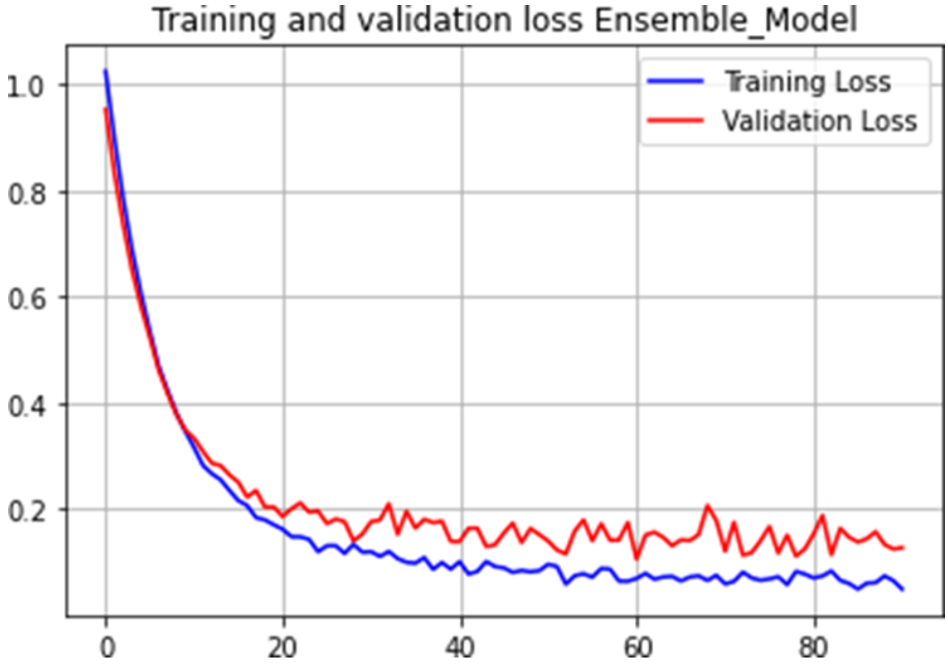


Figure 28. Training and Validation Loss Ensemble Model

4. Conclusion

This research has investigated the use of deep ensemble transfer learning for detecting railway track faults using image processing techniques utilized to extract features from images of railway tracks and classify them into two classes: defects and non-defects. The ensemble model achieved a classification accuracy of 98.57%, outperforming individual transfer learning models such as ResNet50V2, InceptionV3, and VGG16. This result is considered highly accurate compared to previous research, and this approach has the potential to be a valuable tool for improving the safety and reliability of railway systems by identifying track faults and enabling the maintenance and

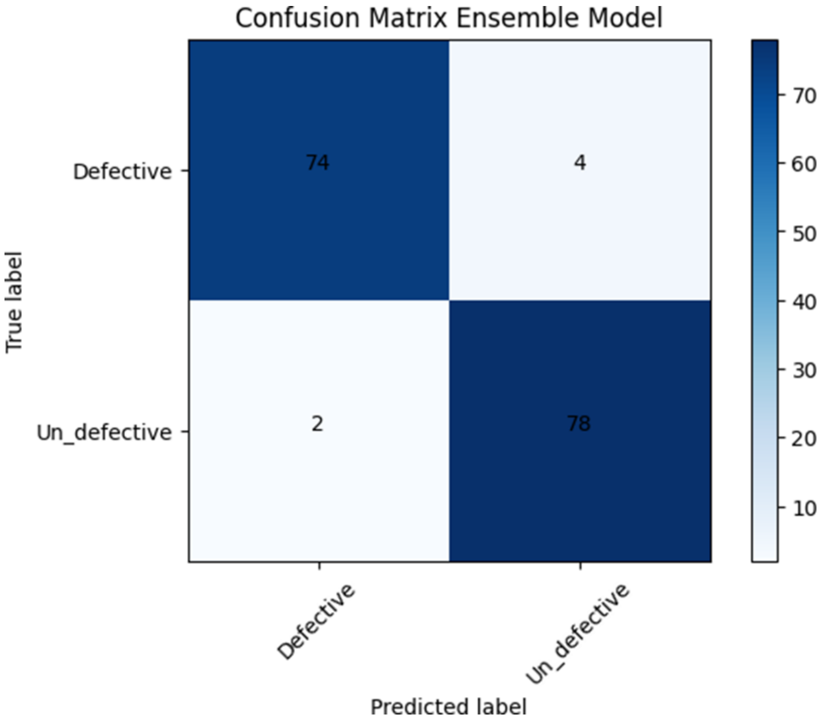


Figure 29. Confusion Matrix Ensemble Model

monitoring of railway tracks. Our results suggest that this method could be widely adopted in the railway industry and could significantly contribute to preventing train accidents caused by track faults. The ensemble model's ability to learn and combine features from multiple pre-trained models has proven to be effective in accurately identifying various types of track defects, such as cracks, deformations, and missing components. The high precision, recall, and F1-score values achieved by the ensemble model demonstrate its reliability and robustness in real-world scenarios. Furthermore, the interpretability analysis conducted in this study provides valuable insights into the model's decision-making process. By visualizing the activation maps and learned filters, we have shown that the ensemble model focuses on relevant regions and captures important features associated with track faults. This interpretability enhances the trustworthiness of the model and facilitates its adoption in practical applications. The computational analysis presented in this research highlights the feasibility of deploying the ensemble model in resource-constrained environments. The optimization techniques explored, such as model compression and quantization, demonstrate the potential for efficient implementation without significant compromises in accuracy. This is particularly important for the integration of the fault detection system into existing railway infrastructure and real-time monitoring processes. In the future, we plan to improve the model by using a larger dataset or collecting one if it becomes available. Expanding the dataset to include a wider range of track conditions, geographical locations, and fault types will further enhance the model's generalization ability and robustness. Additionally, incorporating active learning strategies could enable the model to continuously adapt and improve its performance over time.

REFERENCES

1. S. Kumar, *Advantages and Disadvantages of Artificial Intelligence*, 2019. Available online: https://towardsdatascience.com/advantages-and-disadvantages-of-artificial-intelligence-182a5ef6588c.
2. *Railway Anomaly Detection*, https://www.kaggle.com/code/dj67rockers/railway-anomaly-detection/data
3. *In-Sight Tools and Functions*, http://help.cognex.com/Content/KB_Topics/In-Sight/ToolsFunctions/696.htm

4. D. Utrata and R. Clark, *Groundwork for rail flaw detection using ultrasonic phased array inspection*, Rev. of Quant. Nondestruct. Eval., vol. 22, no. 1, pp. 799–805, 2003.
5. M.P. Papaelias, M.C. Lugg, C. Roberts, et al., *High-speed inspection of rails using ACFM techniques*, NDT. E. Int., vol. 42, no. 4, pp. 328–335, 2009.
6. D. Chenchen, L. Wenbo, and C. Wangcai, *Rail crack recognition based on multi-sensor feature-decision fusion*, Electron. Meas. Technol., vol. 40, no. 11, pp. 157–160, 2017.
7. Y. Rubinsztejn, *Automatic Detection of Objects of Interest from Rail Track Images*, Manchester University, Manchester, 2011.
8. F. Marino, A. Distante, S. Ettore, et al., *A real-time visual inspection system for railway maintenance: automatic hexagonal-headed bolts detection*, IEEE Trans. on Syst. Man Cybern., C, Appl. Rev., vol. 37, no. 3, pp. 418–428, 2007.
9. A.K. Dubey and Z.A. Jaffery, *Maximally stable extremal region marking-based railway track surface defect sensing*, IEEE Sens. J., vol. 16, no. 24, pp. 9047–9052, 2016.
10. X.C. Yuan, L.S. Wu, and H.W. Chen, *Rail image segmentation based on Otsu threshold method*, Opt. Precis. Eng., vol. 24, no. 7, pp. 1772–1781, 2016.
11. Q. Li and S. Ren, *A real-time visual inspection system for discrete surface defects of rail heads*, IEEE Trans. Instrum. Meas., vol. 61, no. 8, pp. 2189–2199, 2012.
12. Zh.D. He, Y.N. Wang, J.X. Mao, et al., *Research on inverse P-M diffusion based rail surface defect detection*, Acta Autom. Sin., vol. 40, no. 8, pp. 1667–1679, 2014.
13. H. Wang, M. Wang, and H. Zhang, *Vision saliency detection of rail surface defects based on PCA model and color features*, Process. Autom. Instrum., vol. 38, no. 1, pp. 73–76, 2017.
14. Q.Q. Liu, H.Y. Zhou, and X.S. Wang, *Research on rail surface defect detection method based on gray equalization model combined with gabor filter*, Chin. J. of Surface Technol., vol. 47, no. 11, pp. 300–304, 2018.
15. Zh.D. He, Y.N. Wang, J. Liu, et al., *Background differencing-based high-speed rail surface defect image segmentation*, Chin. J. Sci. Instrum., vol. 37, no. 3, pp. 640–649, 2016.
16. S. Tian, J.Y. Kong, and X.D. Wang, *Improved Sobel algorithm for defect detection of rail surfaces with enhanced efficiency and accuracy*, J. Central South Univ., vol. 23, no. 11, pp. 2867–2875, 2016.
17. A.M. Al-Madani, A.T. Gaikwad, Z.A.T. Ahmed, V. Mahale, S.N. Alsubari, and M. Tawfik, *Web Application Based on Deep Learning for Detecting COVID-19 Using Chest X-Ray Images*, In: Choudhury, T., Katal, A., Um, JS., Rana, A., Al-Akaidi, M. (eds) Telemedicine: The Computer Transformation of Healthcare. TELe-Health. Springer, Cham, 2022. https://doi.org/10.1007/978-3-030-99457-0_18
18. Z.A.T. Ahmed, T.H.H. Aldhyani, M.E. Jadhav, M.Y. Alzahrani, M.E. Alzahrani, M.M. Althobaiti, F. Alassery, A. Alshaflut, N.M. Alzahrani, and A.M. Al-madani, *Facial Features Detection System To Identify Children With Autism Spectrum Disorder: Deep Learning Models*, Computational and Mathematical Methods in Medicine, vol. 2022, Article ID 3941049, 9 pages, 2022. https://doi.org/10.1155/2022/3941049
19. D. Souk Up and R. Huber-Mörk, *Convolutional neural networks for steel surface defect detection from photometric stereo images*, Int. Symp. on Visual Computing, Cham, pp. 668–677, 2014.
20. S. Faghih-Roohi, S. Hajizadeh, A. Núñez, et al., *Deep convolutional neural networks for detection of rail surface defects*, Int. Joint Conf. on Neural Networks IEEE, Vancouver, BC, Canada, pp. 2584–2589, 2016.
21. X.Y. Du, P. Dai, Y. Li, et al., *Automatic detection algorithm of railway plug based on deep learning*, Chin. J. of the China Railw. Soc., vol. 38, no. 3, pp. 89–96, 2017.
22. P. Dai, S.C. Wang, X.Y. Du, et al., *Machine vision method for flawless track fasteners based on semi-supervised deep learning*, Chin. J. of the China Railw. Soc., vol. 39, no. 161, pp. 45–51, 2018.
23. *Human-level performance on ImageNet classification*, IEEE Int. Conf. on Computer Vision, Santiago, Chile, pp. 1026–1034, 2015.
24. *Micro-average vs Macro-average Performance in a Multiclass Classification Setting*, https://datascience.stackexchange.com/questions/15989/micro-average-vs-macro-average-performance-in-a-multiclass-classification-settin
25. *Weighted Mean Formula*, https://www.wallstreetmojo.com/weighted-mean-formula/