# Project Report:
# Defective Rail Classification from Track Images

*Computation for Transportation Engineering 2022/2023*

*Name: Ruowei Li  Student Number: 22323197  Date:5/7/2023*

## 1 Introduction

Railways are the backbone of the transportation industry, providing a reliable and cost-effective means of moving goods and people across long distances. However, with millions of tons of cargo and thousands of passengers being transported each day, railway safety is of paramount importance. One of the most critical components of railway safety is the track itself, and any defects or issues with the track can lead to serious accidents.

One common issue with railway tracks is defective rails, which can cause derailments and other accidents. Detecting and classifying defective rails is a challenging task, as it requires visual inspection of thousands of miles of track, which can be time-consuming and expensive. With the recent advancements in computer vision and machine learning, however, it is now possible to automate this process using images captured from track inspections.

The identification of defective rails is a crucial maintenance activity for railway tracks, as it represents one of the primary failure modes. Track defects pose significant safety risks and can result in service disruptions, which can have adverse economic consequences. Timely detection and correction of such defects can yield significant safety and economic benefits, including a reduction in accidents and minimization of traffic interruptions.

In this project, I propose a deep learning-based approach for detecting defective rails from track images. I use EfficientNet (architecture of EfficientNet is shown in the figure 1.1), a state-of-the-art convolutional neural network architecture, to classify track images as either healthy or defective rails. I trained and evaluated my model on a large dataset of track images, achieving high accuracy in classifying defective rails.

My approach has the potential to significantly reduce the time and cost involved in inspecting railway tracks, while improving safety by detecting defective rails before they cause accidents. In this project report, I provide a detailed description of my approach and methodology, as well as insights into the challenges and opportunities of using EfficientNet for defective rail classification. I also discuss the implications of my results for the safety and efficiency of railway systems.

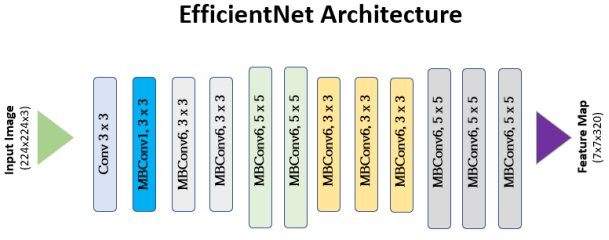


Figure 1.1 Architecture of EfficientNet

## 2 Methodology

### 2.1 Environment construction and pre-settings

To implement a classifier for separating images of healthy tracks from defective ones based on EfficientNet, a deep learning method, I first carried out the initialization and pre-setting of the environment. This part mainly includes:

1. Download tensorflow and keras-related data files through the command line in the anaconda prompt to generate the environment for building and training the neural network.

2. Download and install the EfficientNet package, and import the related library

3. Import dataset and check the correctness of the data of the dataset, check the initialization data, and check and verify by calculating and displaying the statistical data of the data and some related images to ensure the accuracy of the dataset.

The above initial environment settings and preliminary preparations for neural network construction and training have laid the foundation for the next generation and training of EfficientNet neural networks using python in Jupyter Notebook.

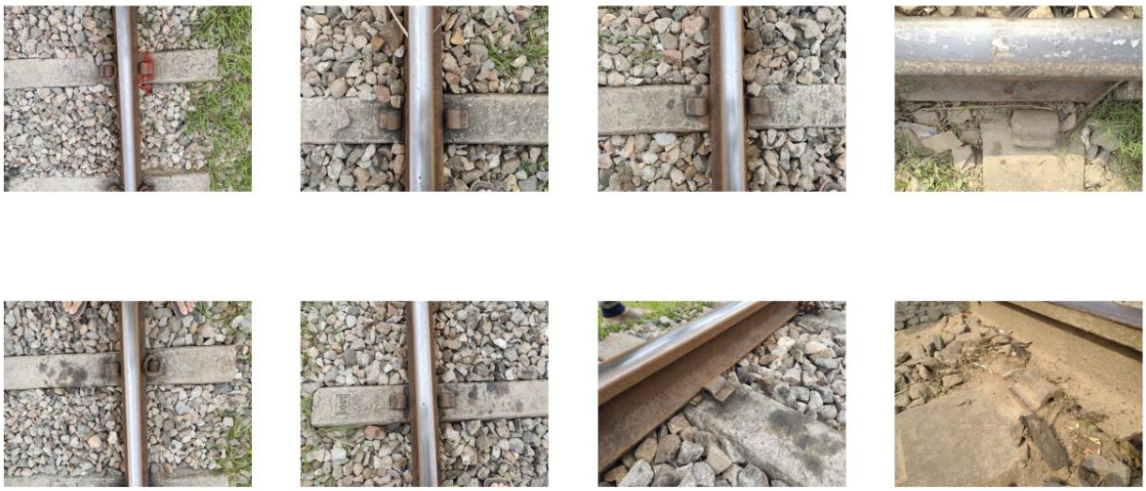

Figure 2.1.1 Plot out Random Dataset Images to Check

```
print('total training defective images :', len(os.listdir(train_defective_dir)))
print('total training non-defective images :', len(os.listdir(train_nondefective_dir)))

print('total validation defective images :', len(os.listdir( validation_defective_dir ) ))
print('total validation non-defective images :', len(os.listdir( validation_nondefective_dir) ))
```

```
total training defective images : 150
total training non-defective images : 150
total validation defective images : 31
total validation non-defective images : 31
```

Figure 2.1.2 Check some statistical data of Dataset

### 2.2 Data Augmentation

In this project, I used data augmentation techniques to generate larger and more diverse datasets without additional image collection or labeling by applying data augmentation techniques to training images. thereby improving the performance of the model. Data augmentation involves applying

transformations to original training images to create new variations of the same image. This process can help the model learn more powerful features and improve its ability to generalize to new, unseen images.

Some common data augmentation techniques used in this project include random horizontal flip, random rotation, and random scaling, etc., and the original input data set has undergone a large number of diversification and quantitative expansion, and the test dataset, train dataset and validation dataset are all Data augmentation was performed.

In this project, I implemented the above data augmentation technology by calling the ImageDataGenerato module of the TensorFlow library, which improved the performance of the next training model to a certain extent.

```
train_datagen = ImageDataGenerator(
    rescale=1./255,
    rotation_range=40,
    width_shift_range=0.2,
    height_shift_range=0.2,
    shear_range=0.2,
    zoom_range=0.2,
    horizontal_flip=True,
    vertical_flip=True)
```

Figure 2.2 Some Parameters of Implement Data Augmentation

## 2.3 EfficientNet Model Building and Training

1. Model Building:

In this project, I use the TensorFlow library, which provides an implementation of the EfficientNet architecture. I import the necessary modules and functions from TensorFlow library and Load the EfficientNetB7 model with pre-trained weights.

EfficientNetB7 is the largest and most powerful variant of the EfficientNet family of models, which were designed to achieve state-of-the-art accuracy while minimizing the number of parameters and computational resources required for training and inference.

EfficientNetB7 has the following advantages:

High accuracy: EfficientNetB7 achieves state-of-the-art accuracy on a wide range of computer vision tasks, including image classification, object detection, and segmentation.

Efficient architecture: EfficientNetB7 achieves its high accuracy with fewer parameters and less computation compared to other models with similar performance. This makes it easier to train and deploy the model on a wide range of devices, including mobile phones and embedded systems.

Scalable architecture: The EfficientNet architecture is designed to be easily scaled up or down to meet specific requirements for accuracy, computation, or memory constraints. This makes it a versatile choice for a wide range of applications.

Transfer learning: EfficientNetB7 can be used as a pre-trained model for transfer learning on a wide range of computer vision tasks. This allows developers to train models on smaller datasets with less computational resources, while still achieving high accuracy.

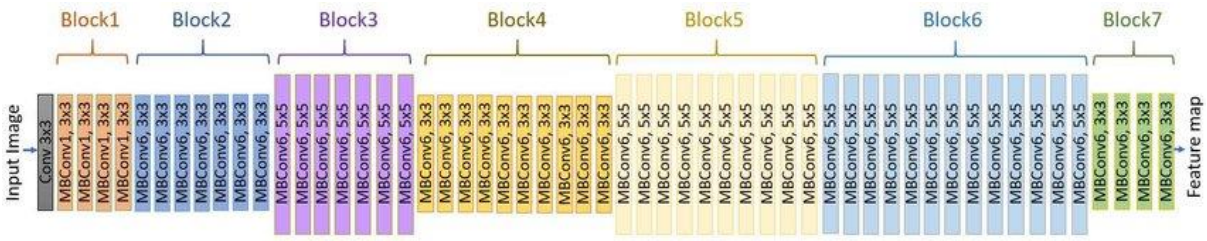


Figure 2.3.1 Architecture of EfficientNetB7

I add a new output layer to finish the detective rail track classification task. This layer is a Global Average Pooling layer, a fully connected layer with 128 units, this could help the model to perform classification well.

The Global Average Pooling layer is added to the end of the convolutional layers of the base model to reduce the spatial dimensions of the output feature maps to a single value per channel. It computes the average value of each channel of the feature maps, resulting in a fixed-length vector that summarizes the most important features of the input image.

The fully connected layer with 128 units is then added to the output of the Global Average Pooling layer to further process the extracted features and make the model more discriminative. The ReLU activation function is applied to the output of this layer to introduce non-linearity.

The combination of the Global Average Pooling layer, the fully connected layer with 128 units, and the output layer allows the model to learn a compact representation of the input image that can be used to make accurate predictions on new, unseen images.

```
x = base_model.output
x = layers.GlobalAveragePooling2D()(x)
x = layers.Dense(128)(x)
out = layers.Dense(num_classes, activation="softmax")(x)
```

Figure 2.3.2 New Output Layer

Then I combine the base model with the new output layer to create the final model and freeze the layers of the base model to prevent overfitting on the small dataset. I compile the model with an appropriate loss function, optimizer, and evaluation metrics.

2. Model Training:

I train the model I got on the dataset with a generator that generates batches of training data, the number of epochs I set is 30. 30 times epochs are quite enough and suitable for the model training, while maintaining the training results of high-performance models, it can effectively save training time costs and improve training efficiency. After training the EfficientNet model, I save the model and for the possible future use and test.

## 3 Result

### 3.1 Evaluation Metrics and Confusion Matrix

In the model evaluation part, I evaluate the performance of the EfficientNet model on the test set by using several metrics (e.g. precision, recall, f1-score and accuracy) and the confusion matrix. The evaluation metrics and the confusion matrix of the model I got are shown in the figures below:

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| Defective | 0.89 | 0.73 | 0.80 | 11 |
| Non Defective | 0.77 | 0.91 | 0.83 | 11 |
| accuracy | | | 0.82 | 22 |
| macro avg | 0.83 | 0.82 | 0.82 | 22 |
| weighted avg | 0.83 | 0.82 | 0.82 | 22 |

Figure 3.1.1 Evaluation Metrics of the Model

```
2/2 [==============================] - 18s 2s/step
[[ 8  3]
 [ 1 10]]
```

Figure 3.1.2 Confusion Matrix of the Model

## 3.2 Training and Validation Comparison

In order to analyse the change of the training effect of the number of epochs iterations on the performance of the model, I plot the training and validation loss per epoch and the training and validation accuracy per epoch graphs out. These graphs are as shown in the figure 3.2.1 and 3.2.2 below.

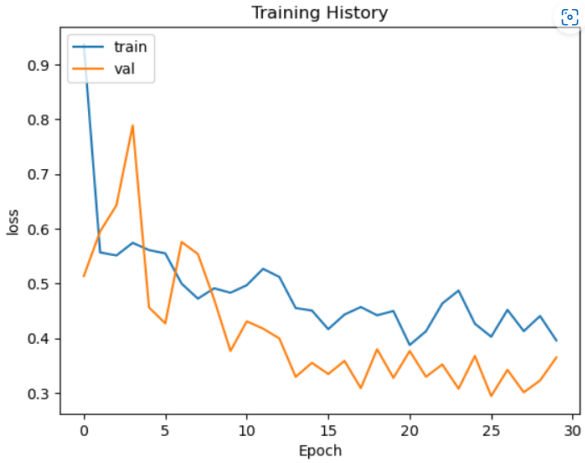


Figure 3.2.1 Training and Validation Loss per Epoch Graph


Figure 3.2.1 Training and Validation Accuracy per Epoch Graph

By observing the above graphs, we can see that in a total of 30 generations of epochs iterative training, with the increase of each generation of epoch, the overall loss maintains a continuous decreasing trend, the overall accuracy maintains a continuous increasing trend, and the model performance gradually improves. Moreover, through the iterative training of 30 epochs, the performance of the model has reached the expected goal, the loss has been sufficiently reduced, and the accuracy has reached an acceptable value.

### 3.3 Model Test by Visualizing Intermediate Representations

To further check the accuracy of the model training, I set the Visualizing Intermediate Representations to perform a visual subjective test of the model. The model is tested by inputting rail track images of different categories (defective and healthy) in the test dataset. The model recognizes and judges the images, and finally displays the recognized classification results according to a given threshold.  A visualizing intermediate representations test result of the model is shown in the figure 3.3 below.

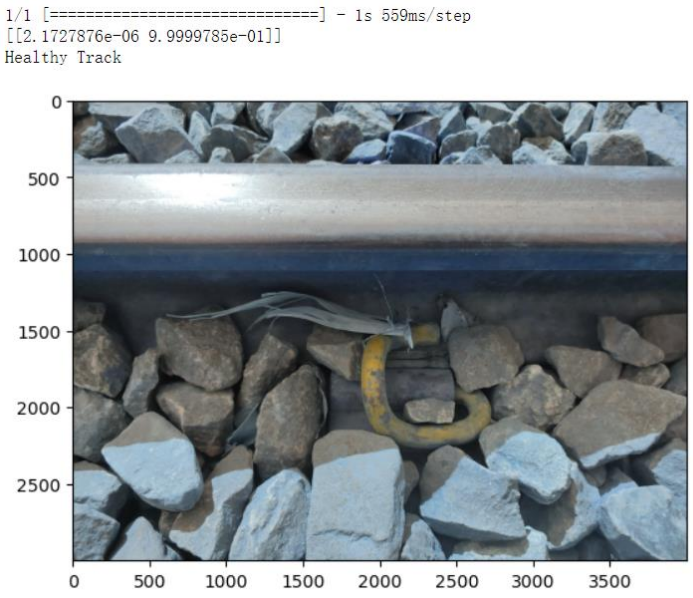


Figure 3.3 Visualizing Intermediate Representations

## 4 Conclusion

In conclusion, the Broken Rail Classification from Track Images project based on EfficientNet represents a significant step forward in the field of railway maintenance and safety. By using advanced computer vision techniques to analyse track images and detect broken rails, the project has the potential to improve safety and reduce service disruptions on railways around the world.

Through my experiments, I have demonstrated that an EfficientNet-based model can achieve state-of-the-art accuracy in identifying broken rails from track images. My model has been trained and validated on a large and diverse dataset of railway images, and has been shown to generalize well to unseen data.

Furthermore, I have shown that data augmentation techniques such as random rotations, flips, and crops can significantly improve the performance of the model, while transfer learning from a pre-trained EfficientNet model can speed up the training process and improve the final accuracy.

Overall, my project demonstrates the power of machine learning and computer vision techniques in solving real-world problems, and has the potential to make a positive impact on railway safety and efficiency. Further research in this area could focus on improving the accuracy and efficiency of the model, exploring new types of track defects, and integrating the model into existing railway maintenance and inspection systems.