Trinity College Dublin
Coláiste na Tríonóide, Baile Átha Cliath
The University of Dublin

School of Engineering

# Defective Rail Classification using a simple CNN and VGG16 Based Classifiers

Author: Eoghan Ó Laoghaire

Student Number: 18319011

Lecturer: Dr. Andrea Staino

April 2024

A report submitted in partial fulfilment of the degree of MAI (Mechanical & Manufacturing Engineering)

# Declaration

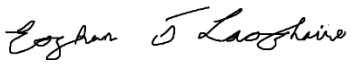
I hereby declare that this report is entirely my own work and that it has not been submitted as an exercise for a degree at this or any other university.

I have read and I understand the plagiarism provisions in the General Regulations of the University Calendar for the current year, found at http://www.tcd.ie/calendar.

I have completed the Online Tutorial on avoiding plagiarism `Ready Steady Write', located at http://tcd-ie.libguides.com/plagiarism/ready-steady-write..

I consent to the examiner retaining a copy of this thesis beyond the examining period, should they so wish (EU GDPR May 2018).

I agree that this report will not be publicly available but will be available to TCD staff and students in the University's open access institutional repository on the Trinity domain only, subject to Irish Copyright Legislation and Trinity College Library conditions of use and acknowledgement.

Signed:  Date: 21/04/2024

# Contents

# Figures

# Tables

# Equations

# Nomenclature

| | |
|---|---|
| CNN | Convolutional Neural Network |
| NN | Neural Network |
| SGD | Stochastic Gradient Descent |
| ANN | Artificial Neural Network |

# Abstract

Two convolutional neural network models, a custom CNN and a VGG16 based model, were trained on a dataset of defective and nondefective rails and their performance was evaluated, with the classification task treated as binary classification. The model based on VGG16 significantly outperformed the custom convolutional neural network with training times that were half the length. Convolutional Neural Networks with high performance could be used to create automatic defective rail detection systems that reduce or remove the need for manual identification of railway defects, which are a leading cause of railway accidents and deaths on railways worldwide.

# 1 Introduction

Defective rails are a leading cause of accidents on railways around the world and are potentially responsible for many deaths annually. The detection of broken rails is not an easy task, as it often requires that rails be identified manually. Manual detection requires that an operator travel the length of the railway and identify defects by sight. This is problematic in that it assumes that the operator can accurately classify defects at a high rate, which may not be the case, and due to the large distances that woul need to be covered to ensure that the entire length of a track has been surveyed.

Convolutional Neural Networks have been used for a number of image classification tasks throughout the years, and their ability to accurately classify images and detect anomalies at a higher rate than humans, especially for specialised tasks such as tumour segmentation, is well documented. Convolutional Neural Networks have been applied to the task of broken rail classification in the past, and state of the art models are capable of reaching accuracies above 93% as well as integration with visualisation platforms which allow anomalies detected by CNNs to be segmented [1].

## 1.1 Objectives

The objectives of this report were to design and train a custom convolutional neural network and a VGG16 based model and compare their performance at classifying images of defective and nondefective rails.

## 1.2 Structure

The remainder of this report is structured as follows. Chapter 2 is a brief literature review of the techniques used in the design and training of the models presented in this report, to support the design choices, as well a brief review of anomaly detection and classification using CNNs. Chapter 3 presents the design and architecture of both models. Chapter 4 presents the results of training as well as performance metrics. Chapter 5 contains discussion of results and limitations of this project. Chapter 6 contains the conclusions of this report. Chapter 7 contains the bibliography of this report. Two jupyter notebooks which contain the models and visualisation of model architectures respectively, as well as a trained model of the VGG16 method employed were uploaded alongside this report.

# 2 Background

## 2.1 Image Classification using Convolutional Neural Networks

Convolutional neural networks are Artificial Neural Networks which have been used for image classification for some time now, and have shown to be capable of classifying images more successfully than humans in many specific applications [2, 3]. Convolutional Neural Networks work by extracting features from an image that become increasingly complex as the layers get deeper while also significantly reducing the number of parameters of a model compared to a fully connected model with the same number of layers. Deeper layers in a CNN will contain information that is related more to the class of an image than the visual image itself [4].

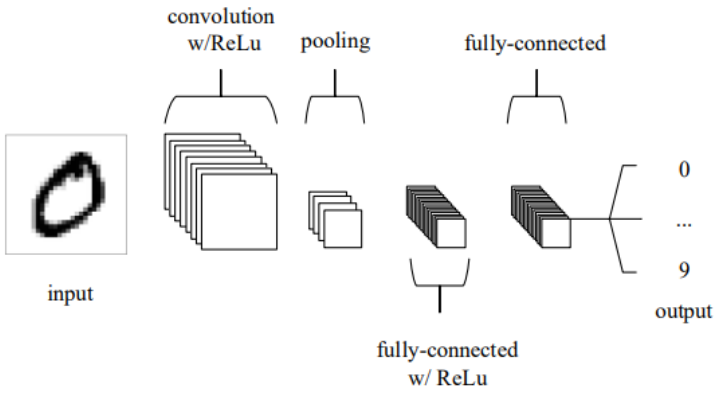



*Figure 2-1 Simple CNN for MNIST data set*

The input layer stores pixel values, while the convolutional layers determine the values of an activation function by applying convolutional kernels to an image to create a 2D activation map. This 2D map enables the CNN to learn which features cause the activation function to "fire" i.e. the most relevant features in an image related to its class [5, 6]. The pooling layer downsamples the images to further reduce the complexity of a model [5]. The mathematics of the convolutional operation of a CNN are not covered in detail in this literature review.

### 2.1.1 Hyperparameters

Hyperparameters are parameters that directly control a models ability to learn and there are a large number of hyperparameters within any deep learning model [7]. For the purposes of this report the literature review focuses only on the hyperparameters which were tweaked during the training process of the models presented in the next section. Hyperparameter tuning is often used to combat overfitting, which occurs when a model learns to use a highly variable function to fit data and thus generalises poorly to unseen data

[8].

### 2.1.2 Optimisation Algorithm

Optimisation functions are functions which improve the performance of a deep learning model. For many years the gradient descent function was the most popular choice of optimisation algorithms and remains in common use today. The gradient descent function updates the model weights on each iteration based on the gradient and an appropriately chosen learning rate [9].

$$w \coloneqq w - lr\frac{\partial L}{\partial w}$$

*(2.1) Gradient Descent Function*

Stochastic gradient descent is a form of mini batch gradient descent where the batch size is equal to 1, i.e. it is based ona single randomly chosen sample. Stochastic gradient descent algorithms exist for a number of learning systems including K-Means, SVM and Neural Networks [9]. Adaptive optimisation functions such as Adam, which is an adaptive version of the SGD, also exist, and are capable of updating the learning rate for each weight of a deep learning model theoretically improving performance although Adam does not always outperform SGD.

### 2.1.3 Learning Rate

Learning rate is a scalar variable which determines the step size of each iteration in the process of minimising the loss function, it is typically chosen by the designer of a model ahead of time [10].

### 2.1.4 Regularisation

Traditional regularisation is a technique through which a function is applied to the cost function of a model which penalise certain outputs. L1 and L2 regularisation are commonly used in deep learning to help combat overfitting. L1 has the effect of setting weights to zero while L2 penalises large weights [11].

$$\sum_{i=1}^{N}\left(y_i - \sum j\,\beta_j x_{ij}\right) + \lambda \sum j\,|\beta_j|$$

*(2.2) L1 Regularisation*

$$\sum_{i=1}^{N}\left(y_i - \sum j\,\beta_j x_{ij}\right) + \lambda \sum j\,\beta_j{}^2$$

*(2.3) L2 Regularisation*

Alternative methods such as dropout and batch normalisation have gained significant popularity in recent years. Dropout works by randomly switching off a predetermined number of neurons at times during training thereby altering the structure of a neural

network, which may help a model to reduce bias in its learning process caused by weights which are learning something that is not helpful to the overall task, potentially reducing overfitting [12]. Batch normalisation is a technique which reduces training time by normalising the values of each layer in a neural network while also having a regularising effect, and may sometimes reduce or eliminate the need to use dropout [12].

### 2.1.5 Cost/Loss Functions

The cost or loss function of a deep learning model is a function which computes the loss between the actual value and predicted value of a model. Minimising the loss function is key to machine learning. The binary cross entropy cost function is used for binary classification [13].

$$\text{Loss}_{\text{Binary } CE} = -\frac{1}{N}\sum_{i=1}^{N}\left(y_i \, log\,(\text{h}_{\text{w},i}) + (1-y_i) log\,(1-\text{h}_{\text{w},i})\right)$$

*(2.4) Binary Cross Entropy*

## 2.2 Transfer Learning

Transfer learning is a technique whereby neural networks which have been trained on a large dataset and have achieved very high peformance, such as VGG16, are applied to new problems by freezing the prelearned model weights and applying a new densely connected layer on top of the frozen layers and using this "new" model to classify images. Transfer Learning can yield significantly better results than a simple CNN, especially when trying to train a CNN on a small dataset, although typically require tuning to achieve full potential [4].

## 2.3 Data Augmentation

Data augmentation is a technique which seeks to increase the size of a dataset by applying augmentations to the images, which could be color space, rotational or other kinds of augmentations, to create more data on which to train a model [8]. Data augmentation is most useful for training models on datasets which are very small. Deep learning performance is often reliant on large datasets and therefore data augmentation is sometimes a necessity, although some models such as UNET perform exceptionally well for their specific tasks on small datasets [8, 14].

## 2.4 Performance Classification

For many years classifier performance was gauged using only its accuracy at classifying validation images. In recent years other metrics such as precision, recall and f1 have been used to gauge the performance of a model. The f1 score is the harmonic mean of the

precision and recall scores of a model and may better represent the actual accuracy of a model [15].

$$F1 = 2\,x\,\frac{\text{precision x recall}}{\text{precision} + \text{recall}} = \frac{2\text{TP}}{2\text{xTP} + \text{FP} + \text{FN}}$$

*(2.5) F1 score*

Where TP, FP and FN are true positives, false positives and false negatives predicted by the model respectively.

## 2.5 Automatic Broken Rail Detection

Convolutional neural networks for anomaly detection exist for a number of applications from medical applications such as tumour detection, segmentation and classification to industrial applications such as surface inspection [14, 16]. The ability of CNNs to accurately classify images at a higher rate than humans, especially for specialised tasks such as broken rail detection, makes their application for these purposes desireable [1]. Broken rails are typically identified manually, which can be an arduous process considering the distances and time involved in travelling along a railway while simultaneously trying to identify problematic rails [1]. Identification of broken rails can be a lifesaving intervention, as many lives are lost annually due to train accidents [1, 17]. Accidents due to broken rails can also lead to severe downtimes of a track for the process repairing the rail and moving damaged/derailed trains [18]. Faulty rails are the second leading cause of rail accidents in the US [17].

Acoustic sensing has also been proposed as a potential method for the automatic detection of broken rails [18]. This method is advantageous as acoustic sensors can be monitored remotely and could potentially be integrated with a CNN model that is capable of classifying the recordings obtained using acoustic sensing, removing the need to image rails entirely.

## 2.6 Broken Rail Detection Dataset

The dataset of defective and non defective rails used as part of this report was generated for use in training a deep learning based automatic rail detection system. The dataset is split into training, test and validation images. The types of faults in the images are differentiated as rail faults and fastener faults, although they are not split up this way in the dataset by default, and are catagorised only as either defective or nondefective [1]. There are 299, 62 and 22 images in the Train, Validation and Test datasets respectively.

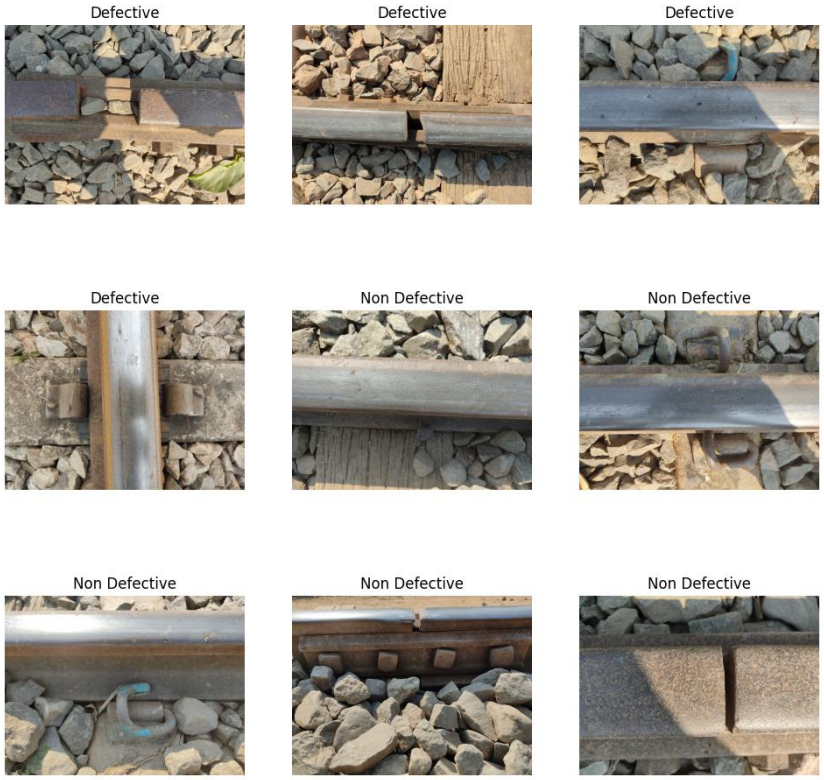



*Figure 2-2 Defective and Nondefective Rails from the Rail Dataset*

The authors of the paper associated with the dataset created a ResNet based model named ECARRnet which they claim accurately detects faults at a rate of 93%, which is greater than current state of the art methods of automatic rail defect detection [1].

# 3 Model Architecture

## 3.1 Custom CNN

The architecture of the final simple CNN model is shown below in Figure 3-1 and Figure 3-2 below.

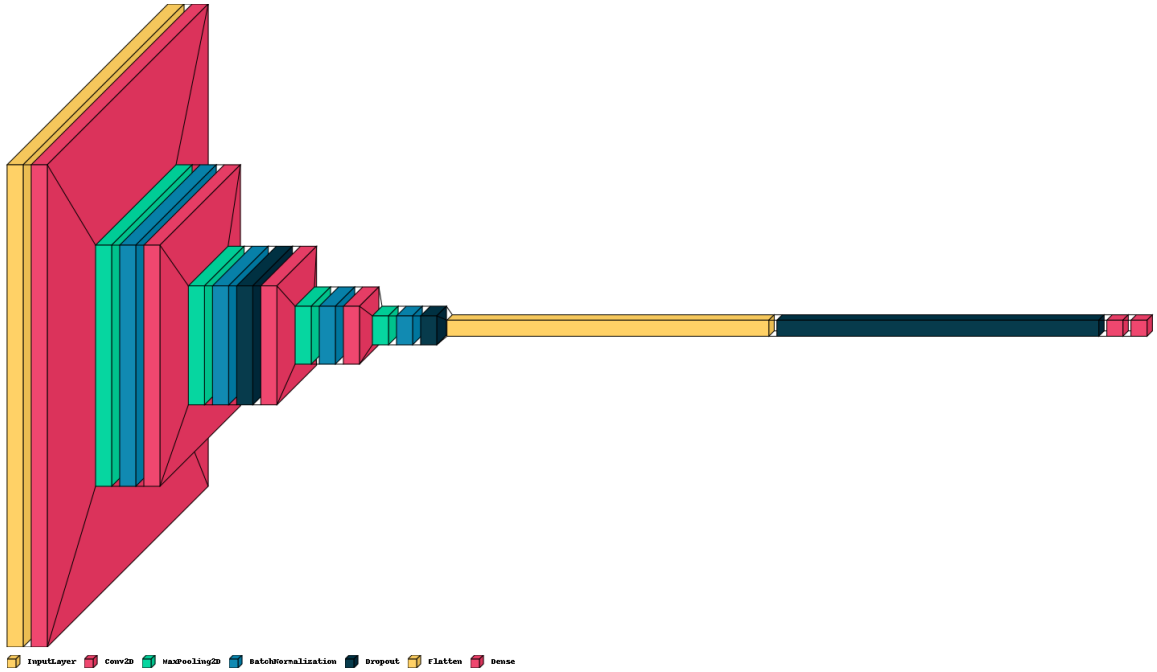



*Figure 3-1 Visualisation of Final simple CNN architecture*

A number of tweaks were made in the design process of the simple CNN, outlined in the table below, a number of smaller optimisations were made in the design process, including experimenting with the learning rate value. L2 regularisation (l=0.01) was used on the second to last dense layer of the CNN.

*Table 3-1 Design Changes Over Main Model Iterations*

| Iteration Number | Learning Rate | Dropout (Conv Layers) | Batch Norm | Input Size | Parameters |
|---|---|---|---|---|---|
| 1 | 0.01 | None | None | (300, 300) | 2715681 |
| 2 | 0.01 | 0.5 | Keras Default | (150, 150) | 725025 |
| 3 | 0.01 | 0.3 | Keras Default | (150, 150) | 725025 |
| 4 (Final) | 0.001 | 0.3 | Keras Default | (150, 150) | 725025 |

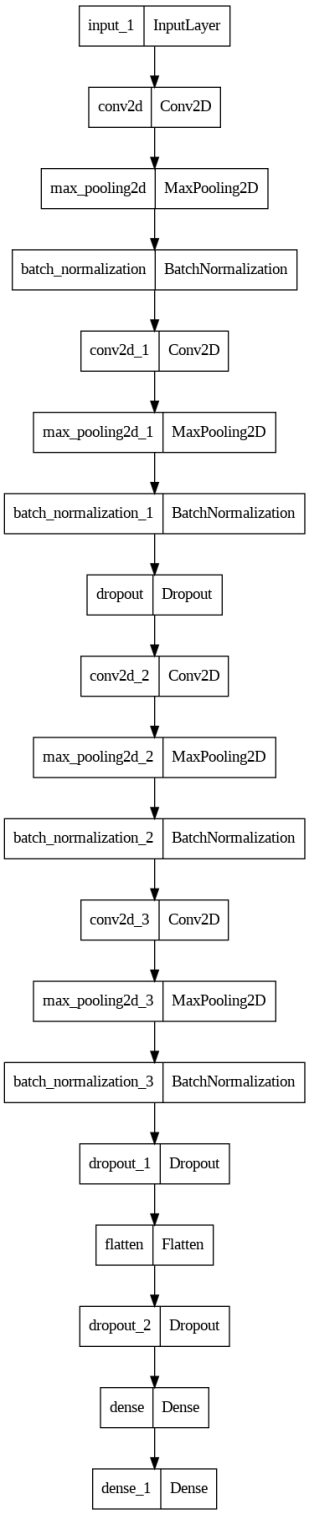



*Figure 3-2 Final simple CNN Architecture*

### 3.1.1 Unused Second model

A second model with a similar structure was also tested which did not utilise any batch normalisation and had dropout applied to every convolutional layer. This model performed well on the training set but converged to a local minimum on the validation set nearly every

time.

### 3.1.2 Choice of Optimiser and Activation Functions

Stochastic Gradient Descent with decay and momentum was used as SGD is known to perform well on image classification tasks despite slow computation times, and also due to familiarity with the use of SGD. Adaptive methods such as Adam may lower the training times of a model, but SGD is generalises well and is still a good choice of optimiser. SGD may make up for the slow computation times by achieving shorter convergence times than other optimisers [19]. Relu activation was used on the convolutional and dense layers, except for the final dense layer where sigmoid activation was used as the task was treated as a binary classification between defective and nondefective rails for the purposes of this project.

## 3.2 Transfer Learning Model

A transfer learning model using VGG16 with 3 dense (1 prediction) layers on top was also employed. This model was tested using both SGD and Adam as an optimiser.

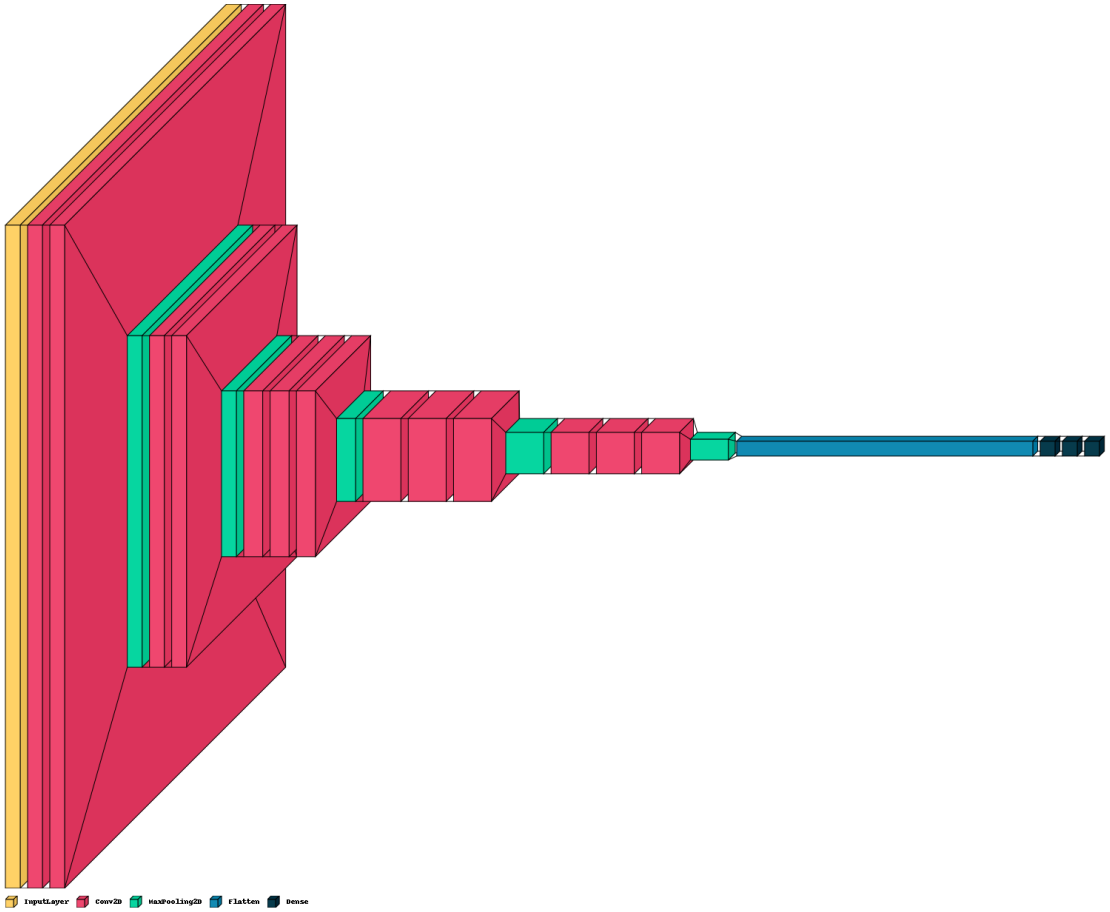



*Figure 3-3 Transfer Learning Model (VGG16) Visualisation*

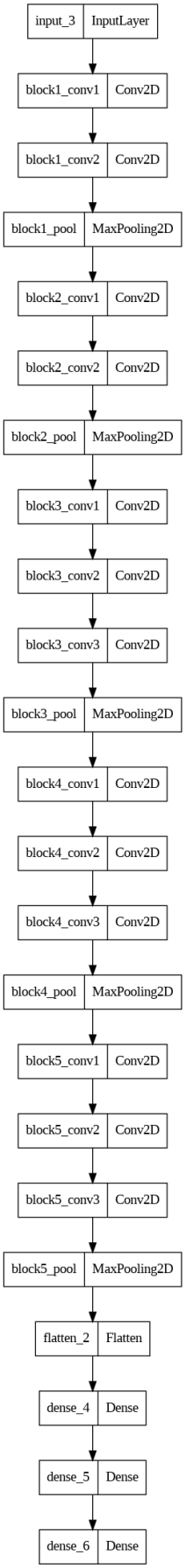



*Figure 3-4 Transfer Learning Model Architecture*

# 4 Results

## 4.1 Simple CNN Classifier

The simple CNN was first tested without regularisation of the convolutional layers over 200 epochs to monitor its longterm behaviour. Note that data augmentation was used to increase the training, validation and test set sizes. Types of augmentation used included positional shifts, rotations and flips.

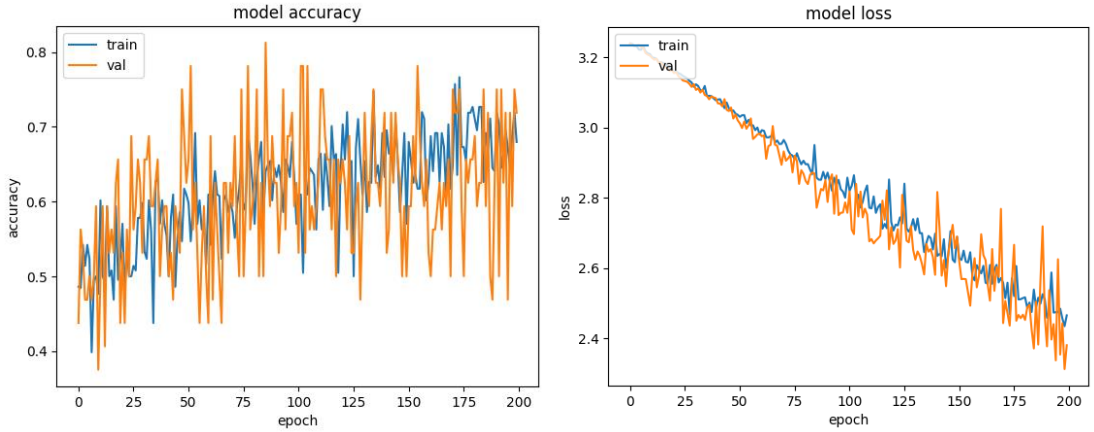



*Figure 4-1 Accuracy and Loss Plots for CNN model without Regularisation or Batch Norm*

After testing without regularisation batchnormalisation was added to each convolutional layer and dropout was added to alternating layers.

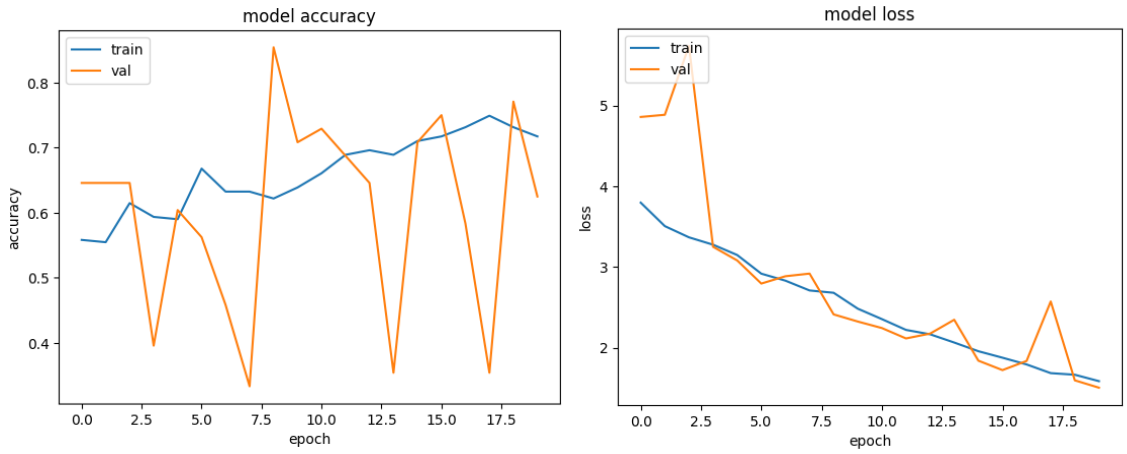



*Figure 4-2 Accuracy and Loss Plots for CNN model with Dropout and Batch Norm*

Finally the level of dropout on the convolutional layers was decreased from 0.5 to 0.3 and tested over 20 epochs.

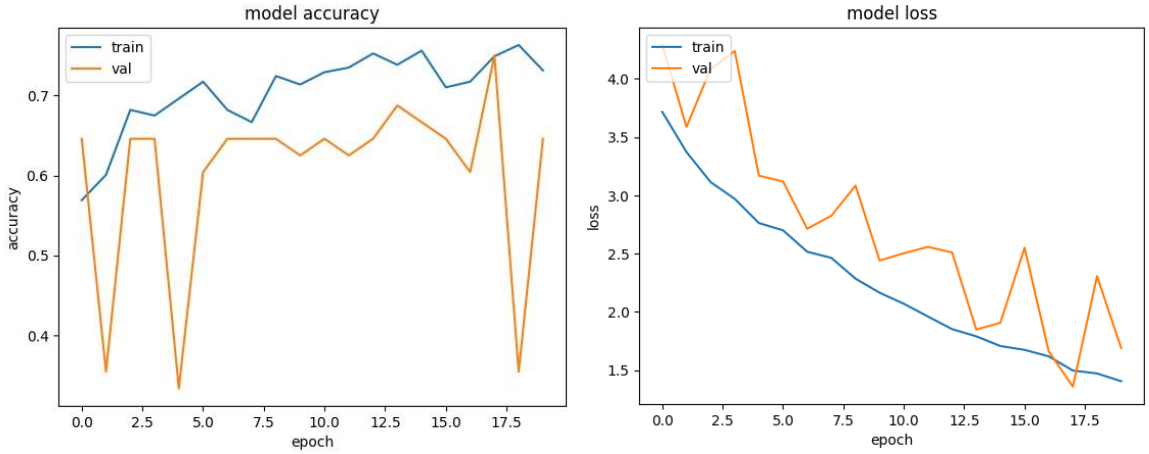



*Figure 4-3 Simple CNN Accuracy and Loss Plots with Reduced Dropout after 20 Epochs*

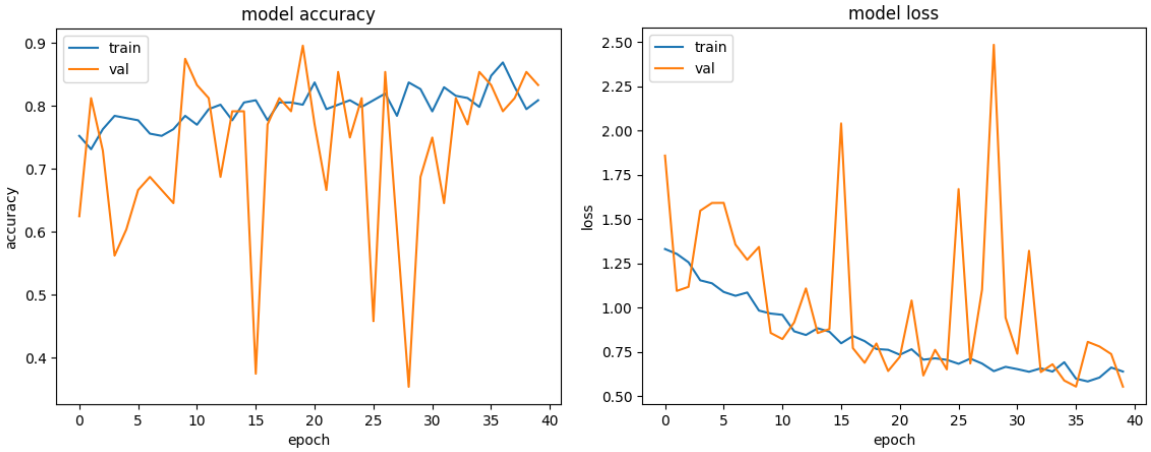



*Figure 4-4 Simple CNN Performance Plots after another 40 epochs*

The simple CNN model was also tested on a smaller test set of images after training and validation. Accuracy and Loss values taken below were the final recorded values for 20 and 60 epochs of training respectively.

*Table 4-1 Simple CNN performance On Test Set after 20 and 60 epochs of training*

| Iteration | Loss | Accuracy | Training Time |
|---|---|---|---|
| **Simple CNN (20 Epochs) (Validation)** | 1.7077 | 0.6458 | 16.6mins |
| **Simple CNN (60 Epochs) (Validation)** | 0.6409 | 0.8245 | 50 mins |
| **Simple CNN (20 Epochs) (Test Set)** | 1.5335 | 0.6875 | 16.6mins |
| **Simple CNN (60 Epochs) (Test Set)** | 0.7200 | 0.6875 | 50 mins |

The learning rate of the model was then changed by a factor of 10 to 0.001. Due to time constraints this version was tested for only 30 epochs.

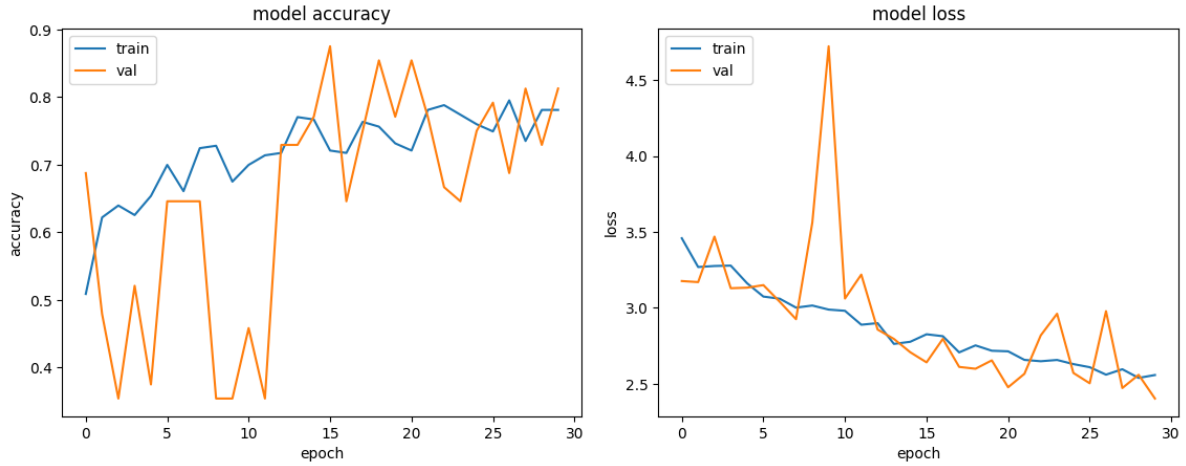



*Table 4-2 Validation Loss and Accuracy of simple CNN with learning rate of 0.001*

| Iteration 4 | Loss | Accuracy | Training Time |
|---|---|---|---|
| Simple CNN (30 Epochs) (Validation) | 2.4027 | 0.8125 | 25 mins |

## 4.2 Transfer Learning

The transfer learning model was initially tested using as an optimiser SGD, and then again with Adam.

*Table 4-3 Transfer Learning Model Performance on Test Set, with SGD*

| TL Model (VGG16) | Loss | Accuracy | Training Time |
|---|---|---|---|
| VGG16 (10 Epochs) (Test Set) | 0.5986 | 0.75 | 8.3 mins |

The model was then trained again using Adam as an optimiser for 30 epochs, where it appears to have retained some learning from the initial training with SGD.

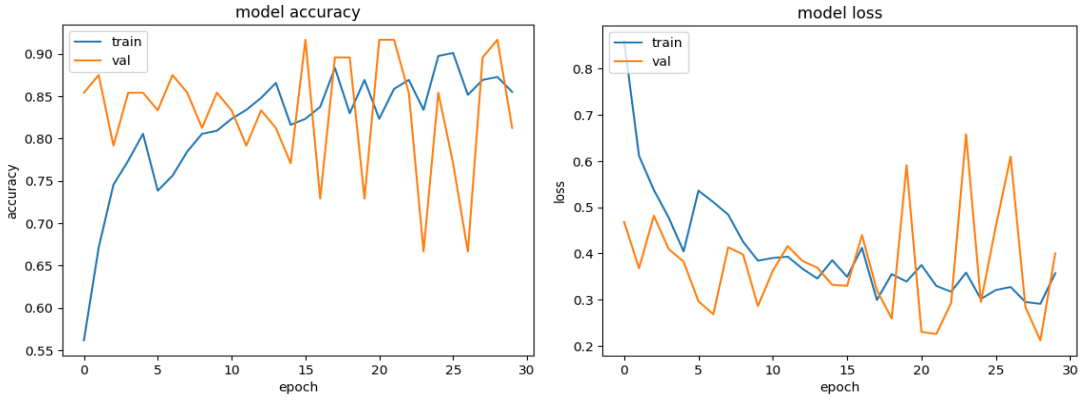



*Figure 4-5 Accuracy and Loss Plots for TF Model after 30 epochs, with Adam*

*Table 4-4 Transfer Learning Model Performance with Adam*

| TL Model (VGG16) | Loss | Accuracy | Training Time |
|---|---|---|---|
| VGG16 (30 Epochs) (Test Set) | 0.4584 | 0.8750 | 25 mins |
| VGG16 (30 Epochs) (Validation) | 0.4003 | 0.8125 | 25 mins |

## 4.3 Metrics

Finally predictions were made with both models to generate some metric scores using model.predict() and sklearn classification metrics.

*Table 4-5 Metrics for CNN Iteration 4 (30 Training Epochs)*

| CNN | Precision | Recall | F1 | Accuracy |
|---|---|---|---|---|
| Defective | 0.88 | 0.64 | 0.74 | 0.77 |
| Non Defective | 0.71 | 0.91 | 0.80 | |

*Table 4-6 Metrics for Transfer Learning (30 Epochs)*

| Transfer Learning | Precision | Recall | F1 | Accuracy |
|---|---|---|---|---|
| Defective | 0.90 | 0.82 | 0.86 | 0.86 |
| Non Defective | 0.83 | 0.91 | 0.87 | |

The ROC curve of both models is plotted in Figure 4-6 below.

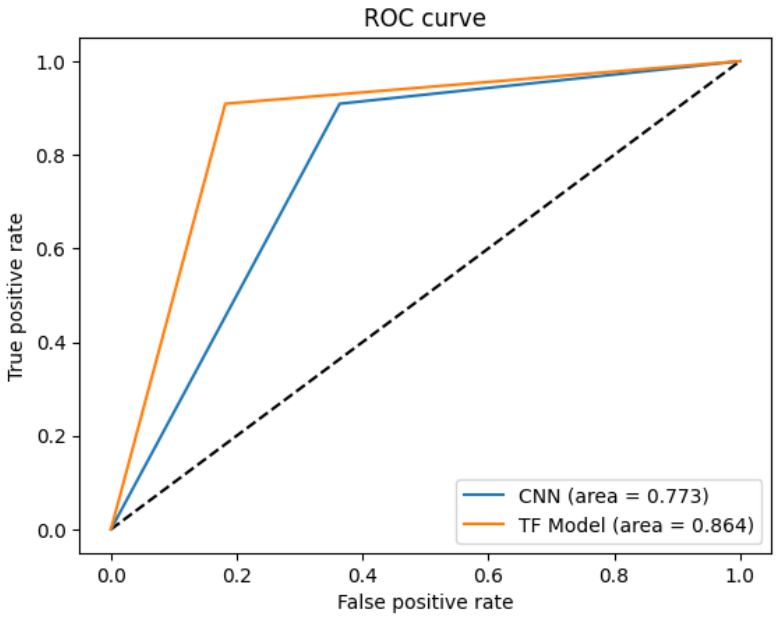



*Figure 4-6 ROC curves for transfer learning model and CNN*

## 4.4 Visualisation of the Transfer Learning Model Layer Activation

Below are some visualisations of what the transfer learning model activation for some layers look like. The first convolutional layer appears to be detecting edges. Note also the clear effect of downsampling in the max pooling layer with a drop in visual quality, seen in Figure 4-7 below.

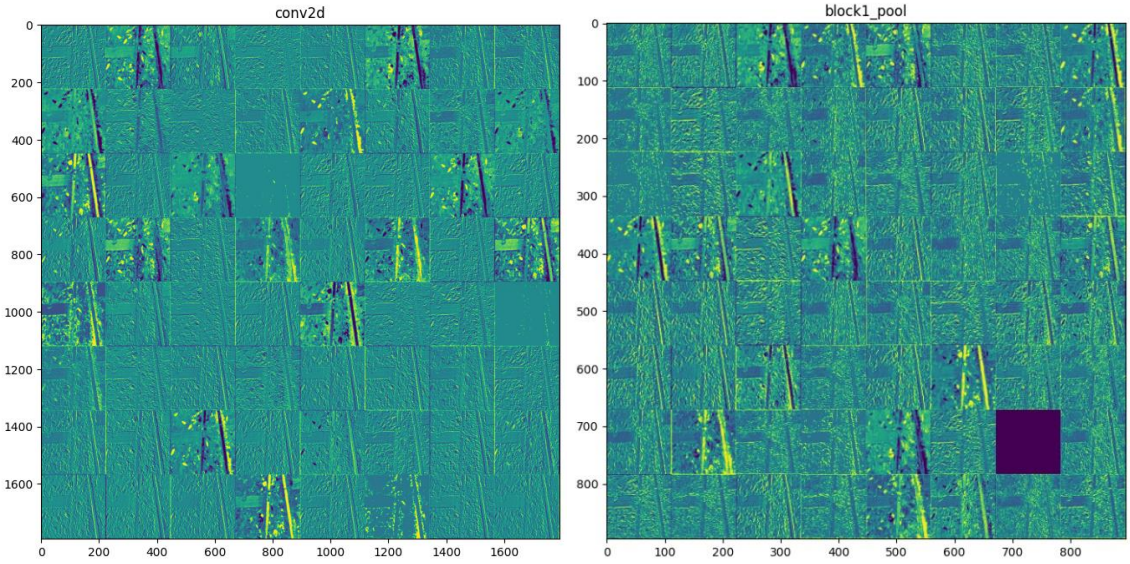



*Figure 4-7 Activation of The First Convolutional and Max Pool Layers of the TL Model*

Once the the images reach the 3rd block of the transfer learning model, the activation images have clearly already become much more complex. The deeper layers of a convolutional neural network model increasingly contain information that is related more to

the class of an image than the image itself [4].

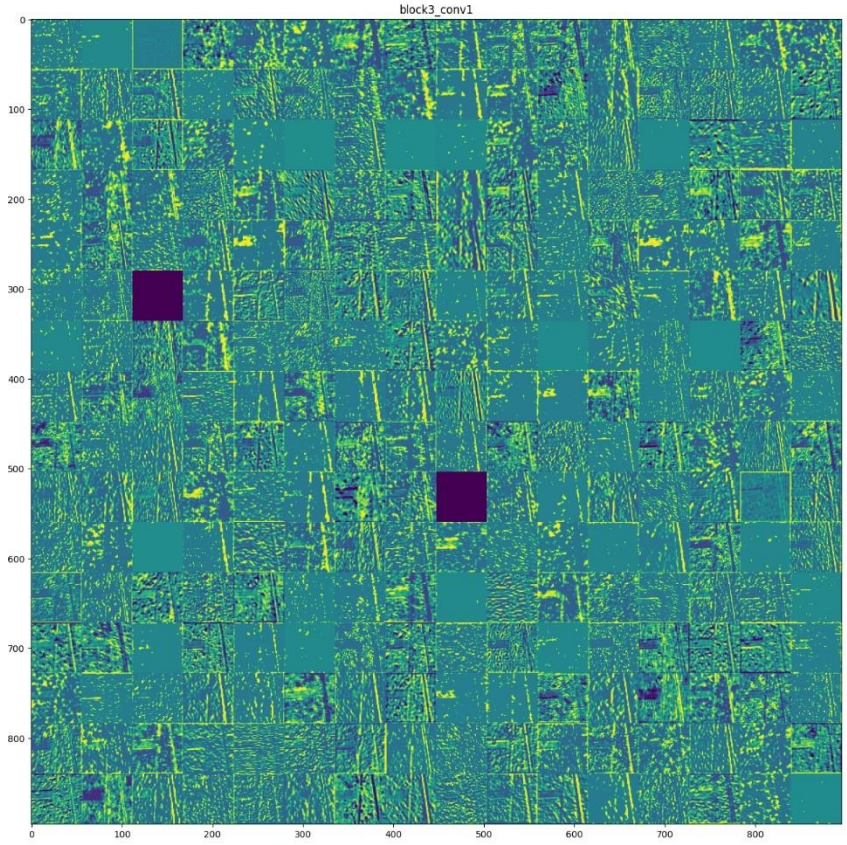



*Figure 4-8 First convolutional layer of the 3rd block in TL Model*

# 5 Discussion & Analysis

## 5.1 Analysis of Results

The results clearly show that the transfer learning based model is significantly more performant than the simple CNN although both were capable of reaching accuracies above 80%. The transfer learning model was much more accurate when evaluated on the test set (75% after 10 epochs and 87.5% after 30 epochs with Adam as optimiser) than the simple CNN (68.75% accurate for both 20 and 60 total epochs) however both models appear to be somewhat unstable based on the graphs obtained.

For the simple CNN, it was clear that the learning rate was too high for the first three iterations, better performance could be achievable using a lower learning rate, as the model appeared to be more stable in the fourth and final iteration

Although both models have high precision for defective images, the f1 score is significantly higher for the transfer learning model. F1 is the harmonic mean of precision and recall, and is potentially more indicative of the performance of a model than accuracy.

## 5.2 Limitations

There are a number of important limitations to consider when interpreting the results presented in this report. The task was treated as a binary classification task for simplicity despite the fact that the defective rails are classified either as rail defects or fastener defects. This likely makes it more difficult for the model to accurately classify images as the actual defective and non defective classes could be said to contain two subclasses each, nondefective and defective rails and nondefective and defective fasteners.

Colour augmentation was not used in the augmented data generators. Augmentation using colour could potentially make certain defects more or less prominent within an image, helping the models to learn.

Training time was a limiting factor as the training times for both models were quite long, which meant that it was not possible to push the models to their potential limits without training over very lengthy periods.

Hyperparameter optimisation using methods such as grid seaerch was not performed, the hyperparameters of the model were tuned through trial and error. The transfer learning model was also not fine tuned after the initial rounds of training.

The loss and accuracy for both models would sometimes decrease at the same time indicating that both the transfer learning model and the simple CNN are overfitting the data.

The accuracy plot for the transfer learning model has very high accuracy even on the first epoch (>80%), this plot was obtained after using the model with SGD as an optimiser for 10 epochs and then switching to adam to retrain for 30 epochs however it likely that learning from the first training run may have remained in the model somehow, even though the model should have been starting from scratch, the performance is strong regardless.

# 6 Conclusions

The main aim of this project was to develop CNN based classifiers that could accurately classify defective and nondefective rails. Of the two models proposed, the pretrained model significantly outperformed the simple CNN on the training, validation and test datasets. Both models were capable of occasionally producing above 80% during training and validation although only the pretrained model was capable of reaching accuracies on the validation set greater than 90% and reaching accuracy above 80% consistently. The transfer learning model had significantly higher recall and f1 scores for defective rails than the custom CNN, however precision was comparable. It is clear from the graphical data obtained that the models require more fine-tuning and hyperparameter optimisation as they appear to be unstable.

Smaller learning rates may be needed for both models to find optimal and stable solutions and reduce overfitting. A larger dataset of rail images could potentially improve the results as well as data augmentation using colour augments that make defects in the rails more easily identifiable. Transfer learning using an alternative pretrained model such as ResNet or InceptionV3 could potentially also yield superior results to the VGG16 model used as part of this project. Longer training times may also improve the performance of both models.

# 7 Bibliography


[1] S. I. Eunus *et al.*, "ECARRNet: An Efficient LSTM-Based Ensembled Deep Neural Network Architecture for Railway Fault Detection," *AI,* vol. 5, no. 2, pp. 482-503, Apr 8 2024. [Online]. Available: https://www.mdpi.com/2673-2688/5/2/24.

[2] Q. Li, W. Cai, X. Wang, Y. Zhou, D. D. Feng, and M. Chen, "Medical image classification with convolutional neural network," in *2014 13th International Conference on Control Automation Robotics & Vision (ICARCV)*, Dec 10-12 2014, pp. 844-848, doi: https://doi.org/10.1109/ICARCV.2014.7064414.

[3] I. D. Shikhar Tuli, Erin Grant, Thomas L. Griffiths, "Are Convolutional Neural Networks or Transformers more like human vision?," presented at the CogSci 2021 Conference, Virtual, Jul 1, 2021. doi:https://doi.org/10.48550/arXiv.2105.07197

[4] F. Chollet, "Chapter 5 *Deep Learning For Computer Vision*," in *Deep Learning With Python*: Manning Publications, 2018.

[5] R. N. Keiron O'Shea, "An Introduction to Convolutional Neural Networks.," *CoRR,* Dec 2 2015, doi: https://doi.org/10.48550/arXiv.1511.08458.

[6] F. Chollet, "Chapter 4 *Introduction to convnets*," in *Deep Learning With Python*: Manning Publications, 2018.

[7] K. Nyuytiymbiy. "Parameters and Hyperparameters in Machine Learning and Deep Learning." 2020. https://towardsdatascience.com/parameters-and-hyperparameters-aa609601a9ac (accessed 19/04/2024, 2024).

[8] C. Shorten and T. M. Khoshgoftaar, "A survey on Image Data Augmentation for Deep Learning," *Journal of Big Data,* vol. 6, no. 1, p. 60, Jul 06 2019, doi: https://doi.org/10.1186/s40537-019-0197-0.

[9] L. Bottou, "Stochastic Gradient Descent Tricks," in *Neural Networks: Tricks of the Trade: Second Edition*, G. Montavon, G. B. Orr, and K.-R. Müller Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012, pp. 421-436. doi: https://doi.org/10.1007/978-3-642-35289-8_25.

[10] K. P. Murphy, *Machine Learning: A Probabilistic Perspective*. MIT Press, 2012.

[11] R. Tibshirani, "Regression Shrinkage and Selection Via the Lasso," *Journal of the Royal Statistical Society: Series B (Methodological),* vol. 58, no. 1, pp. 267-288, 2018, doi: https://doi.org/10.1111/j.2517-6161.1996.tb02080.x.

[12] C. Garbin, X. Zhu, and O. Marques, "Dropout vs. batch normalization: an empirical study of their impact to deep learning," *Multimedia Tools and Applications,* vol. 79, no. 19, pp. 12777-12815, Jan 22 2020, doi: https://doi.org/10.1007/s11042-019-08453-9.

[13] E. Schultheis and R. Babbar, *Unbiased Loss Functions for Multilabel Classification with Missing Labels*. 2021. [Online]. Available: https://www.researchgate.net/publication/354801910_Unbiased_Loss_Functions_for_Multilabel_Classification_with_Missing_Labels

[14] P. F. Olaf Ronneberger, Thomas Brox, "U-Net: Convolutional Networks for Biomedical Image Segmentation," presented at the MICCAI 2015 Conference, Munich, Germany, May 18, 2015. doi:https://doi.org/10.48550/arXiv.1505.04597

[15] S. A. Hicks *et al.*, "On evaluation metrics for medical applications of artificial intelligence," (in eng), *Sci Rep,* vol. 12, no. 1, p. 5979, Apr 8 2022, doi: https://doi.org/10.1038/s41598-022-09954-8. PMCID: PMC8993826

[16] B. Staar, M. Lütjen, and M. Freitag, "Anomaly detection with convolutional neural networks for industrial surface inspection," *Procedia CIRP,* vol. 79, pp. 484-489, 2019/01/01/ 2019, doi: https://doi.org/10.1016/j.procir.2019.02.123.

[17] Sireesha, Ajay Kumar, Mallikarjunaiah, and B. Kumar, "Broken Rail Detection System using RF Technology," *SSRG International Journal of Electronics and Communication Engineering,* Apr 2015. [Online]. Available: https://www.internationaljournalssrg.org/IJECE/2015/Volume2-Issue4/IJECE-V2I4P103.pdf.

[18] A. Wagner, A. Nash, F. Michelberger, H. Grossberger, and G. Lancaster, "The Effectiveness of Distributed Acoustic Sensing (DAS) for Broken Rail Detection," *Energies,* vol. 16, no. 1, p. 522, 2023. [Online]. Available: https://www.mdpi.com/1996-1073/16/1/522.

[19] I. Kandel, M. Castelli, and A. Popovič, "Comparative Study of First Order Optimizers for Image Classification Using Convolutional Neural Networks on Histopathology Images," (in eng), *J Imaging,* vol. 6, no. 9, Sep 8 2020, doi: https://doi.org/10.3390/jimaging6090092. PMCID: PMC8321140

END OF REPORT