Classification of Faults in Railway Ties using Computer Vision and Machine Learning

Amruta Kulkarni

Thesis submitted to the Faculty of the
Virginia Polytechnic Institute and State University
in partial fulfillment of the requirements for the degree of

Master of Science
in
Computer Engineering

Kevin B. Kochersberger, Chair
Devi Parikh
A. Lynn Abbott

May 15, 2017
Blacksburg, Virginia

Keywords: Railway ties, Computer Vision, Machine Learning

Classification of Faults in Railway Ties using Computer Vision and Machine Learning

Amruta Kulkarni

ABSTRACT

This work focuses on automated classification of railway ties based on their condition using aerial imagery. Four approaches are explored and compared to achieve this goal - handcrafted features, HOG features, transfer learning and proposed CNN architecture. Mean test accuracy per class and Quadratic Weighted Kappa score are used as performance metrics, particularly suited for the ordered classification in this work. Transfer learning approach outperforms the handcrafted features and HOG features by a significant margin. The proposed CNN architecture caters to the unique nature of the railway tie images and their defects. The performance of this approach is superior to the handcrafted and HOG features. It also shows a significant reduction in the number of parameters as compared to the transfer learning approach. Data augmentation boosts the performance of all approaches. The problem of label noise is also analyzed. The techniques proposed in this work will help in reducing the time, cost and dependency on experts involved in traditional railway tie inspections and will facilitate efficient documentation and planning for maintenance of railway ties.

This work received support from Bihrle Applied Research Inc.

Classification of Faults in Railway Ties using Computer Vision and Machine Learning Techniques

Amruta Kulkarni

GENERAL AUDIENCE ABSTRACT

Railway tracks and their components need to be frequently inspected for defects or design non-compliances. Manual inspections involve long hours, high cost and dependency on the availability of experts. Previous efforts to automate the inspection of the railway track inspections are focused towards either other components of the railway track or towards using custom designed ground vehicles. This work presents four approaches to automate the inspection of wooden railway ties by categorizing them into one of three categories based on their condition. Images of the track are taken by an aerial vehicle, in which the track is left untouched. The techniques of computer vision and machine learning used in this work outperform the baselines. The efforts are directed towards making the algorithm learn from the labeled data. The labeled data is also artificially enlarged and enriched, which boosts the performance of the classifiers. The performance metrics used to evaluate the classification approaches are particularly suited for the task at hand. The problem of inconsistency in labeling between two human labelers is analyzed. Potential further directions for research are also identified.

To Swanand - for his patience, advice and faith

Acknowledgments

I would like to express my sincere gratitude towards Dr. Kevin Kochersberger for giving me the opportunity to pursue this research. His guidance and support has helped me complete this work. His confidence in me has been my motivation throughout. I am grateful to be a part of the Unmanned Systems Lab under his guidance.

I would like to thank Dr. Devi Parikh for her inspiring lectures in Computer Vision which motivated me to pursue this field. I would also like to thank Dr. Abbott for his valuable comments.

I extend my sincere thanks to Dr. John Bird and Dr. Jia Bin Huang for their valuable insights that helped me enrich my work.

I am grateful to Jennifer Player for giving me this opportunity to work with Bihrle Applied Research Inc. I would like to thank Stanton Coffey and the rest of the team at Bihrle for providing the dataset and consistent guidance required to complete this work.

I express a heartfelt thanks to my friends at Virginia Tech, who are my family away from home. Most of all, I would like to thank my parents, for raising me to be the person I am and my brother, Anish for always encouraging me.

Contents

List of Figures **ix**

List of Tables **xii**

1 Introduction **1**

2 Related Work **5**

3 Background in Machine Learning techniques **7**

3.1 Support Vector Machine 7

3.2 k-Nearest Neighbors 8

3.3 Decision tree 9

3.4 Naive Bayes 9

3.5 Linear Discriminant Analysis 10

3.6 Logistic Regression 10

3.7 Ensemble Approach 11

3.8 Convolutional Neural Networks 13

4 Description of Dataset **15**

4.1 Image Acquisition 15

4.2 Image Categorization 16

4.3 Data Augmentation and Cleaning 17

5 Ballast Detection **18**

5.1 Sobel Filter Approach 19

5.2 Gabor Filters Approach 19

5.3 Blurring the Ballast 19

6 Features **21**

6.1 Handcrafted Features 21

6.2 HOG Features 24

6.3 Pre-trained CNN Features 25

7 Performance Metrics for Classification **28**

8 Proposed CNN Architecture **32**

9 Experiments and Results **34**

9.1 Execution Environment 34

9.2 Handcrafted Features 34

9.3 HOG Features 35

9.4 CNN Features 37

9.5 Finetuned AlexNet 38
9.6 Proposed Architecture 40
9.6.1 1 convolutional layer 40
9.6.2 2 convolutional layers 41
9.7 Label Noise 42
9.8 Comparison of Classification Approaches 44
9.9 Qualitative Results 47

10 Conclusions **48**

11 Recommendations **50**

11.1 Dealing with label noise 50
11.2 Dealing with Ballast 50
11.3 Working with other infrastructural datasets 51
11.4 Sensor fusion 51

12 Bibliography **52**

Appendices **57**

A Confusion Matrices **58**

B Train-Val-Test Split **63**

List of Figures

1.1 Railway Track image provided by Bihrle Applied Research Inc. 1

1.2 TieInspect [1] 2

3.1 Optimal Hyperplane with maximum margin for SVM [2] 8

3.2 Ensemble Classifier [3] 12

3.3 CNN Architecture 13

4.1 Categories of Ties - Broken, Cracked, Good 16

5.1 Ballast Blurring 20

6.1 Ties pre-processed for computation of Handcrafted features 22

6.2 Railway Tie - field side and gauge side 23

6.3 CNN as Feature Extractor [4] 26

7.1 Sample Confusion Matrix 29

7.2 Guide to interpret QWK score [5] 31

9.1 Confusion Matrix for Naive Bayes classifier with Handcrafted features 35

9.2 Confusion Matrix for SVM classifier with HOG features (Cell Size 8 × 8) . . 37

9.3 Pearson Coefficients between predicted labels on the test set by pairs of different classifiers 38

9.4 Confusion Matrix for SVM classifier with CNN features 39

9.5 Confusion Matrix for Finetuned AlexNet 40

9.6 Training and Validation Curves for Proposed Architecture with 1 convolutional layer 41

9.7 Confusion Matrix for Proposed CNN architecture with 1 convolutional layer 42

9.8 Training and Validation Curves for Proposed Architecture with 2 convolutional layers 43

9.9 Confusion Matrix for Proposed CNN architecture with 2 convolutional layers 44

9.10 Performance of all approaches on test set 45

9.11 Per class accuracies for all approaches on test set 46

9.12 Trade-off between architecture and performance 46

9.13 Misclassifications by SVM trained on CNN features 47

A.1 Confusion Matrix for LDA with Handcrafted Features 58

A.2 Confusion Matrix for LR with Handcrafted Features 59

A.3 Confusion Matrix for SVM with Handcrafted Features 59

A.4 Confusion Matrix for kNN with Handcrafted Features 60

A.5 Confusion Matrix for Decision Tree with Handcrafted Features 60

A.6 Confusion Matrix for LDA with CNN Features 61

A.7 Confusion Matrix for Naive Bayes with CNN Features 61

A.8 Confusion Matrix for kNN with CNN Features 62

A.9 Confusion Matrix for Decision Tree with CNN Features 62

List of Tables

6.1 AlexNet Layers 27

8.1 Proposed CNN Layers 33

9.1 Performance of different classifiers with Handcrafted Features on test set . . 36

9.2 Performance of SVM classifier on validation set with HOG features of different cell sizes 36

9.3 Performance of different classifiers with CNN Features on test set 40

B.1 Train-Val-Test Split 63

Chapter 1

Introduction

Figure 1.1: Railway Track image provided by Bihrle Applied Research Inc.

Railway ties are the rectangular supports for the tracks[6]. See Figure 1.1. They transfer load from the rails to the ground, hold the rails upright and maintain correct distance between the tracks. They are generally made from wood, but concrete, steel and plastic composites are also used in making railway ties these days. However, according to statistics provided by the Railway Tie Association, as of 2015, wood maintains a 93%+ share of the market for ties installed in North America[7]. These wooden ties are susceptible to damage due to

friction, weather and pests. As they age, they develop cracks, which allow sparks to lodge easily, leading them to catch fire. In order to effectively and efficiently maintain the railroads track structure, railroad maintenance officers need to have accurate knowledge of the exact condition of the track and its key components. While inspection tools used for measurement of rail and track geometry condition have been around for many years, inspection tools for railway ties are only now becoming available to supplement and complement the traditional tie inspector's calibrated eye. This is in spite of the fact that cross-ties represent the second largest cost area, accounting for between 20% and 40% of railroad track maintenance costs[1].

The traditional method of inspecting railway ties involves experts who walk the track, visually inspecting the condition of the ties. They keep track of the bad ties by using a mechanical counter, merely categorizing the ties as good vs. bad. Also, since the experience and skills of the tie inspectors varies, the categorization is neither accurate nor consistent. In recent years, several new systems for monitoring and recording tie condition have been introduced. Inspection systems such as the TieInspect (see Figure 1.2) are used to record the tie condition and plan replacement activities accordingly. This hand-held computerized data collection and analysis system allows the tie inspector to record the condition of each tie individually but still depends on the tie inspector's visual observations. Recently, ground vehicles equipped with various sensors are being developed for such inspections. But the use of ground vehicles blocks the track for the railways. Moreover, these vehicles need to be custom designed and could be expensive.

Figure 1.2: TieInspect [1]

The aim of this work is to automate the inspection of the condition of railway ties using

computer vision and machine learning techniques. The work is motivated towards reducing the constant human supervision by employing machine learning models to imitate the expert's knowledge of categorizing railway ties. By using Unmanned Aerial Vehicle for image capture, the track is left untouched. Four different approaches are explored in this work for categorizing the railway ties:

1. Handcrafted Features with different classifiers
2. Histogram of Oriented Gradients features with Support Vector Machine
3. Transfer Learning using AlexNet
4. Proposed Convolutional Neural Network using domain knowledge

The performance of the approaches is measured using metrics that are compatible with the ordered classification task. Data Augmentation is used to artificially enlarge and enrich the dataset. The use of transfer learning for this task of categorizing railway ties has been explored. A modified CNN architecture is proposed using domain knowledge and is trained and tested on the dataset used in this work. The problem of label noise is analyzed by comparing labels from two labelers. Potential improvements in the proposed approaches are also discussed and new areas of further research are identified. It is believed that such an automated inspection can extend to other infrastructures as well and detailed reports for the infrastructural conditions can be generated, thus reducing time, cost and continuous dependency on human experts.

The structure of the rest of this document is as follows: Chapter 2 discusses the related work in this field, Chapter 3 gives relevant background information in Machine Learning techniques, Chapter 4 describes the dataset, Chapter 5 discusses the approach taken to detect and deal with ballast as an occlusion, Chapter 6 describes the feature space, Chapter 7 includes background of classifiers and related techniques as well as performance metrics used in this work, Chapter 8 explains the proposed CNN architecture, Chapter 9 focuses on the

experiments conducted in this work and the corresponding results along with comparative results across all approaches, Chapter 10 draws conclusions based upon the results and Chapter 11 recommends potential future directions of work.

Chapter 2

Related Work

Most of the railway inspection work still needs human supervision, even though autonomous techniques have been researched. This chapter describes some of the previous works that were aimed towards automating the inspection of railway track and its components.

Resendiz et al.[8] focus on developing a machine-vision system to supplement the visual inspections. They propose the use of a Video Track Cart that takes continuous video of the track underneath and localizes the track components using edge detection, texture information and template matching. Their efforts are focused on inspection of cut-spikes, rail anchors and turnout components. They use the technique of Gabor filtering for detecting ballast vs. non-ballast areas. In a follow up work[9], they have explored the use of MUSIC algorithm to detect the periodically occurring railway track components. Their focus is on detection of parts of the railway track rather than classification. The work in [10] is very similar.

A technical report by the FRA[11] describes the work by Computer Vision Laboratory at University of Maryland in developing algorithms for railway inspection. They propose a crack detector based on decomposing images into edge and texture components, a missing/broken fastener algorithm based on computer vision features and a statistical classifier,

and a crumbling/chipped tie detector based on a material classifier that uses a deep convolutional neural network architecture. They work with concrete ties dataset. They state that their crack detector is not mature enough to distinguish between cracks and edges. They identify automated tie grading as a future work.

The work in [12] deals with using SVM classifier with handcrafted features. Cracks are detected using image pre-processing and morphological operations and are then categorized based on their length and width. These features are used to categorize the ties as good vs. bad. However the crack detection is dependent on various thresholds for the morphological processing and its generality is questionable.

The system proposed for gauge detection in[13] is being used extensively in the field. The remaining algorithms for bolt detection, missing fasteners, detection of scarring of crossties, etc. have been developed and tested in a controlled laboratory environment. Grading of the ties has not been explored. Scarring in the ties is segmented using orientational filters and no machine learning techniques are used in the tie related work.

[14], [15] and [16] deal mostly with detection of railway track components only. Gibert et al. [17] show that a multi-task learning framework gives improved detection of the track components and also results in better accuracies for detecting defects in railway ties and fasteners. They detect crumbled and chipped tie conditions using a fully convolutional network trained on different classes of materials. Their multi-level classification is focused more on the condition of the fasteners. They do not perform graded classification for the ties.

Chapter 3

Background in Machine Learning techniques

3.1 Support Vector Machine

Support vector machines (SVMs) are a subset of supervised machine learning methods. SVMs are popular for their success in classification tasks. SVM constructs a hyperplane or a set of hyperplanes in a high-dimensional feature space, which is then used to create a decision boundary for classification[18]. The hyperplane maximizes the distance to the nearest training-data point of any class as shown in Figure 3.1. Twice this distance is defined as the margin of the SVM. Larger the margin, lesser is the generalization error of the SVM classifier. SVMs can efficiently perform a non-linear classification using the kernel trick. SVMs are very effective in high dimensional spaces, especially in cases where number of dimensions is greater than the number of samples. Since SVMs use only a subset of training points in the decision function (called support vectors), they are also memory efficient. The ability to define kernel functions makes them versatile. The disadvantage of support vector machines is that if the number of features is much greater than the number of samples, the method is likely to give poor performances[19].

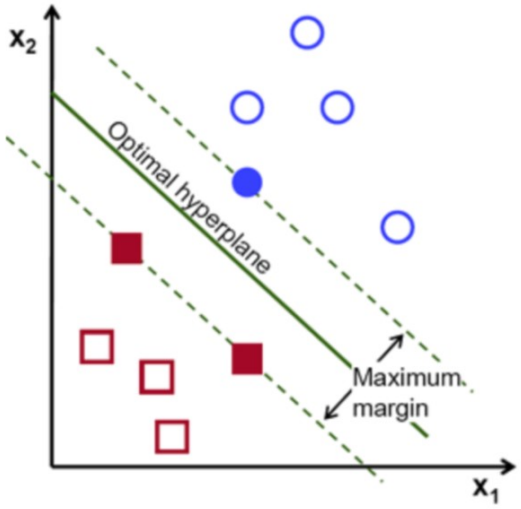


Figure 3.1: Optimal Hyperplane with maximum margin for SVM [2]

3.2 k-Nearest Neighbors

k-nearest neighbors (k-NN) is a non-parametric machine learning technique used for classification. In k-NN classification, the test instance is classified based on the classes of its k nearest neighbors in the feature space[20]. Either a majority vote or a weighted majority vote of the neighbors is considered, which results in the test instance being assigned to the most common class among its k nearest neighbors. If k = 1, then the test instance is simply assigned to the class of the single nearest neighbor. The distance function used and the value of k are the hyperparameters in this case, they determine the classifier's output.

1-NN is not very robust. The classification decision of each test sample relies on the class of a single training sample, which may be incorrectly labeled or atypical. kNN for $k > 1$ is more robust. kNN is not suitable for high-dimensional data. In high dimensional input space, similar datapoints may still be separated by large distances. This problem is referred to as the Curse of Dimensionality.

3.3 Decision tree

Decision Tree is a non-parametric supervised machine learning method used for classification. The objective is to create a model that predicts the value of a target variable by learning simple decision rules inferred from the training data features[21]. The decision tree algorithm builds classification models in the form of a tree structure. The core algorithm for building decision trees called ID3 was proposed by J. R. Quinlan which uses entropy and information gain to construct a decision tree[22]. An advantage of Decision trees are that they are simple to interpret and visualize. However, they are prone to overfitting. Mechanisms such as pruning, setting the minimum number of samples required at a leaf node or setting the maximum depth of the tree are effective in dealing with this problem. Decision tree learners create biased trees in case of unbalanced classes. It is therefore recommended to balance the dataset prior to fitting with decision tree.

3.4 Naive Bayes

Naive Bayes methods are supervised machine learning algorithms based on applying Bayes theorem with some naive assumptions. One of those assumptions is independence between every pair of features. Naive Bayes also assumes positional independence between the features. Given a class variable y and a dependent feature vector x_1 through x_n, Bayes theorem states the relationship according to Baye's rule of independent events. Some variations of naive Bayes classifiers have various distinct assumptions regarding the distribution of $P(x_i \mid y)$[23]. Even though Naive Bayes classifiers are easy to implement and have some assumptions about the data, they have worked quite well in many real-world situations, particularly text-document classification and spam email detection. One of the main advantages is that they require small sized training data to estimate the model parameters compared to other similar techniques. This is because parameters here can be found in linear time by calculating a closed form solution using Maximum likelihood technique. On the other

hand more expensive techniques perform iterative approximation. Another reason for speed is that this classifier solves problems rising from curse of dimensionality by expressing the ability to estimate feature distribution independently as a one dimensional distribution[24].

3.5 Linear Discriminant Analysis

Linear Discriminant Analysis (LDA) is a generalized version of Fisher's linear discriminant, a classical machine learning method used for classification tasks to find a linear combination of data features that is used for creating a decision boundary which then separates two or more classes of data samples. The resulting combination may be used as a linear classifier. LDA is also commonly used for dimensionality reduction before performing classification[25].

The representation of LDA consists of statistical properties observed from the given data distribution, calculated for each class label. For a single input variable (x) it is the mean and the variance of the variable for each class label. For multiple variables, it is easily extended to multivariate Gaussian, namely the means and the covariance matrix. This is then used to generate the LDA equation which makes the predictions. LDA also has some assumptions about the data as it simplifies the problem. First is that the data is Gaussian. It also assumes that each attribute has the same variance[26]. In reality, the class means and covariances are unknown. They are approximated from the training data using the maximum likelihood estimation or the maximum a posteriori estimation techniques.

3.6 Logistic Regression

Logistic regression, is a machine learning model where the dependent variable is a class label. It is a case of a binary dependent variable that is, where it can take only two values, 0 and 1[27]. The binary logistic model calculates the probability of a binary response based on data features. These are assumed to be independent of each other. Multinomial logistic regression

is a classification method that generalizes logistic regression to multi label problems. Here the model predicts the probabilities of the different possible outcomes of a class label, given a set of independent data features[28].

3.7 Ensemble Approach

Ensemble methods are learning algorithms that construct a set of classifiers and then classify new data points by taking a weighted vote of their predictions [29]. Ensemble learning is primarily used to improve the performance of a model, or reduce the likelihood of an unfortunate selection of a poor one. Using an ensemble of such models - instead of choosing just one - and combining their outputs by - for example, simply averaging them - can reduce the risk of an unfortunate selection of a particularly poorly performing classifier. Combining classifiers may not necessarily beat the performance of the best classifier in the ensemble, but it certainly reduces the overall risk of making a particularly poor selection. If proper diversity is achieved, a different error is made by each classifier, strategic combination of which can then reduce the total error. Figure 3.2 graphically illustrates this concept, where each classifier - trained on a different subset of the available training data - makes different errors (shown as instances with dark borders), but the combination of the (three) classifiers provides the best decision boundary[3]. Here, two approaches to Ensemble Learning are considered.

Weighted Majority Voting

Voting based methods of Ensemble learning operate on only the labels predicted by the set of classifiers in the ensemble[3]. Here, the predictions of the individual classifiers are weighted based on their performance. The highest performing classifier gets the maximum weight assigned to its prediction. The weights decrease gradually for the other classifiers based on their performance. This can essentially be thought as a scenario where each classifier votes

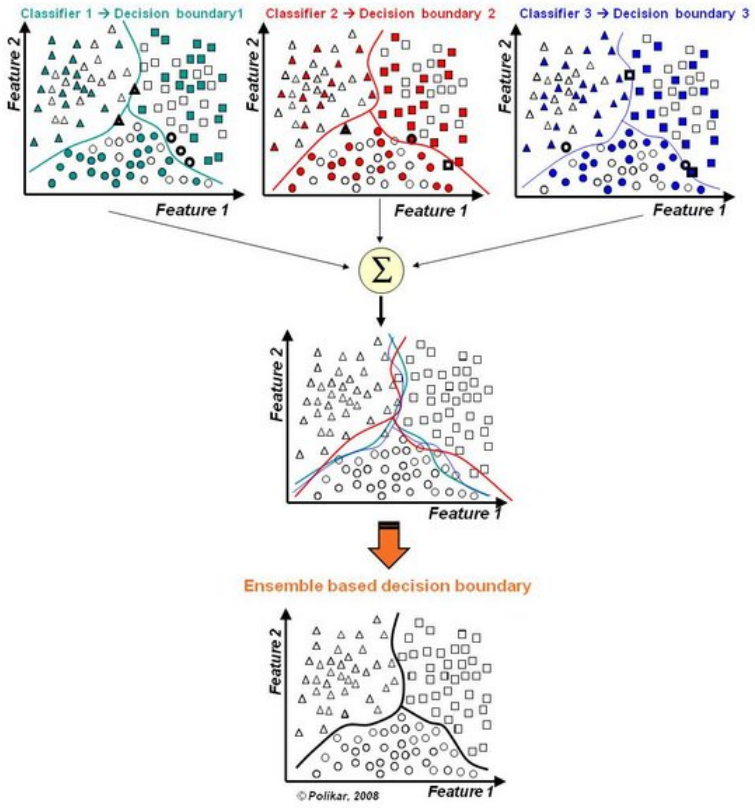


Figure 3.2: Ensemble Classifier [3]

for the correct label for each query, with the better performing classifiers getting more votes.

Stacking

Stacking, also called stacked generalization, is the technique of training a learning algorithm to combine the predictions of the set of classifiers in the ensemble. The first step is to train the individual classifiers on the training data. This is followed by training another classifier on the predictions made by the individual classifiers. Stacking has been used in various regression, classification tasks in both supervised and unsupervised learning environments. It has been shown to outperform the technique of averaging the classifiers' predictions in [30]. Logistic regression model is a popular stacker model.

3.8 Convolutional Neural Networks

Convolutional Neural Networks may be considered as a special case of Neural Networks where there is an explicit assumption that the input is an image. CNNs have neurons with three dimensions- width, height and depth. They thus generate a response based on a local region in the image. Parameter sharing is used in CNNs to control the number of parameters when dealing with the high dimensional input data (images). This takes advantage of the fact that local regions in an image are correlated. Like Neural Networks, CNNs are also formed by stacking of hidden layers of the neurons, but they have some special layers as shown in Figure 3.3[31].

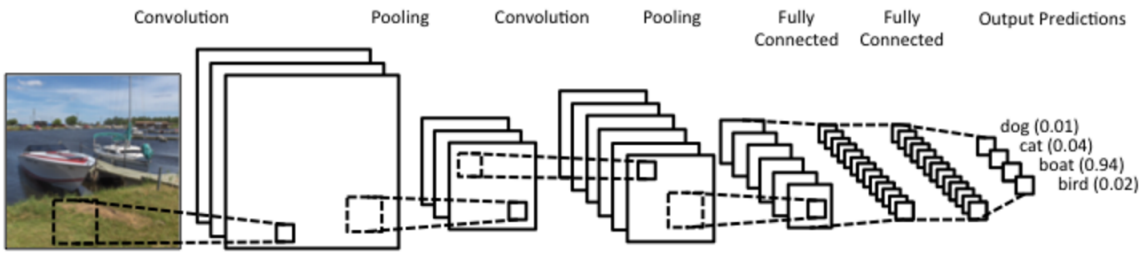


Figure 3.3: CNN Architecture

The convolutional layer is the core layer of the CNN, consisting of convolutional filters of specified width, height and depth. The number of filters, their spatial extent, stride of convolution and padding are the hyperparameters for this layer. These filters are responsible for learning representations such as edges or blobs in the earlier layers to more complex image regions such as wheels or honeycombs in the further layers. Pooling layers are periodically inserted between successive convolutional layers in a CNN architecture, with the intention of reducing the amount of parameters and computation in the network, and hence control overfitting. They can be implemented using either the max or the average function. Normalization layers are used for brightness or contrast normalization. Fully connected layers are essentially the layers used in Neural Networks with connections to all activations of

the previous layer. Apart from this, Rectified Linear Units and its variations are used in CNN architectures to enable fast learning and to address the problem of vanishing gradients. Dropout layers are used for regularization so as not to overfit[32]. In this technique, the output of each neuron is set to zero with a probability of 0.5. The result thus works like an ensemble. These layers are repeated to form deeper networks that achieve better performances, until a certain depth.

CNNs have gained huge popularity in the recent years, largely due to the availability of large datasets, GPUs, sophisticated regularization techniques which enable CNNs to achieve near-human or sometimes better performance in a large number of tasks. The technique of Transfer Learning has enabled the use of CNNs on small datasets too, either as feature extractors or by finetuning.

Chapter 4

Description of Dataset

4.1 Image Acquisition

The images used in this work are provided by Bihrle Applied Research Inc. An Unmanned Aerial Vehicle (UAV) is flown over the railway tracks and an imaging sensor mounted on the UAV takes pictures of the tracks underneath. The UAV used in this case is a small hybrid fixed-wing aerial system weighing less than 55 lbs. It has a payload capacity of 5 lbs and an endurance of 5 hours. The sensor is monochrome, 35 mm format 28.8 Mega Pixel CCD sensor. It has global shutter that ensures high-quality images of moving objects without blur or distortion. The sensor is mounted on a stabilized tracking gimbal to account for vibration of the drone or sudden movements due to poor weather conditions. The target altitude for the UAV is 400 feet Above Ground Level and the cruise speed is approximately 40 knots airspeed. No image processing is implemented to account for the motion blur. Images are captured only during the daylight hours and no special lighting equipment is used. Each image is tagged with the corresponding latitude/longitude/altitude information along with the gimbal mount rotation. From the image of the entire track, each tie is cropped out. These cropped out tie images constitute the dataset used in this work. All images in this dataset are of wooden ties.

4.2 Image Categorization

A number of factors contribute to faults in the railway ties. Breaks and splits are caused in the ties due to load or impact acting cross-wise and parallel to the grain of the wood. Cuts are caused in the ties either due to the movement of plate over the tie or due to equipment (railways) moving across the grain of the wood. Apart from these faults, the wood of the tie might also get crushed or rot due to weather and pests. These faults are factored in while inspecting the tie condition and the tie is categorized depending on the severity of these defects. The condition of the tie indicates the expected remaining life of the tie and thus determines whether it needs to be replaced and when.

In this work, the dataset is split into 3 categories namely 'broken', 'cracked' and 'good' as shown in Figure 4.1. The guidelines for the labeling were provided by Bihrle Applied Research Inc.

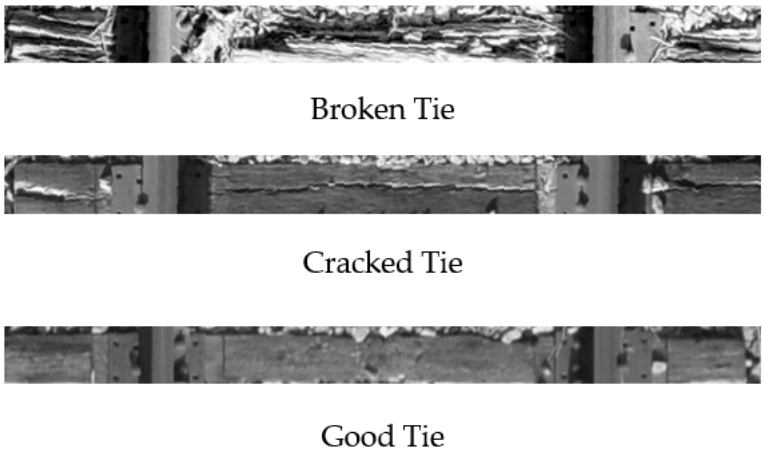


Figure 4.1: Categories of Ties - Broken, Cracked, Good

4.3 Data Augmentation and Cleaning

Data Augmentation is the technique of artificially enlarging a dataset by using label-preserving transformations[32]. This helps in reducing the chance of overfitting small datasets and also makes the models more generalized to variations in the data. The data augmentation techniques used in this work include horizontal flip, vertical flip, random brightness and contrast enhancement. These transformations are applied to randomly selected images from each category of the dataset. Apart from increasing the size of the data n-fold, this also generates virtual datapoints for different lighting and orientation of camera. As depicted through the results, the data augmentation technique gave a boost of 10% in Quadratic Weighted kappa score compared to non-augmented data in the evaluation metric chosen (QWK score described in section 6.7).

Machine learning models unveil the knowledge already hidden in the data. Thus, the quality of the data that is fed into the model plays a key role in determining the ability of the model to learn intricate relationships in the data. Data Cleaning is the process of identifying and correcting or removing inaccurate datapoints in order to ensure high quality of data. This work assumes that the pre-processing methods applied to generate the cropped out tie images cater to rotated images. However, it was observed that some of the images in the dataset contained only small portions of the railway tie due to the tie being extremely angled. As the feature extraction algorithms work on the entire portion of the image, such images would lead to spurious representations of the category. In some other images, the background seemed to occupy a significant part of the image. This would cause the classifier to learn on features from the background. In order to avoid these scenarios, such images were eliminated from the dataset.

Once the cleaned dataset is obtained, it is split into training, validation and test set for experimentation. The details of the split are included in Appendix B.

Chapter 5

Ballast Detection

Ballast, made out of crushed stone is an important part of the railroad infrastructure. It forms the trackbed upon which the ties are laid. The ballast serves the functionalities of bearing the load from the railroad ties, facilitating drainage of water and keeping away vegetation that might interfere with the track structure. It gives strength and rigidity to the railway track structure and at the same time allows flexibility for when the trains pass over the tracks.

However, sometimes the ballast ends up on top of the ties. This poses as an occlusion for the faults classification task as it may hide the cracks underneath or produce spurious filter responses. To avoid this, the ballast is first detected and then blurred out. This does not solve the problem of hidden cracks but curbs the spurious filter responses during feature extraction for the classification task. The approaches taken to detect the ballast and their results are discussed in the following sections. Note that the results are visually compared since no ground truth labeling for the ballast was available.

5.1 Sobel Filter Approach

This approach uses the multiplication of the responses of horizontal and vertical Sobel filters applied on the tie image. The assumption is that the cracks will show up in the response of just the Sobel Y filter, whereas the ballast will show up in the responses of both the Sobel X and Sobel Y filters. The binary image obtained as a product of applying both the filters is then subjected to the morphological operation of opening to clean up spurious results. This approach can detect the ballast but suffers from false positives. This is particularly harmful because it might blur out potential cracks in the image; for instance when a crack is jagged.

5.2 Gabor Filters Approach

Following the intuition of the Sobel filter approach, this approach uses Gabor filters in 5 directions and 3 scales. The responses of the varying directional filters are multiplied (AND operation) and those of the varying scale filters are summed (OR operation), followed by morphological operations to get the final result. The intuition behind multiplying the directional filter responses and summing the scale filter responses is that the ballast at a particular scale will show up in all directions. This approach works better at identifying the ballast for the subsequent blurring task.

5.3 Blurring the Ballast

The ballast regions detected using the Gabor approach are blurred using Gaussian blurring function. As seen in the Figure 5.1, the blurring is not completely seamless and is dependent on the performance of the ballast detection.

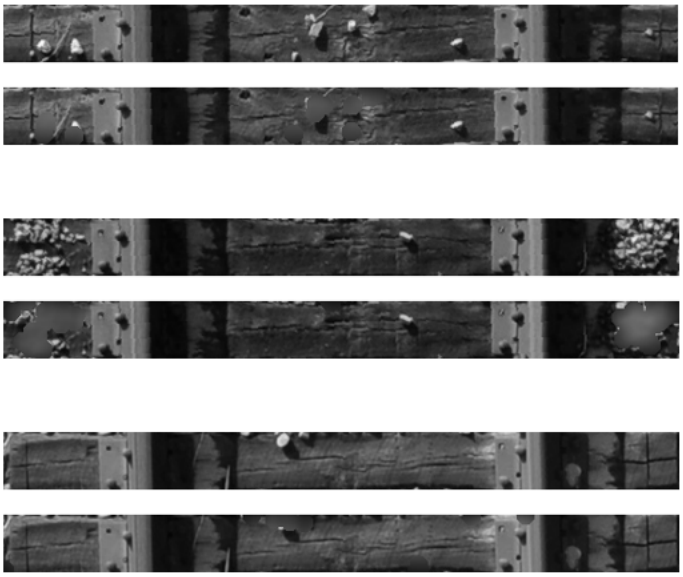


Figure 5.1: Ballast Blurring

Chapter 6

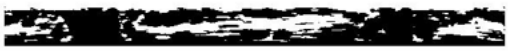
Features

The input data, images in this case, is typically high dimensional, redundant and variable. Features are low dimensional, mathematically and computationally convenient representations of the raw input data which facilitate learning for further classification/regression tasks. Each feature encodes a certain measurable property of the phenomenon being observed. Feature engineering is the process of selecting informative, discriminating and independent features. Traditional handcrafted features rely on domain knowledge. Features can also be learned themselves through a technique called 'feature learning'. This automates the process of feature engineering and also makes the features more generalized to various datasets of the same task. When labeled input data is used for feature learning, it is called supervised feature learning. The following sections detail the different features experimented for the classification task.

6.1 Handcrafted Features

In this approach, the tie images are first subjected to median filtering to remove noise while still preserving the edges. In each 10×10 window of the image, the median of all the

pixel values is calculated and placed at the center of the window in the corresponding output image. This is followed by applying the Sobel X filter to find edges in the horizontal direction which correspond to the cracks. The response image is thresholded to get the binary image as shown in Figure 6.1. The following set of hand-designed features are then extracted from these binary images.



Broken Tie pre-processed for Handcrafted features

Cracked Tie pre-processed for Handcrafted features

Good Tie pre-processed for Handcrafted features

Figure 6.1: Ties pre-processed for computation of Handcrafted features

1. Density of bright pixels

   This is the ratio of bright pixels to total size of the image.

2. Mean of areas of the connected components

   Connected components are found in the binary images using the 4 and 8 connectivity. The areas of the regions formed are calculated and their mean is considered for this feature. Regions with areas smaller than a threshold are rejected while computing the mean.

3. Mean of eccentricities of the connected components

Eccentricity is a measure of how much the region deviates from being a perfect circle. Higher values of eccentricity are indicative of the region being more like a line segment. Hence, the cracks will tend to have higher eccentricity values. Eccentricity values are calculated for each connected component and their mean is considered for this feature. Regions with eccentricity values smaller than a threshold are rejected while computing the mean.

4. Density of bright pixels in three parts of the ties

   The three parts of the ties are left field side, gauge side and right field side of the tie as shown in the Figure 6.2. The ratio of bright pixels to the size of the part is calculated for each of the three parts. This feature is a vector of these three values.

Figure 6.2: Railway Tie - field side and gauge side

5. Mean of width of bounding boxes of connected components

   Bounding boxes are found for each region defined by the connected components. The width of the bounding boxes is an indication of the thickness of the crack. Mean of the width of the bounding boxes in the concerned tie image is considered in this feature.

6. Standard deviation of width of bounding boxes of connected components

   The standard deviation of the width of the bounding boxes is a measure of how much the thickness of the cracks varies in the concerned tie image.

6.2 HOG Features

Histogram of Oriented Gradients(HOG) has become a popular feature descriptor in computer vision since its success in Human Detection [33]. HOG descriptors are used for object detection, classification and tracking. The intuition behind using HOG features here is to capture the gradient orientation in localized rectangular parts of the tie images. The steps involved in the computation of the HOG features are described as follows[33].

In the first step, the gradient values at each pixel are computed using derivative masks. The second step creates the orientation histograms for each cell of the image. The number of bins and the cell size determines the level of fine details that are captured. Large cell-sizes are used to capture large-scale information whereas small cell-sizes capture the finer details of the image. Larger number of bins lead to encoding of finer details but increase the processing time. In the next step, the cells are grouped into larger blocks which typically are made to overlap and the gradient values are locally normalized. The size of the block determines the ability of the feature descriptor to account for illumination and contrast changes. An overlap of half the block size has been shown to achieve adequate contrast normalization. The resultant HOG descriptor is a vector formed by the concatenation of the normalized cell histograms from all the block regions. The length of this HOG descriptor is given by N_HOG such that -

$$N\_HOG = NumBlocks \times BlockSize \times NumBins$$

where Numblocks is the number of blocks in the image, BlockSize is the number of cells in a block and NumBins is the number of bins in the orientation histogram computed for each cell[34].

These parameter values are tweaked to cater to the tie images. The HOG descriptors are thus used as representations of the tie images that can be fed into the classifiers.

6.3 Pre-trained CNN Features

CNNs are typically trained using large collections of diverse images. From these large collections, CNNs can learn rich feature representations for a wide range of images. Yosinksi et al.[35] have shown that many deep neural networks trained on natural images exhibit a curious phenomenon in common: on the first layer they learn features similar to Gabor filters and color blobs. Such first-layer features appear not to be specific to a particular dataset or task, but general in that they are applicable to many datasets and tasks. The features transition from being general to specific as the depth increases with the last layer being completely specific to the task at hand. This phenomenon allows us to take a pre-trained CNN, remove the last fully-connected layer and then treat the rest of the CNN as a fixed feature extractor for the new dataset. These features are referred to as CNN codes. Razavian et al.[36] have shown that these feature representations performed superior to the highly tuned state-of-the-art systems on a variety of tasks. They concluded saying that deep learning with CNN was to be considered as the primary candidate in essentially any visual recognition task thereon. Using a pre-trained CNN as a feature extractor is also an easy way to leverage the power of CNNs, without investing time and effort into training. For small datasets, like the one in this work, this is often a great tool to use the CNNs without overfitting.

In this work, AlexNet[32] pre-trained on ImageNet data is used as the base network. The architecture of this network is as shown in Table 6.1. The last fully-connected layer of AlexNet is designed to output 1000 class scores for the ImageNet data. This layer is eliminated and the rest of the network is used as a feature extractor. It gives a 4096-dimensional vector for every tie image that contains the activations of the fc7 layer. This feature vector is the representation of the tie image that is further used for training the classifiers.

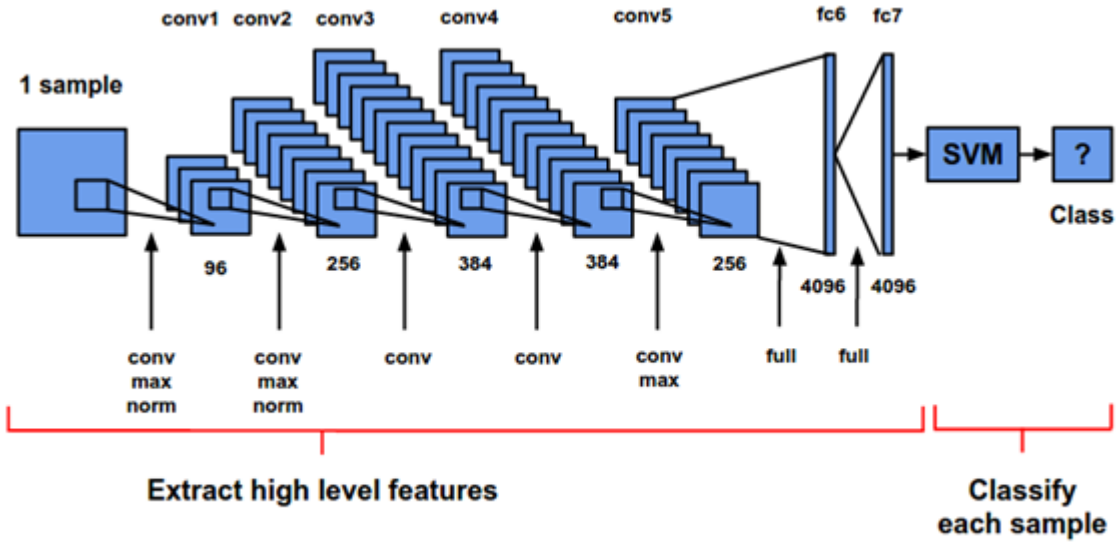


Figure 6.3: CNN as Feature Extractor [4]

Table 6.1: AlexNet Layers

Name	Description
data	Image Input - 227x227x3 images with 'zerocenter' normalization
conv1	Convolution - 96 11x11x3 convolutions with stride [4 4] and padding [0 0]
relu1	ReLU
norm1	Cross Channel Normalization - cross channel normalization with 5 channels per element
pool1	Max Pooling - 3x3 max pooling with stride [2 2] and padding [0 0]
conv2	Convolution - 256 5x5x48 convolutions with stride [1 1] and padding [2 2]
relu2	ReLU
norm2	Cross Channel Normalization - cross channel normalization with 5 channels per element
pool2	Max Pooling - 3x3 max pooling with stride [2 2] and padding [0 0]
conv3	Convolution - 384 3x3x256 convolutions with stride [1 1] and padding [1 1]
relu3	ReLU
conv4	Convolution - 384 3x3x192 convolutions with stride [1 1] and padding [1 1]
relu4	ReLU
conv5	Convolution - 256 3x3x192 convolutions with stride [1 1] and padding [1 1]
relu5	ReLU
pool5	Max Pooling - 3x3 max pooling with stride [2 2] and padding [0 0]
fc6	4096 fully connected layer
relu6	ReLU
drop6	50 per cent dropout
fc7	4096 fully connected layer
relu7	ReLU
drop7	50 per cent dropout
fc8	1000 fully connected layer
prob	Softmax
output	Classification Output - 'tench', 'goldfish', and 998 other classes

Chapter 7

Performance Metrics for Classification

Performance metrics in classification are an important tool to assess the quality of the learned models. Ferri et al.[37] talk about different performance measures and their choices with respect to specific applications in. The number of correct classifications made among all the predictions by the classifier is called the accuracy. The accuracy, by itself is often a deceiving measure of the performance. This is because, a classifier might be getting a good accuracy by just predicting the most frequent class label in the training set. When the test labels are available, building a confusion matrix allows better visualization of the performance of the classifier. A confusion matrix shows the predicted and actual classifications. For L distinct labels, the size of the confusion matrix is L $\times$ L[38]. From the confusion matrix, the accuracy can be calculated for each class. This is obtained as a ratio of number of times the label was correctly predicted to the total number of times that label was predicted by the classifier. The mean of such accuracies per class thus serves a better measure of the performance of the classifier. This is one of the performance metrics that is used in this work.

The classes considered in this work are ordered classes. This means that each class depicts a degree of presence of the faults. Thus, it is essential to consider the distance, in terms of labels, between the correct label and the predicted label. Assuming consecutive classes to be separated uniformly by a distance of 1, it follows that if the true label is 1 and the classifier

predicts 2, the distance is 1. Whereas, if the true label is 1 and the classifier predicts 3, the distance is 2. This degree of agreement between the true labels and predicted labels can be considered using the Kappa score. The Kappa score is the measure of degree to which two judges concur in sorting N items into k mutually exclusive categories[39]. The judges can be individual humans or computer programs. Originally developed to measure the agreement between two judges[40], the Kappa score has also been used to measure the performance of a classifier individually by comparing it to the true labels [41]. The Kappa score is given by the formula:

$$K = \frac{p_o - p_e}{1 - p_e} \tag{7.1}$$

In this case, p_o is the accuracy of the classifier and p_e is the probability that the agreement between the predicted labels and the true labels is by chance. The p_e is given by:

$$p_e = \frac{1}{N^2} \sum_k n_{k1} n_{k2} \tag{7.2}$$

where, N is the number of items being sorted into k categories. n_{ki} is the number of times judge i predicted category k.

Consider the confusion matrix shown in Figure 7.1 for further understanding of this concept.

		True Labels
Predicted Labels	20	5
	10	15

Figure 7.1: Sample Confusion Matrix

p_o is the sum of the diagonal elements of the confusion matrix divided by the total of all elements. It is the observed agreement between the predicted and true labels. Thus,

$$p_o = \frac{20 + 15}{50} = 0.7$$

According to the equation 7.2, p_e is calculated as -

n_{11} is the number of times label 1 appears in the true labels = 30

n_{21} is the number of times label 2 appears in the true labels = 20

n_{12} is the number of times the classifier predicts label 1 = 25

n_{22} is the number of times the classifier predicts label 2 = 25

And N = 20 + 10 + 15 + 5 = 50

$$p_e = \frac{1}{50^2}(30 \times 25 + 20 \times 25) = 0.5$$

Then, by applying equation 7.1,

$$\kappa = \frac{0.7 - 0.5}{1 - 0.5} = 0.4$$

It can be noted that this Kappa score treats disagreements between all categories equally. However, as mentioned before, the categories in the classification problem in this work are ordinal. It is thus potentially meaningful to consider not just the diagonal elements representing absolute concordances but also the off-diagonal elements representing relative agreements. To enable this, each cell in the confusion matrix is now assigned a weight based on its distance from the diagonal. This modified Kappa is called the Weighted Kappa[42]. The weighing function can be linear or quadratic. This work uses the Quadratic Weighted Kappa score as a performance metric for the classifiers. The weights are give by-

$$weights = 1 - \frac{distance^2}{max\_distance^2}$$

where max_distance is the maximum possible distance for the problem. In this work, the distance between consecutive labels is assumed to be the same, thus max_distance is equal to the number of categories in the classification problem. The confusion matrix is weighed according to the weighing scheme and the rest of the calculations are similar to the simple Kappa. Figure 7.2 shows a guide to interpret the QWK scores.

Value of QWK	Strength of agreement
< 0.20	Poor
0.21 - 0.40	Fair
0.41 - 0.60	Moderate
0.61 - 0.80	Good
0.81 - 1.00	Very good

Figure 7.2: Guide to interpret QWK score [5]

In addition to these absolute metrics, the performance of a classifier can also be evaluated by comparing it to a baseline and measuring the percent improvement over the baseline. This work also includes such comparisons.

Chapter 8

Proposed CNN Architecture

The proposed architecture is a modification of AlexNet, considering that it performs well on the dataset with the transfer learning approach as shown in the upcoming chapter. The three main modifications are: input layer, depth and convolutional filter size for the first convolutional layer.

The input layer for popular CNN architectures is symmetrical - 32×32, 227×227 and so on. While using these CNNs on images of other sizes, the images are resized to the size of the input layer. The images of railway ties are rectangular with a width of about 46 pixels and length of around 580 pixels. They become very distorted when resized to be compatible with AlexNet's input layer size and could potentially loose important image details. Thus, the input layer in the proposed architecture is defined to be $46 \times 580 \times 1$, since the images are gray-scale. Secondly, the entire depth of the AlexNet and thereby the large number of parameters might not be essential for this straightforward task which does not including complex semantic details. Thus, the proposed architecture uses 1 convolutional layer in one version and 2 convolutional layers in the other. Their performance is compared in Chapter 9. Thirdly, the convolutional filters for popular architectures are also symmetric and two-dimensional. In this work, the tie images are categorized based on the cracks in them, which are horizontal in nature. Thus, 1-dimensional convolutional filters are proposed for the first

convolutional layer. This would also lead to lesser parameters.

With these intuitions, the proposed architecture with two convolutional layers is as shown in Table 8.1. In case of 1 convolutional layer, just the first set of conv-relu-crossnorm-maxpool layers is retained followed by layers from fc-1 onwards. The use of cross-channel normalization, ReLU and dropout layers is inspired from the AlexNet architecture[32].

Table 8.1: Proposed CNN Layers

Name	Description
imageinput	Image Input - 46x580x1 images with zerocenter normalization
conv-1	Convolution - 100 1x10x1 convolutions with stride [4 4] and padding [1 1]
relu-1	ReLU
crossnorm-1	Cross Channel Normalization - cross channel normalization with 5 channels per element
maxpool-1	Max Pooling - 2x2 max pooling with stride [2 2] and padding [0 0]
conv-2	Convolution - 200 5x5x100 convolutions with stride [1 1] and padding [2 2]
relu-2	ReLU
norm2	Cross Channel Normalization - cross channel normalization with 5 channels per element
maxpool-2	Max Pooling - 2x2 max pooling with stride [2 2] and padding [0 0]
fc-1	10 fully connected layer
relu-3	ReLU
dropout	50 per cent dropout
fc-2	4096 fully connected layer
softmax	Softmax
classoutput	Classification Output - label1-broken, label2-cracked, label3-good

Chapter 9

Experiments and Results

9.1 Execution Environment

The experiments in this work are executed in two environments - GPU and non-GPU. The less computationally intensive experiments such as those involving the handcrafted features are performed using Intel Core i5 CPU with two cores, 8 GB RAM and a 64-bit Windows OS. The more computationally intensive experiments involving training or finetuning of CNNs or hyperparamter optimization with high dimensional features are performed using GeForce 940M with a compute capability of 5, 8 GB RAM and 64-bit Windows OS. MATLAB 2015a, 2016b, 2017a versions are used as a platform with the Statistics and Machine Learning toolbox, Optimization toolbox, Image Processing Toolbox, Computer Vision Toolbox, Image Acquisition toolbox and Neural Network Toolbox over the course of the work.

9.2 Handcrafted Features

Handcrafted features as described in section 5.1 are extracted for each image in the training and test set. The features from the training set are used to train 6 different classifiers and

their performance is compared on the test set to get the best model. 5-fold cross validation is used for tuning the hyperparameters for each classifier. The performance of the classifiers is summarized in Table 9.1. The QWK score is calculated on the test set. The confusion matrix for the best performing classifier, Naive Bayes in this case, is shown in Figure 9.1. The confusion matrices for the other classifiers are given in Appendix A.

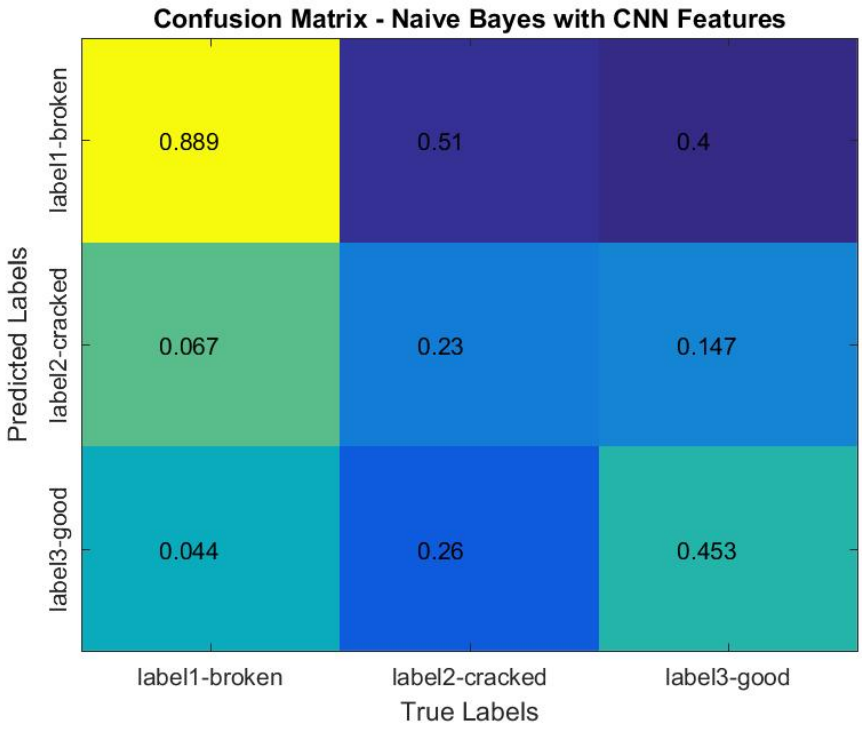


Figure 9.1: Confusion Matrix for Naive Bayes classifier with Handcrafted features

9.3 HOG Features

HOG Features with different cell sizes are extracted from the training set of images. SVM classifier with linear kernel is trained on these features. The intuition behind using SVM is that it can deal with large number of dimensions of the HOG features. The performance of the classifier for different window sizes is compared on the HOG features extracted from

Table 9.1: Performance of different classifiers with Handcrafted Features on test set

Classifier	Mean accuracy per class	QWK Score
Logistic Regression	49.65%	0.1996
LDA	50.33%	0.2122
kNN	49.78%	0.2128
Decision Tree	50.37%	0.2212
SVM	52.35%	0.2252
Naive Bayes	52.41%	0.2521

Table 9.2: Performance of SVM classifier on validation set with HOG features of different cell sizes

Cell Size	Mean accuracy per class
4×4	68.59%
8×8	70.71%
16×16	65.59%
4×8	68.36%
6×8	70.09%

images in the held-out validation set. This comparison is summarized in Table 9.2. The best performance is obtained for the 8×8 cell size. The performance of this model on the test set is reported as 73.07% mean accuracy per class and 0.5858 QWK score. The confusion matrix for this is shown in Figure 9.2. The images with the ballast blurred as described in the chapter 5 were used for HOG feature extraction. But this approach did not give better results.

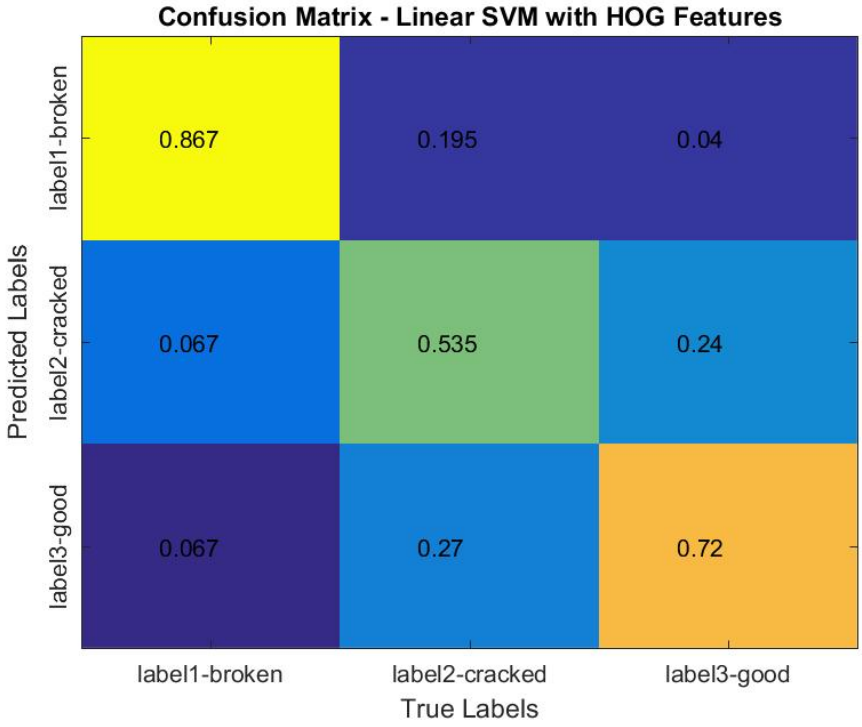


Figure 9.2: Confusion Matrix for SVM classifier with HOG features (Cell Size 8×8)

9.4 CNN Features

CNN features are extracted from the training and test set at the 'fc7' layer of AlexNet pretrained on ImageNet. Different classifiers are trained on the training set features. The parameters for these classifiers are tuned using hyperparameter optimization. Performance of these classifiers is compared on the test set in Table 9.3. The confusion matrix for the best performing classifier, SVM, is shown in Figure 9.4. The confusion matrices for the other classifiers are included in Appendix A. Activations from earlier parts of the network are also explored as CNN features, but do not yield good results.

Ensemble techniques of Weighted Majority Voting and Stacking are applied to labels predicted by each classifier. For the Weighted Majority Voting technique, the best performing algorithm (SVM in this case) is given the highest weight and the weights are reduced gradually for the other algorithms based on their performance. Weights are tuned on the validation

set. However, this technique does not yield better results than the best performing individual model in this case. The reason for this is that the predictions are not weakly correlated. Figure 9.3 shows the Pearson correlation coefficients for each pair of classifiers. This coefficient ranges from -1 to +1 with -1 indicating no correlation and +1 indicating very high correlation. All the values in Figure are almost 0.5 or above, indicating significantly strong correlation. Thus, the ensemble does not contribute to better results.

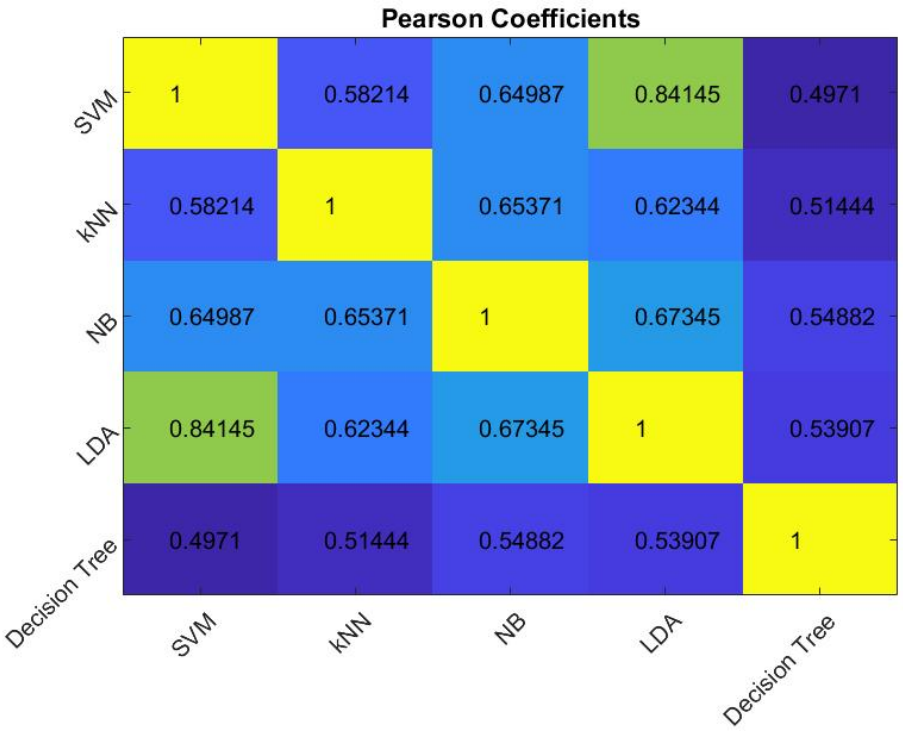


Figure 9.3: Pearson Coefficients between predicted labels on the test set by pairs of different classifiers

9.5 Finetuned AlexNet

AlexNet pretrained on ImageNet data is finetuned on the railway ties dataset. The last three layers are replaced by a fully connected layer, softmax layer and classification output layer to

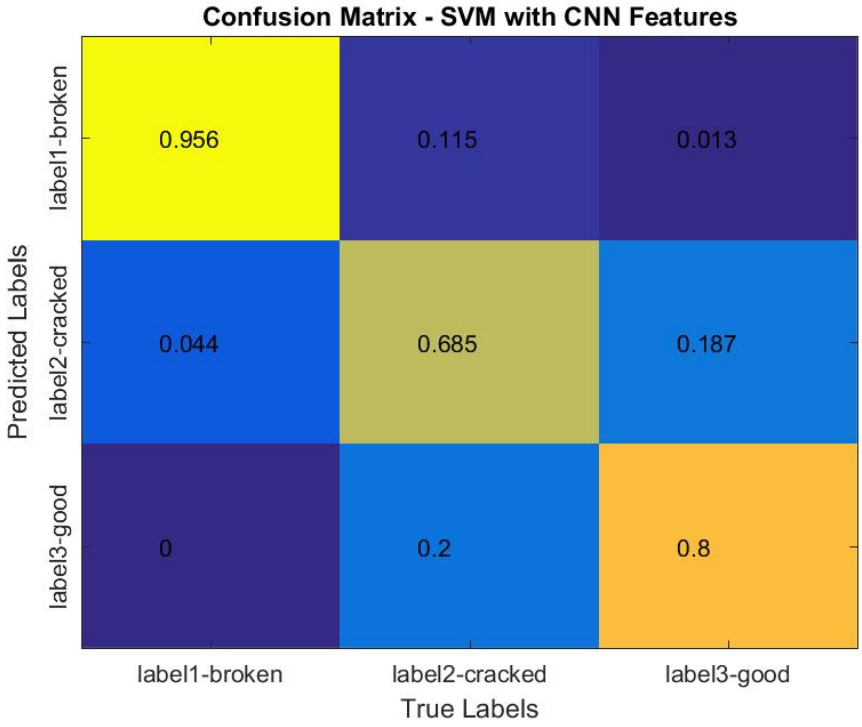


Figure 9.4: Confusion Matrix for SVM classifier with CNN features

cater to the three class classification problem instead of the original 1000 class classification for ImageNet data. The fully connected layer is set to have a size of 3. The learning rate of the weights and bias for this layer is set to be 20 times more than the global learning rate to speed up training. This network, formed as a concatenation of earlier layers from AlexNet and problem-specific layers at the end, is trained using Stochastic Gradient Descent with a momentum of 0.9. The learning rate is set at 0.0001. Dropping the learning rate after fixed number of epochs was experimented but did not yield better results in this case. The performance of the finetuned AlexNet at each epoch was validated on the validation set and the best performing state of the model was tested on the test set. Mean accuracy per class on the test set is 81.63% and the QWK score is 0.7006. The confusion matrix for this model is as shown in Figure 9.5.

Table 9.3: Performance of different classifiers with CNN Features on test set

Classifier	Mean accuracy per class	QWK Score
Decision Tree	61.28%	0.4699
Naive Bayes	73.05%	0.5834
kNN	75.11%	0.6011
LDA	79.56%	0.6786
SVM	81.35%	0.7044

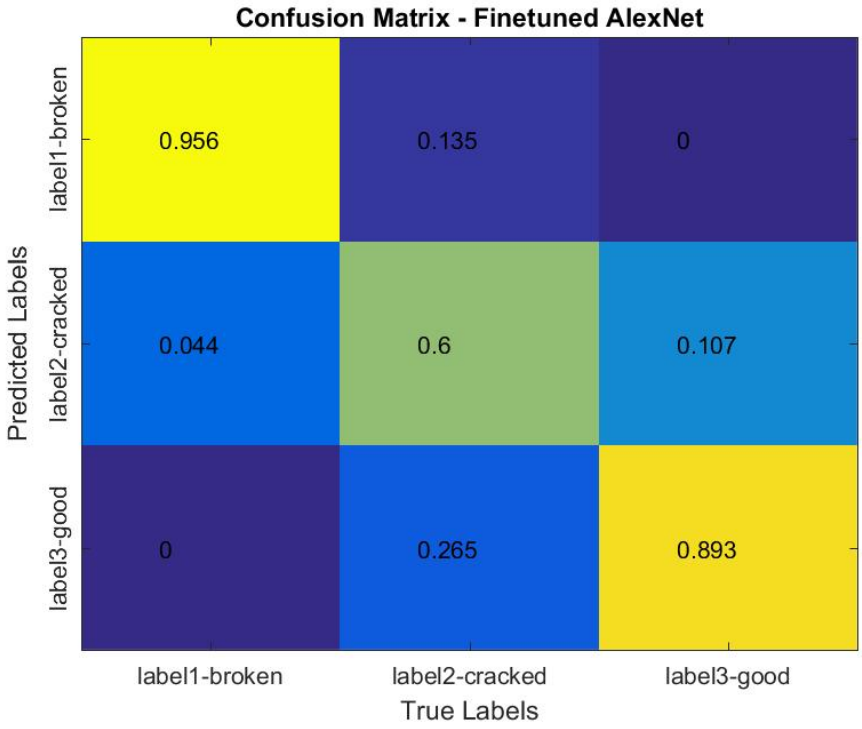


Figure 9.5: Confusion Matrix for Finetuned AlexNet

9.6 Proposed Architecture

9.6.1 1 convolutional layer

The proposed architecture with 1 convolutional layer as discussed in Chapter 8 is trained on the training set. Stochastic gradient descent with momentum of 0.9 and L2 regularization of

0.0001 are used for training. The learning rate is set at 0.0001. A learning rate drop schedule was experimented but yielded similar results in this case. The training and validation accuracy curve is as shown in Figure 9.6. The state of the network that performs best on the validation set is chosen to classify the test set. The mean per class accuracy on the test set is 61.43% and the QWK score is 0.4739. The confusion matrix is as shown in the Figure 9.7.

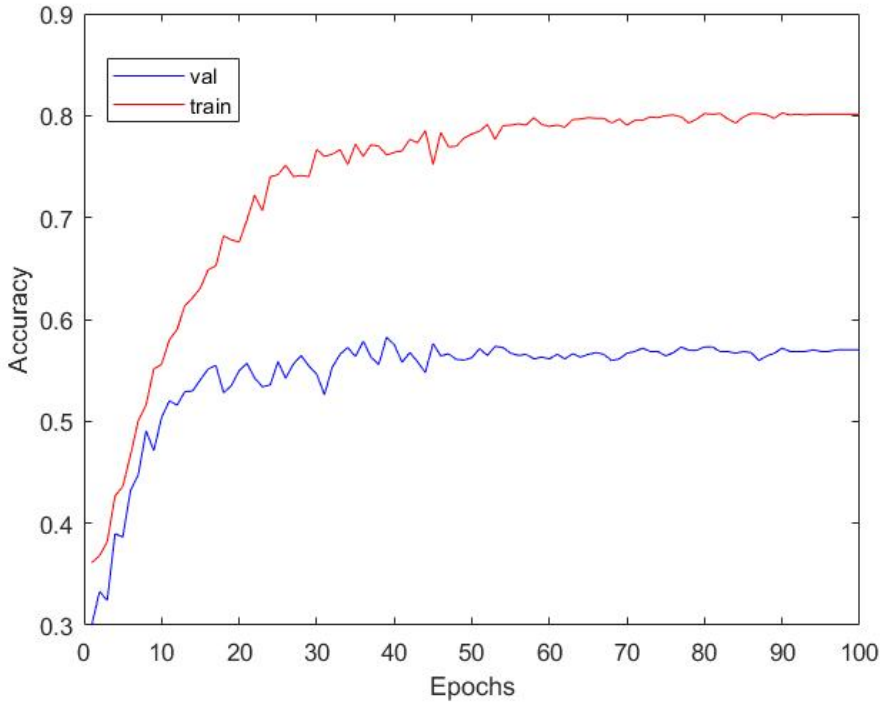


Figure 9.6: Training and Validation Curves for Proposed Architecture with 1 convolutional layer

9.6.2 2 convolutional layers

The proposed architecture with 2 convolutional layers as discussed in Chapter 8 is trained on the training set. Stochastic gradient descent with momentum of 0.9 and L2 regularization of 0.001 are used for training. The initial learning rate is set at 0.001 and is dropped by a factor

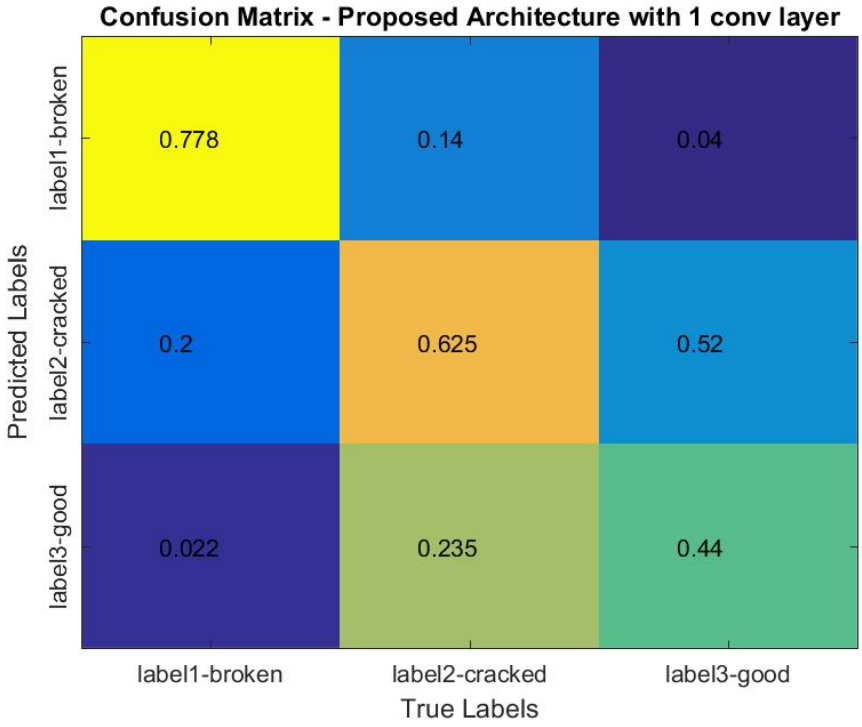


Figure 9.7: Confusion Matrix for Proposed CNN architecture with 1 convolutional layer

of 0.2 after every 25 epochs to deal with the problem of validation accuracy oscillations. These training options are determined empirically after several iterations of experimentation. The training and validation accuracy curve is as shown in the Figure 9.8. For the best performing state of the network on the validation set, the mean per class accuracy on the test set is 73.72% and the QWK score is 0.6294. The confusion matrix is as shown in the Figure 9.9.

9.7 Label Noise

The performance of supervised machine learning techniques depends on the quality of the data that is used for training the machine learning algorithms. The labels reflect the mapping between the data and the underlying concept in the mind of the labeler. These underlying concepts may vary from expert to expert, even more so in case of ordered categories. The

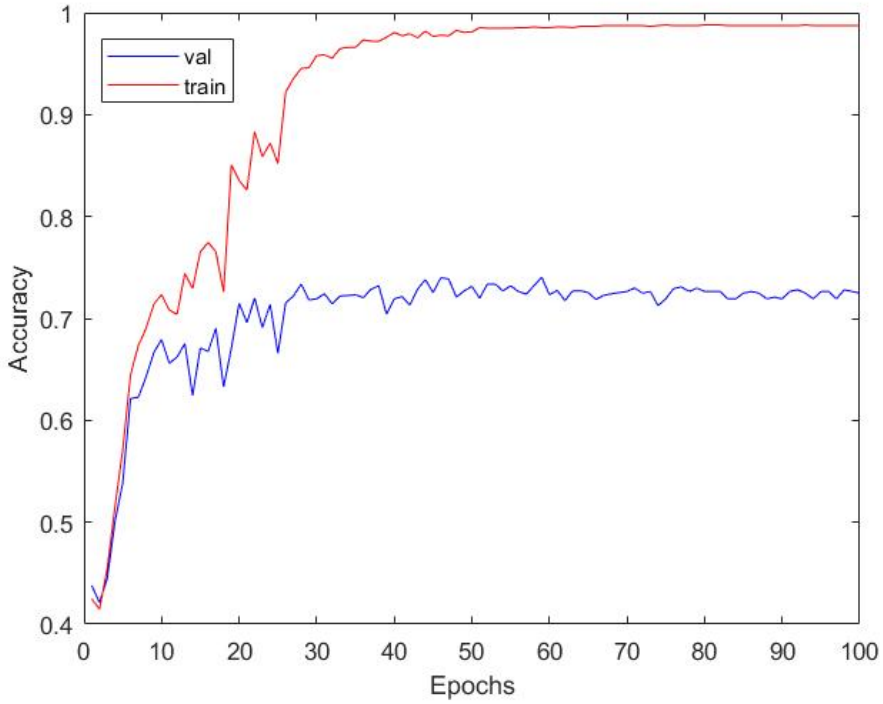


Figure 9.8: Training and Validation Curves for Proposed Architecture with 2 convolutional layers

labeling quality of an expert depends on many factors like the knowledge of the expert, his attentiveness and judging ability during labeling and the ambiguity in the data itself[43]. In this work, the confusion between label 2 and label 3 is evident from the images as well as from the confusion matrices for various algorithms. Note that the Naive Bayes classifier trained on handcrafted features (Figure 9.1), for instance, is able to classify the label1-broken tie images very well but it also classifies half of the cracked images as broken. In an attempt to study this confusion further, a part of the data is labeled by another expert. The degree of agreement between the two experts on this data is then measured using the Quadratic Weighted Kappa Score. The QWK score obtained is 0.5436.

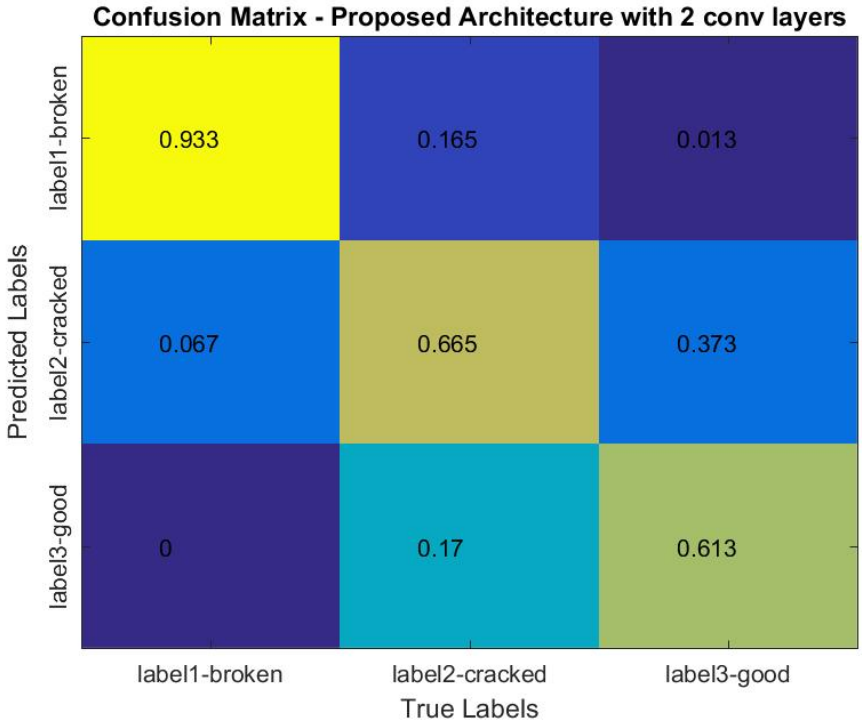


Figure 9.9: Confusion Matrix for Proposed CNN architecture with 2 convolutional layers

9.8 Comparison of Classification Approaches

The best performing models from each approach are compared. Figure 9.10 summarizes all the classification approaches based on their performance on the test set measured by mean accuracy per class and QWK score. The proposed architecture with 1 convolutional layer outperforms the handcrafted features and the one with 2 convolutional layers outperforms the HOG features. Though the difference in their mean per class accuracies is not much, the QWK score is higher by 4 points and goes from being moderate agreement to good agreement according to the guide for interpretation.

Figure 9.11 shows the per class accuracies for all approaches. It can be seen that the Naive Bayes classifier with handcrafted features works well for the Broken category but performs poorly for the other two categories. The SVM trained with HOG features performs well for the Broken and Good categories but has low performance on the Cracked category. The

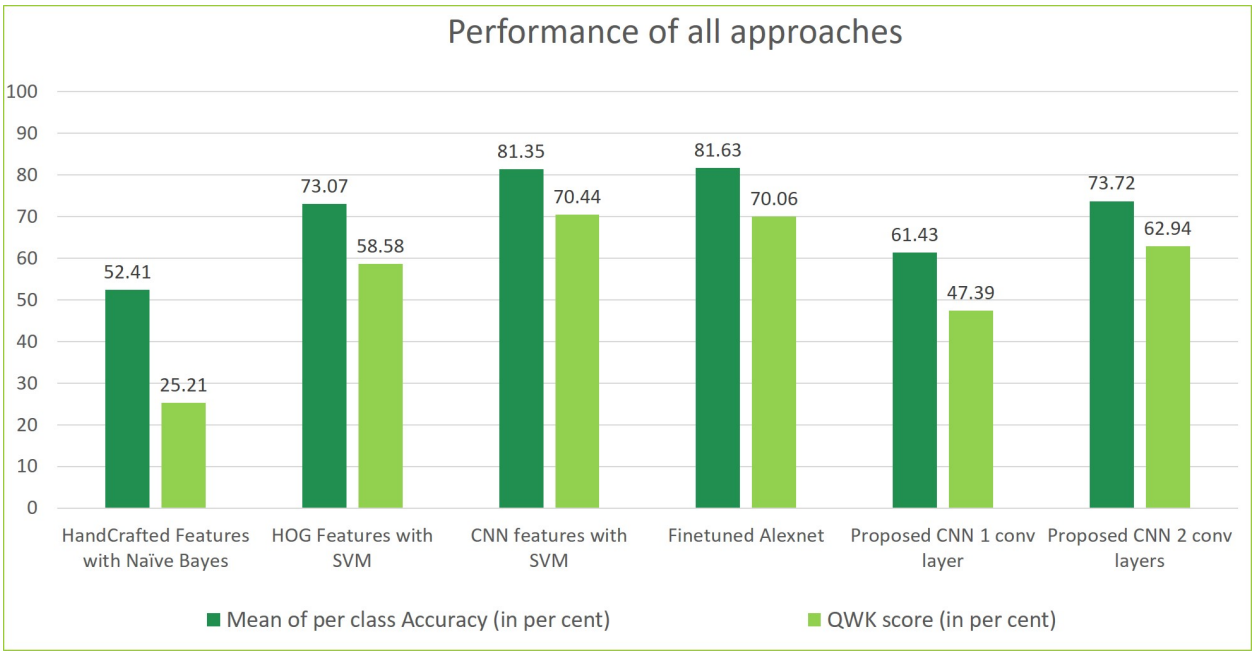


Figure 9.10: Performance of all approaches on test set

proposed CNN architecture with 1 convolutional layer performs moderately for the Broken and Cracked categories but poorly for the Good category. Proposed CNN architecture with 2 convolutional layers performs really well for the Broken category and moderately for the other 2 categories. Finally, the SVM with CNN features and the Finetuned CNN perform equally well on the Broken category giving the highest performance seen for this category. The finetuned AlexNet gives better performance on the Good category by 9% whereas the SVM with CNN features performs better on the Cracked category by 8.5%.

Figure 9.12 shows the trade-off between architecture in terms of number of parameters and performance in terms of mean per class accuracy. Finetuned AlexNet has about 56 million parameters whereas the proposed architecture with 2 convolutional layers has only 0.7 million parameters. The difference in their performance is very less as compared to this.

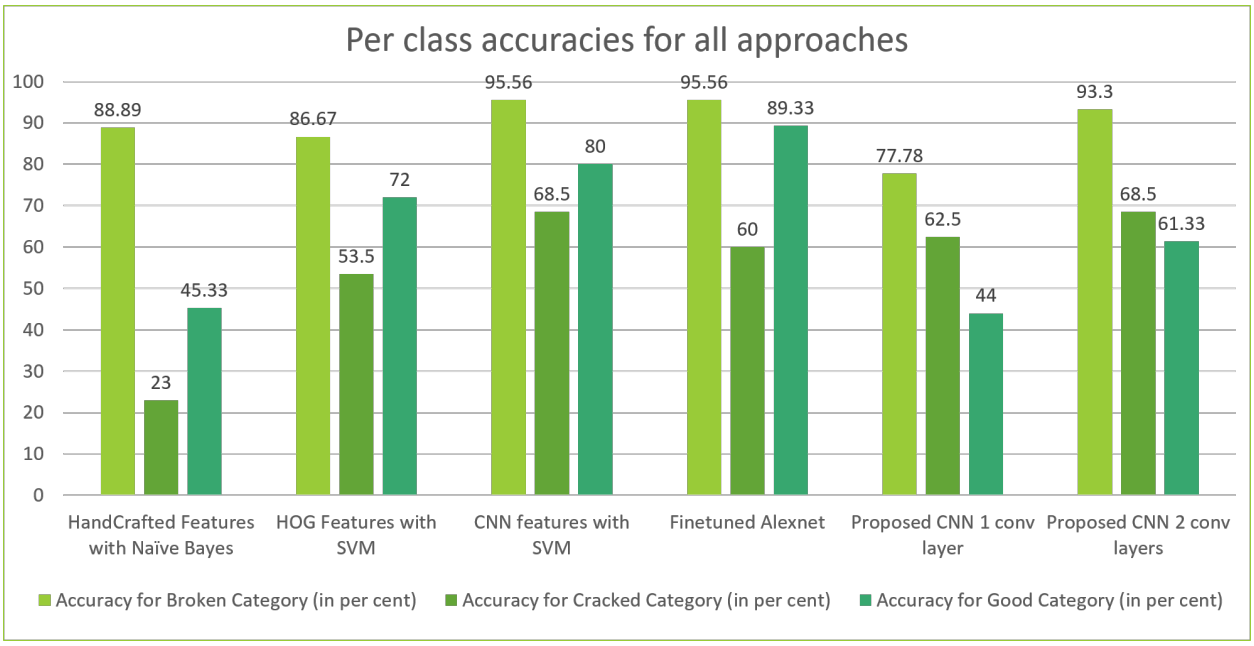


Figure 9.11: Per class accuracies for all approaches on test set

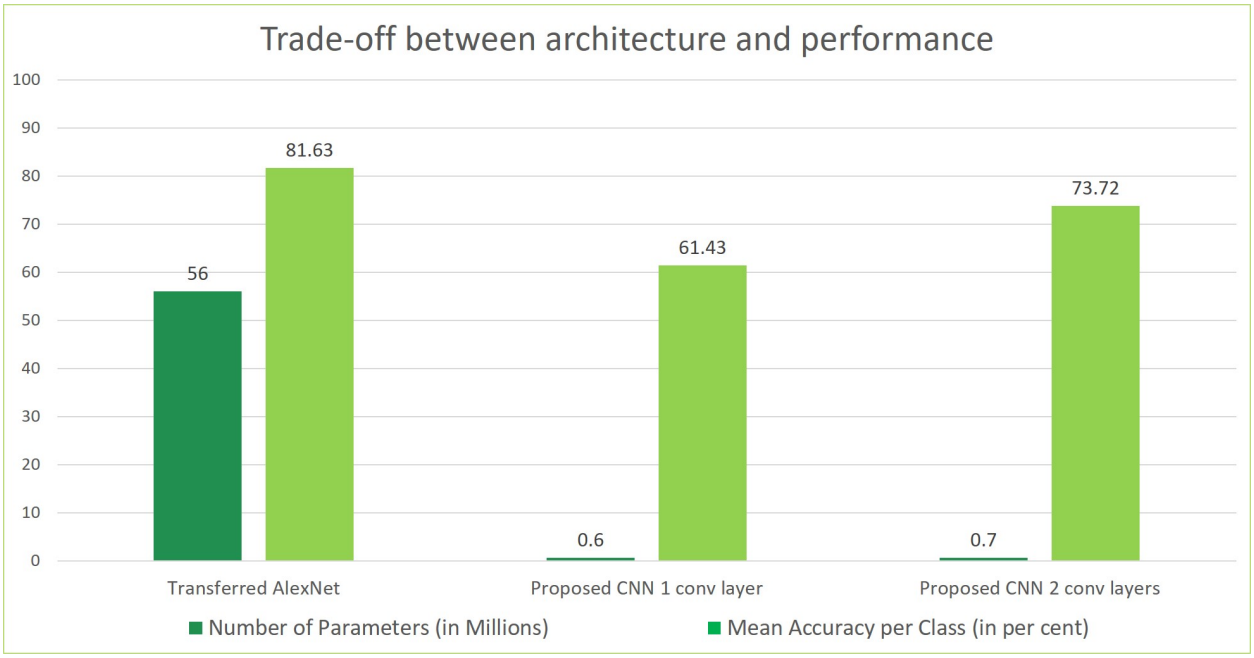


Figure 9.12: Trade-off between architecture and performance

9.9 Qualitative Results

Every classification technique implemented here makes some misclassifications. No particular pattern was observed for the misclassifications by different techniques. All techniques are observed to make the similar kind of mistakes, dealing with bad lighting and ballast occlusion in varying degrees. As a representation, the misclassifications done by the SVM trained on CNN features are shown in Figure 9.13.

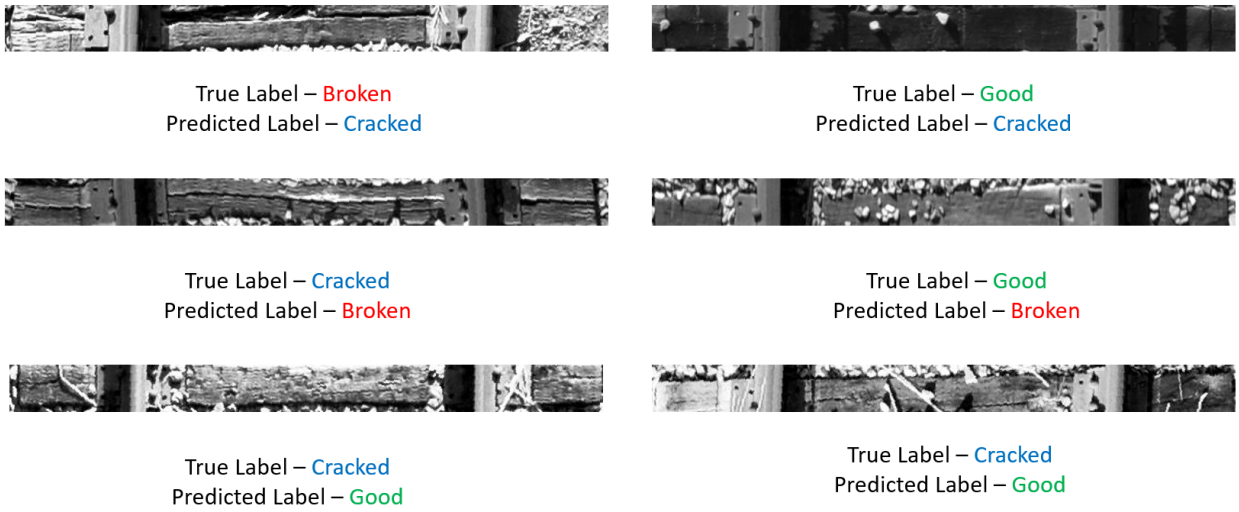


Figure 9.13: Misclassifications by SVM trained on CNN features

Chapter 10

Conclusions

This work presents a comparison between 4 approaches to classify railway ties into 3 categories based on their condition. It deals with automated grading of railway ties which has been identified as a potential future work by previous work in this domain. The performance metrics used in this work are particularly suited for the ordered classification task. The technique of transfer learning has been explored in two ways - using a pretrained CNN as a feature extractor and finetuning of a pretrained CNN. The CNN features and finetuned CNN are shown to outperform traditional handcrafted features and HOG features. The SVM trained on CNN features gives a mean per class accuracy of 81.35% and QWK score of 0.7044 while the finetuned CNN gives mean per class accuracy of 81.63% and QWK score of 0.7006. Both these models are the best performing approaches for the tie-grading task on the given dataset. The modified CNN architecture, proposed by taking into account the unique nature of the railway tie images, shows a significantly large reduction in parameters - from 56 million parameters for AlexNet to 0.7 million parameters for proposed architecture with comparatively less deterioration in performance. The performance of the proposed CNN architecture is better than the handcrafted and HOG features. It is thus concluded that the use of domain knowledge in the design of CNN architecture is beneficial. The use of data augmentation technique to enlarge and enrich the small dataset has benefited all

the approaches. The problem of label noise is also analyzed and the agreement between two labelers is found to have a QWK score of 0.5436.

For further improvement, the problem of label noise needs to be addressed using ensemble or importance reweighting techniques. Knowledge distillation technique can be used to improve the performance of the smaller CNN by learning from a larger CNN. CNNs pretrained on materials data may be used as an initialization for the finetuning. The problem of occlusion due to ballast can be dealt with in a more sophisticated way using techniques of seam carving or inpainting. The CNN architecture can also be modified to recognize the ballast and avoid it. Details of these recommendations are discussed in the next chapter.

Chapter 11

Recommendations

11.1 Dealing with label noise

As addressed in section 8.7, label noise is a concern in this supervised classification task. Ensemble techniques can be used to combine labels from several experts. Majority voting or weighted majority voting can be explored, with higher weights for more experienced or skilled experts. Other techniques such as structured labeling[43] and importance reweighting[44], [45] to deal with label noise can also been explored.

11.2 Dealing with Ballast

Performance of the ballast detection algorithm can be judged and improved by using ground truth annotation for ballast. Assuming that the ballast detection works well, techniques of inpainting[46] and seam carving[47] can be explored to remove the ballast occlusion. The proposed CNN architecture can be improved to deal with ballast by including a second channel in the input image which indicates the position of detected ballast.

11.3 Working with other infrastructural datasets

Techniques from this work can be applied to detect condition of roads or bridges. Patches are inherent to the railway tracks in the form of ties. Such patches can be extracted from the road or bridge images and can be classified into categories based on their condition. A detailed report of the infrastructural condition can thus be generated in a cost and time efficient manner to facilitate quicker restoration or replacement.

11.4 Sensor fusion

Data from other sensors such as laser, pressure sensors, acoustic sensors and so on can be fused with the data from the image sensor to get a better, more confident classification.

Chapter 12

Bibliography

[1] Rta tie report. http://www.rta.org/assets/docs/TieReports/tiereport3.pdf.

[2] Opencv documentation - svm. http://docs.opencv.org/2.4/doc/tutorials/ml/introduction_to_svm/introduction_to_svm.html.

[3] R. Polikar. Ensemble learning. *Scholarpedia*, 4(1):2776, 2009. revision #91224.

[4] Cnn as feature extractor. https://jeremykarnowski.wordpress.com/.

[5] Interpreting kappa. https://www.medcalc.org/manual/kappa.php.

[6] Railway ties. https://en.wikipedia.org/wiki/Railroad_tie.

[7] Railway tie association faq. http://www.rta.org/faqs-main.

[8] Esther Resendiz, Luis Fernando Molina, John M Hart, J Riley Edwards, Steven Sawadisavi, Narendra Ahuja, and CPL Barkan. Development of a machine-vision system for inspection of railway track components.

[9] E. Resendiz, J. M. Hart, and N. Ahuja. Automated visual inspection of railroad tracks. *IEEE Transactions on Intelligent Transportation Systems*, 14(2):751–760, June 2013.

[10] J Riley Edwards. *Advancements in railroad track inspection using machine-vision technology*. PhD thesis, University of Illinois at Urbana-Champaign, 2009.

[11] Technical report - fra. `https://www.fra.dot.gov/eLib/details/L16634`.

[12] Mohammad Sajjad Pasha. Machine vision for automating visual inspection of wooden railway sleepers, 2007.

[13] Mubarak Shah. Automated visual inspection/detection of railroad track. 2010.

[14] Ying Li, Hoang Trinh, Norman Haas, Charles Otto, and Sharath Pankanti. Rail component detection, optimization, and assessment for automatic rail track inspection. *IEEE Transactions on Intelligent Transportation Systems*, 15(2):760–770, 2014.

[15] Hoang Trinh, Norman Haas, Ying Li, Charles Otto, and Sharath Pankanti. Enhanced rail component detection and consolidation for rail track inspection. In *Applications of Computer Vision (WACV), 2012 IEEE Workshop on*, pages 289–295. IEEE, 2012.

[16] Ying Li, Charles Otto, Norm Haas, Yuichi Fujiki, and Sharath Pankanti. Component-based track inspection using machine-vision technology. In *Proceedings of the 1st ACM International Conference on Multimedia Retrieval*, page 60. ACM, 2011.

[17] Xavier Gibert, Vishal M Patel, and Rama Chellappa. Deep multitask learning for railway track inspection. *IEEE Transactions on Intelligent Transportation Systems*, 18(1):153–164, 2017.

[18] Support vector machine. `https://en.wikipedia.org/wiki/Support_vector_machine`.

[19] Support vector machine-scikit-learn documentation. `http://scikit-learn.org/stable/modules/svm.html`.

[20] knn. `https://en.wikipedia.org/wiki/K-nearest_neighbors_algorithm`.

[21] Decision tree-scikit-learn documentation. `http://scikit-learn.org/stable/modules/tree.html`.

[22] J. Ross Quinlan. Induction of decision trees. *Machine learning*, 1(1):81–106, 1986.

[23] Naive bayes. `https://en.wikipedia.org/wiki/Naive_Bayes_classifier`.

[24] Naive bayes-scikit-learn documentation. `http://scikit-learn.org/stable/modules/naive_bayes.html`.

[25] Lda. `https://en.wikipedia.org/wiki/Linear_discriminant_analysis`.

[26] Lda. `http://machinelearningmastery.com/linear-discriminant-analysis-for-machine-learn`

[27] Logistic regression. `https://en.wikipedia.org/wiki/Logistic_regression`.

[28] Multinomial logistic regression. `https://en.wikipedia.org/wiki/Multinomial_logistic_regression`.

[29] Thomas G Dietterich. Ensemble methods in machine learning. In *International workshop on multiple classifier systems*, pages 1–15. Springer, 2000.

[30] Bertrand Clarke. Comparing bayes model averaging and stacking when model approximation error cannot be ignored. *Journal of Machine Learning Research*, 4(Oct):683–712, 2003.

[31] Cs231n convolutional neural networks for visual recognition. `http://cs231n.github.io/convolutional-networks/`.

[32] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems*, page 2012.

[33] Navneet Dalal and Bill Triggs. Histograms of oriented gradients for human detection. In *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, volume 1, pages 886–893. IEEE, 2005.

[34] Extract histogram of oriented gradients (hog) features. `https://www.mathworks.com/help/vision/ref/extracthogfeatures.html`.

[35] Jason Yosinski, Jeff Clune, Yoshua Bengio, and Hod Lipson. How transferable are features in deep neural networks? In *Advances in neural information processing systems*, pages 3320–3328, 2014.

[36] Ali Sharif Razavian, Hossein Azizpour, Josephine Sullivan, and Stefan Carlsson. Cnn features off-the-shelf: an astounding baseline for recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pages 806–813, 2014.

[37] César Ferri, José Hernández-Orallo, and R Modroiu. An experimental comparison of performance measures for classification. *Pattern Recognition Letters*, 30(1):27–38, 2009.

[38] Ron Kohavi and Foster Provost. Glossary of terms. *Machine Learning*, 30(2-3):271–274.

[39] Kappa as a measure of concordance in categorical sorting. `http://vassarstats.net/kappa.html`.

[40] Jacob Cohen. A coefficient of agreement for nominal scales. *Educational and Psychological Measurement*, 20(1):37–46, 1960.

[41] Ian H Witten, Eibe Frank, Mark A Hall, and Christopher J Pal. *Data Mining: Practical machine learning tools and techniques*. Morgan Kaufmann, 2016.

[42] Jacob Cohen. Weighted kappa: Nominal scale agreement provision for scaled disagreement or partial credit. *Psychological bulletin*, 70(4):213, 1968.

[43] Todd Kulesza, Saleema Amershi, Rich Caruana, Danyel Fisher, and Denis Charles. Structured labeling for facilitating concept evolution in machine learning. In *Proceedings of the Conference on Human Factors in Computing Systems (CHI 2014)*. ACM, May 2014.

[44] Tongliang Liu and Dacheng Tao. Classification with noisy labels by importance reweighting. *IEEE Transactions on pattern analysis and machine intelligence*, 38(3):447–461, 2016.

[45] Nagarajan Natarajan, Inderjit S Dhillon, Pradeep K Ravikumar, and Ambuj Tewari. Learning with noisy labels. In *Advances in neural information processing systems*, pages 1196–1204, 2013.

[46] A. Criminisi, K. Toyama, and P. P?rez. Object removal by exemplar-based inpainting. *2013 IEEE Conference on Computer Vision and Pattern Recognition*, 02:721, 2003.

[47] Shai Avidan and Ariel Shamir. Seam carving for content-aware image resizing. In *ACM Transactions on graphics (TOG)*, volume 26, page 10. ACM, 2007.

Appendices

Appendix A

Confusion Matrices

This section contains the confusion matrices for the different classifiers experimented for handcrafted features and CNN features.

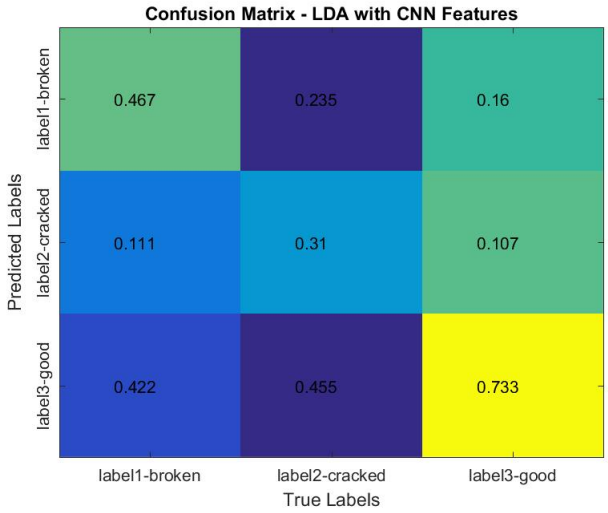


Figure A.1: Confusion Matrix for LDA with Handcrafted Features

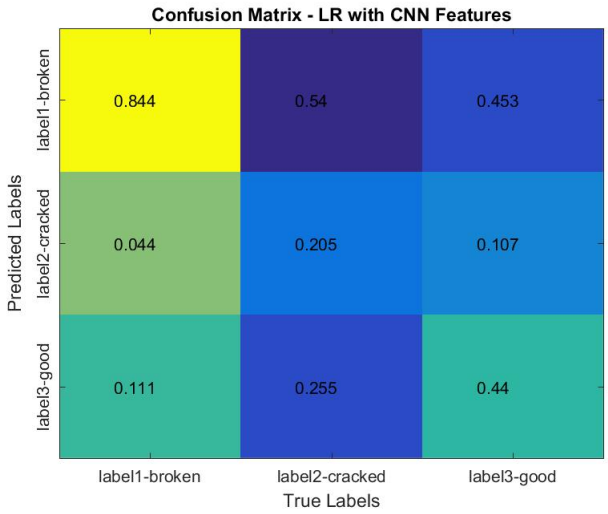


Figure A.2: Confusion Matrix for LR with Handcrafted Features

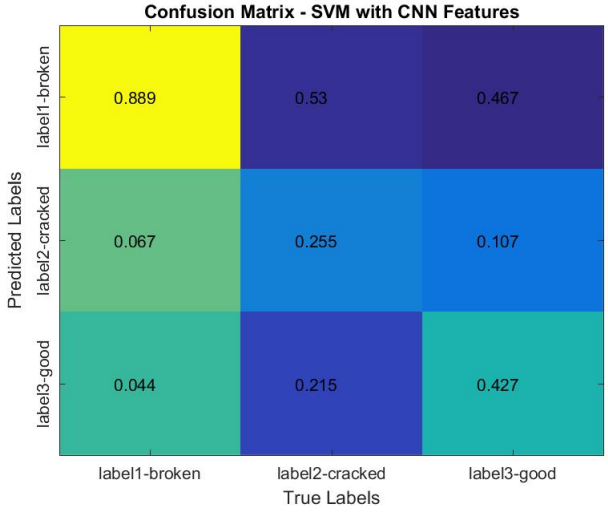


Figure A.3: Confusion Matrix for SVM with Handcrafted Features

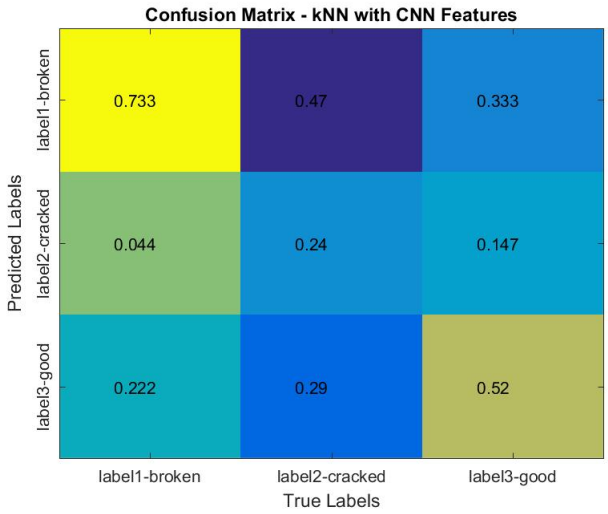


Figure A.4: Confusion Matrix for kNN with Handcrafted Features

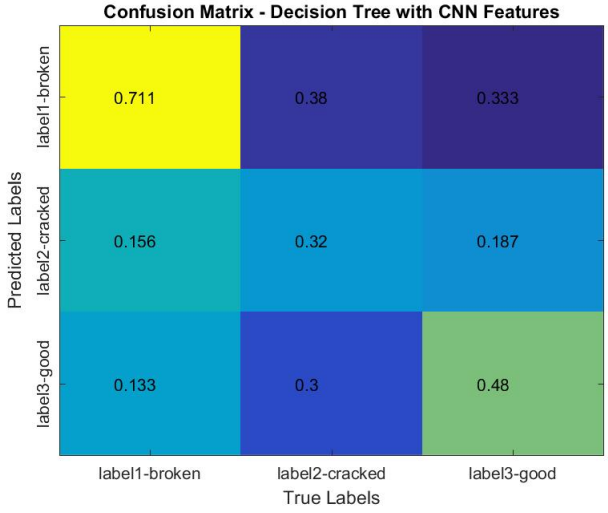


Figure A.5: Confusion Matrix for Decision Tree with Handcrafted Features

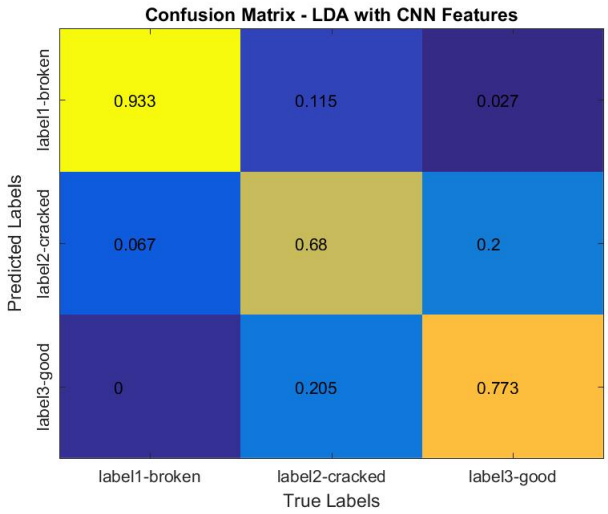


Figure A.6: Confusion Matrix for LDA with CNN Features

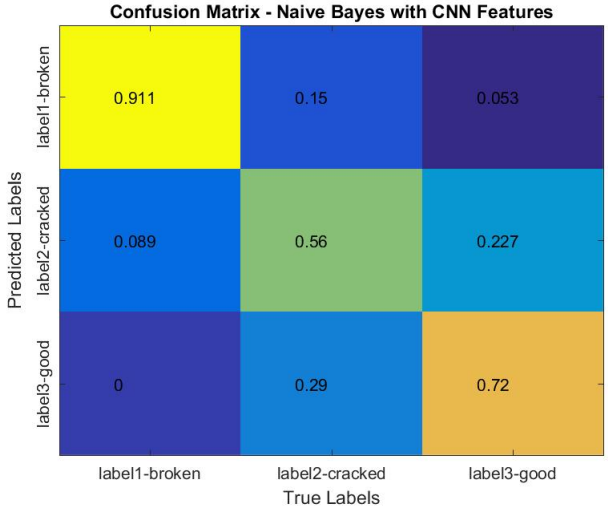


Figure A.7: Confusion Matrix for Naive Bayes with CNN Features

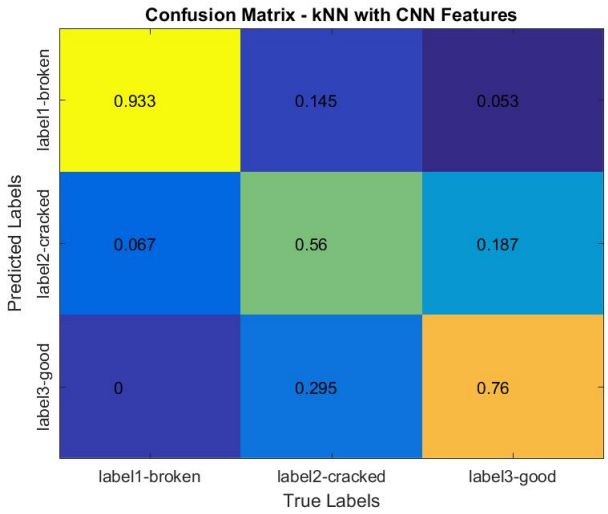


Figure A.8: Confusion Matrix for kNN with CNN Features

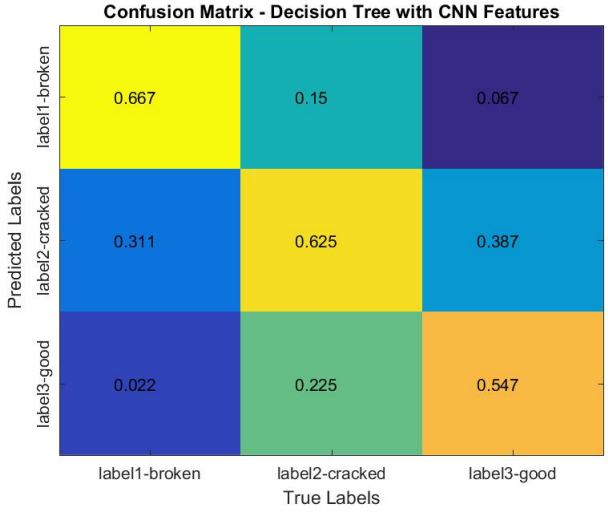


Figure A.9: Confusion Matrix for Decision Tree with CNN Features

Appendix B

Train-Val-Test Split

Table B.1: Train-Val-Test Split

Category	Number of Training Images	Number of Validation Images	Number of Test Images
Broken	500	46	45
Cracked	500	200	200
Good	500	75	75