Deep Convolutional Neural Networks for Detection of Rail Surface Defects

Shahrzad Faghih-Roohi*, Siamak Hajizadeh†, Alfredo Núñez†, Robert Babuska* and Bart De Schutter*

*Delft Center for Systems and Control, Delft University of Technology
Mekelweg 2, 2628 CD Delft, The Netherlands
Email: S.Faghihroohi@tudelft.nl; R.Babuska@tudelft.nl; B.DeSchutter@tudelft.nl
†Section of Railway Engineering, Delft University of Technology
Stevinweg 1, 2628 CN Delft, The Netherlands
Email: S.Hajizadeh@tudelft.nl; A.A.NunezVicencio@tudelft.nl

Abstract—In this paper, we propose a deep convolutional neural network solution to the analysis of image data for the detection of rail surface defects. The images are obtained from many hours of automated video recordings. This huge amount of data makes it impossible to manually inspect the images and detect rail surface defects. Therefore, automated detection of rail defects can help to save time and costs, and to ensure rail transportation safety. However, one major challenge is that the extraction of suitable features for detection of rail surface defects is a non-trivial and difficult task. Therefore, we propose to use convolutional neural networks as a viable technique for feature learning. Deep convolutional neural networks have recently been applied to a number of similar domains with success. We compare the results of different network architectures characterized by different sizes and activation functions. In this way, we explore the efficiency of the proposed deep convolutional neural network for detection and classification. The experimental results are promising and demonstrate the capability of the proposed approach.

I. INTRODUCTION

Feature learning by using deep neural networks has recently been applied to a variety of computer vision and classification problems, and has proved successful in many domains. The classification accuracy over several benchmark vision data sets, commonly with a large number of samples per class, has been improved over the shallow classical approaches with hand-crafted features [1]–[3]. Shallow learning approaches are based on general assumptions that ignore the characteristics of the given real data. In comparison to these methods, deep learning techniques help to move away from hand-crafted features design towards automated learning of problem-specific features, directly from the data. Convolutional neural networks are based on this strategy. Normally large feature learning networks such as convolutional neural nets have hundreds or thousands of parameters, which requires large data sets for training. In many real-world applications however, not all the collected big data are good sample data sets of the target classes. Therefore, the question whether features can be efficiently learned for classification of such data sets is important for such applications.

Automated rail defect detection using video cameras is an example of a problem where the number of target defects in the available data set is much smaller than the number of healthy samples. Rail surface defects occur due to different reasons, for example as a result of fatigue, due to the repetitive passings of rolling stock over rail components such as welds, joints, and switches, or because of the impacts from damaged wheels. If the rail defects grow and are treated late, they may lead to high maintenance costs. Therefore, automatic and facilitated detection of defects is important [4]–[6].

Recently, the use of video cameras for the inspection of rail tracks has become popular [7], [8], due to the error-prone, costly, and time-consuming process of manual rail monitoring. Detection models based on both learned (as in [9]) and predefined features (as in [10]) have been applied to different aspects of rail defect detection. However, the use of video cameras for the detection of rail surface defects caused by rolling contact fatigue (RCF) has been mostly studied using hand-crafted or predefined features [8], [11]. Other methods based on spatial correlation statistics, gradient-based, and hand-crafted features have been used in [12]–[14]. Compared to these feature learning methods, convolutional neural networks use relatively little pre-processing.

In this paper, we present an application of deep convolutional neural networks (DCNNs) for automatic detection of rail surface defects. Our data resembles that of [8] for visual inspection of rails. One immediate advantage of using a DCNN is that unlike [8], we do not have to go into an elaborate procedure for the extraction of features. We can rather use raw images as input to the classification model, which is subsequently optimized using a mini-batch gradient descent method for the entire network. We compare three DCNNs with different structures (i.e. different in size and number of parameters) for their classification accuracy and computation time.

This paper is organized as follows. In Section II, we review some related work on rail defect detection. In Section III, we describe the structure and operation of the convolutional neural network that is used to detect defects. In Section IV, we describe our data sets and present the proposed deep convolutional neural network. Section V presents the experimental results together with a comparison of different training strategies. Section VI concludes the paper with a brief discussion.

II. Related Work

In the last decade, different object recognition methods have been used in the field of rail defect detection. In [4], [15]–[17], some signal processing techniques such as noise reduction and wavelet transforms have been used for estimating irregularities and detecting defects in railway tracks. Unlike signal processing, the use of image processing techniques and image data analysis is a very recent approach for detection of rail defects. This is due to the new developments in the technology of video cameras and machine vision for rail monitoring. In [18] and [19], image processing techniques have been used for steel defect detection. In [18], a convolutional neural network is trained on a database of photometric stereo images. By means of differently colored light-sources illuminating the rail surfaces, the defects are made visible in a photometric dark-field setup. Moreover, in [19], a max-pooling convolutional neural network is applied for steel defect classification. In both papers, it is indicated that the data sets for training and testing are quite small, and the training is prone to over-fitting.

Classically in detection from visual data, gradient-based features such as the histogram of oriented gradients (HoG), scale-invariant feature transforms (SIFT), spacial pyramids, and basis functions representations such as Gabor filters are among the common choices of features (see e.g. [20]). In recent years, the expansion of deep feature learning schemes such as deep convolutional neural nets has provided such applications with a better tool for extracting features that are specifically tailored for each domain. Deep convolutional neural nets have been developed rapidly in the field of object recognition since the breakthrough work of [1]. In [21], [22], deep convolutional neural networks have been used for rail fastening condition monitoring. While [21], [22] have focused on identification of track components such as ballast, concrete, wood, and fastener, in this paper we focus on the detection and classification of different defects that occur at the rail surface. Moreover, we are interested in evaluating different structures of deep convolutional neural networks that can provide good accuracy rates on classification of rail defects compared to classical learning methods.

III. Classification Method

A. Deep Convolutional Neural Network

A deep convolutional neural network (DCNN), based on the classical convolutional neural network proposed by LeCun et al. [23] consists of three main components:

1) Convolution: A convolution layer is connected to the next layer in a similar manner as the traditional bipartite multi-layer neural network, with the key difference that the weights are shared between sets of connections. Each set of weight sharing connections forms a filter that is convoluted with the input data. There are usually several such filters trained in parallel at each layer. Convolution filters slide over small local receptive fields of input image data in image classification applications. Every

filter acts as a feature detector. The result of applying a convolution across an image forms a feature map.

2) Activation function: This function facilitates the discrimination between image classes by being imposed on the convolution filter output and performing a non-linear transformation of a data space. Some examples of activation functions are the hyperbolic tangent function (Tanh), the sigmoid function, and rectified linear units (ReLU) [24].

3) Max-pooling: The feature maps resulting from a convolution layer are sub-sampled in a pooling layer. In a max-pooling layer, the dimensions of the feature maps are reduced by merging local information (selecting maximum values) within a neighborhood window [19].

Convolutional layers and max-pooling layers are laid successively to create the DCNN architecture. Compared to shallow architectures, a DCNN has multiple layers that can represent complex functions with higher efficiency and generalization accuracy.

B. Training Methods

A batch (standard) gradient descent method involves optimizing the error over the entire training set. Since this procedure can be computationally extremely expensive for a large network, often for DCNN training an approximation method called mini-batch stochastic gradient descent method is used [2]. Here the difference is that instead of calculating the gradient of the error over the entire training set, each iteration calculates the gradient of the error for a small part of the samples called the mini-batch. We denote with b and n the size of mini-batch and the total number of training samples, respectively. For the mini-batch gradient descent, there are in total $T = n/b$ iterations per training epoch. The weight parameters $\mathbf{w}$ are therefore obtained through optimization of the approximated expected value of an error function f defined as:

$$E_t[f(\mathbf{w})] = \frac{1}{b} \sum_{i=(t-1)b+1}^{tb} f(\mathbf{w}; x_i) \qquad (1)$$

where $t \in \{1, ...T\}$ is the iteration index and x_i is the ith training sample. At each iteration the weights are adjusted using the gradient descent update rule:

$$\mathbf{w}^{(t+1)} = \mathbf{w}^{(t)} - \mu \nabla_{\mathbf{w}} E_t[f(\mathbf{w}^{(t)})] \qquad (2)$$

with μ being the learning rate.

While batch gradient descent runs through all the samples in the training set to obtain a single update for $\mathbf{w}$ in each iteration, stochastic gradient descent uses only a single training sample, and mini-batch gradient descent method uses b (i.e. the mini-batch size) samples at each iteration. Stochastic gradient descent and mini-batch gradient descent are computationally much cheaper than batch gradient descent. Mini-batch gradient descent can often be as fast as stochastic gradient descent if appropriate vectorization is applied in computing the derivative terms of (2). For problems with non-convex objective functions, stochastic gradient descent has sometimes shown

the ability to escape from local optima where batch gradient descent is trapped [2]. Therefore, stochastic gradient descent may perform better in applications such as DCNN training.

IV. IMPLEMENTATION

A. Data description

Our data sets are images of rail tracks that are collected from a camera with a high frame rate. This camera is mounted on a measurement vehicle and captures the top view of the rail tracks. The video data covers approximately 350 kilometers of track, equivalent to 700 kilometers of rail. Among the collected frames, we manually labelled 22408 objects as belonging to 1 out of 6 classes (normal, weld, light squat, moderate squat, severe squat, and joint). The weld class corresponds to those parts of the track surface where the rails are welded together to form one continuous rail, and in most of the cases in the images, these are hardly distinguishable from the normal rail surface even by the human eye when the weld is in good health condition. Insulated joints electrically separate two consecutive track sections with an insulating material that is easily seen in the images. Squats are a type of surface-initiated track defects [4]. A sample of images from the different types of squats is shown in Figure 1. There are different classifications of the squat types based on their severity and size, but often there is no rigid distinction between the types, since squats have a gradual growth process. Our data set contains 985 welds, 938 light squats and smaller trivial defects, 562 moderate and severe squats, and 755 rail joints. These images are obtained from the original images of rail tracks, after segmentation of the track from the ballast and other background textures surrounding the rails.

B. DCNN Architecture

In this paper, three DCNN structures (small, medium, and large) are considered. Table I summarizes the information of each DCNN structure. The parameters of the DCNNs are determined by implementing each network with various combinations of parameters such as the number of feature maps, the sizes of the filters, the number of layers, and the number of nodes of fully-connected layers. Then, the parameters that lead to the highest classification accuracy have been selected for building the DCNN models. Model parameters such as the learning rate and the class weights are also adjusted during the initial test runs. Out of each class, we reserve 10 percent of the samples for testing and use the remaining 90 percent for training.
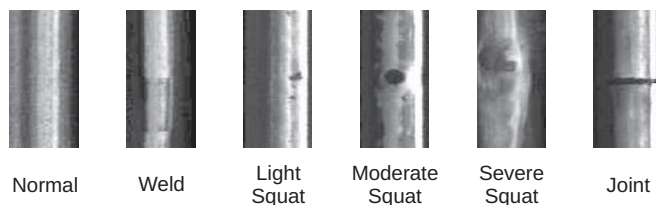

Figure 1. A sample of images of defects and non-defects

TABLE I
THE STRUCTURES OF THE DEEP CONVOLUTIONAL NEURAL NETWORKS
CONSIDERED IN THIS PAPER

Deep convolutional neural network				
Type of layer		Small	Medium	Large
Convolutional layer 1	Feature maps	6	10	20
	Filter size	17×9	9×5	9×5
Max-pooling layer 1	Filter size	3×3	2×2	2×2
Convolutional layer 2	Feature maps	10	20	40
	Filter size	8×3	9×6	9×6
Max-pooling layer 2	Filter size	3×3	2×2	2×2
Convolutional layer 3	Feature maps	-	30	60
	Filter size	-	4×2	4×2
Max-pooling layer 3	Filter size	-	2×2	2×2
Fully-connected layer 1	Number of nodes	70	120	320
Fully-connected layer 2	Number of nodes	18	30	80
Fully-connected layer 3	Number of nodes	-	-	8
Estimated number of parameters (weights)		22000	120600	644200

Figure 2 illustrates the response of the filters trained at the 3rd convolutional layer of the large DCNN. For visualization purposes, the filters are scaled. A number of filters have been trained to capture the wave-like patterns of the defects. This filtering acts very similar to predefined feature extraction functions such as the Gabor family of functions or the Cosine transform.

For more details, the structure of the medium DCNN is illustrated in Figure 3. This DCNN model consists of three convolutional layers, three max-pooling layers, and three fully-connected layers. Input images are of size 100×50 pixels with 2 color channels (gray scale). The first convolution layer takes a normalized image and filters it with kernels of size
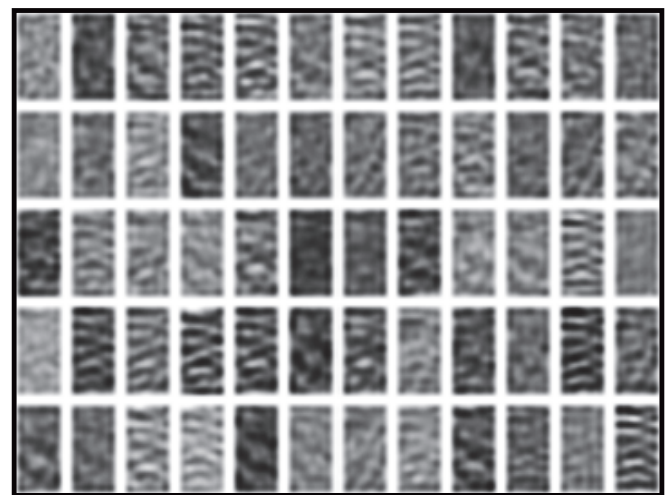

Figure 2. Visualization of the response of the last convolutional layer to random stimuli, in the large DCNN
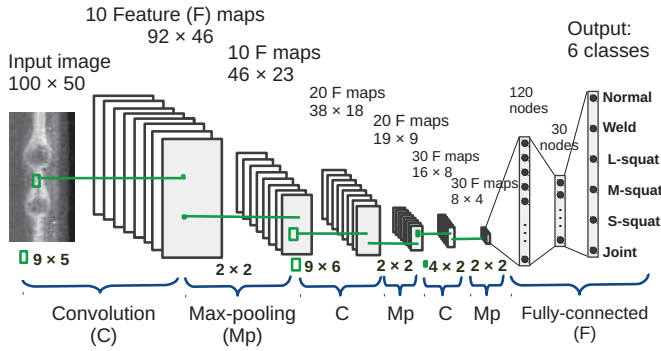
Figure 3. Architecture of the proposed medium DCNN

9×5 pixels. The second convolution layer takes the pooled feature map of the first layer and filters it with kernels of size 9×6 pixels. The kernel size of the third convolution layer is 4×2 pixels. In this model, max-pooling units of size 2×2 pixels are used. We use the hyperbolic tangent function (Tanh) and rectified linear units (ReLU), respectively, as activation function. After three convolutional and max-pooling layers, the high-level reasoning in the convolutional neural network is performed via fully-connected layers. In this network, we use two fully-connected layers which have 120 nodes and 30 nodes, respectively. The output of the network classifies the input image using the 6 classes described in Section IV.A.

V. EXPERIMENT RESULTS

A. Experimental setup

The implementation is based on the framework in Torch 7 [25]. To decrease the unwanted variation due to different lighting and rail texture conditions, we apply a simple normalization to all samples. The learning rate (μ) is initially set to 10^{-3} with a decay factor of 10^{-5} to help avoid overfitting to the training data. From the results acquired in our parameter adjustment runs, we set the mini-batch size (b) to 8 for all three DCNNs. Then, we train each network over 40 epochs. For each test we shuffle all the existing samples and divide each class into 10 sets. For 10 rounds of cross-validation, we test each time on one out of 10 sets and use the rest for training. For the testing, we randomly under-sample the normal class to 250 test samples per round. This is done due to the huge imbalance of the class sizes, which can severely bias the test results if all normal test samples are evaluated. The final results are averaged over the 10 rounds. In order to convert the multi-class classification results to the binary classification of normal samples versus anomalies, we simply regard all the non-normal classes as one and compute the numbers of true positives (TP), true negatives (TN), false positives (FP), and false negatives (FN). The binary classification accuracy: $(TP+TN)/(TP+TN+FP+FN)$ and F1-score: $2TP/(2TP+FP+FN)$ are then defined based on the reduced classification matrices.

B. Classification results

We report two types of results from the experiments. Initially the networks are trained to classify the data into 6 classes. We compare the confusion matrices of the classification results trained on the three structures (small, medium, and large DCNN). Then, we retrain the networks using both Tanh and ReLU activation functions to classify samples into 3 classes. The first class represents the normal rail that has both weld and normal samples. The second class contains all types of small defects and squats, and finally the third class only consists of rail joints.

The confusion matrices of the classification results trained on the three DCNN are presented in Tables II-IV. The rows of the matrices correspond to the correct classes and the columns correspond to the predicted classes. In Table III for instance, the percentage of correctly classified severe squats is 48.13, while 33.12 percent of the actual severe squats are classified in the medium squat class. Similarly, in Table IV, the percentage of correctly classified welds is 61.95 percent, while 30.97 percent of the actual welds are classified in the normal class. This false classification value is relatively high compared to the total number of welds. The comparison shows that in the assessment of the binary and multi-class classification accuracy, the two classes of normal and weld should be integrated into one class (Normal). Also, all three classes of light, medium, and severe squat should be combined together as a defect class (Defect). As a result, there are

Table II
CONFUSION MATRIX OF THE SMALL DCNN (%)

	Normal	Weld	L-squat	M-squat	S-squat	Joint
Normal	94.62	3.06	1.82	0.19	0.08	0.23
Weld	31.85	59.62	3.20	1.55	0.29	3.49
L-squat	21.09	3.27	66.93	7.42	0.10	1.19
M-squat	5.95	3.81	27.62	53.57	7.62	1.43
S-squat	6.25	5.62	2.50	37.50	44.38	3.75
Joint	3.13	5.75	1.12	1.75	2.12	86.13

Table III
CONFUSION MATRIX OF THE MEDIUM DCNN (%)

	Normal	Weld	L-squat	M-squat	S-squat	Joint
Normal	95.74	1.94	1.78	0.11	0.04	0.39
Weld	29.61	63.01	3.11	0.88	0.19	3.20
L-squat	22.47	3.17	65.05	7.72	0.30	1.29
M-squat	8.81	2.86	22.86	56.19	6.90	2.38
S-squat	8.12	4.37	3.13	33.12	48.13	3.13
Joint	2.25	5.50	1.62	1.00	1.00	88.63

Table IV
CONFUSION MATRIX OF THE LARGE DCNN (%)

	Normal	Weld	L-squat	M-squat	S-squat	Joint
Normal	96.32	1.82	1.32	0.15	0.08	0.31
Weld	30.97	61.95	2.52	0.68	0.29	3.59
L-squat	22.08	3.17	64.36	9.40	0.10	0.89
M-squat	6.90	3.33	24.76	56.67	7.15	1.19
S-squat	8.12	2.50	3.13	34.37	50.00	1.88
Joint	2.50	4.37	1.00	1.75	2.00	88.38

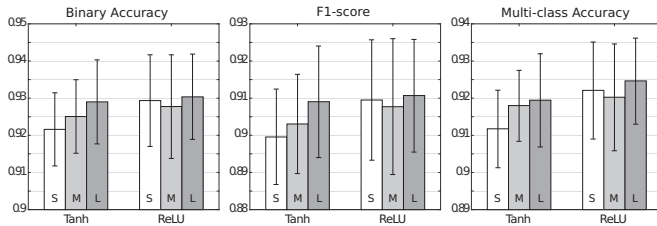| Activation Function	DCNN	Binary accuracy	Binary F1-score	Multi-class accuracy	Relative run time
Tanh	Small	0.9216	0.8995	0.9117	1.005
	Medium	0.9250	0.9030	0.9180	1.370
	Large	0.9290	0.9091	0.9195	2.022
ReLU	Small	0.9293	0.9095	0.9221	1.000
	Medium	0.9277	0.9077	0.9203	1.375
	Large	0.9304	0.9107	0.9247	2.027

Figure 4. Comparison of the three network structures (S: Small; M: Medium; L: Large) for performance and accuracy

have compared the classification performance and training times of three DCNN structures and identified the most accurate results. With the proposed small, medium, and large DCNNs, the rail defect classes can be successfully classified with almost 92% accuracy. From the performance results of the DCNN models, we conclude that the large DCNN model performs better for the classification task than the small and medium DCNN model, although the network training takes a longer time.

In the current context, detection and classification of all types of rail defects is often carried out while spending much time and cost. With this in mind, exploring a deep learning approach that would be general enough to be used for automatic detection of other types of rail defects is our immediate future work of interest. In particular, we will explore the use of auto-encoders and other deep networks for this purpose.

ACKNOWLEDGMENT

three classes (Normal, Defect, and Joint) considered for the performance evaluation of the DCNN models.

The performance results of the three DCNN models (small, medium, and large) for the multi-class (3 classes) and binary (normal and anomaly) classifications are summarized in Table V. The relative computation time of training the networks for each network structure and activation function is also reported in Table V.

To summarize the results in Table V, Figure 4 compares the obtained accuracy results of the three DCNNs from 20 runs, based on the means of the computed performance metrics and the corresponding standard deviations. In general, using the ReLU activation functions for training of the network produces better results. Also, the computation times of ReLU networks are almost the same as those of the Tanh networks, and the differences are not significant.

It can be observed from Table V and Figure 3 that the large DCNN mostly performs better than the rest. However, this improvement comes at the price of doubling the computation time as opposed to the small DCNN, e.g. in the case of using ReLU activation functions. Interestingly the medium-sized network is outperformed by both the small and large networks, but with a higher computation time compared to the small network.

VI. CONCLUSIONS

In this paper, we have presented a deep learning approach for automatic detection of rail surface defects. Due to the huge amount of image data, we employed a DCNN to efficiently extract and recognize image features, and to automatically detect rail defects. We took advantage of the DCNN to skip elaborate procedures of feature extractions required in classical learning approaches; instead we use raw images as the only input to the classification model, and subsequently optimize the network using a mini-batch gradient descent method. We

REFERENCES

[1] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in Neural Information Processing Systems*, pp. 1097–1105, 2012.

[2] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *CoRR*, vol. abs/1409.1556, 2014.

[3] Y. Hou, H. Zhang, and S. Zhou, "Convolutional neural network-based image representation for visual loop closure detection," *CoRR*, vol. abs/1504.05241, 2015.

[4] M. Molodova, Z. Li, A. Nunez, and R. Dollevoet, "Automatic detection of squats in railway infrastructure," *IEEE Transactions on Intelligent Transportation Systems*, vol. 15, no. 5, pp. 1980–1990, 2014.

[5] Z. Li, M. Molodova, A. Núñez, and R. Dollevoet, "Improvements in axle box acceleration measurements for the detection of light squats in railway infrastructure," *IEEE Transactions on Industrial Electronics*, vol. 62, no. 7, pp. 4385–4397, 2015.

[6] M. Oregui, M. Molodova, A. Núñez, R. Dollevoet, and Z. Li, "Experimental investigation into the condition of insulated rail joints by impact excitation," *Experimental Mechanics*, vol. 55, no. 9, pp. 1597–1612, 2015.

[7] Q. Li and S. Ren, "A visual detection system for rail surface defects," *IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews*, vol. 42, no. 6, pp. 1531–1542, 2012.

[8] Q. Li and S. Ren, "A real-time visual inspection system for discrete surface defects of rail heads," *IEEE Transactions on Instrumentation and Measurement*, vol. 61, no. 8, pp. 2189–2199, 2012.

[9] X. Gibert, V. M. Patel, and R. Chellappa, "Robust fastener detection for autonomous visual railway track inspection," in *2015 IEEE Winter Conference on Applications of Computer Vision (WACV)*, pp. 694–701, IEEE, 2015.

[10] Y. Li, H. Trinh, N. Haas, C. Otto, and S. Pankanti, "Rail component detection, optimization, and assessment for automatic rail track inspection," *IEEE Transactions on Intelligent Transportation Systems*, vol. 15, no. 2, pp. 760–770, 2014.

[11] P. De Ruvo, G. De Ruvo, A. Distante, M. Nitti, E. Stella, and F. Marino, "A visual inspection system for rail detection &; tracking in real time railway maintenance," *Open Cybernetics & Systemics Journal*, vol. 2, pp. 57–67, 2008.

[12] R. Huber-Mörk, M. Nölle, A. Oberhauser, and E. Fischmeister, "Statistical rail surface classification based on 2d and 21/2d image analysis," in *Advanced Concepts for Intelligent Vision Systems*, pp. 50–61, Springer, 2010.

[13] P. Liang, G. Teodoro, H. Ling, E. Blasch, G. Chen, and L. Bai, "Multiple kernel learning for vehicle detection in wide area motion imagery," in *2012 15th International Conference on Information Fusion (FUSION)*, pp. 1629–1636, IEEE, 2012.

[14] N. Dalal and B. Triggs, "Histograms of oriented gradients for human detection," in *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, vol. 1, pp. 886–893, IEEE, 2005.

[15] B. Liang, S. Iwnicki, A. Ball, and A. E. Young, "Adaptive noise cancelling and time–frequency techniques for rail surface defect detection," *Mechanical Systems and Signal Processing*, vol. 54, pp. 41–51, 2015.

[16] X. Zhang, N. Feng, Y. Wang, and Y. Shen, "An analysis of the simulated acoustic emission sources with different propagation distances, types and depths for rail defect detection," *Applied Acoustics*, vol. 86, pp. 80–88, 2014.

[17] J. S. Lee, S. Choi, S.-S. Kim, C. Park, and Y. G. Kim, "A mixed filtering approach for track condition monitoring using accelerometers on the axle box and bogie," *IEEE Transactions on Instrumentation and Measurement*, vol. 61, no. 3, pp. 749–758, 2012.

[18] D. Soukup and R. Huber-Mörk, "Convolutional neural networks for steel surface defect detection from photometric stereo images," in *Advances in Visual Computing*, pp. 668–677, Springer, 2014.

[19] J. Masci, U. Meier, D. Ciresan, J. Schmidhuber, and G. Fricout, "Steel defect classification with max-pooling convolutional neural networks," in *The 2012 International Joint Conference on Neural Networks (IJCNN)*, pp. 1–6, IEEE, 2012.

[20] P. Gehler and S. Nowozin, "On feature combination for multiclass object classification," in *IEEE 12th International Conference on Computer Vision*, pp. 221–228, IEEE, 2009.

[21] X. Gibert, V. M. Patel, and R. Chellappa, "Deep multi-task learning for railway track inspection," *CoRR*, vol. abs/1509.05267, 2015.

[22] X. Gibert, V. M. Patel, and R. Chellappa, "Material classification and semantic segmentation of railway track images with deep convolutional neural networks,," in *IEEE International Conference on Image Processing (ICIP)*, 2015.

[23] Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel, "Backpropagation applied to handwritten zip code recognition," *Neural Computation*, vol. 1, no. 4, pp. 541–551, 1989.

[24] Y. Kim and T. Moon, "Human detection and activity classification based on micro-doppler signatures using deep convolutional neural networks," *IEEE Geoscience and Remote Sensing Letters*, vol. 13, no. 1, pp. 8–12, 2016.

[25] R. Collobert, K. Kavukcuoglu, and C. Farabet, "Torch7: A matlab-like environment for machine learning," in *BigLearn, NIPS Workshop*, no. EPFL-CONF-192376, 2011.