applied sciences

MDPI

Article

Developing Machine Learning-Based Models for Railway Inspection

Chunsheng Yang [1,*], Yanmin Sun [2], Chris Ladubec [1] and Yan Liu [1]

[1] National Research Council Canada, Ottawa, ON K1A 0R6, Canada; Christopher.Ladubec@nrc-cnrc.gc.ca (C.L.); Yan.Liu@nrc-cnrc.gc.ca (Y.L.)
[2] Defence Research and Development Canada, Ottawa, ON K1A 0Z4, Canada; Yanmin.Sun@forces.gc.ca
* Correspondence: Chunsheng.Yang@nrc-cnrc.gc.ca; Tel.: +1-613-993-0262

Abstract: Smart railway maintenance is crucial to the safety and efficiency of railway operations. Successful deployment of technologies such as condition-based monitoring and predictive maintenance will enable railway companies to conduct proactive maintenance before defects and failures take place to improve operation safety and efficiency. In this paper, we first propose to develop a classification-based method to detect rail defects such as localized surface collapse, rail end batter, or rail components—such as joints, turning points, crossings, etc.—by using acceleration data. In order to improve the performance of the classification-based models and enhance their applicability in practice, we further propose a deep learning-based approach for the detection of rail joints or defects by deploying convolutional neural networks (CNN). CNN-based models can work directly with raw data to reduce the heavy preprocessing of feature engineering and directly detect joints located on either the left or the right rail. Two convolutional networks, ResNet and fully convolutional networks (FCN), are investigated and evaluated with the collected acceleration data. The experimental results show both deep neural networks obtain good performance, which demonstrate that the deep learning-based methods are effective for detecting rail joints or defects with the expected performance.

Keywords: rail defects; rail joint; switch; acceleration data; wavelet; machine learning algorithms; deep neural network; CNN

Citation: Yang, C.; Sun, Y.; Ladubec, C.; Liu, Y. Developing Machine Learning-Based Models for Railway Inspection. *Appl. Sci.* **2021**, *11*, 13. https://dx.doi.org/10.3390/app11010013

Received: 17 November 2020
Accepted: 17 December 2020
Published: 22 December 2020

1. Introduction

The railway industry spent almost 40% of revenue on maintaining, renewing and expanding infrastructure [1,2] over past decades. Advanced railway maintenance is crucial to improve safety and reduce operational costs. Recently, condition monitoring of railway infrastructure has become more and more important, leading railway companies to take advantage of artificial intelligence (AI) based technologies. Fleet reliability is a key lever for increasing efficiency and reducing total cost for smart railway operation. Predictive maintenance represents a great opportunity to yield the next big efficiency leap in maintenance—reducing the number of failures, the amount of unplanned maintenance and, eventually, the required level of reserve asset capacity for rail operators [3].

Wheel failures and broken rails are two main factors that cause train derailments in today's railway operations. Wheel failures, which account for half of all train derailments, cost billions of dollars to the North America rail industry [1,4]; and another main cause of derailments is broken rails due to rail defects. To minimize rail breaks and help avoid catastrophic events such as derailments [5], railways are now closely monitoring the performance of wheels and trying to remove them before they start imparting high impact forces to the rails. Many techniques for detecting wheel flats and out of round issues have been developed [6] and installed at strategic locations on the rail network. These detectors measure the vertical force or impact of each passing wheel. One of them is called Wheel Impact Load Detectors (WILD) [7]. A central system receives the data in

real-time and advises the staff when the impact of a given wheel is too high. In the North American railway network, a set of threshold values has been developed and implemented to flag the bad actor wheels since 2004. Building on WILD techniques, we have developed the WILD Predictor [8–10] to predict wheel failures before they reach these threshold values. The WILD predictor can predict wheel flats, out of round wheels, and estimate the time for wheels to reach the impact force thresholds using the machine learning-based predictive models.

There are many available techniques developed by the railway industry and community to detect rail surface defects [6,11–14]. One cost-effective method of detecting rail surface defects such as shelling, squats, split heads, engine burns, etc. [15] is to use accelerometers mounted to the bogie side frames or wheel axles. The collected acceleration data from the sensors were processed with a pre-determined threshold value to judge the rail surface defects. The technique is simple, useful, and applicable for detecting rail joints or broken rails. However, much more work remains to be done in order to distinguish the joint (or broken rail) from various track surface defects since these surface defects generate high amplitude vertical accelerations in the axles and side frames as well. Due to such limitations, the existing threshold-based algorithms usually require additional visual inspection of these areas of track, which greatly increase the cost and complexity of the techniques. Therefore, more intelligent algorithms are needed so that the accelerometers can automatically distinguish the rail defects from rail joint (or broken rail) and other special track features with discontinuities. As the first step to distinguish rail surface defects from other track features, the present work is focused on the detection of rail joints on both sides of the track. To this end, we first developed machine learning methods to build the models to detect the rail joints using the accelerating data collected from an inspection vehicle [16]. The developed models can detect the rail joints with high accuracy. However, this method has two weaknesses: (1) the model requires the feature data from raw data and (2) each side of the track needs a separate model for detection. This creates difficulties to deploy or apply the models in railway operations. To address these limitations, we propose to develop deep learning-based models by applying convolutional neural networks (CNN).

The state-of-the-art developments with deep learning neural networks, especially the CNN, demonstrate an end-to-end time series classification approach, which is able to deal with raw data without any data preprocessing. Meanwhile, it is possible to develop one single model for identifying rail joints on both sides of the railway. After introducing the classification-based method for rail joint detection, the paper investigates this approach for the detection of rail joints with acceleration data.

Following this section, the paper first provides an overview of rail joint detection technologies; then Section 3 introduces the developed feature-based classification method. Section 4 investigates the proposed deep learning-based methods. Section 5 presents the preliminary results for joint identification. Section 6 discusses the results and future work; and Section 7 concludes the paper.

2. Overview of Railway Defect Detection Techniques and Acceleration Data

2.1. Techniques of Rail Defect Detection

There have been many methods and techniques developed in order to reduce the number of derailments. One way is to improve the quality of rail steel. For example, harder steels will lead to longer life and reduce the probability of rail defects. However, with the increasing of traffic on the railroad, the failure of rail still remains and seems inevitable. Early detection of rail defects is still necessary in order to repair the rail track before it causes any serious problems such as derailments. Rail defects are mainly detected by using an instrumented train car [11], a railway-monitoring vehicle or an instrumented in-service train car [12]. Such instrumented vehicles contain various sensors or devices, including video cameras, accelerometers on the car body, acoustic sensors, and ultrasonic wave sensors, GPS, linear potentiometers, strain gauges on the top chord, and so on. A rail defect detection system consists of not only an instrumented vehicle, a sensor suite,

but also a data acquisition system, data analytics tool, and knowledge extraction algorithms. For rail inspection, the methods can be grouped into vision-based methods such as laser scanner [14,17–19] and video image processing [17–19], and motion-based methods such as acceleration [20], which measures the vertical motion of the vehicles to detect rail defects and other track components.

2.2. Acceleration Data

An accelerometer is a sensor capable of detecting the impact of rail surface defects to wheel or vehicle body when mounted on the wheel or the bogie of an in-service railway vehicle. When a train passes over track with rail surface irregularities such as joints and surface defects, the impact force that occurs at the wheel/rail interface will generate a peak of high acceleration to the wheelsets, axle box, bogie, and vehicle body. The abnormal motion can be monitored by the accelerometer.

In this study, the investigation uses the acceleration data collected by two accelerometers installed on a rail inspection vehicle (as shown in Figure 1) owned by Transport Canada. The two sensors measure acceleration signals in vertical directions at two locations—i.e., left and right—as shown in Figure 2. A segment of track with many rail joints was selected to collect the acceleration data.

Figure 1. Transport Canada rail inspection vehicle.

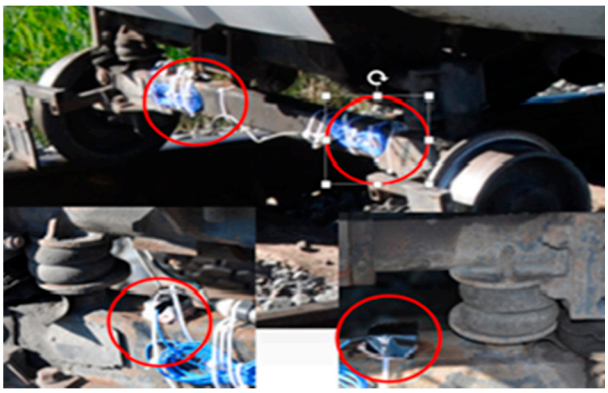


Figure 2. Test vehicle and accelerometers.

In order to collect enough data points during the high frequency impact due to the joint discontinuity, a sampling rate of 464 Hz was used in the data acquisition system. This particular sampling rate was set by the acceleration data system. Figure 3 shows a typical piece of the time series data collected through a test run. As multiple accelerometers are mounted on wheels of 'right' and 'left' sides, a joint (or a surface defect) on one side of the track can also be observed in the data of the other side and vice versa [21,22]. If the data for the right and left rails is analyzed separately for the defect detection on each side, extra verification tests have to be considered to decrease the false alarm rate. Therefore,

a robust algorithm should take the data of both sides into account and identify if there is a joint (or defect) at the location and on which side of the track the joint (or defect) is located.

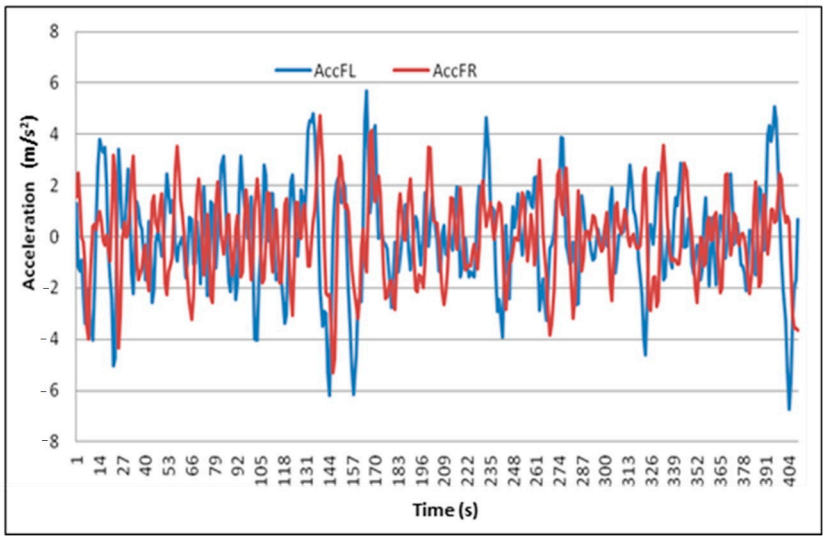


Figure 3. Original time histories of acceleration on two sides of a vehicle.

Other data collected during the test runs include vehicle speed by wheel axle encoder, as well as the latitude and longitude coordinates by a GPS sensor. The speed channel will be used later to convert the acceleration data into the distance domain, and the GPS data is useful to map the locations of the identified joints (or defects). The Figure 3 shows an example of the acceleration data collected from the rail inspection vehicle and Figure 4 shows a rail inspection technician inspecting a rail joint in the field. It is a time consuming task to manually inspect the rail joint for maintenance decision-making.

Figure 4. Manual joint inspection by a technician in the field.

3. Feature-Based Classification Method

In this section, we provide an overview of classification-based methods for rail defect detection. The main task is to build the classifiers (models) from the acceleration dataset

obtained from the rail inspection car. These models must accurately recognize the particular data signatures that indicate rail defects or other rail components. The classification-based method consists of five main processes: data labeling, feature extraction, model building, model evaluation, and the decision rules of rail component identification. The following subsections describe each process in detail.

3.1. Labeling

We cast the rail defect or joint detection problem as a binary classification task with two class values: joint/defect (1) and not joint/defect (0). Many supervised learning techniques can be used to address this problem but they require that each instance in a given acceleration dataset is pre-assigned to one of the class values. In this work, we use the signal analytic result (threshold) to automatically label each instance into one of the class values (1 or 0). These labels will become the ground truth in the model building and evaluation process.

3.2. Wavelet Feature Extraction

As in all challenging machine learning applications, the quality of the representation of the data input is a key factor for building high-performance models. In this work, the acceleration data of both sides of the track was sampled with a given frequency of 464 Hz as mentioned above. The data was transformed by signal processing in the data acquisition system mounted on the rail inspection car. After the exploration of acceleration data, we believe it is necessary to perform the wavelet analysis in order to augment the data by generating new features. Wavelet analysis is a signal processing technique that has emerged over the last 30 years as an alternative to traditional signal processing methods, like the Fourier transform. The wavelet transform [23,24] can extract local information in the time-domain and frequency-domain. Discrete wavelet transform (DWT) is a common method of wavelet analysis, which decomposes low/high-frequency components from the original signals. The following describes the foundation of the standard discrete wavelet transformation.

In DWT, a discrete signal can be expressed using a scaling function and a wavelet function [24]. Using the Haar scaling function, and the Daubechies wavelet function, the signal ($f[n]$) can be expressed in Equation (1).

$$f[n] = \frac{1}{\sqrt{M}} \sum_{k=0}^{\infty} W_{\varnothing}[j_0, k] \varnothing_{j0,k}[n] + \frac{1}{\sqrt{M}} \sum_{j=j0}^{\infty} \sum_{k}^{M} W_{\psi}[j, k] \psi_{j,k}[n], \quad (1)$$

where $f[n]$, $\phi_{j0,k}[n]$, and $\psi_{j,k}[n]$ are discrete functions defined in $[0, M-1]$, with a total of M points. By taking the inner product to obtain the wavelet coefficients: approximation coefficients $\left(W_{\phi}[j0,\ k]\right)$ and detailed coefficients ($W_{\psi}[j, k]$) are given in Equations (2) and (3), respectively.

$$W_{\phi}[j_0, k] = \frac{1}{\sqrt{M}} \sum_{n=1}^{M} f[n] \phi_{j_0,k}[n], \quad (2)$$

$$W_{\psi}[j, k] = \frac{1}{\sqrt{M}} f[n] \psi_{j,k}[n] j \geq j_0, \quad (3)$$

In this study, we generate the wavelet features based on Equations (2) and (3). The computed wavelet features will be added on to the original data set as the new features for the model-building step.

3.3. Model Building

Our goal is to build a classification model to classify acceleration data into rail defects or other rail components such as joints, crossings, and turnouts. Many classification algorithms are available from machine learning research, including instance-based learning (IBL), naïve Bayes, support vector machine (SVM), decision trees, and neural networks. In this study, simple algorithms such as decision trees, random forest decision trees, and

naïve Bayes are preferable over more complex algorithms because they are quick and produce models that are easily explained. We systematically apply the same algorithm several times with varying attribute subsets (with wavelet features, without wavelet features, etc.) to obtain a set of heterogeneous models for the model selection step to evaluate and choose the final model for identifying rail defects or joints.

3.4. Decision Rules for Identificaion

With the outputs from the selected model, we can apply the following rules (Equation (4)) to identify the final detection results.

$$RID = \begin{cases} R_{com} \; if \; N \geq N_c \; and \; \{x_i \subseteq w\} \\ R_{Def} \; \; if \; N_d \; \leq N < N_c \; and \; \{x_i \subseteq w\} \end{cases}, \quad (4)$$

where, RID is the identification results for acceleration data given a specific window size (w) of rail track length; R_{com} and R_{Def} represent rail components such as joint and rail defect in the rail inspection. N is the positive detection numbers in (w) which is determined based on the sampling rate and speed of inspection car. N_d is a constant decided based on the signature of a rail defect, and N_c is a constant decided based on the signature of rail component such as a joint. For example, we decided N_c = 5 based on the statistical result of rail joints corresponding to the equipment used in this study.

3.5. Model Evaluation

In this work, the main task is to detect rail joints in which there may contain multiple positive detections. In terms of the rule of identification, whenever minimum number of positive acceleration data are greater than N_c in a given length of rail track, this piece of rail track will be detected as a joint. We define a detection rate as detected joint numbers divided by total joint numbers in the testing dataset. Eventually, we use accuracy and detection rate as the criteria for model evaluation and selection.

Following the described methods, we conducted extensive experiments to demonstrate the feasibility and usefulness of the feature-based classification methods. The details can be found in [16]. As mentioned above, the acceleration data were collected from a rail inspection car. The data only covered a specific 64 km segment of rail track. There is not any rail defect detection-related data. All data are related to rail joints. In the experiments, we separated the collected data into a training dataset and a testing dataset. The training dataset contains 11 joints on the left side of the track, and 12 joints on the right side of the track. The testing dataset contains 19 left joints and 22 right joints. In the experiment, the window size for the joint is set as 4 cm based on common railway practice. Table 1 shows the experimental results of the model performance for four decision tree classifiers named by feature combinations. In Table 1, MA is trained with a decision tree learning algorithm running on feature data set, OrigFeature-L (left rail only), MB is trained with a decision tree learning algorithm running on feature data set, OrigFeature-R (right rail only). Similarly, MC and MD are trained with decision trees on feature data set, Wavelet-R and Wavelet-L. Since we applied a well-defined rule to combine the prediction from each model (N_c = 5), all false alerts are successfully filtered. Therefore, no false alerts of joint detection are returned by any of the models.

Table 1. Performance of the developed models.

Model	Feature Set	Accuracy	Detection Rate	False Alerts
MA	OrigFeature-L	0.99860	90%	0
MB	OrigFeature-R	0.99834	89.5%	0
MC	Wavelet-R	0.99832	84%	0
MD	Wavelet-L	0.99834	100%	0

The experimental results indicate that the models can detect rail joints with good performance. However, there are two deficiencies: (1) the model requires feature engineering for new feature generation and (2) each side of the track needs a dedicated model for joint detection. This creates complexity in the deployment of the described models and requires other solutions to identify the location of defects, including the side of track, and the position of the defect on the railway. To address these issues we propose a deep learning-based approach to develop one model for defect detection on both rails using the raw data directly.

4. Deep Learning-Based Methods

As discussed above, the deep learning-based methods are developed to address the modeling issues existing in the feature-based classification methods. The ultimate goal is to develop one global model for detecting rail defects directly using raw data without complex feature engineering. Deep neural networks are a proven technology for this problem. We can cast detecting rail defects or components such as rail joints as a time series classification task from the viewpoint of machine learning. Therefore, we have to represent the time series of acceleration raw data in order to apply deep neural networks such as convolutional neural networks. In this section, we present an overview of time series classification using deep learning networks along with a proposed data representation method.

4.1. Time Series Classification

A time series is a sequence of data points (measurements) which has a natural temporal ordering. The goal of time series classification is to derive a label to an interval of data points over time based on its behavior. The acceleration data used in this study is a type of time series data as shown in Figure 3. Detecting joints from the acceleration data can be fulfilled through the time series classification approach [25].

Conventional time series classification approaches can be grouped under two main types: similarity-based and feature-based methods [26]. Similarity-based approaches are based on similarity measurements—e.g., Euclidean distance—such that the testing samples are assigned with class labels of the corresponding training samples where their similarity measurements are minimized. 1-nearest neighbor (1-NN) and dynamic time warping (DTW) are two widely used methods of this category. Feature-based methods transfer the time series instances to a feature space where a set of features are analyzed to represent the patterns of the time series instances. Classification systems run on these features to label the time series instances. Both of these two approaches conduct heavy computations to assess similarity measures for the similarity-based approach and feature extractions for the feature-based approach.

Recently some works have demonstrated deep neural networks, especially convolutional neural networks (CNN), for end-to-end time series classification [27,28]. Unlike the traditional feature-based classification framework, CNN does not require handcrafted features. Both features and classifier are learned jointly in one model. There are three basic components to define a basic CNN: (1) the convolutional layer, (2) the pooling layer, and (3) the output layer. Interested readers please refer to [29] for a more detailed introduction to CNN. A convolution layer applies a sliding a filter over the time series [17]. The filter is a generic non-linear transformation of a time series. The output of a convolution layer (one filter) on an input time series can be considered as another time series that underwent a filtering process to obtain discriminative features useful for classification tasks. When applying several filters on an input time series, multiple discriminative features are learned automatically. The pooling operation aggregates the time series over the entire time dimension resulting in a single real value. Usually a global aggregation is useful to reduce the number of parameters in a model thus decreasing the risk of overfitting. The final discriminative layer takes the features from the convolution and pooling layers and generates a probability distribution over the class labels for the input time series. In this

paper, the state-of-the-art technology of the CNN is investigated to build an end-to-end time series classification model to detect joints from acceleration data.

4.2. Deep Learning Networks

Convolutional neural networks have achieved groundbreaking performance in image recognition and there are numerous variants of CNN architectures reported in the literature. In [27], several deep learning networks were empirically studied for the task of univariate and multivariate time series classification. These studies suggest ResNet [30,31] and fully convolutional networks (FCN) [28] achieve better performances than the others. Therefore, in this paper ResNet and FCN are investigated. The same network structures of ResNet and FCN presented in [28] are adopted. They are reconfigured for bivariate time series classification. Figure 5 shows their structures.

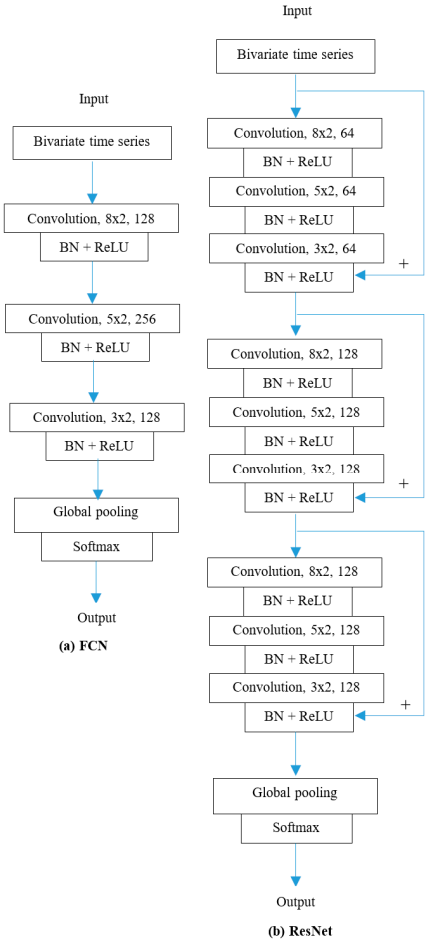


Figure 5. Network structures of FCN and ResNet.

FCN is built with three convolution blocks followed by a global average pooling layer and a soft-max layer. The basic convolution block is a convolutional layer followed by a batch normalization (BN) layer and a ReLU activation layer. The kernel sizes are $\{8 \times 2, 5 \times 2, 3 \times 2\}$ and filter sizes are {128, 256, 128} respectively to the three convolution blocks (Figure 5a). ResNet is one of the most popular architectures in various computer vision tasks. ResNet extends the neural networks to a very deep structure by adding a shortcut connection between consecutive convolutional layers. These additional shortcut connections link the output of residual blocks to their inputs thus enable the flow of the gradient directly through these connections. In this structure, there are three residual blocks with a shortcut connection, a global average pooling layer, and a soft-max layer. A basic unit of a residual block is composed of a convolutional layer followed by a batch

normalization (BN) layer and a ReLU activation layer. There are three basic units in each residual block with the kennel sizes of $\{8 \times 2, 5 \times 2, 3 \times 2\}$. The filter sizes of these three residual blocks are set as {64, 128, 128} respectively (Figure 5b).

The input to the networks is a bivariate time series of acceleration data of 'right' and 'left' sides; the output of the network is probability distributions over three class labels of {"None-joint", "Left-joint", "Right-joint"} for three circumstances: no joint detected, a joint on "left" side, and a joint on "right" side. The class label is determined with the largest probability. These models are implemented in Python taking advantage of the deep learning package "Keras".

4.3. Representation Method of Raw Data

To develop a deep network for smart railway inspection directly using raw acceleration data, we have to provide an efficient data representation. To this end, we propose a 'distance-domain data representation' method for data generation. The main reasons are: (1) the acceleration data is collected with a constant sampling time interval, but the speed of the inspection car varies, causing the data points to be unequally spaced in the time domain; and (2) the goal of this research is to develop a global model to detect defects either on the left rail or right rail. Therefore, we have to generate a data object that contains equally spaced data points. The developed 'distance-domain data representation' method consists of four steps: data segmentation, data interpolation, data normalization, and data resampling.

4.3.1. Data Segmentation

Each data point, indexed by the distance measure from the start point, is a vector of two acceleration values of left and right sides of the rail. The whole dataset is ordered by the distance measure in increasing order. Three class labels ("None-joint", "Left-joint", "Right-joint") are considered. To train a deep learning neural network model, the data has to be formulated as $D = \{(x_1, y_1), (x_2, y_2) \dots (x_n, y_n)\}$. Each pair (x_i, y_i) $(1 \leq i \leq n)$ is one instance, with n instances in the data set. Each x_i is an interval of time series with fixed length, i.e., each x_i has the same number of data points and each point has the same number of elements; y_i is a categorical value representing the class label. To manipulate the acceleration dataset into the required format, the bivariate time series data is segmented by sliding windows with each segmentation as a sample. The sliding window uses a fixed distance interval. With the raw data (in the time domain), the number of data points in a fixed distance interval varies at different locations due to the varied speeds of the inspection vehicle. To resolve this issue, a data interpolation method is adopted. The interpolation method is discussed in the next subsection. Associated with x_i, y_i is the class label taking a value from {"None-joint", "Left-joint", "Right-joint"}.

4.3.2. Data Interpolation

The acceleration data was collected with a sampling time interval of 2.15 ms. The data was converted into the distance domain by integrating the speed signal collected simultaneously with the two acceleration channels. Thus, the data points are indexed by distance from the starting point. The running speed of the inspection vehicle is changing and the data was collected with a fixed frequency, so the distance interval is not unique. To train a model for time series classification, input samples require fixed-length signals. If the raw data is directly divided with the same number of data points in each sample, the sequence features of samples are not aligned.

If the distance interval between two adjacent data points is large, some data points may be missed between them. Data interpolation is a method of constructing new data points within the range of a discrete set of known data points. The method creates a function by curve-fitting the data points; then the function is used to generate points anywhere along the curve. In this work, the system was implemented in Python. Therefore the

"interp1d" class in the Python package "SciPy-interpolate" is a convenient method for this task. Suppose there is a sample x_i by sliding window, which is expressed in Equation (5).

$$x_i = \begin{bmatrix} x_i^l \\ x_i^r \end{bmatrix} = \begin{bmatrix} x_i^{l1} \, x_i^{l2} \ldots x_i^{lm} \\ x_i^{r1} \, x_i^{r2} \ldots x_i^{rm} \end{bmatrix}, \tag{5}$$

where, x_i is a bivariate time series of acceleration data of left and right track respectively. There are m data points in this sample, which is different from sample to sample. There is another index vector $d_i = \{d_i^1, \, d_i^2, \, \ldots \, d_i^m\}$ with each d_i^j indicating the location of the data point $\begin{pmatrix} x_i^{lj} \\ x_i^{rj} \end{pmatrix}$. The distance between two adjacent data points might be different but the distance interval of the sample $d = d_i^m - d_i^1$, fixed by the sliding window size, is the same for every sample. The first step is to create the curve-fitting functions by the "interp1d" class from Python package "SciPy-interpolate" [32] for left and right separately as expressed in Equation (6) and Equation (7).

$$f_l = interp1d(d_i, \, x_i^l, \, kind = cubic), \tag{6}$$

$$f_r = interp1d(d_i, \, x_i^r, \, kind = cubic), \tag{7}$$

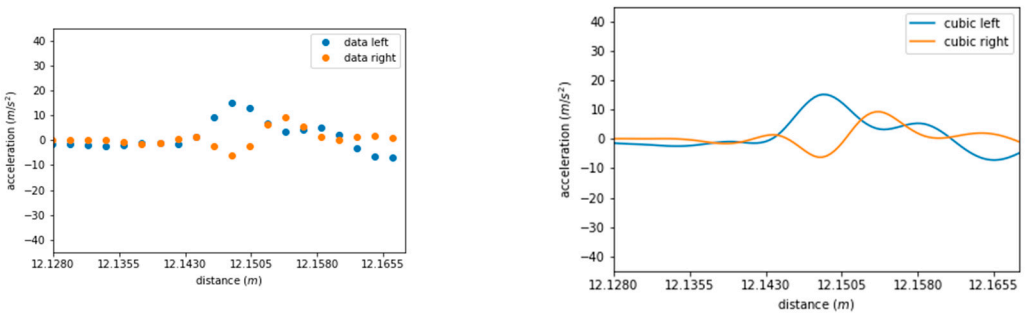
The second step is to generate a fixed number of data points evenly distributed in this distance interval. Suppose M is the fixed number, Equation (8) expresses new data vector.

$$d_i^{new} = \left\{ d_i^{1,new}, d_i^{2,new}, \, \ldots \, d_i^{M,new} \right\}, \tag{8}$$

where, d_i^{new} is a new distance vector contains M points equally spaced between the start point of d_i^1 and the end point d_i^m. The new data points can be generated as $x_i^{l,\,new} = f_l \, (d_i^{new})$ and $x_i^{r,\,new} = f_r(d_i^{new})$. Through data interpolation, each sample can be expressed as Equation (9).

$$x_i^{new} = \begin{bmatrix} x_i^{l,new} \\ x_i^{r,new} \end{bmatrix} = \begin{bmatrix} x_i^{l1,new} \, x_i^{l2,new} \ldots x_i^{lM,new} \\ x_i^{r1,new} \, x_i^{r2,new} \ldots x_i^{rM,new} \end{bmatrix}, \tag{9}$$

An example to show the data representation method described above is given in Figure 6. Figure 6 shows a data sample that contains 19 data readings (points) given the window size is 4 cm from our application. Figure 6a is the original data points for two sides of rail; and Figure 6b is the curve after fitting with the "SciPy-interpolate" algorithm. With the fitted spline, we resample the curve with 100 data points, which are equally spaced (0.4 mm).



(**a**) (**b**)

Figure 6. An example of data sample given a fixed window size (4 cm). (**a**) Original data; (**b**) cubic interpolations.

4.3.3. Data Normalization

To make the raw data have a consistent scale, the data need to be normalized before training. Standardization is a widely used normalization method where the data have zero-mean and unit-variance. The method calculates the mean and standard deviation from a data column. Then the mean was subtracted from each data point of the data column and the result is divided by the standard deviation.

The dataset was split into a training set and a testing set. The training set was for learning the model and the testing set was used for validating the model. Generally, the training set and testing set are assumed to be drawn from the same population and have approximately the same mean and standard deviation values. The testing set represents data from the real-world in the future. The testing set was normalized by the mean and standard deviation calculated from the training set. Otherwise, if the mean and standard deviation of the whole dataset were used, future information would be introduced in model learning.

4.3.4. Data Resampling

Most locations along the track contain no joints and only a few locations have joints on either the left or right side. When the data was manipulated, most of the samples will be associated with the class label of "None-joint" and only a few are with the class label of "Left-joint" or "Right-joint". Consequently, the data set is unbalanced where the "None-joint" class has a large amount of samples and the other two joint classes have very few samples. Trained with the unbalanced data set, a classification model's performance on the small classes is usually very poor. The reason is the classification model was trained to maximize the classification accuracy. Since the small classes have very little influence on the classification accuracy, the model is prone to optimize for accuracy in the prevalent class. To prevent the unbalanced data problem, the "None-joint" class is down-sampled when training the model. The dataset is composed of a randomly selected subset of the "None-joint" class mixed with the whole samples from the other two classes. What is an appropriate ratio of down sample is uncertain. However, if the down sampling size is not enough, the dataset is still unbalanced; if the down sampling size is too much, the model will be prone to the other two classes resulting in a large number of false alarms. The down sampling size can be decided through some empirical methods such as trial-and-error.

5. Experimental Results

There is no label information within the original datasets. To conduct these experiments, another column was manually added to indicate whether or not there is a joint the interval; and if there is a joint, which side track it is on (left or right). Three classes are coded as 0—"None-joint", 1—"Left-joint", 2—"Right-joint". There are two datasets collected from two different locations. For the first dataset, approximately 110,000 data points were labeled with 25 joints on left and 29 joints on right. This data set was separated into two parts with the front part as the training set and the latter part as the testing dataset. 15 left joints and 17 right joints are in the training set; and 10 left joints and 9 right joints are in the testing dataset. For the second dataset, the data partition method is the same as for the first data set with 134 joints on the left and 142 joints on the right in the training set; and 50 joints on the left and 47 joints on the right in the testing set.

Table 2 shows the basic statistical results of the training dataset and testing dataset samples for each rail joint. From the statistical results the data points for each joint varies with a minimum value of 11 points and a maximum value of 19 in the training dataset and 15 in the testing dataset. As mentioned in the data representation algorithm, each data object for deep nets will be resampled with identical value of 100 data points after applying "SciPy-interpolate" algorithm.

Table 2. Statistical results of the data samples (raw data).

Data Points	Training Dataset	Testing Dataset
Min	11	11
Max	19	15
Mean	13.34	13.32

The training data was manipulated into the required format. Since a joint affected zone is around 2 cm wide on the rail, the sliding window interval is set up as 4 cm expanding around 1 cm to each end. After the data manipulation the training dataset is a matrix of three dimensions $100 \times 2 \times n$, where 100 is the number data points in each sample after applying interpolation algorithm; 2 indicates the bivariate time series; and n is the number of samples. Most samples are labeled with 0 ("None-joint" class). After several empirical tests, this class was randomly downsized to 200 for the first data set and 1000 for the second dataset. Therefore n = 200 + 15 + 17 = 232 for the first dataset and n = 1000 + 50 + 47 = 1097 for the second dataset. This data matrix is normalized before training the model. The class label vector $y = [y_1, y_2, \ldots y_n]$ with each y_i taking a value from $[0, 1, 2]$ as the class label of sample x_i.

The test dataset was segmented with overlapping sliding windows. For the training data, the joint locations were assumed to be known such that each joint can be segmented into one sample inclusively. However, the testing data was assumed unseen from the real world. By using the same segmentation method of sliding windows, it is possible that a sequence of data points representing a joint may be divided into several adjacent intervals and generate duplicate data objects for an identical joint. Consequently, a joint can be difficult to identify correctly. To prevent this occurrence, overlapping sliding windows are used, i.e., a sliding window with an overlap of front half with the previous window and an overlap of latter half with the next window. In this way, a joint can be fully covered by a window. The same interval of 4 cm is used thus the overlapped windows with an overlap of 2 cm with the previous window and an overlap of 2 cm with the next window. Each window was taken as one test sample. The output is a vector of class labels corresponding to each sample. However, with the overlapping sliding windows, a joint can be segmented in multiple successive windows (samples). These samples might be labeled with a joint label 1 or 2 but not consistent. This indicates that there a joint in this area but not sure on which side. To deal with this inconsistency, the samples with labels of joint classes (1 or 2) were taken. If several samples are from several successive overlapped windows (can be decided by their start and end locations), these samples will be merged as one sample. These selected samples will be re-classified to get the final class labels.

The training set was composed by randomly downsizing the "None-joint" class. This factor may vary the models' performance. Ten models are trained by 10 different subsets of the "None-joint" class plus the two joint class samples, and tested with the same testing dataset. Thus the average results from the ten models can be reported. Both ResNet and FCN are trained and tested with the same datasets in all runs in order to compare their performance fairly. Finally, the model performance evaluated by the recall, precision and F-measure values on the two joint classes is presented in Table 3. From the results, it is obvious that both ResNet and FCN achieve good results on the two testing datasets. ResNet performs slightly better than FCN for the first dataset and they both achieve similar results on the second dataset.

Table 3. Performance of deep learning-based models.

		ResNet		
		Recall	**Precision**	**F-Measure**
Data_1	Left (10 joints)	0.83 ± 0.07	0.83 ± 0.08	0.83 ± 0.06
	Right (9 joints)	0.94 ± 0.06	0.89 ± 0.06	0.91 ± 0.08
Data_2	Left (134 joints)	0.98 ± 0.04	0.77 ± 0.03	0.85 ± 0.04
	Right (142 joints)	0.89 ± 0.03	0.87 ± 0.02	0.88 ± 0.03
		FCN		
		Recall	**Precision**	**F-Measure**
Data_1	Left (10 joints)	0.82 ± 0.12	0.71 ± 0.13	0.74 ± 0.15
	Right (9 joints)	0.96 ± 0.08	0.81 ± 0.13	0.86 ± 0.13
Data_2	Left (134 joints)	0.98 ± 0.03	0.76 ± 0.06	0.87 ± 0.05
	Right (142 joints)	0.95 ± 0.04	0.83 ± 0.05	0.89 ± 0.05

6. Discussion

Based on the preliminary results obtained from the experiments, it is obvious that the feature-based method can detect the joint with high detection rate but requires two independent models for right and left rail respectively. The deep learning-based method can address this limitation with one global model to detect a joint on either the left rail or right rail with the expected performance. Another advantage is deep neural networks can directly deploy the raw time history data as the model input without complex data engineering which makes the model more useful in practice and applicable in real applications. The experimental results shown in Table 3 demonstrate that ResNet and FCN both achieved good performance in joint detection, with ResNet performing slightly better than FCN on the first dataset and similarly on the second dataset. It is worth mentioning that the deep learning-based model will unable to detect the joints that appear simultaneously on the both side since the models were trained only for each side of the rails.

It is worth noting that there are a few parameters that require special attention in the proposed methods: the window size, the number of data points in the data object, and the length of the joint. These parameter settings or configurations will have a great impact on the model performance and applicability of the developed models. In this work, these parameters are set up based on the requirement of rail joint detection. In the future, if we want to use the model to detect other rail components—such as turning points, crossings, etc.—these parameters have to be reconfigured based on the new requirements.

Although this preliminary study was focused on detecting rail joints, we believe that the developed approach can be easily extended to other railway inspections such as rail defect detection or rail component identification. The next step of the research is to apply the approach to other rail defects such as localized surface collapse (LSC), rail end batter (REB), or crushed head (CH) prior to rail failure.

7. Conclusions

This paper presented machine learning-based methods for railway inspection, including a feature-based method and a deep neural network-based method using acceleration data. The deep learning-based method is an end-to-end time series classification approach for the detection of rail joints on railway track by training convolutional neural networks. It enhances the classification-based detection method with applicability and easy-deployment features. The advantages of the deep neural network-based approach are: (1) a significant reduction in the heavy data preprocessing for feature extraction; and (2) ability to detect joints on left or right rail using one global model. Our approach overcomes the interference issue with acceleration data, i.e., a discontinuity (joint or defect) on one side of rail also generates high amplitude of the acceleration signal on the other side

of the rail. This interference effect can increase the false alarm rate if a separate model for each side of the rail is used. With a unified model trained by data collected on both sides, this effect is learned in the model such that the model is able to distinguish joints on left or right rail specifically. The experimental results demonstrated that ResNet and FCN both achieved good performance in joint detection.

Although this study was focused on detecting rail joints, we believe that the developed approach can be easily extended to other conditions on rail surface. The next step of the research is to apply the approach to other rail surface defects [33]—such as localized surface collapse (LSC), rail end batter (REB), or crushed head (CH)—prior to rail failure. These defects have been identified by a recent derailment investigation as leading indicators for rail failure and derailment.

Author Contributions: The contribution for each author is as follows. Conceptualization: Y.L., C.Y. and Y.S.; Methodologies: C.Y. and Y.L. are the major contributors to the feature-based classification method; Y.S. and Y.L. are the major contributors to the deep-learning based method; Validation: C.Y., Y.S., C.L. and Y.L.; Data curation: Y.L.; Writing—original draft preparation: C.Y. and Y.S.; Writing—review and editing: Y.L. and C.L. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Acknowledgments: We are grateful to Transport Canada for providing a rail inspection car to collect rail joint data. We appreciate the suggestions and valuable discussion from our colleagues and the coop students for helping with the experiments.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Salient Systems Inc. Preventing Train Derailment. Available online: http://www.salientsystems.com (accessed on 21 December 2020).
2. Accident Trends—Summary Statistics. Building Energy Data Book, Department of Energy, United States. 2011. Available online: http://safetydata.fra.dot.gov/officeofsafety/publicsite/summary.aspxDOE (accessed on 21 December 2020).
3. How Can Rail Operators and OEMs Benefit from the Disruption Brought on by Digitization? Available online: https://www.mckinsey.com/media/McKinsey/Industries/Public/the-rail-sectors-hanging-maintenance-game.pdf (accessed on 21 December 2020).
4. Mosleh, A.; Montenegro, P.; Costa, P.A.; Calçada, R. An approach for wheel flat detection of railway train wheels using envelope spectrum analysis. *Struct. Infrastruct. Eng.* **2020**. [CrossRef]
5. Kalay, S. Research Implementation—Improved Rail Safety & Efficiency. In Proceedings of the 16th Annual AAR Research Review, Pueblo, CO, USA, 15–16 March 2011.
6. Barke, D.; Chiu, W.K. Structure Health Monitoring in the Railway Industry: A Review. *Struct. Health Monit.* **2005**, *4*, 81–94. [CrossRef]
7. Lechowicz, S.; Hunt, C. Monitoring and Managing Wheel Condition and Loading. Transportation Recording: 2000 and Beyond. In Proceedings of the International Symposium on Transportation Recorders, Arlington, VA, USA, 3–5 May 1999.
8. Yang, C.; Létourneau, S. Learning to Predict Train Wheel Failures. In Proceedings of the 11th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD 2005), Chicago, IL, USA, 21–24 August 2005; pp. 516–525.
9. Yang, C.; Létourneau, S. Model Evaluation for Prognostics: Estimating Cost Saving for the End Users. In Proceedings of the 6th International Conference on Machine Learning and Applications (ICMLA 2007), Cincinnati, OH, USA, 13–15 December 2007; pp. 304–309.
10. Yang, C.; Létourneau, S. Two-stage classifications for improving time-to-failure estimates: A case study in prognostic of train wheels. *Int. J. Appl. Intell.* **2009**, *31*, 255–266. [CrossRef]
11. Connell, D. Rail Safety and Rail Life Extension. In Proceedings of the 16th Annual AAR Research Review, Pueblo, CO, USA, 15–16 March 2011.
12. ENSCO—Track Inspection Vehicles. Available online: http://www.ensco.com/userfiles/file/Products_Services_PDF/07_Rail/Track-Inspection-Systems/100059-Track-Inspection-Vehicles-ENSCO-Rail.pdf (accessed on 21 December 2020).
13. Research and Traffic Group. *Railway Safety Technologies*; Technical Report for Railway Safety Act Review Secretariat; Research and Traffic Group: Glenburnie, ON, Canada, 2007.
14. Elberink, S.O.; Khoshelham, K.; Arastounia, M.; Diaz Benito, D. Rail Track Detection and Modeling in Mobile Laser Scanner Dat. *ISPRS Ann. Photogramm. Remote Sens. Spat. Inf. Sci.* **2013**, *2*, 223–228. [CrossRef]
15. Transportation Safety Board of Canada. *Rail Way Investigation Report*; R15H0013; Transportation Safety Board of Canada: Gatineau, QC, Canada, 2017.

16. Yang, C.; Tran, M.; Liu, Y. A Machine Learning-based Method for Rail inspection. In Proceedings of the 2017 International Conference on Knowledge, Information and Creativity Support Systems (KICSS 2017), Nagoya, Japan, 9–11 November 2017.
17. Berry, A.; Nejikovsky, B.; Gibert, X.; Tajaddini, A. High speed video inspection of joint bars using advanced image collection and processing techniques. In Proceedings of the 8th World Congres on Railway Research, Seoul, Korea, 18–22 May 2008; pp. 1–13.
18. Trinh, H.; Haas, N.; Li, Y.; Otto, C.; Pankanti, S. Enhanced rail component detection and consolidation for rail track inspection. In Proceedings of the 2012 IEEE Workshop on the Applications of Computer Vision (WACV), Breckenridge, CO, USA, 9–11 January 2012; pp. 289–295.
19. Alippi, C.; Casagrande, E.; Scotti, F.; Piuri, V. Composite Real-time Image Processing for Railways Track Profile Measurement. *IEEE Trans. Instrum. Meas.* **2000**, *49*, 559–564. [CrossRef]
20. Shafiullah, G.M.; Shawkat Ali, A.B.M.; Thompson, A.; Wolfs, P.J. Predicting Vertical acceleration of Railways Wagons Using Regression algorithms. *IEEE Trans. Intell. Transp. Syst.* **2010**, *11*, 290–299. [CrossRef]
21. Li, Z.; Molodova, M.; Núñez, A.; Dollevoet, R. Improvements in axle box acceleration measurements for the detection of light squats in railway infrastructure. *IEEE Trans. Ind. Electron.* **2015**, *62*, 4385–4397. [CrossRef]
22. Molodova, M.; Li, Z.; Núñez, A.; Dollevoet, R. Automatic Detection of Squats in Railway Infrastructure. *IEEE Trans. Intell. Transp. Syst.* **2014**, *15*, 1980–1990. [CrossRef]
23. Nason, G.P.; Silverman, B.W. The stationary wavelet transform and some statistical applications. *Science* **1995**, *103*, 281–299.
24. Liu, C.L. A Tutorial of the Wavelet Transform. 2010. Available online: http://disp.ee.ntu.edu.tw/tutorial/WaveletTutorial (accessed on 21 December 2020).
25. Sun, Y.; Liu, Y.; Yang, C. Railway Joint Detection using Deep Convolutional Neural Networks. In Proceedings of the IEEE 15th International Conference on Automation Science and Engineering (CASE), Vancouver, BC, Canada, 22–26 July 2019.
26. Hatami, N.; Gavet, Y.; Debayle, J. Classification of Time-Series Images Using Deep Convolutional Neural Networks. In Proceedings of the Tenth International Conference on Machine Vision 2017, Vienna, Austria, 13–15 November 2017.
27. Ismail Fawaz, H.; Forestier, G.; Weber, J.; Idoumghar, L.; Muller, P.A. Deep learning for time series classification: A review. *Data Min. Knowl. Disc.* **2019**, *33*, 917–963. [CrossRef]
28. Wang, Z.; Yan, W.; Oates, T. Time series classification from scratch with deep neural networks: A strong baseline. In Proceedings of the 2017 International Joint Conference on Neural Networks (IJCNN 2017), Anchorage, AK, USA, 14–19 May 2017; pp. 1578–1585.
29. Krizhevsky, A.; Sutskever, I.; Hinton, G.E. ImageNet Classification with Deep Convolutional Neural Networks. In Proceedings of the 25th International Conference on Neural Information Processing Systems, Lake Tahoe, NV, USA, 3–6 December 2012; Volume 1, pp. 1097–1105.
30. He, K.; Zhang, X.; Ren, S.; Sun, J. Identity Mappings in Deep Residual Networks. In *European Conference on Computer Vision*; Elsevier: Amsterdam, The Netherlands, 2016; pp. 630–645.
31. He, K.; Zhang, X.; Ren, S.; Sun, J. Deep residual learning for image recognition. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Las Vegas, NV, USA, 26 June–1 July 2016; pp. 770–778.
32. Jones, E.; Oliphant, E.; Peterson, P. SciPy: Open Source Scientific Tools for Python. 2001. Available online: http://www.scipy.org/ (accessed on 21 December 2020).
33. FRA (Federal Railroad Administration). *Track Inspector Rail Defect Reference Manual*; Office of Railroad Safety, US Deptartment of Transportation: Washington, DC, USA, 2015; pp. 1–82.