sensors

MDPI

Article

Defect-Repairable Latent Feature Extraction of Driving Behavior via a Deep Sparse Autoencoder

Hailong Liu [1,2,*], Tadahiro Taniguchi [3], Kazuhito Takenaka [4] and Takashi Bando [5]

1 The Graduate School of Information Science and Engineering, Ritsumeikan University, Kusatsu, Shiga 525-8577, Japan
2 Research Fellow (DC) with the Japan Society for the Promotion of Science, Tokyo 102-0083, Japan
3 The College of Information Science and Engineering, Ritsumeikan University, Kusatsu, Shiga 525-8577, Japan; taniguchi@em.ci.ritsumei.ac.jp
4 The Corporate R&D Div.1, Sensing System R&D Dept., DENSO CORPORATION, Aichi 448-8661, Japan; kazuhito_takenaka@denso.co.jp
5 The DENSO International America, Inc., San Jose, CA 95110, USA; takashi_bando@denso-diam.com
* Correspondence: liu@em.ci.ritsumei.ac.jp; Tel.: +81-077-561-5745

Received: 15 January 2018; Accepted: 9 February 2018; Published: 16 February 2018

Abstract: Data representing driving behavior, as measured by various sensors installed in a vehicle, are collected as multi-dimensional sensor time-series data. These data often include redundant information, e.g., both the speed of wheels and the engine speed represent the velocity of the vehicle. Redundant information can be expected to complicate the data analysis, e.g., more factors need to be analyzed; even varying the levels of redundancy can influence the results of the analysis. We assume that the measured multi-dimensional sensor time-series data of driving behavior are generated from low-dimensional data shared by the many types of one-dimensional data of which multi-dimensional time-series data are composed. Meanwhile, sensor time-series data may be defective because of sensor failure. Therefore, another important function is to reduce the negative effect of defective data when extracting low-dimensional time-series data. This study proposes a defect-repairable feature extraction method based on a deep sparse autoencoder (DSAE) to extract low-dimensional time-series data. In the experiments, we show that DSAE provides high-performance latent feature extraction for driving behavior, even for defective sensor time-series data. In addition, we show that the negative effect of defects on the driving behavior segmentation task could be reduced using the latent features extracted by DSAE.

Keywords: feature extraction; defects repairing; deep learning; driving behavior analysis

1. Introduction

Driving behavior can be measured as multi-dimensional time-series data, i.e., the driving behavior data generated by a variety of sensors in a vehicle, e.g., the speed of the wheels, steering angle, and accelerator opening rate. However, these multi-dimensional time-series data often include redundant information, i.e., a variety of sensor time-series data could present the same feature of driving behavior. A simple example is that the sensor time-series data of the speed of the wheels and meter readings of the velocity, as well as the engine speed can all present the velocity of the vehicle, even though these data have been obtained from different sensors or different sensing process involving different degrees of noise, artifacts, and influences from other variables, e.g., a gear ratio. Meanwhile, time-series data measured by a sensor can also be generated by fusing the original one-dimensional time-series data. For example, the sensor time-series data of the yaw rate can be regarded as data generated by fusing latent time-series data related to the velocity and the change in the driving direction (steering). For that reason, we can assume that some sensor time-series data are generated by fusing a variety of the

latent low-dimensional time-series data. Redundant information can be expected to complicate the data analysis, e.g., more factors need to be analyzed, and more calculations are needed; even varying levels of redundancy can influence the results of the analysis. Therefore, reducing the redundant information besides retaining the latent low-dimensional time-series data is a main task for this study. If the low-dimensional time-series data can be extracted from the driving behavior data, then we can use them in a variety of driving behavior analyses, e.g., recognition, prediction, and visualization.

A sensor, however, introduces noise, outliers, and defects into the time-series data by interacting with the environment and other sensors when a failure of the sensor occurs. Such defects in the time-series may negatively affect the extracted low-dimensional time-series data, because incorrect values are often measured in a period of frames. In addition, those defects also destroy the relevancy between data before and after them, e.g., the context of driving behaviors. For example, if we were to use a Hidden Markov model (HMM) for driving behavior analysis, defective data would result in the rupture of the latent Markov chain and produce unsatisfactory results. Thus, an important and challenging problem emerges from the above discussion: how to extract the low-dimensional time-series data of driving behavior. To address this problem, we need a method that is able to automatically extract low-dimensional time-series data from the measured multi-dimensional driving behavior data even when the data include redundant dimensions. Moreover, this method should also reduce the negative effect of defects in sensor time-series data on the extracted low-dimensional time-series data.

In this study, we start from the premise that defects can be detected because there are many existing methods that can detect outliers and defects [1–3]. We focus on and propose a method to extract low-dimensional time-series data of driving behavior with defects along with repairing the defective data. The proposed method is a deep learning method using a deep sparse autoencoder (DSAE), which can also repair defects introduced in measured driving behavior data. The DSAE is piled up by a number of autoencoder that reduces dimensionality by finding redundancy and detecting repetitive structures of data automatically [4]. The low-dimensional time-series data can be extracted as time-series of the latent features of a neural network, i.e., the activation patterns on hidden layers in the DSAE. The extracted time-series of latent features can represent driving behavior data. The advantage of this method is that DSAE only needs to be trained once and it can accomplish two tasks: extracting time-series of latent features and repairing defects in sensor time-series data. We reported brief descriptions and preliminary results of this study in [5,6]. In this paper, we present a complete description and experimental evaluation of our method and demonstrate the validity of our proposed method based on several experiments. We also propose using DSAE with a back propagation method (DSAE-BP) to repair the defects in sensor time-series data. The main contributions of this paper are as follows.

- We show that DSAE can extract highly correlated low-dimensional time-series of latent features by reducing various degrees of redundancy in different multi-dimensional time-series data of driving behavior.
- We verify that DSAE can reduce the negative effect of defects on the extracted time-series of latent features by repairing the defective sensor time-series data using a BP method.
- We find that the time-series of latent features extracted from the repaired time-series sensor data by DSAE have segmentation results similar to those of non-defective sensor time-series data.

As a result, we show that DSAE can be used as a highly useful and defect-repairable feature extraction method for driving behavior.

The remainder of this paper is organized as follows. Section 2 presents background material related to this research. Section 3 describes the proposed method including the feature extraction process and defect repair process. Sections 4 and 5 verify the effectiveness of the proposed method in terms of feature extraction and defect repair. Section 6 provides an application of the proposed method

to driving behavior segmentation. Section 7 discusses the advantages and limitations of proposed method. Section 8 concludes this paper.

2. Background

This section presents background material on three aspects of the research: feature extraction methods for driving behavior analysis, a feature extraction method by deep learning for intelligent vehicles, and defect repair for driving behavior analysis.

2.1. Feature Extraction for Driving Behavior Analysis

In most studies about driving behavior analysis, the feature vectors that are regarded as input data of the analysis method were selected manually from the measured sensor data [7–9] and designed manually [10]. For example, Taniguchi et al. selected the velocity, steering angle, brake pressure, accelerator position, and designed the temporal difference between the velocity and steering angle as input data of the HMM for driving behavior prediction [8]. Li et al. designed a feature for drunk driving detection by using the respective slopes of the lateral position of the vehicle, steering angle, and time interval [10]. However, we consider the feature design and selection to rely on expert experience, and finding an appropriate way of feature representations for each dataset is often difficult, especially for driving behaviors that are obtained from a driver–vehicle system including a large number of variables, e.g., the impact of the external environment on the driver, the driver's driving intention, and the vehicle dynamics. Compared with manual selection and manual feature design, we can apply an automatic feature extraction method for obtaining low-dimensional time-series data that can represent multi-dimensional data from many sensors. Therefore, we selected an unsupervised feature extraction method that can extract low-dimensional time-series of latent features from multi-dimensional sensor time-series data automatically.

Principal component analysis (PCA) is extensively used for unsupervised feature extraction [11]. PCA can project data into a low-dimensional feature space and find the principal components by maximizing the variance of the data in the projected feature space. Chatzigiannakis et al. proposed using PCA to detect traffic incidents, e.g., traffic accidents and low-speed vehicle flow [12]. Meanwhile, Calderó-Bardají et al. also used PCA to extract the latent features of electroencephalography and used a support vector machine to detect the steering direction prior [13]. In addition, Lin et al. proposed using an independent component analysis (ICA) method for predicting the driver's state of drowsiness [14]. The ICA can extract independent latent features from multivariate signals [15]. This method assumes that the measured signals are generated via linear transformations from source signals. However, ICA is very sensitive to outliers when it uses the kurtosis measure as an optimization criterion [16]. Therefore, ICA is not robust for extracting latent features of driving behavior when the defects were detected.

PCA and ICA cannot use a nonlinear transformation to extract time-series of latent features from driving behavior data because vehicle dynamics and human driving behavior have nonlinear properties. This problem could be solved by using kernel principal components analysis (KPCA) [17]. For example, Zhao et al. used KPCA to extract latent features for identifying a driver's mental fatigue. This method had a higher accuracy rate than PCA [18]. KPCA uses a nonlinear kernel function that involves a nonlinear transformation to map the data onto a high-dimensional space. However, when there is a large volume of driving behavior data, the computational cost of the KPCA method is high, because the kernel method must compute a gram matrix in $\mathbb{R}^{N\times N}$, where N is the total number of frames of data. In this paper, we propose using DSAE for extracting the time-series of latent features for driving behavior. DSAE has the advantages of being able to process both nonlinear data and large datasets.

2.2. Feature Extraction by Deep Learning for Intelligent Vehicles

In recent years, feature extraction methods employing deep learning approaches have attracted considerable attention. Deep learning methods use neural networks with a deep structure, i.e., they have three or more layers. Deep learning has previously been used in the field of intelligent vehicles for a variety of tasks. For example, Dong et al. proposed a regularized network that includes a stacked recurrent numerous network (RNN) and an autoencoder to use the GPS signal to analyze driving styles [19]. Hori et al. used a long short-term memory RNN to detect the confusion state of a driver [20]. Although the above two studies included feature extraction for time-series analysis, those features were extracted by supervised learning models for respective tasks. We used an unsupervised learning method for feature extraction in this study, because, if the time-series of latent features extracted via an unsupervised learning method are able to represent driving behavior, they can support varied tasks including tasks based on the unsupervised method, e.g., time-series segmentation [21] and data visualization [22].

Meanwhile, we consider that the feature extraction for driving behaviors in the lower layer of the deep learning model can be used to access the latent features of relatively simple driving behaviors, e.g., turning left and accelerating forward. As the number of layers increases, the latent features in the lower layer will be fused to form latent features that can represent more complex driving behaviors, e.g., turning left while accelerating. For example, [22] showed that a visualization result based on time-series of latent features extracted by using a deep learning method, i.e., DSAE, could determine complex driving behaviors more effectively than by using a feature extraction methods with a shallow structure, e.g., a sparse autoencoder (SAE), PCA, ICA, and KPCA. However, [22] evaluated the performance of DSAE in a task of driving behavior visualization. In this paper, we rigorously verified the capability of DSAE to extract the time-series of latent features of driving behavior.

2.3. Defect Repair for Driving Behavior Analysis

The presence of noise, outliers, and defective input data has a negative effect on many kinds of analyses. In the field of intelligent vehicles, many studies have proposed detection methods for outliers and defects. For example, Tagawa et al. proposed defect detection by using a structured denoising autoencoder, which they effectively applied to real driving data [1]. Krishnaswami et al. used a parity equation residual generation method to detect defective sensor information for diagnosing engine faults [2]. Malhotra et al. used long short-term memory networks to detect an anomaly in time-series, and they reported that this method had a promising result on multi-sensor engine data [3]. Therefore, we started with the premise that defects and outliers exist in driving behavior data.

To repair those outliers and defects, Liu et al. used a windowing process as the smoothing process to smooth noise in a pre-processing step of driving behavior visualization [22]. However, a windowing process cannot easily solve the problems caused by outliers and defects. Instead, a windowing process allows outliers and defects to have a negative effect on the time axis. Kaneda et al. used a robust Kalman filter to remove the outliers of vehicle control [23]. Jang alleviated traffic congestion by using a modified median filter to remove outliers of dedicated short-range communications probe data in an advanced traveler information system [24]. Usually, measurements do not only frequently contain outliers, but the defects in sensor time-series data also include incorrect measurements over a period of frames. The above two methods are unable to repair the data for a period in which many defects are found. Therefore, in this study, our approach to repairing a period of defects is to use prediction by a trained DSAE. The advantage of our method is that the DSAE can extract time-series of latent features and repair defects in sensor time-series data by training the DSAE model only one time.

Two previous studies used a deep learning method to repair defects in the field of intelligent transportation systems. Duan et al. used stacked denoising autoencoders for the imputation of defective traffic data [25]. Ku et al. also used stacked denoising autoencoders that were constructed for road segments to extract their spatial-temporal relationships to use them for imputing defects [26]. The above two studies both used a deep learning method based on a stacked denoising autoencoder.

Because the denoising autoencoder added some noise and defects to the training data when the model was being trained, both of the above studies led to the development of a model with a generalization of noise and defects. In addition, it should be noted that these models [25,26] focused on simply repairing defects in the input data. In contrast, the purpose of this study is to extract time-series of latent features that can represent driving behaviors, even those containing defects, in the measured sensor time-series data.

3. Proposed Method

We propose the use of the deep learning method DSAE to extract the time-series of latent features for driving behavior. DSAE can also reduce the negative effects on feature extraction when the sensor time-series data contain defects, which starts from the premise that defects can be detected. Our proposed method is illustrated in Figure 1, which shows two steps: the training process and defect-repairing process. The training process trains the DSAE with non-defective sensor time-series data of driving behavior. The defect-repairing process repairs defects in the sensor time-series data and extracts the time-series of latent features by using the trained DSAE.

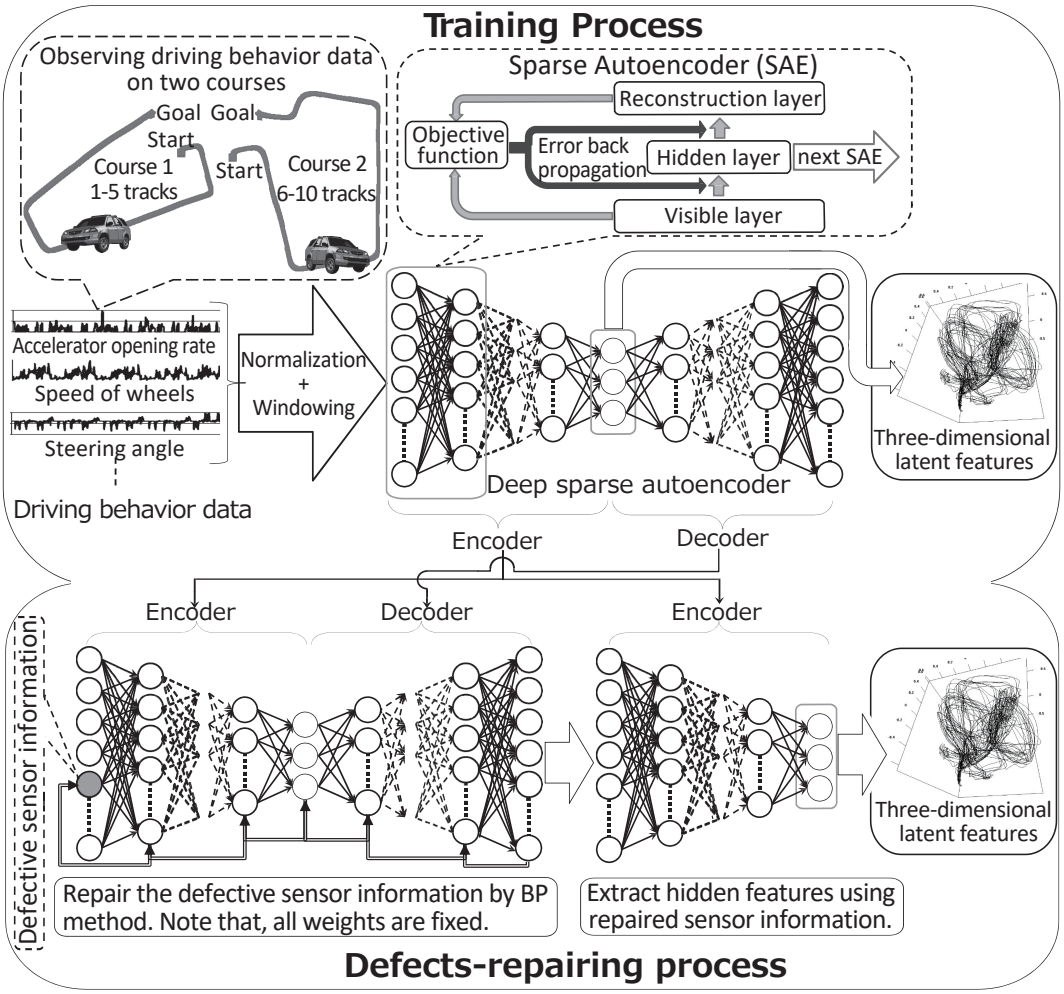

Figure 1. Feature extraction and defect-repairing processes of the deep sparse autoencoder (DSAE), which repairs defects in sensor time-series data by using a back propagation (BP) method.

3.1. Training Process

In the training process, the DSAE is trained to extract a low-dimensional time-series latent feature from driving behavior data. Before data are input into the DSAE, we use two pre-processing steps: normalization and windowing processing. Normalization is used to deal with dimensionless quantity because the unit of each kind of sensor time-series data of driving behavior is different. We define the measured driving behavior data as $\mathbf{Y} \in \mathbb{R}^{D_Y \times N_Y}$, where each dimension represents one kind of sensor time-series data and where D_Y is the dimensionality and N_Y is the quantity of data (frames) in $\mathbf{Y}$. The t-th datum is defined as $\mathbf{y}_t = (y_{t,1}, y_{t,2}, \ldots, y_{t,D_Y})^T \in \mathbb{R}^{D_Y}$.Note that, the activation function we employ in the DSAE uses a hyperbolic tangent function of which the output range is $(-1, 1)$. However, the range of the measured driving behavior data in the dataset $\mathbf{Y}$ is $(-\infty, \infty)$. To reconstruct the input data with tanh, we independently normalize each dimension of the input data into $(-1, 1)$ by using the maximum and minimum values. Therefore, the t-th normalized datum is expressed as $\mathbf{x}_t = (x_{t,1}, x_{t,2}, \ldots, x_{t,D_Y})^T \in \mathbb{R}^{D_Y}$, where

$$x_{t,d} = 2\Big(\frac{y_{t,d} - \min(y_{1:N_y,d})}{\max(y_{1:N_y,d}) - \min(y_{1:N_y,d})}\Big) - 1 \tag{1}$$

In addition, to extract the latent features with a property of time-series, a windowing process enables the DSAE to fully connect a period of time steps of driving behavior data in the input layer.

In the windowing process, we aggregate the normalized data with a slide window to convert the data of w frames into a vector. Therefore, the t-th frame of the windowing time-series data $\mathbf{h}_t^{(1)}$ is expressed as $\mathbf{h}_t^{(1)} = (\mathbf{x}_{t-w+1}^T, \mathbf{x}_{t-w+2}^T, \ldots, \mathbf{x}_{D_V}^T)^T \in \mathbb{R}^{D_{\mathbf{H}}^{(1)}}$,where $D_{\mathbf{H}}^{(1)} = w \times D_Y, t \geq w$. We notate these time-series data as $h_t^{(1)}$ with superscript (1) because the time-series data is used as an input for the first layer in the network of the DSAE. Finally, we obtain the windowing time-series data, which are $\mathbf{H}^{(1)} = \{\mathbf{h}_1^{(1)}, \mathbf{h}_2^{(1)}, \cdots, \mathbf{h}_{N_{\mathbf{H}^{(1)}}}^{(1)}\} \in \mathbb{R}^{D_{\mathbf{H}^{(1)}} \times N_{\mathbf{H}^{(1)}}}$, when the slide window moves along the time axis in a step-by-step manner. The frames in $\mathbf{H}^{(1)}$ are $N_{\mathbf{H}^{(1)}} = N_Y - w + 1$.

The DSAE is a deep neural network with a symmetrical structure and an odd number of layers. The DSAE extracts time-series of latent features by minimizing the error between the input time-series data and decoded time-series data through an encoding–decoding process. The DSAE is presented diagrammatically in the upper part of Figure 1. If DSAE has L layers ($L \in \{2n+1 \mid n \in \mathbb{Z}^+\}$), then the layers from the first layer to the middle layer, notated as the m-th layer, where $m = (L+1)/2$, are the encoding layers. Meanwhile, the decoding layers are those from the m-th layer to the L-th layer. The time-series data of the middle layer (the m-th layer) of the DSAE are the time-series of the latent features of the input data.

The mappings between all pairs of adjacent layers can be expressed by the same equation. For example, the t-frame data $\mathbf{h}_t^{(l)}$ of the l-th layer are mapped to the $l+1$-th layer via

$$\mathbf{h}_t^{(l+1)} \quad = \quad f(\mathbf{W}^{(l)}\mathbf{h}_t^{(l)} + \mathbf{b}^{(l)}) \in \mathbb{R}^{D_H^{(l)}}, \tag{2}$$

in which the function f is an activation function of DSAE and $l \in \mathbb{Z}^+$, $l < L$. In this study, we selected a hyperbolic tangent function $f(\cdot) = tanh(\cdot)$ as the activation function because the hyperbolic tangent function outperforms the sigmoid function and radial bases function [27]. Meanwhile, $\mathbf{W}^{(l)} \in \mathbb{R}^{D_H^{(l)} \times D_H^{(l+1)}}$ and $\mathbf{b}^{(l)} \in \mathbb{R}^{D_H^{(l)}}$ are the weight matrix and bias vector. $D_H^{(l)}$ is the dimensionality of the vector of the l-th layer. The objective function of a DSAE with L layers is

$$J(\Phi) \quad = \quad \frac{1}{2N_V}\sum_{t=1}^{N_V}||\mathbf{h}_t^{(L)} - \mathbf{h}_t^{(1)}||_2^2 + \frac{\alpha}{2}\sum_{l=1}^{L-1}||\mathbf{W}^{(l)}||_2^2 + \beta\sum_{i=1}^{D_H^{(l)}}\mathrm{KL}(\omega||\bar{h}_i^{(m)}), \tag{3}$$

where $\Phi = \{\mathbf{W}^{(1)}, \cdots, \mathbf{W}^{(L-1)}, \mathbf{b}^{(1)}, \cdots, \mathbf{b}^{(L-1)}\}$. The first term is an error term that shows the square error between the input data and decoded data. The second term is a penalty term that limits the elements of all the weights $\mathbf{W}^{(l)}$ with the L2 norm to prevent over-fitting. The parameter α can control the strength of the penalty term. The third term is a sparse term to ensure data sparsity in the m-th layer. It is because this term would enable more obvious features to be obtained. The sparse term is a Kullback–Leibler divergence between two Bernoulli random variables with ω and $\bar{h}_i^{(m)}$ [28].

$$\mathrm{KL}(\omega||\bar{h}_i^{(m)}) = \omega \log \frac{\omega}{\bar{h}_i^{(m)}} + (1-\omega) \log \frac{1-\omega}{1-\bar{h}_i^{(m)}}, \tag{4}$$

$$\bar{h}_i^{(m)} = \frac{1}{2}\Big(1 + \frac{1}{N_V}\sum_{t=1}^{N_V} h_{t,i}^{(m)}\Big), \tag{5}$$

where ω is the sparsity target of the hidden layer and $h_{t,i}^{(m)}$ is the i-th element of $\mathbf{h}_t^{(m)}$. The value of β can be used to control the strength of the sparse term. If the sparse term was minimized, then $\bar{h}_i^{(m)}$ will be close to ω.

Finally, we use a back propagation (BP) method [29] to minimize the objective function for training the DSAE. The BP method performs partial differentiations of the weight matrices and biases for the objective function Equation (3) via a chain rule. Therefore, the equations to update the weight matrix $\mathbf{W}^{(l)}$ and the bias vector $\mathbf{b}^{(l)}$ between the l-th and $(l+1)$-th layers are:

$$\begin{aligned} \mathbf{W}^{+(l)} &\leftarrow \mathbf{W}^{(l)} - \lambda^{(l)} \frac{\partial J(\Phi)}{\partial \mathbf{W}^{(l)}}, \\ &= \mathbf{W}^{(l)} - \lambda^{(l)} \Big(\frac{1}{N_v}\sum_{t=1}^{N_v} \mathbf{h}_t^{(l)} \boldsymbol{\gamma}_t^{(l)} + \alpha \mathbf{W}^{(l)}\Big), \end{aligned} \tag{6}$$

$$\begin{aligned} \mathbf{b}^{+(l)} &\leftarrow \mathbf{b}^{(l)} - \lambda^{(l)} \frac{\partial J(\Phi)}{\partial \mathbf{b}^{(l)}}, \\ &= \mathbf{b}^{(l)} - \lambda^{(l)} \Big(\frac{1}{N_v}\sum_{t=1}^{N_v} \boldsymbol{\gamma}_t^{(l)}\Big), \end{aligned} \tag{7}$$

where $(\cdot)^+$ signifies the value that has been updated. The vector $\boldsymbol{\gamma}_t^{(l)} \in \mathbb{R}^{D_H^{(l)}}$ is calculated via

$$\boldsymbol{\gamma}_t^{(l)} = \begin{cases} -\mathbf{D}^{(l+1)}(\mathbf{h}_t^{(1)} - \mathbf{h}_t^{(L)}) & (l{=}L{-}1) \\ \mathbf{D}^{(l+1)}(\mathbf{W}^{(l+1)}\boldsymbol{\gamma}_t^{(l+1)}) + \beta\boldsymbol{\epsilon} & (l{=}m{-}1) \\ \mathbf{D}^{(l+1)}(\mathbf{W}^{(l+1)}\boldsymbol{\gamma}_t^{(l+1)}) & (\text{others}), \end{cases} \tag{8}$$

where $\mathbf{D}^{(l)}$ is a diagonal matrix shown by

$$\mathbf{D}^{(l)} = \mathrm{diag}\left(1 - \big(h_{t,1}^{(l)}\big)^2, 1 - \big(h_{t,2}^{(l)}\big)^2, \cdots, 1 - \big(h_{t,D_H^{(l)}}^{(l)}\big)^2\right); \tag{9}$$

and the i-th element of the vector $\boldsymbol{\epsilon} \in \mathbb{R}^{D_H^{(l)}}$ is

$$\epsilon_i = \frac{1-\omega}{1-\bar{h}_i^{(m)}} - \frac{\omega}{\bar{h}_i^{(m)}}.$$

In the above updating equations, $\lambda^{(l)}$ is the dynamic learning rate of $\mathbf{W}^{(l)}$ and $\mathbf{b}^{(l)}$. The dynamic learning rate is changed in every iterative cycle via a line search, for which we define the searching

distance θ. The dynamic learning rate $\lambda^{(l)}$ and searching distance θ are initialized by small positive numbers. Therefore, θ is updated by:

$$\theta^{+} = \begin{cases} -0.5\theta & \left(J^{+}(\Phi) > J(\Phi)\right) \\ \theta & \left(J^{+}(\Phi) < J(\Phi)\right). \end{cases} \tag{10}$$

Then, the appropriated dynamic learning rate is ${\lambda^{(l)}}^{+} = \lambda^{(l)} + \theta$. The line search is set to terminate when the value of $|J^{+}(\Phi) - J(\Phi)|$ becomes smaller than a certain small threshold.

To prevent the weight and bias from converging to inaccurate local optima, we use a procedure known as greedy layer-wise training. We divide the DSAE into a number of SAEs that include three layers. Each SAE is responsible for optimizing a set of weights and bias between every two layers of DSAE. Meanwhile, the SAE also outputs the data in its m-th layer as the input data of the next SAE. After using SAE to train all the weights and bias layer by layer, we use those trained weights and bias as initialization for the DSAE, and re-train them through the multi-layer neural network of DSAE.

3.2. Defect-Repairing Process

We consider a trained DSAE capable of repairing the defects in sensor time-series data via the encoding-decoding process because it is trained by minimizing the squared error between the input data and reconstructed data, which is the decoded data. In the work presented in this paper, we assume that the defective sensor is given. In [6], we used a DSAE with a fixed-point iterative method (DSAE-FP) to reduce the negative effect of defects on the driving behavior in a segmentation task. In addition, we propose the DSAE with back propagation method (DSAE-BP) in this section.

3.2.1. DSAE-FP

The DSAE-FP, which inputs sensor time-series data with defects into the trained DSAE, and outputs the reconstructed time-series data, is illustrated in Figure 2. Then, the defects are updated by the corresponding elements of the reconstructed time-series data. Several iterations of the above steps are necessary for convergence to a fixed point. However, DSAE-FP only uses the potential of data reconstruction to repair the defects in sensor time-series data. Therefore, DSAE-FP would easily cause the defects to converge to different fixed points if the initial defect values were different.

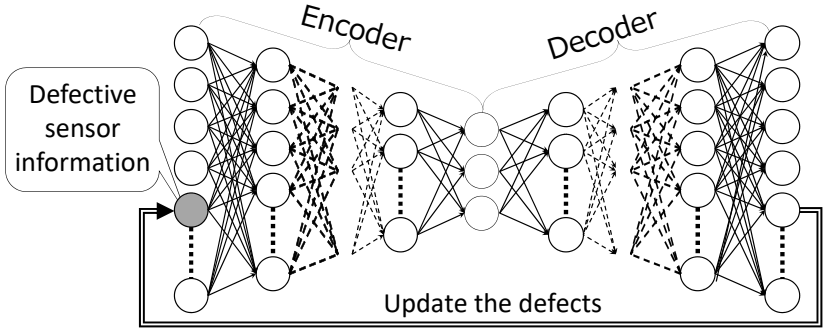

Figure 2. DSAE-FP: Repair the defects in sensor time-series data by updating to the reconstructed time-series data.

3.2.2. DSAE-BP

An overview of DSAE-BP is shown in Figure 3. The advantage of using the BP method is that it guarantees that the defects are updated in the direction in which the reconstruction error is minimized using the BP method. The reconstruction error of t-th time step is

$$E(\Phi)_t = \frac{1}{2}||\mathbf{h}_t^{(L)} - \mathbf{h}_t^{(1)}||_2^2. \tag{11}$$

We update the defective element $h_{t^{(1)},d}$ in the sensor time-series data via

$$\left(h_{t,d}^{(1)}\right)^{+} = h_{t,d}^{(1)} - \psi\left(\frac{\partial E(\Phi)_t}{\partial \mathbf{h}_t^{(1)}}\right)_d, \tag{12}$$

where ψ is the learning rate and $\left(\frac{\partial E(\Phi)_t}{\partial \mathbf{h}_t^{(1)}}\right)_d$ is the d-th element of the vector $\frac{\partial E(\Phi)_t}{\partial \mathbf{h}_t^{(1)}}$. This partial differential is expanded as,

$$\begin{aligned}
\frac{\partial E(\Phi)_t}{\partial \mathbf{h}_t^{(1)}} &= \frac{1}{2}\frac{\partial E(\Phi)_t}{\partial (\mathbf{h}_t^{(1)} - \mathbf{h}_t^{(L)})}\frac{\partial (\mathbf{h}_t^{(1)} - \mathbf{h}_t^{(L)})}{\partial \mathbf{h}_t^{(1)}} \\
&= (\mathbf{h}_t^{(1)} - \mathbf{h}_t^{(L)})\frac{\partial (\mathbf{h}_t^{(1)} - \mathbf{h}_t^{(L)})}{\partial \mathbf{h}_t^{(1)}} \\
&= (\mathbf{h}_t^{(1)} - \mathbf{h}_t^{(L)}) - (\mathbf{h}_t^{(1)} - \mathbf{h}_t^{(L)})\frac{\partial \mathbf{h}_t^{(L)}}{\partial \mathbf{h}_t^{(1)}}, \\
\frac{\partial \mathbf{h}_t^{(L)}}{\partial \mathbf{h}_t^{(1)}} &= \frac{\partial \mathbf{h}_t^{(L)}}{\partial \mathbf{h}_t^{(L-1)}}\frac{\partial \mathbf{h}_t^{(L-1)}}{\partial \mathbf{h}_t^{(L-2)}} \times \cdots \times \frac{\partial \mathbf{h}_t^{(2)}}{\partial \mathbf{h}_t^{(1)}},
\end{aligned} \tag{13}$$

through the chain rule. The mapping between each two layers of the DSAE is the same as in Equation (2); therefore, the partial derivative of the l-th layer with respect to the $(l-1)$-th layer can be expressed by

$$\frac{\partial \mathbf{h}_t^{(l)}}{\partial \mathbf{h}_t^{(l-1)}} = \mathbf{D}^{(l)}\mathbf{W}^{(l-1)}. \tag{14}$$

Overall,

$$\frac{\partial E(\Phi)_t}{\partial \mathbf{h}_t^{(1)}} = (\mathbf{h}_t^{(1)} - \mathbf{h}_t^{(L)}) - (\mathbf{h}_t^{(1)} - \mathbf{h}_t^{(L)})\mathbf{D}^{(L)}\mathbf{W}^{(L-1)}\mathbf{D}^{(L-1)}\mathbf{W}^{(L-2)} \times \cdots \times \mathbf{D}^{(2)}\mathbf{W}^{(1)}. \tag{15}$$

As a result, a defect $h_{t,d}^{(1)}$ is updated via Equation (12) with several iterations required to repair it.

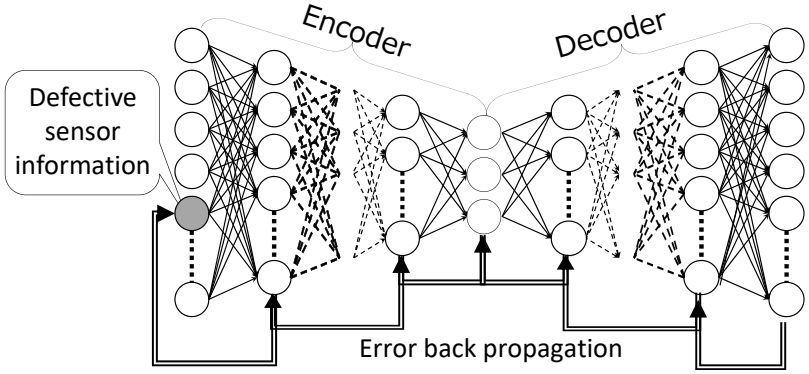

Figure 3. DSAE-BP: Repair the defects in sensor time-series by a BP method. Weights and biases of DSAE are not changed.

4. Experiment 1: Feature Extraction

The purpose of this experiment is to verify the ability of the DSAE to extract the time-series of latent features. Driving behavior data were obtained from multiple sensors with an actual vehicle. In this experiment, the DSAE extracts the time-series of latent features from different datasets by preparing 12 datasets, D1 to D12, that include part or all of the measured sensor time-series data. PCA was also used as a comparative method. In the experiment, we evaluated the reconstruction error

and the extracted latent features by using the two models. Additional details of the experiments are provided in the following subsections.

4.1. Experimental Conditions

We asked a participant to drive the vehicle through two types of courses shown in Figure 1. Pedestrians, other moving vehicles, and parked vehicles, were found along the courses. The vehicle traveled each course five times to obtain the driving behavior data. The data of circuits 1–5 corresponded to the first course, and data of circuits 6–10 corresponded to the second course, as shown in Figure 1. In total, we measured 12,958 frames of driving behavior data with a frame rate of 10 fps. Each frame of data included nine kinds of sensor information, which are listed in Table 1.

Table 1. Measured sensor information and the assumed latent features of driving behavior.

Measured Sensor Information		
I_1: Accelerator opening rate	I_2: Brake master-cylinder pressure	I_3: Steering angle
I_4: Speed of wheels	I_5: Meter readings of velocity	I_6: Engine speed
I_7: Longitudinal acceleration	I_8: Lateral acceleration	I_9: Yaw rate
Assumed latent features		
V: The feature is related to the velocity		
A: The feature is related to the acceleration		
D: The feature is related to a change in the driving direction		

We considered the physical meaning of these nine kinds of sensor information. For the discussion in this paper, we divided the sensor information into three categories for reference. The sensor information measured for the accelerator-opening rate, brake master-cylinder pressure, and longitudinal acceleration were considered to be related to the acceleration of the vehicle; the sensor information obtained for the speed of the wheels, and the meter readings of the velocity and engine speed were considered to relate to the velocity of the vehicle; the steering angle can represent the change in the driving direction; and the longitudinal acceleration and yaw rate contained information about the acceleration of the vehicle and the change in the driving direction. Therefore, we assumed that the nine kinds of sensors information were generated from the latent features of the driving behavior, which were related to the velocity of the vehicle (V), the acceleration of vehicle (A), and the change in driving direction (D). Note that we did not demonstrably affirm the acceleration, velocity, and the change in the driving direction to be latent features of the driving behavior; instead, we assumed that the latent features were related to them.

Twelve datasets (D1 to D12) composed of different measured sensor information, as listed in Table 2, were prepared for the purpose of verifying and comparing the feature extraction performance. In Table 2, the included sensor information I_1 to I_9 are marked by "$\surd$" for each dataset; and "$\circ$" shows the assumed latent features V, A, and D relating to each dataset. For example, dataset D1 included the accelerator opening rate and brake master-cylinder pressure, and was assumed to involve latent features in relation to A. D5 included the steering angle and speed of wheels that were assumed to involve latent features both in relation to V and D. Meanwhile, we assumed that D7 to D12 involved all the latent features in relation to A, V, and D.

We employed 12 PCAs and 12 DSAEs to extract the three-dimensional time-series of the latent features from D1 to D12 independently. The experimental results of [22] showed that, when using window size $w = 10$, the extracted three-dimensional time-series of the latent features could represent a variety of driving behaviors and the transformations between driving behaviors by driving behavior visualization. Thus, for each PCA and DSAE, we set the time window size to $w = 10$ (one second) for windowing processing. The PCAs extracted three-dimensional time-series of the latent features from the windowing data of each dataset, of which the structure is shown in the last column of Table 2. When DSAEs were used to extract three-dimensional data from each dataset, the dimensions of each

layer were approximately half of the dimensions of the upper layer. The structure of the encoder of each DSAE is shown in the third column of Table 2. The set of parameters for the DSAEs is as follows: $\alpha = 0.03$, $\beta = 0.7$, and $\omega = 0.5$, which were set based on the experience we gained during many previous experiments.

Table 2. Sensor information included in the 12 datasets we prepared and the PCA and DSAE designs for each of them.

Data Sets	Included Sensor Information									Assumed Latent Features			Encoder Structure of DSAE (with Window Size: 10)	The Structure of PCA (with Window Size: 10)
	I_1	I_2	I_3	I_4	I_5	I_6	I_7	I_8	I_9	V	A	D		
D1	√	√									○		2D×10=20D→10D→5D→3D	2D×10=20D→3D
D2			√									○	1D×10=10D→5D→3D	1D×10=10D→3D
D3				√						○			1D×10=10D→5D→3D	1D×10=10D→3D
D4	√	√		√						○	○		3D×10=30D→15D→7D→3D	3D×10=30D→3D
D5			√	√						○		○	2D×10=20D→10D→5D→3D	2D×10=20D→3D
D6	√	√	√								○	○	3D×10=30D→15D→7D→3D	3D×10=30D→3D
D7	√	√	√	√						○	○	○	4D×10=40D→20D→10D→5D→3D	4D×10=40D→3D
D8	√	√	√	√	√					○	○	○	5D×10=50D→25D→12D→6D→3D	5D×10=50D→3D
D9	√	√	√	√	√	√				○	○	○	6D×10=60D→30D→15D→7D→3D	6D×10=60D→3D
D10	√	√	√	√	√	√	√			○	○	○	7D×10=70D→35D→17D→8D→3D	7D×10=70D→3D
D11	√	√	√	√	√	√	√	√		○	○	○	8D×10=80D→40D→20D→10D→3D	8D×10=80D→3D
D12	√	√	√	√	√	√	√	√	√	○	○	○	9D×10=90D→45D→22D→11D→3D	9D×10=90D→3D

4.2. Evaluation of Model Training via Data Reconstruction

In our view, reconstructing driving behavior data from the extracted time-series of latent features is important for evaluating the feature extraction model, especially for the DSAE, which extracts the time-series of latent features by minimizing the reconstruction error with an encode-decode process. Meanwhile, data reconstruction is also important for repairing the defects in sensor time-series data because we can use it to compensate for the defects. In this regard, PCA is similar to an autoencoder in that it can be considered as an autoencoder model with a linear active function [30]. Therefore, PCA can reconstruct the input data by

$$\mathbf{h}_t^{(L)} = \mathbf{W}_E^T(\mathbf{W}_E\mathbf{h}_t^{(1)}), \tag{16}$$

where $\mathbf{W}_E \in \mathbb{R}^{D_{H^{(1)}} \times D_{H^{(m)}}}$ is a matrix of eigenvectors. We reconstructed the windowing time-series data by decoding the time-series of latent features by using PCA and DSAE. The square error between the windowing time-series data and the reconstructed data can be used as a measure to evaluate the effectiveness of training. Considering the different dimensionalities of the data in D1 to D12, we used the average of the square error as an evaluation measure.

The averaged square error for D1 to D12 by using PCA and DSAE is shown in Figure 4. We can see that, when using the DSAEs, the square error between the windowing time-series data and the reconstructed time-series data was generally smaller than when using PCAs. Although the square errors of D2, D3, and D5 were smaller when using PCAs than when using DSAEs, these errors were insignificantly small regardless of whether PCAs or DSAEs were used. The above results showed that DSAE was more effective to reconstruct windowing time-series data from the time-series of latent features than PCA.

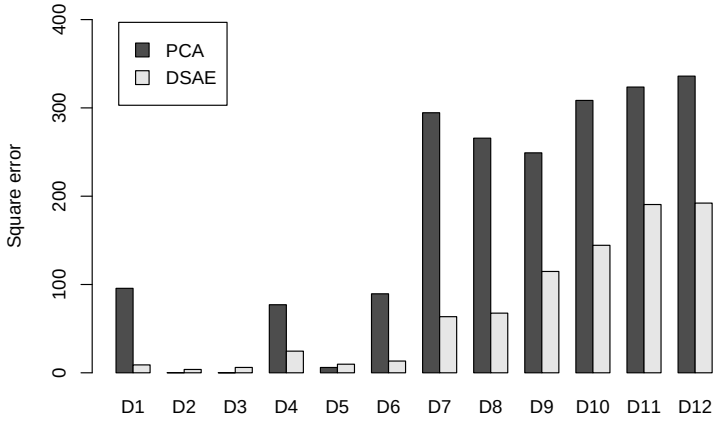

Figure 4. Averaged square error between windowing time-series data and reconstructed time-series data of datasets D1 to D12.

4.3. Evaluation of Latent Feature Extraction of Time-Series Using CCA

We consider that, if the method can obtain a similar extraction result from different datasets including the assumed three latent features of V, A, and D, then the extraction performance of this method is high. Here it is necessary to note that unsupervised learning methods, such as PCA and DSAE, extract time-series of latent features according to the distribution of the input data and generate the feature space automatically. This leads to a certain degree of freedom (the rotation and displacement of axes) between the feature spaces generated from the different datasets. For example, Figures 5 and 6 show that the distributions of the extracted time-series of latent features from D7 to D12, which is expected to be similar, is rotated with the feature space generated by PCA and DSAE. Therefore, the standard correlation analysis could not be used to evaluate the experimental results directly because the time-series of latent features that were extracted from different datasets are not in the same space. Instead, we used canonical correlation analysis (CCA) [31] to analyze the correlation as a method to evaluate the similarity between the data from the two different spaces. CCA seeks a pair of linear transformations, one for each dataset, such that when the dataset is transformed, the corresponding coordinates are maximally correlated. The two optimized linear transformations reduce the influence of the degree of freedom. This process enables us to evaluate the correlation with a pair of extracted time-series of latent features. We use the canonical correlation coefficient as the evaluation measure to present the correlation. The range of the canonical correlation coefficient is $[0, 1]$. As all the canonical correlation coefficients approach 1, the strength of the correlation between the two datasets increases. Likewise, if all the canonical correlation coefficients are close to 0, then the two datasets are uncorrelated. In both of these cases, both datasets are transformed using two optimized vectors.

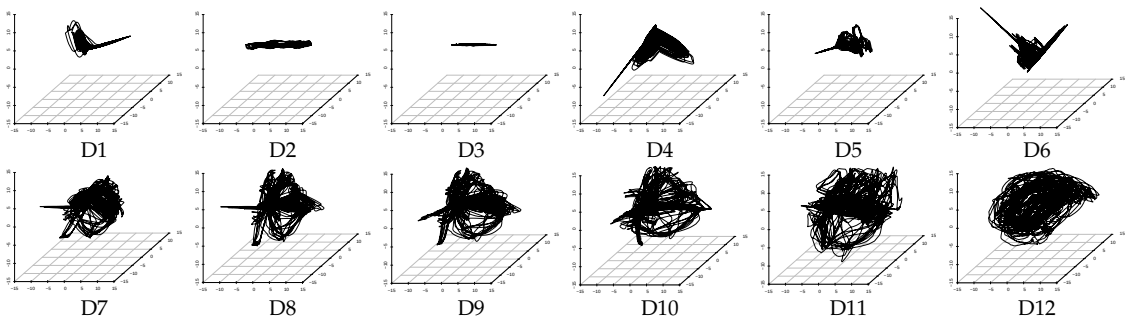

Figure 5. Three-dimensional time-series of latent features extracted from datasets D1 to D12 by PCA in the three-dimensional feature space.

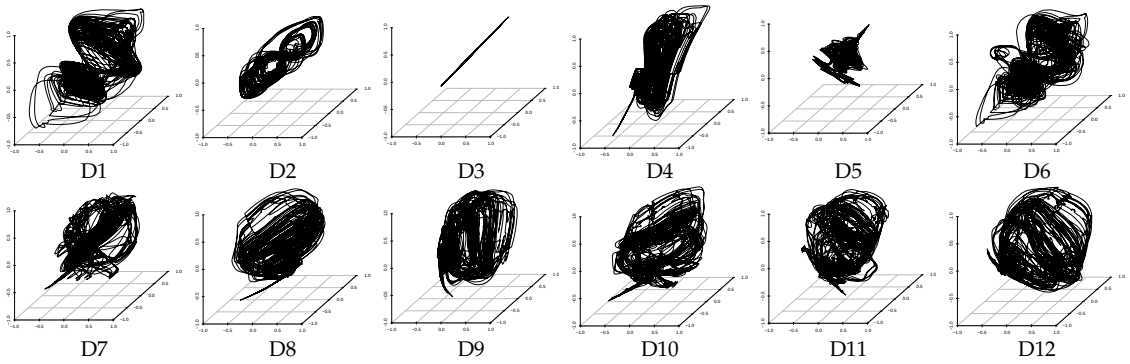

Figure 6. Three-dimensional time-series of latent features extracted from datasets D1 to D12 by DSAE in the three-dimensional feature space.

We considered all pairs of datasets from D1 to D12, and used CCA to evaluate the correlation of features extracted from these pairs. The relationship between the time-series of latent features extracted from different datasets was evaluated by defining the distance of correlation (*DOC*) using the canonical correlation coefficients; in particular,

$$DOC = 1 - \frac{1}{D_H^{(m)}} \sum_{k=i}^{D_H^{(m)}} CCC_k, \tag{17}$$

where CCC_k is the k-th canonical correlation coefficient, and $D_H^{(m)}$ is the dimensionality of the extracted time-series of latent features; for this evaluation, $D_H^{(m)} = 3$. If the time-series of latent features extracted from two datasets have a strong correlation, *DOC* is close to 0; conversely, *DOC* is close to 1.

We used Equation (17) to evaluate the correlation between each pair of time-series of latent features extracted from the 12 datasets we prepared. The distances of correlation among the time-series of latent features extracted from D1 to D12 by using the PCAs and DSAEs are shown in Figures 7 and 8. In both figures, the distances of correlation among the extracted time-series of latent features from D1 to D6 were larger than 0.2. This result is in agreement with our assumptions, namely that D1 to D6 did not involve the same assumed latent features shown in Table 2. Meanwhile, we assumed D7 to D12 to involve the same latent features related to V, A, and D, which are also shown in Table 2. Therefore, we considered that the *DOC*s among the time-series of latent features extracted from D7 to D12 should be as small as possible. This assumption is verified by using the PCAs and DSAEs because the *DOC*s among the time-series of latent features extracted from D7 to D12 were less than 0.11. However, in Figure 7, the *DOC*s among the time-series of latent features extracted by using PCAs from D7 to D12 were larger than by using DSAEs. Particularly, when using PCA, the *DOC*s among D7-D11, D7-D12, and D9-D12 were obviously larger than when using DSAE. Finally, we measured the average value of *DOC*s among D7 to D12 by using both PCAs and DSAEs. The result is shown in Figure 9. The figure shows that, in contrast to PCA, the average *DOC* of DSAE was more than 2.5 times lower than that of PCA. In summary, we verified the ability of DSAE to extract the time-series of latent features from the driving behavior datasets with various degrees of redundancy, and especially that it outperformed PCA.

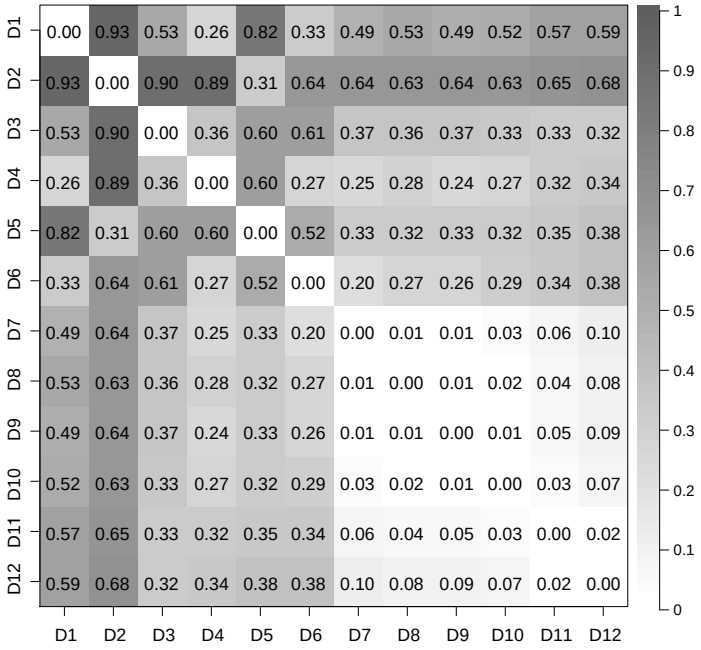

Figure 7. Distance of correlation between each pair of time-series of latent features extracted from the 12 datasets by using PCAs.

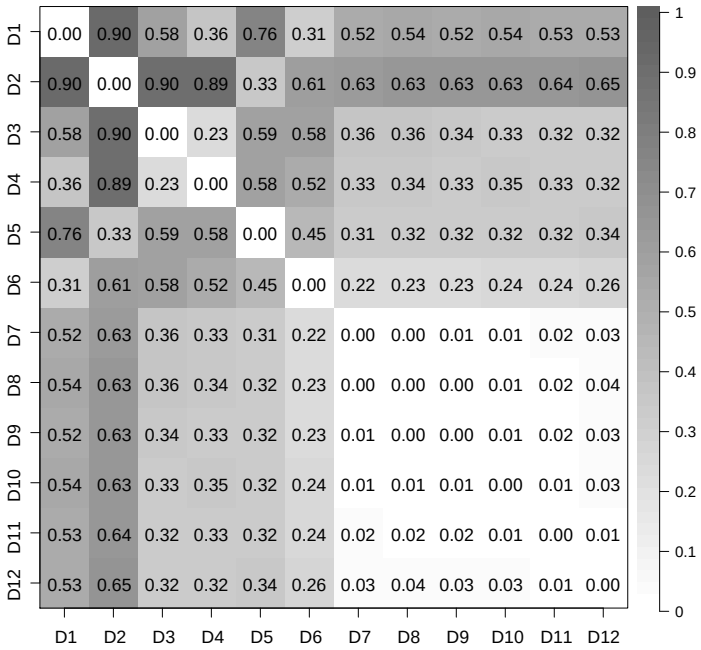

Figure 8. Distance of correlation between each pair of time-series of latent features extracted from the 12 datasets by using DSAEs.

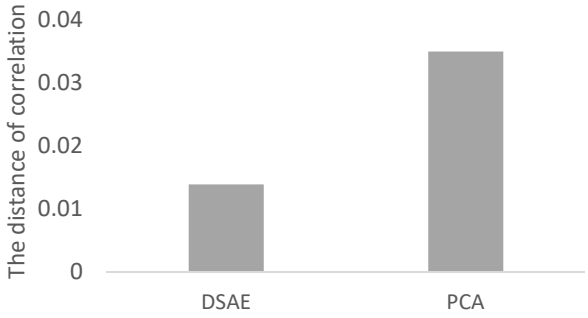

Figure 9. Averaged distance of correlation among pairs of time-series of latent features extracted from D7 to D12 by using PCAs and DSAEs.

5. Experiment 2: Reducing the Negative Effect of Defects for Feature Extraction

The purpose of this experiment is to verify that our proposed method can reduce the negative effect of defects on the feature extraction of driving behavior. The sensor time-series data obtained when measuring actual driving behavior often includes defects. First, we manually created defects in different non-defective sensor time-series datasets. Second, we used a DSAE that was trained on non-defective sensor time-series data to repair the defects, and extracted the time-series of latent features from the repaired time-series data. Finally, we evaluated the ability to reduce the negative effect of defects with two criteria: the similarity between the repaired data and the non-defective data, and the similarity between latent features from the repaired data and the non-defective data.

5.1. Experimental Conditions

We assumed that each sensor-measured time-series dataset includes a periodical defect. Therefore, we synthetically introduced some defects into dataset D12 listed in Table 2, which included nine kinds of intact sensor time-series data. Based on D12, we prepared eight datasets, C1 to C8, which included defects in different sensor time-series data. However, we coordinated the time of the defects of longitudinal and lateral acceleration in C7, because the same acceleration sensor measured both of them. Details of those datasets are presented in Table 3. Non-defective sensor time-series data are shown as "$\surd$", and sensor time-series data with defects are shown as "$(\surd)$". Apparently, a defect persisting for a short period is easier to repair by linear interpolation because some of the driving behavior changes linearly in a short period, for example when the driver turns the steering left with uniform angular velocity. We consider our proposed method to be effective for defects that persist for long periods. Therefore, we created defects of 100 frames (10 s) every 300 frames (30 s) on D12. In this way, we introduced 4620 defective frames in the sensor time-series data.

Table 3. Sensor time-series data included in eight prepared datasets.

Data Sets	Included Sensor Time-Series Data								
	I_1	I_2	I_3	I_4	I_5	I_6	I_7	I_8	I_9
C1	$(\surd)$	$\surd$	$\surd$	$\surd$	$\surd$	$\surd$	$\surd$	$\surd$	$\surd$
C2	$\surd$	$(\surd)$	$\surd$	$\surd$	$\surd$	$\surd$	$\surd$	$\surd$	$\surd$
C3	$\surd$	$\surd$	$(\surd)$	$\surd$	$\surd$	$\surd$	$\surd$	$\surd$	$\surd$
C4	$\surd$	$\surd$	$\surd$	$(\surd)$	$\surd$	$\surd$	$\surd$	$\surd$	$\surd$
C5	$\surd$	$\surd$	$\surd$	$\surd$	$(\surd)$	$\surd$	$\surd$	$\surd$	$\surd$
C6	$\surd$	$\surd$	$\surd$	$\surd$	$\surd$	$(\surd)$	$\surd$	$\surd$	$\surd$
C7	$\surd$	$\surd$	$\surd$	$\surd$	$\surd$	$\surd$	$(\surd)$	$(\surd)$	$\surd$
C8	$\surd$	$\surd$	$\surd$	$\surd$	$\surd$	$\surd$	$\surd$	$\surd$	$(\surd)$

We expected the different settings of the defect value to affect the model in different ways. For example, the defect values of -1 and 1 are the extremes for types of sensor time-series data of which the average is near 0 when the dataset is normalized to a range of $[-1, 1]$, such as in the case of the steering angle, longitudinal and lateral acceleration, and yaw rate. Meanwhile, there are types of sensor time-series data that include many values of -1, such as the accelerator opening rate and the brake master-cylinder pressure. For these cases, the defect value of -1 is less extreme than defect values of 1 and 0. Therefore, we set three types of defect values: 1, 0, -1.

5.2. Evaluation of Data Repair of Sensor Time-Series Data.

This subsection describes our verification and comparison of the abilities of DSAE-FP and DSAE-BP to repair defects. We iteratively updated DSAE-FP 500 times to repair defects. Meanwhile, DSAE-BP was updated 2000 times with a learning rate of $\psi = 0.01$ to repair the defects of each dataset in the series C1 to C8.

We calculated the square error between the updated defect value and the true value at each iterative cycle. Figures 10 and 11 compare the results of using DSAE-FP and DSAE-BP. To display the convergence process of the square error more clearly, Figures 10 and 11 only show the first 100 iterative cycles of the update for results of DSAE-FP. Note that the convergence speed is controlled by the learning rate when DSAE-BP is used, which means that a comparison of the convergence speed between DSAE-FP and DSAE-BP would have no significance. Therefore, we focused on the converged value of the square error. Figures 10 and 11 show that the square error of each dataset was reduced and converged by using DSAE-FP and DSAE-BP, irrespective of whether the initial defect value was 1, 0, or -1. Except for C4, for the other results obtained with DSAE-FP the square error of the initial defect value of 1, 0, and -1 converged to similar values. This indicated that DSAE-FP repaired the defect value to a convergence value, but DSAE-FP easily enabled the defect values to converge to different local converged values if the initial defect values were different, such as the result of C4. In addition, Figure 10 shows that the square errors of C3 and C4 decreased before increasing, and finally converged when the initial defect value was set to 0. This also showed that the converged value is not necessarily the minimum of the square error in those cases when using DSAE-FP. Meanwhile, when we used DSAE-BP, the different initial values of the defect value can also be reduced and converge to a similar value. For most of the results, the converged values of the square error were small when using DSAE-BP compared to when using DSAE-FP, especially for C4. This result is also shown in Figure 12, which shows the converged values of the squared error with a different initial value for each dataset when DSAE-FP and DSAE-BP were used. To visualize the data repaired by DSAE-FP and DSAE-BP, Figure 13 shows part of the repaired sensor time-series data of the steering angle in C3 and the ground truth thereof when the defect value was 1. It shows that the repaired sensor time-series data using DSAE-BP were closer to the ground truth than when using DSAE-FP. In summary, the above experimental results show that DSAE-BP is more effective than DSAE-FP for repairing defective sensor time-series data of driving behavior.

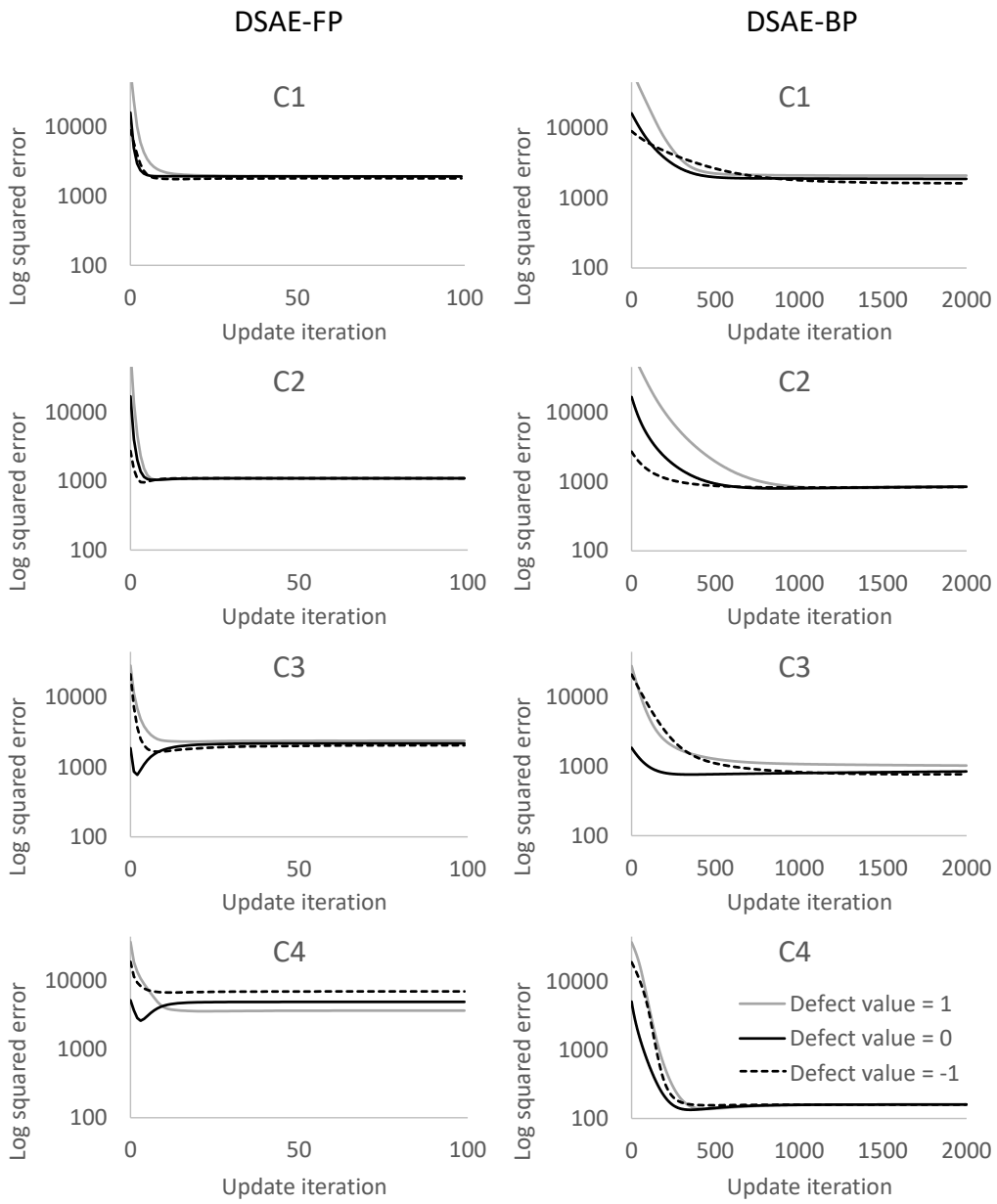

Figure 10. Square error between the repaired sensor time-series data and non-defective sensor time-series data for C1 to C4 at each update iteration. The plots in the columns on the left and right represent the results obtained by using DSAE-FP and DSAE-BP, respectively.

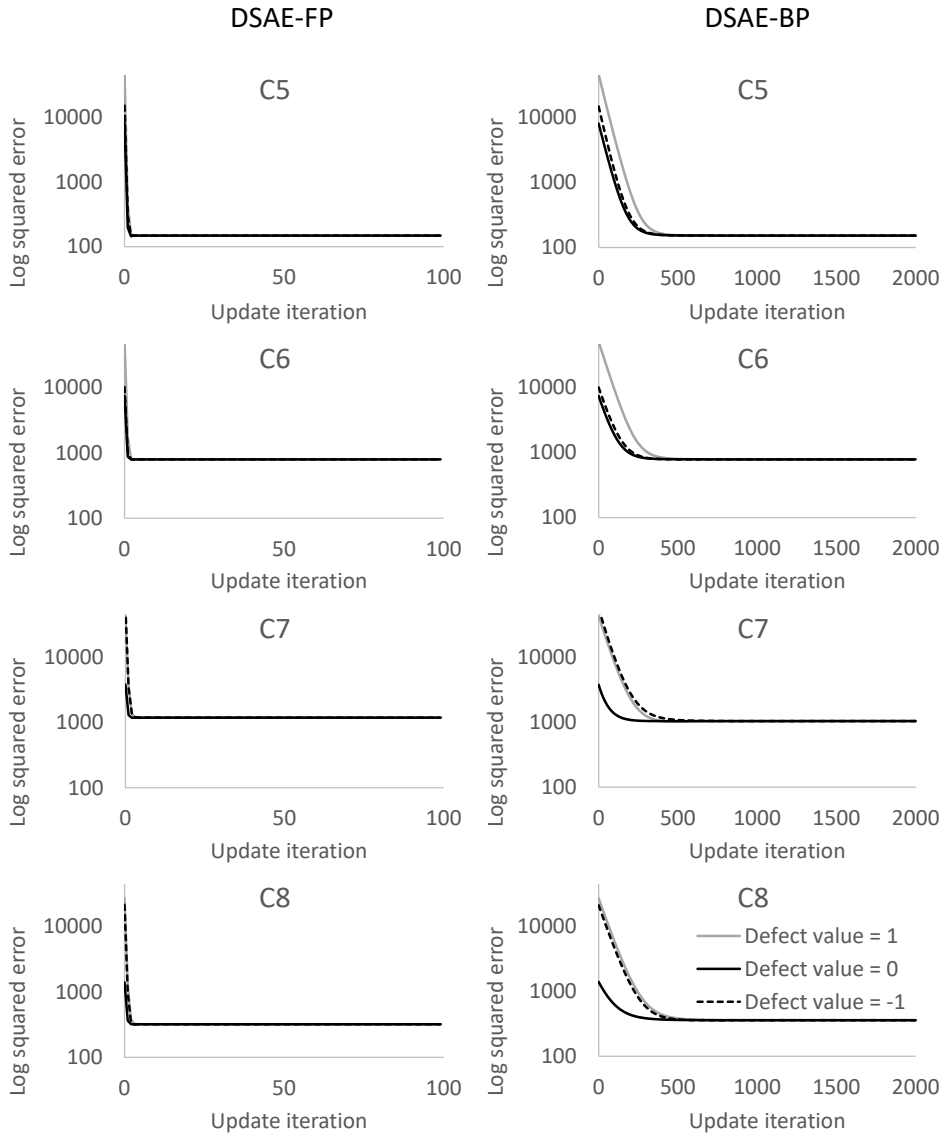

Figure 11. Square error between the repaired sensor time-series data and non-defective sensor time-series data for C5 to C8 at each update iteration. The plots in the columns on the left and right represent the results obtained by using DSAE-FP and DSAE-BP, respectively.

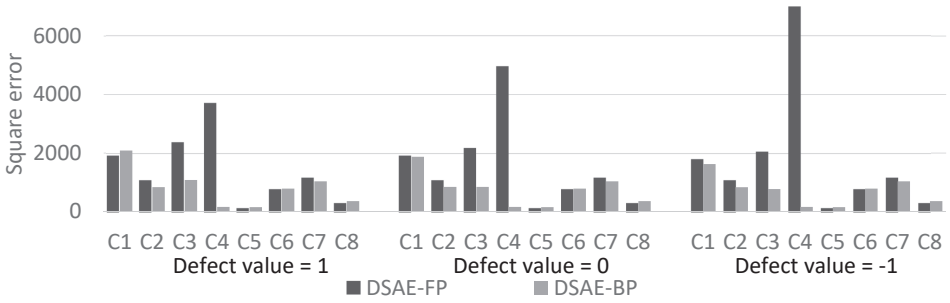

Figure 12. Convergence values of squared error with different initial value for each of the datasets when DSAE-FP and DSAE-BP were used.

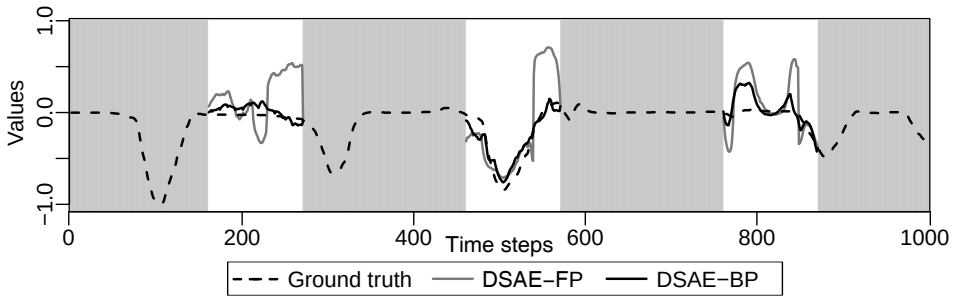

Figure 13. Example of defect repair of the steering angle by DSAE-FP and DSAE-BP for a part of C3 in the time-series, when the defect value was 1. A white background indicates the period of defects. The values in parentheses in the legend represent the defect values.

5.3. Evaluation of Feature Extraction with Defective Data

We focused on feature extraction from datasets that include defective sensor time-series data. Especially, we evaluated the similarity of the time-series of latent features extracted from a dataset including defective and non-defective sensor time-series data. In the first evaluation experiment, we used the feature extraction methods PCA and DSAE without data repairing to show the negative effects of defect data on them. Second, we used the typical defect repair methods, linear interpolation (LI) and median filter (MF) with a window size of 100, before applying PCA or DSAE as the baseline to reduce the negative effect of defects on feature extraction. Note that LI and MF do not depend on the defect value because they use the data before and after the interval of defects to repair defects alone. Third, we used our proposed method DSAE-BP to extract the latent features from defective and non-defective sensor time-series data. We also used PCA-FP, PCA-BP and DSAE-FP as comparative methods. When we used PCA-FP, the defective sensor information was updated by its reconstructed data via PCA by Equation (16). Meanwhile, PCA-BP used the same defect-repairing algorithm as DSAE-BP, to update the defective element $h_{t^{(1)},d}$ in the sensor time-series data via Equation (12), but $\frac{\partial E(\Phi)_t}{\partial \mathbf{h}_t^{(1)}} = (\mathbf{h}_t^{(1)} - \mathbf{h}_t^{(L)}) - (\mathbf{h}_t^{(1)} - \mathbf{h}_t^{(L)})(\mathbf{W}_E \mathbf{W}_E^T)$.

We consider C1 to C8 to be the defective datasets based on D12. Meanwhile, the time-series of latent features of C1 to C8 were extracted through a trained model by using D12 to ensure that the feature spaces of C1 to C8 and D12 were the same. This enabled us to directly calculate the square error between the time-series of latent features extracted from defective and non-defective sensor time-series data to evaluate the negative effect on the extracted time-series of latent features. However, the feature spaces generated by PCA and DSAE were different, particularly their scales. Thus, the use of the square error to directly evaluate PCA and DSAE would not be sensible. Therefore, we used the coefficient of determination (R^2) [32] to evaluate the latent feature extraction of time-series by PCA and DSAE with and without the defects. This is because we return this evaluation as a regression problem, which is the time-series of latent features of D12 as the response variable, C1 to C8 as the explanatory variable, and the time-series of latent features extracted from C1 to C8 as the predicted values. We also considered the periods without defects in defective sensor time-series data to be the same as the corresponding periods of data in non-defective time-series data. Therefore, the time-series of latent features extracted from those periods were also the same, because the transformations of latent feature extraction are the same. To evaluate the negative effects caused by the defects only, we defined a sub-dataset $S \subset H^{(m)}$, which included the extracted time-series of latent features in the periods of defects, where $S \in \mathbb{R}^{D_{H^{(m)}} \times N_S}$ and N_S is the number of frames of defects, i.e., 4620. We defined R^2 (see Appendix A), which can be regarded as an opposite manifestation of the square error normalized by the variance of $H^{(m)}$ of D12. When the value of R^2 closely approximates 1, the time-series of latent features extracted from defective and non-defective sensor time-series data are similar. Conversely, the similarity is poor when R^2 is reduced. Especially when the time-series of

latent features extracted from defective and non-defective sensor time-series data are very different, the numerator of the second term will be greater than the denominator in Equation (A1), thus R^2 will become a negative number.

The evaluation results of datasets C1 to C8 are presented in Table 4. The highest value of R^2 is presented in bold font and the second highest value is underlined for each defect value. First, to show the negative effects of defects on feature extraction, we used PCA and DSAE without defect repair to extract the latent features from defective and non-defective sensor time-series data. Table 4 shows that many of the values of R^2 were negative when the PCA and DSAE were used with defect values 1 and -1. When the defect value was 0, the values of R^2 were below 0.75 when PCA was used for C2 and DSAE was used for C1 and C2. This shows that the negative effect of defects on feature extraction was considerable. Second, we additionally used LI and MF before PCA and DSAE as the baseline to reduce the negative effect of defects on the feature extraction. Table 4 shows that all the values of R^2 were higher than the direct use of feature extraction when the LI and MF were used before the PCA and DSAE, regardless as to whether the defect value was 1, 0 or -1. Even though LI and MF could be used for this task, the values of R^2 for some of the datasets were below 0.9, e.g., C7 by LI and MF with PCA; C1 and C7 by LI with DSAE; and C1 and C4 by MF with DSAE. Third, we used our proposed method DSAE-BP and the comparative methods PCA-FP, PCA-BP, and DSAE-FP to extract the latent features from defective and non-defective sensor time-series data. Table 4 also shows that both the DSAE-FP and the DSAE-BP could update the defective sensor information to enable it to converge to a similar value, regardless of whether the defect value was selected as 1, 0, or -1. Meanwhile, most of the R^2 values of PCA-FP were negative. It is noteworthy that, when we evaluated R^2 of C7 and C8 with the defect values 1, 0, and -1 by using PCA-FP, the R^2 values became "–" that means the defective sensor information was diverged via update. That is, even though the FP method was used before the PCA, the effect of defects on the extracted latent features by PCA-FP was huge. DSAE-FP also used the FP method but its R^2 values were much higher than PCA-FP. Almost all of the R^2 values of DSAE-FP were higher than 0.9. However, for C4, the R^2 values of DSAE-FP were lower than 0.7 when the defect values were 1, 0, and -1. Table 4 shows that when we used PCA-BP and DSAE-BP, the R^2 values were higher than 0.95 for all the datasets and all the defect values. This shows that the use of the BP method to repair defective data was effective. In addition, the values presented in bold font in Table 4 shows that the highest value of R^2 was obtained when using DSAE-BP for most cases in this experiment, irrespective of whether the defect value was 1, 0 or -1.

Even if DSAE-BP did not achieve the highest value, its performance was the second best, as is evident from the underlined values in Table 4. In summary, the experimental results showed that the DSAE-BP can extract time-series of latent features that are similar to the real values when the sensor time-series data are defective. In other words, DSAE-BP can reduce the negative effect of defects on extracted time-series of latent features and it is superior to other comparative methods.

Table 4. R^2s between the extracted time-series of latent features of D12 and C1 to C8. The highest value of R^2 is presented in bold font; the second highest value is underlined for each defect value.

Defect Values	Mehods	C1	C2	C3	C4	C5	C6	C7	C8
1	LI+PCA	0.958	0.970	0.962	0.990	0.989	0.976	0.825	0.948
1	LI+DSAE	0.864	0.945	0.938	0.944	0.997	0.988	0.890	0.992
1	MF+PCA	0.950	0.952	0.957	0.951	0.945	0.976	0.893	0.960
1	MF+DSAE	0.855	0.916	0.933	0.725	0.989	0.990	0.924	0.995
1	PCA	0.581	−0.173	0.520	0.637	0.595	0.481	−0.269	0.174
1	DSAE	−0.233	−0.671	0.213	−0.791	0.916	0.729	0.579	0.874
1	PCA-FP	0.832	−1.01	−177	−0.858	−0.055	0.337	–	–
1	DSAE-FP	0.968	0.975	0.906	0.750	**1.00**	**0.997**	0.987	**0.999**
1	PCA-BP	**0.985**	0.975	**0.987**	**0.998**	0.996	0.989	0.953	0.986
1	DSAE-BP	0.964	**0.983**	0.974	0.992	**1.00**	**0.997**	**0.990**	**0.999**
0	LI+PCA	0.958	0.970	0.962	0.990	0.989	0.976	0.825	0.948
0	LI+DSAE	0.864	0.945	0.938	0.944	0.997	0.988	0.890	0.992
0	MF+PCA	0.950	0.952	0.957	0.951	0.945	0.976	0.893	0.960
0	MF+DSAE	0.855	0.916	0.933	0.725	0.989	0.990	0.924	0.995
0	PCA	0.904	0.745	0.970	0.949	0.930	0.925	0.912	0.961
0	DSAE	0.641	0.553	0.955	0.692	0.986	0.960	0.947	0.995
0	PCA-FP	0.832	−1.01	−177	−0.858	−0.055	0.337	–	–
0	DSAE-FP	0.968	0.975	0.916	0.653	**1.00**	**0.997**	0.987	**0.999**
0	PCA-BP	**0.985**	0.975	**0.987**	**0.998**	0.996	0.989	0.953	0.986
0	DSAE-BP	0.969	**0.983**	0.981	0.992	**1.00**	**0.997**	**0.990**	**0.999**
−1	LI+PCA	0.958	0.970	0.962	0.990	0.989	0.976	0.825	0.948
−1	LI+DSAE	0.864	0.945	0.938	0.944	0.997	0.988	0.890	0.992
−1	MF+PCA	0.950	0.952	0.957	0.951	0.945	0.976	0.893	0.960
−1	MF+DSAE	0.855	0.916	0.933	0.725	0.989	0.990	0.924	0.995
−1	PCA	0.949	0.961	0.642	0.814	0.870	0.901	−0.581	0.351
−1	DSAE	0.862	0.930	0.385	−0.080	0.976	0.954	0.430	0.904
−1	PCA-FP	0.832	−1.01	−177	−0.858	−0.055	0.337	–	–
−1	DSAE-FP	0.970	0.975	0.921	0.489	**1.00**	**0.997**	0.987	**0.999**
−1	PCA-BP	**0.985**	0.975	**0.987**	**0.998**	0.996	0.989	0.953	0.986
−1	DSAE-BP	0.975	**0.983**	0.982	0.992	**1.00**	**0.997**	**0.990**	**0.999**

6. Application: Driving Behavior Segmentation with Defects

After verifying that DSAE-BP can reduce the negative effect on feature extraction, we show an example of its application to capture transitions in patterns of driving behavior and segment time-series data into segments. We either considered the defects in the time-series data block or altered the transitions in the patterns of driving behavior. Therefore, defects adversely affect the driving behavior segmentation to a large extent. We propose to use DSAE-BP, which is expected to ensure that the segmentation results of defective and non-defective sensor time-series data are similar.

In this experiment, we paired the time-series of latent features extracted from each dataset of C1 to C8 (including defects) with dataset D12 (that does not include defects). To show the effectiveness of the feature extraction method with the different defect values, 1, 0, −1, we also used the nine-dimensional normalized raw data (RAW) of C1 to C8 and D12 without the windowing proses to perform the segmentation task as a baseline. Meanwhile, we also used PCA, DSAE, PCA-BP, and DSASE-FP as comparative methods. Note that, we excluded PCA-FP as a comparison method because of its poor ability to extract latent features with defects in previous experiments. To segment the time-series of driving behaviors, we employed a sticky hierarchical Dirichlet process hidden Markov model (sticky HDP-HMM) [33], which was used for segmenting driving behavior data in previous studies [8,21,34,35]. The RAW data and time-series of latent features extracted from D12 were used to train the sticky HDP-HMM. Then the trained sticky HDP-HMMs were used to estimate the change points of segments from C1 to C8. Finally, we defined a segmentation distance (see Appendix B) to evaluate the similarity

of segmentation between defective and non-defective sensor time-series data. If the segmentation distance is short, it means two segmentation results are similar.

In this experiment, we examined the segmentation distance between D12 and C1 to C8. We set the parameter w of the time window to four, which means the range of the time window included nine frames. We evaluated the segmentation distance for each dataset via each method 10 times because sticky HDP-HMM is a probability model that samples the boundaries of segments via a random process. The results of the average segmentation distance are presented in Table 5. The shortest average segmentation distance (best performance) is shown in bold font; the second shortest average segmentation distance is shown in underlined.

Table 5. Average segmentation distances between D12 and C1 to C8 with defect values of 1, 0, and −1. The shortest average segmentation distance (best performance) is shown in bold font; the second shortest average segmentation distance is shown in underlined.

Defect Value	Methods	C1	C2	C3	C4	C5	C6	C7	C8
1	RAW	381	524	578	402	358	321	312	338
1	PCA	306	408	295	300	304	322	342	326
1	DSAE	351	424	332	344	226	288	325	277
1	PCA-BP	153	167	**98.4**	**87.9**	98.4	151	187	94.3
1	DSAE-FP	132	149	196	235	50.4	89.5	143	59.1
1	DSAE-BP	**126**	**125**	133	142	**46.3**	**81.8**	**132**	**56.5**
0	RAW	325	437	166	321	224	210	108	117
0	PCA	252	329	93.1	199	207	237	231	113
0	DSAE	310	352	126	281	137	213	196	72.6
0	PCA-BP	155	171	**92.3**	**90.0**	94.5	145	194	91.7
0	DSAE-FP	132	155	233	328	46.3	89.1	146	**56.4**
0	DSAE-BP	**123**	**123**	141	142	**43.6**	**84.2**	**132**	58.8
−1	RAW	158	177	352	347	260	203	344	317
−1	PCA	**116**	174	288	263	218	249	342	316
−1	DSAE	137	181	322	377	118	213	332	275
−1	PCA-BP	152	170	**93.5**	**88.3**	95.3	147	189	91.2
−1	DSAE-FP	131	154	228	305	44.1	88.1	144	57.0
−1	DSAE-BP	123	**125**	144	142	**43.8**	**82.9**	**129**	**55.7**

First, we focused on the performance of FP and BP with PCA and DSAE on different defect values. Table 5 shows that regardless of whether the defect value was set to 1, 0, or −1, the average segmentation distances of each dataset processed by using PCA-BP, DSAE-FP and DSAE-BP are more similar to each other than by using other comparative methods. Here, we take C1 as an example. The average segmentation distances of C1 by PCA-BP (153, 155, 152), DSAE-FP (132, 132, 131) and DSAE-BP (126, 123, 123) were more similar to each other than those by RAW (381, 325, 158), PCA (306, 252, 116), and DSAE (351, 310, 137) when the defect values were 1, 0, and −1. The reason is that the square error of each dataset with defect values of 1, 0, and −1 converged to a similar value when FP and BP were used with PCA and DSAE, which can be seen in Figures 10 and 11 for DSAE-FP and DSAE-BP. However, this conclusion does not apply to C4 by using DSAE-FP (235, 328, 305) because the converged values of the square error were not similar by using DSAE-FP for C4 with the initial defect values of 1, 0, and −1 in the Section 5.2.

Next, we compared the performance of DSAE-BP with other comparable methods on different defect values, respectively. The results in Table 5 show that, when the defect value was set to 1, the average segmentation distance between the defective and non-defective sensor time-series data were shortest by using DSAE-BP with DSAE-FP in the second place, except C3 and C4. PCA-BP obtained the best performance on C3 and C4 and DSAE-BP got the second place. Meanwhile, when the defect value was set to 0, DSAE-BP was most effective in terms of achieving the shortest average

segmentation distances between D12 and most of datasets C1 to C8. However, for C3, the result was poor in comparison. This reason is that the sensor time-series data of the steering angle in dataset C3 are defective. The measured sensor time-series data of steering angle included many values of 0, especially in the case of straight-line driving. Therefore, if the defect value of the steering angle was set to 0, then this defect value has a high probability to be equal to the true value. Moreover, when the defect value was set to -1, DSAE-BP achieved the shortest average segmentation distances for C2, C5, C6, C7, and C8, and it also achieved the second shortest average segmentation distances for C1, C3, and C4.

The above results showed that PCA-BP, DSAE-FP, and DSAE-BP have the ability to reduce the negative effect for driving behavior segmentation. Note that, DSAE-BP still outperformed PCA-BP because DSAE-BP could extract the latent features of time-series from defective that were more closer to the latent features extracted from non-defective sensor time-series data than when using PCA-BP. Meanwhile, DSAE-BP outperformed DSAE-FP because DSAE-FP easily enables the defect values to converge to fixed points but it does not guarantee that the fixed points are local optima.

7. Discussion for Advantages and Limitations of Proposed Method

In this study, we propose using DSAE-BP, that based on a trained DSAE, to repair the defective driving behavior data. The advantage of the proposed approach is that DSAE model is trained only one time and two tasks can be completed – latent feature extraction and defective data repairing. The representation of latent features is critical to both tasks. If the latent features of DSAE are difficult to represent one of the driving behaviors, then DSAE-BP will have trouble repairing the defective data when such driving behavior occurs. It is an important topic that is how to design a DSAE that can extract the latent features well. For now, optimizing the structure of the neural network remains an open question. In this study, we still use our experience from our previous study to design the structure of DSAE. In our previous study, we verified the usability of the designed DSAE model by a visualization method [22]. Although it is not sufficient to verify this structure is the best, the experimental results show that it is effective.

In addition, DSAE can be used for a variety of tasks as the pre-training such as driving behavior classification. We repaired the defective data by minimizing the square error in the experiment. However, in the view point of driving behavior classification, square error does not guarantee that the repaired data will not vibrate. For example, the right side of the period of defects in Figure 13 shows that the repaired sensor information of steering angle by DSAE-BP vibrated up and down on the ground truth. If this kind of repaired sensor information will use to classify driving behaviors, then it would lead to misclassification. For this problem, we suggest that DSAE can be fine-tuned for different tasks.

8. Conclusions

We propose a method for extracting low-dimensional time-series of latent features from multi-dimensional driving behavior data using a DSAE. In addition, the DSAE can also repair the defects in sensor time-series data in combination with back propagation (DSAE-BP). In the first experiment, we show that DSAE could extract highly correlated low-dimensional time-series of latent features by reducing the redundancy to various degrees in different multi-dimensional time-series data of driving behavior. Furthermore, we verify that DSAE-BP could repair the defective sensor time-series data and reduce the negative effect of defects on the extracted time-series of latent features in second experiment. We also show in the third experiment that, based on the latent features extracted by DSAE-BP, the negative effect of defects on the driving behavior segmentation task could be reduced.

The proposed defect-repairable latent feature extraction method has a wide range of applications, e.g., the prediction, visualization, segmentation, and estimation of the latent structure of driving behaviors. Many types of data-driven driving behavior analysis methods have been proposed. Most of them benefited from low-dimensional and informative feature vectors. First, by reducing

the dimension of feature vectors by retaining the information contained within the time-series data, we can reduce the computational cost of the post-process, e.g., prediction or segmentation. Second, appropriate feature vectors also increase the generalization performance of the post-process. We show that our proposed method can reduce the dimensionality by retaining the contained information and reducing the redundancy (see Experiment 1). Practically, feature extraction needs to be robust to defects and outliers for its application. Our method can reduce the adverse effect of defects significantly (see Experiment 2). These favorable functions will increase the performance of various applications including driving behavior segmentation (see Experiment 3). We believe that the method has an impact on data-driven driving behavior analyses and contributes to the safety of driving.

In this study, DSAE completed two tasks when only trained once: (1) extracting time-series of latent features; and (2) repairing defects in sensor time-series data by using a back-propagation method. The gradient information for DSAE-BP may also be used to detect the defects of driving behaviors. Based on this idea, it may be possible to detect the occurrence area of defects. However, it is difficult to accurately determine which of the measured sensor information is defective. The reason for this problem is that a variety of sensor information has been fused in the middle layer of the DSAE. How to solve this problem is still an open question. In addition, other unsupervised feature extraction methods would be compared with DSAE such as variational autoencoder [36] and long short term memory [37] for extracting latent features of driving behaviors in the future.

Acknowledgments: This work was supported by JSPS KAKENHI Grant Number JP16J08577. This work was also partially supported by a Grant-in-Aid for Scientific Research on Innovative Areas (16H06569) funded by the Ministry of Education, Culture, Sports, Science, and Technology, Japan.

Author Contributions: Hailong Liu, Tadahiro Taniguchi, Kazuhito Takenaka and Takashi Bando conceived and designed the experiments; Hailong Liu performed the experiments; Hailong Liu and Tadahiro Takenaka analyzed the data; Kazuhito Takenaka and Takashi Bando contributed data; Hailong Liu wrote the paper.

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

Appendix A. Similarity between Two Extracted Time-Series of Latent Features via the Same Trained Model

To evaluate the similarity of time-series of latent features extracted from defective and non-defective sensor time-series data, we defined R^2, which can be regarded as an opposite manifestation of the square error normalized by the variance of time-series of latent features $H^{(m)}$ of D12 (not including defects):

$$R^2 = 1 - \frac{\frac{1}{N_S}\sum_{i=1}^{N_S}(\mathbf{s}_i - \tilde{\mathbf{s}}_i)^T(\mathbf{s}_i - \tilde{\mathbf{s}}_i)}{\sigma^2}, \tag{A1}$$

where $\mathbf{s}_i$ is the i-th vector of the sub-dataset S extracted from the sub-dataset of D12, and $\tilde{\mathbf{s}}_i$ is the i-th vector of the sub-dataset S extracted from C1 to C8 (the sub-dataset $S \subset H^{(m)}$, which included the extracted time-series of latent features in the periods of defects). $\sigma^2 \in \mathbb{R}$ is the variance of $H^{(m)}$ of D12:

$$\sigma^2 = \frac{1}{N_V}\sum_{j=1}^{N_V}(\mathbf{h}_j^{(m)} - \bar{\mathbf{h}}^{(m)})^T(\mathbf{h}_j^{(m)} - \bar{\mathbf{h}}^{(m)}), \tag{A2}$$

where $\bar{\mathbf{h}}^{(m)}$ is a mean vector of $\mathbf{H}^{(m)}$. When the value of R^2 is close to 1, it means that the time-series of latent features extracted from defective and non-defective sensor time-series data are similar. Conversely, the similarity is poor when R^2 is reduced. Especially when the time-series of latent features are extracted from defective and non-defective sensor time-series the data are very different, and then the numerator of the second term will be greater than the denominator in Equation (A1), hence R^2 will become a negative number.

Appendix B. Similarity between Two Segment Results

To evaluate the similarity of the results of segmenting RAW data or extracted time-series of latent features from defective and non-defective sensor time-series data, we introduce an evaluative method in this subsection.

For example, $\mathbf{H}^{(m)} \in \mathbb{R}^{D_{\mathbf{H}^{(m)}} \times N_V}$ is a matrix of the time-series of latent features extracted from non-defective sensor time-series data, and $\tilde{\mathbf{H}}^{(m)} \in \mathbb{R}^{D_{\mathbf{H}^{(m)}} \times N_V}$ is a matrix of the time-series of latent features extracted from sensor time-series data with defects. We used a sticky HDP-HMM trained by using $\mathbf{H}^{(m)}$ (non-defective) to infer their respective hidden states and sample the boundaries of segments many times. $\mathbf{u}^{(l)} = (u_1^{(l)}, u_2^{(l)}, \cdots, u_k^{(l)}) \in \mathbb{R}^k$ is a binary vector that shows the boundaries of segments sampled from $\mathbf{H}^{(m)}$ in l-time of sampling, where $k = N_V - 1$ is the number of intervals. If the hidden state of a sticky HDP-HMM changes at the i-th interval, then $u_i^{(l)} = 1$; else, $u_i^{(l)} = 0$. The vector of the change point of $\tilde{\mathbf{H}}^{(m)}$ in l-time of sampling is shown as $\mathbf{v}^{(l)}$, which is calculated as well as $\mathbf{u}^{(l)}$.

Next, we calculate the segment probability by counting the boundaries of a segment at each interval by sampling many times. A vector of the segment probability can be shown by $\mathbf{p} = (p_1, \cdots, p_m) \in \mathbb{R}^m$ for $\mathbf{H}^{(m)}$, which is calculated by

$$\mathbf{p} = \frac{1}{K} \sum_{j=1}^{K} \mathbf{u}^{(j)}, \tag{A3}$$

where K is the number of samples. Similarly, the vector of the segment probability for $\tilde{\mathbf{H}}^{(m)}$ is represented by $\mathbf{q}$, which is calculated by

$$\mathbf{q} = \frac{1}{K} \sum_{j=1}^{K} \mathbf{v}^{(j)}. \tag{A4}$$

Finally, we evaluate the similarity of the segmentation for $\mathbf{H}^{(m)}$ and $\tilde{\mathbf{H}}^{(m)}$ by comparing $\mathbf{p}$ and $\mathbf{q}$. We consider each sampling time to be a random process. Even if the real probability of the segment at the i-th interval is very high, a boundary of segment $u_i = 1$ was not necessarily sampled at this interval. Therefore, we evaluate the similarity of $\mathbf{H}^{(m)}$ and $\tilde{\mathbf{H}}^{(m)}$ by using the range of the segment probability in $[i - w, i + w]$. In particular, we calculate the average of the segment probability $\bar{p}_i$ and $\bar{q}_i$ in the range by

$$\bar{p}_i = \frac{1}{2w+1} \sum_{a=i-w}^{i+w} p_a, \tag{A5}$$

$$\bar{q}_i = \frac{1}{2w+1} \sum_{a=i-w}^{i+w} q_a, \tag{A6}$$

where w is a time window and $w < i < n - w$. Finally, we define the segmentation distance between $\mathbf{H}^{(m)}$ and $\tilde{\mathbf{H}}^{(m)}$ based on the segment probability:

$$\begin{aligned} D(\bar{\mathbf{p}}, \bar{\mathbf{q}}) &= \frac{1}{\sqrt{2}} \sum_{i=w+1}^{k-w} \sqrt{(\bar{p}_i - \bar{q}_i)^2 + \left((1 - \bar{p}_i) - (1 - \bar{q}_i)\right)^2} \\ &= \frac{1}{\sqrt{2}} \sum_{i=w+1}^{k-w} \sqrt{(\bar{p}_i - \bar{q}_i)^2 + (\bar{q}_i - \bar{p}_i)^2} \\ &= \frac{1}{\sqrt{2}} \sum_{i=w+1}^{k-w} \sqrt{2} |\bar{p}_i - \bar{q}_i \\ &= \sum_{i=w+1}^{k-w} |\bar{p}_i - \bar{q}_i| \end{aligned} \tag{A7}$$

If the segmentation distance is short, it means two segmentation results are similar.

References

1. Tagawa, T.; Tadokoro, Y.; Yairi, T. Structured denoising autoencoder for fault detection and analysis. In Proceedings of the Asian Conference on Machine Learning, Hong Kong, China, 20–22 November 2015; pp. 96–111.
2. Krishnaswami, V.; Luh, G.; Rizzoni, G. Nonlinear parity equation based residual generation for diagnosis of automotive engine faults. *Control Eng. Pract.* **1995**, *3*, 1385–1392.
3. Malhotra, P.; Vig, L.; Shroff, G.; Agarwal, P. Long Short Term Memory Networks for Anomaly Detection in Time Series. In Proceedings of the European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning, Bruges, Belgium, 22–24 April 2015; pp. 89–94.
4. Wang, Y.; Yao, H.; Zhao, S. Auto-encoder based dimensionality reduction. *Neurocomputing* **2016**, *184*, 232–242.
5. Liu, H.; Taniguchi, T.; Tanaka, Y.; Takenaka, K.; Bando, T. Essential Feature Extraction of Driving Behavior Using a Deep Learning Method. In Proceedings of the IEEE Intelligent Vehicles Symposium 2015, Seoul, Korea, 28 June–1 July 2015; pp. 1054–1060.
6. Liu, H.; Taniguchi, T.; Takenaka, K.; Tanaka, Y.; Bando, T. Reducing the Negative Effect of Defective Data on Driving Behavior Segmentation via a Deep Sparse Autoencoder. In Proceedings of the 5th IEEE Global Conference on Consumer Electronics, Kyoto, Japan, 11–14 October 2016; pp. 114–118.
7. Takano, W.; Matsushita, A.; Iwao, K.; Nakamura, Y. Recognition of human driving behaviors based on stochastic symbolization of time series signal. In Proceedings of the IROS 2008 IEEE/RSJ International Conference on Intelligent Robots and Systems, Nice, France, 22–26 September 2008; pp. 167–172.
8. Taniguchi, T.; Nagasaka, S.; Hitomi, K.; Chandrasiri, N.P.; Bando, T. Semiotic prediction of driving behavior using unsupervised double articulation analyzer. In Proceedings of the IEEE Intelligent Vehicles Symposium 2012, Madrid, Spain, 3–7 June 2012; pp. 849–854.
9. Bando, T.; Takenaka, K.; Nagasaka, S.; Taniguchi, T. Generating contextual description from driving behavioral data. In Proceedings of the 2014 IEEE Intelligent Vehicles Symposium, Dearborn, MI, USA, 8–11 June 2014; pp. 183–189.
10. Li, Z.; Jin, X.; Zhao, X. Drunk driving detection based on classification of multivariate time series. *J. Saf. Res.* **2015**, *54*, 61–67.
11. Wold, S.; Esbensen, K.; Geladi, P. Principal component analysis. *Chemom. Intell. Lab. Syst.* **1987**, *2*, 37–52.
12. Chatzigiannakis, V.; Grammatikou, M.; Papavassiliou, S. Extending driver's horizon through comprehensive incident detection in vehicular networks. *IEEE Trans. Veh. Technol.* **2007**, *56*, 3256–3265.
13. Calderó-Bardají, P.; Longfei, X.; Jaschke, S.; Reermann, J.; Mideska, K.; Schmidt, G.; Deuschl, G.; Muthuraman, M. Detection of steering direction using EEG recordings based on sample entropy and time-frequency analysis. In Proceedings of the 2016 IEEE 38th Annual International Conference of the Engineering in Medicine and Biology Society (EMBC), Orlando, FL, USA, 16–20 August 2016; pp. 833–836.
14. Lin, C.T.; Liang, S.F.; Chen, Y.C.; Hsu, Y.C.; Ko, L.W. Driver's drowsiness estimation by combining EEG signal analysis and ICA-based fuzzy neural networks. In Proceedings of the 2006 IEEE International Symposium on Circuits and Systems, Kos, Greece, 21–24 May 2006; pp. 2125–2128.
15. Hyvärinen, A.; Karhunen, J.; Oja, E. *Independent Component Analysis*; John Wiley & Sons: Hoboken, NJ, USA, 2004; Volume 46.
16. Hyvärinen, A.; Oja, E. Independent component analysis: Algorithms and applications. *Neural Netw.* **2000**, *13*, 411–430.
17. Schölkopf, B.; Smola, A.; Müller, K.R. Kernel principal component analysis. In Proceedings of the International Conference on Artificial Neural Networks, Lausanne, Switzerland, 8–10 October 1997; Springer: Berlin/Heidelberg, Germany, 1997; pp. 583–588.
18. Zhao, C.; Zheng, C.; Zhao, M. Classifying driving mental fatigue based on multivariate autoregressive models and kernel learning algorithms. In Proceedings of the 2010 3rd International Conference on Biomedical Engineering and Informatics, Yantai, China, 16–18 October 2010; Volume 6, pp. 2330–2334.
19. Dong, W.; Yuan, T.; Yang, K.; Li, C.; Zhang, S. Autoencoder Regularized Network For Driving Style Representation Learning. *arXiv* **2017**, arXiv:1701.01272.

20. Hori, C.; Watanabe, S.; Hori, T.; Harsham, B.A.; Hershey, J.; Koji, Y.; Fujii, Y.; Furumoto, Y. Driver confusion status detection using recurrent neural networks. In Proceedings of the 2016 IEEE International Conference on Multimedia and Expo (ICME), Seattle, WA, USA, 11–15 July 2016; pp. 1–6.
21. Taniguchi, T.; Nakashima, R.; Liu, H.; Nagasaka, S. Double articulation analyzer with deep sparse autoencoder for unsupervised word discovery from speech signals. *Adv. Robot.* **2016**, *30*, 770–783.
22. Liu, H.; Taniguchi, T.; Tanaka, Y.; Takenaka, K.; Bando, T. Visualization of driving behavior based on hidden feature extraction by using deep learning. *IEEE Trans. Intell. Transport. Syst.* **2017**, *18*, 2477–2489.
23. Kaneda, Y.; Irizuki, Y.; Yamakita, M. Design method of robust Kalman filter via L1 regression and its application for vehicle control with outliers. In Proceedings of the 38th Annual Conference on IEEE Industrial Electronics Society 2012, Montreal, QC, Canada, 25–28 October 2012; pp. 2222–2227.
24. Jang, J. Outlier filtering algorithm for travel time estimation using dedicated short-range communications probes on rural highways. *IET Intell. Transp. Syst.* **2016**, *10*, 453–460.
25. Duan, Y.; Lv, Y.; Kang, W.; Zhao, Y. A deep learning based approach for traffic data imputation. In Proceedings of the 17th International IEEE Conference on Intelligent Transportation Systems, Qingdao, China, 8–11 October 2014; pp. 912–917.
26. Ku, W.C.; Jagadeesh, G.R.; Prakash, A.; Srikanthan, T. A clustering-based approach for data-driven imputation of missing traffic data. In Proceedings of the IEEE Forum on Integrated and Sustainable Transportation Systems 2016, Beijing, China, 10–12 July 2016; pp. 1–6.
27. Karlik, B.; Olgac, A.V. Performance analysis of various activation functions in generalized MLP architectures of neural networks. *Int. J. Artif. Intell. Expert Syst.* **2011**, *1*, 111–122.
28. Ng, A. *Sparse Autoencoder*; CS294A Lecture Notes; Stanford University: Stanford, CA, USA, 2011; pp. 1–19.
29. Rumelhart, D.E.; Hintont, G.E.; Williams, R.J. Learning representations by back-propagating errors. *Nature* **1986**, *323*, 533–536.
30. Magdon-Ismail, M.; Boutsidis, C. Optimal sparse linear auto-encoders and sparse pca. *arXiv* **2015**, arXiv:1502.06626.
31. Hardoon, D.; Szedmak, S.; Shawe-Taylor, J. Canonical correlation analysis: An overview with application to learning methods. *Neural Comput.* **2004**, *16*, 2639–2664.
32. Triacca, U.; Volodin, A. Few remarks on the geometry of the uncentered coefficient of determination. *Lobachevskii J. Math.* **2012**, *33*, 284–292.
33. Fox, E.B.; Sudderth, E.B.; Jordan, M.I.; Willsky, A.S. A sticky HDP-HMM with application to speaker diarization. *Ann. Appl. Stat.* **2009**, *5*, 1020–1056.
34. Nagasaka, S.; Taniguchi, T.; Hitomi, K.; Takenaka, K.; Bando, T. Prediction of Next Contextual Changing Point of Driving Behavior Using Unsupervised Bayesian Double Articulation Analyzer. In Proceedings of the 2014 IEEE Intelligent Vehicles Symposium 2014, Dearborn, MI, USA, 8–11 June 2014; pp. 924–931.
35. Mori, M.; Takenaka, K.; Bando, T.; Taniguchi, T.; Miyajima, C.; Takeda, K. Automatic lane change extraction based on temporal patterns of symbolized driving behavioral data. In Proceedings of the 2015 IEEE Intelligent Vehicles Symposium (IV), Seoul, Korea, 28 June–1 July 2015; pp. 976–981.
36. Kingma, D.P.; Welling, M. Auto-encoding variational bayes. *arXiv* **2013**, arXiv:1312.6114.
37. Sak, H.; Senior, A.; Beaufays, F. Long short-term memory recurrent neural network architectures for large scale acoustic modeling. In Proceedings of the Fifteenth Annual Conference of the International Speech Communication Association, Singapore, 14–18 September 2014.