mathematics

MDPI

Article

Functional Subspace Variational Autoencoder for Domain-Adaptive Fault Diagnosis

Tan Li [1,*], Che-Heng Fung [1], Him-Ting Wong [1], Tak-Lam Chan [1] and Haibo Hu [1,2]

[1] Centre for Advances in Reliability and Safety (CAiRS), Hong Kong, China
[2] Department of Electronic and Information Engineering, Hong Kong Polytechnic University, Hong Kong, China
* Correspondence: tan.li@cairs.hk

Abstract: This paper presents the functional subspace variational autoencoder, a technique addressing challenges in sensor data analysis in transportation systems, notably the misalignment of time series data and a lack of labeled data. Our technique converts vectorial data into functional data, which captures continuous temporal dynamics instead of discrete data that consist of separate observations. This conversion reduces data dimensions for machine learning tasks in fault diagnosis and facilitates the efficient removal of misalignment. The variational autoencoder identifies trends and anomalies in the data and employs a domain adaptation method to associate learned representations between labeled and unlabeled datasets. We validate the technique's effectiveness using synthetic and real-world transportation data, providing valuable insights for transportation infrastructure reliability monitoring.

Keywords: functional data analysis; variational autoencoder; domain adaptation; reliability

MSC: 37M10

Citation: Li, T.; Fung, C.-H.; Wong, H.-T.; Chan, T.-K.; Hu, H. Functional Subspace Variational Autoencoder for Domain-Adaptive Fault Diagnosis. *Mathematics* **2023**, *11*, 2910. https://doi.org/10.3390/math11132910

Academic Editors: Rui Peng, Kaiye Gao and Wen Zhang

Received: 6 June 2023
Revised: 26 June 2023
Accepted: 27 June 2023
Published: 28 June 2023

1. Introduction

The use of data-driven maintenance strategies in sophisticated intelligent transportation systems, which make use of IoT-enabled resources, has gained considerable attention both industry and academia [1]. This is achieved by integrating embedded sensors that collect operational data from the systems being monitored, making it possible to constantly identify or predict any damage to the system. The commonly used approach is to install many interconnected sensors at fixed locations to gather information about facility operations. These sensors are synchronized through a global reference clock, which allows for the simultaneous timestamping of digitized sensor data from different locations. Consequently, conventional machine learning methods can be employed to extract valuable insights through time series data analysis at each location and spatial cross-correlation analysis between different locations [2,3].

In the context of transportation system monitoring, this study focuses on a unique situation, where sensors are deployed on mobile assets, such as vehicles. The vehicle's trajectory is segmented into discrete anchor points, with the path segment between two successive anchor points forming a basic unit for spatial indexing shown in Figure 1. In a path consisting of L segments containing time series data, one can extract data vectors from each segment for further analysis. However, due to uncontrollable motion precision, the time series sensor data from different vehicle journeys will exhibit diverse phase differences or timing misalignment [4]. This issue is further complicated by inconsistencies in the sampling clock rate during each journey. Without an inherent standard for aligning time series data, it becomes a considerable challenge to construct reliable training vectors of time series data, which are crucial for developing machine learning models for monitoring transportation systems.

The process of annotating IoT sensor data in transportation systems also poses a substantial challenge for fully exploiting the capabilities of machine learning models. As the transportation sector increasingly adopts IoT sensor-based monitoring systems, an immense volume of data is generated [5]. While these data have the potential to significantly enhance the training of machine learning models, the scarcity of labeled examples constitutes a major impediment in developing data-driven solutions for these systems [6]. The difficulties stem from various factors, including privacy concerns, disputes over data ownership, the expenses associated with annotation, and the inherent dynamics of transportation systems.

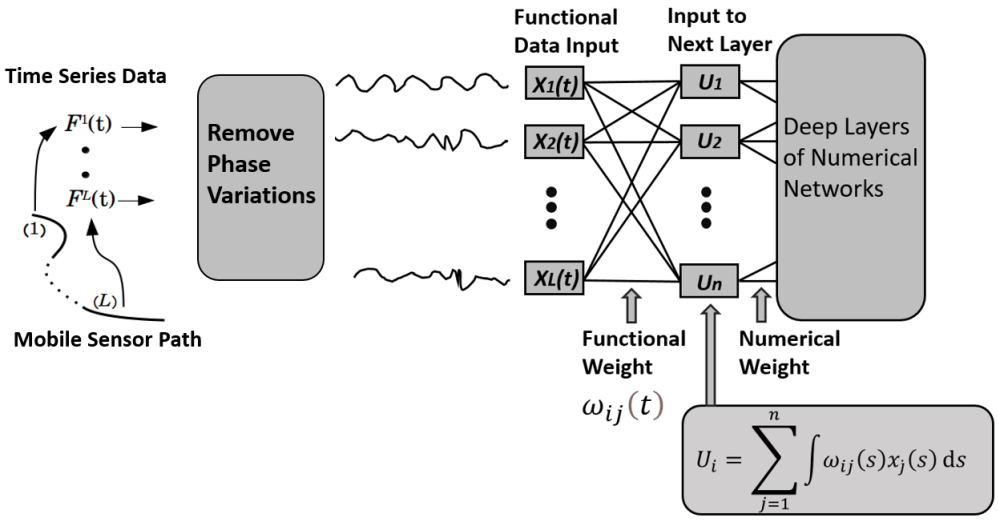

Figure 1. Overview of computation process. From left to right: Raw time series data from different segments are pre-processed to remove timing misalignment (phase variations). The connection between the 1st and the 2nd layer functional weight $\omega(t)_{ij}$, where $i = 1...L$ and $j = 1...n$. The numerical value of u_i, where $i = 1...n$ in the 2nd layer, is the summation of the integration of corresponding functional weights and all the functional input data. The deep layers of numerical networks can take the form of various network architectures, such as feedforward networks and convolutional networks, enabling them to perform clustering and classification of the functional input data.

The current work proposes a novel solution to tackle the challenge of learning from limited labeled data in various domains and resolves data quality issues that arise in real-world data. The challenge of creating an efficient domain adaptation strategy in the current research lies in the absence of complete correlations among latent features across various domains. As shown in Figure 2, our approach address this issue by using split latent subspace encoders in the architecture of a variational autoencoder (VAE) [7,8] and integrating a domain adaptation strategy [9,10] between the source and target domains. Specifically, we associate one VAE with the source domain and another with the target domain. The domain adaptation strategy employs the latent space alignment technique to extend the classification model trained on the source domain, using labeled synthetic data, to the target domain of unlabeled real-world data. To handle the issue of time series misalignment in real-world data that can potentially impact data analysis and machine learning processes, we use the functional data analysis (FDA) technique [11,12]. This technique considers sensor data time series as a sequence of smooth curves or functional data in Hilbert space, assuming that the sensor data points are causally related and behave as samples of smooth functions. This data model of presenting vectorial data as smooth functions allows a formulation of dynamic optimization method [13,14] to address the irregularity in the sensor data sampling interval and phase variations with improved computational efficiency.

The following are the contributions of this paper:

- We propose using an efficient functional data-based dynamic programming method as a machine learning pipeline pre-processing step to remove timing misalignment in moving sensor data.
- We modify learning rules in conventional deep learning models to handle functional data and propose a novel split encoder variational autoencoder that combines functional data classifications and clustering for prognostic management in transport systems.
- We evaluate the proposed technique using synthetic and real-world sensor data by a domain adaptation method and validate its efficacy using different performance metrics.

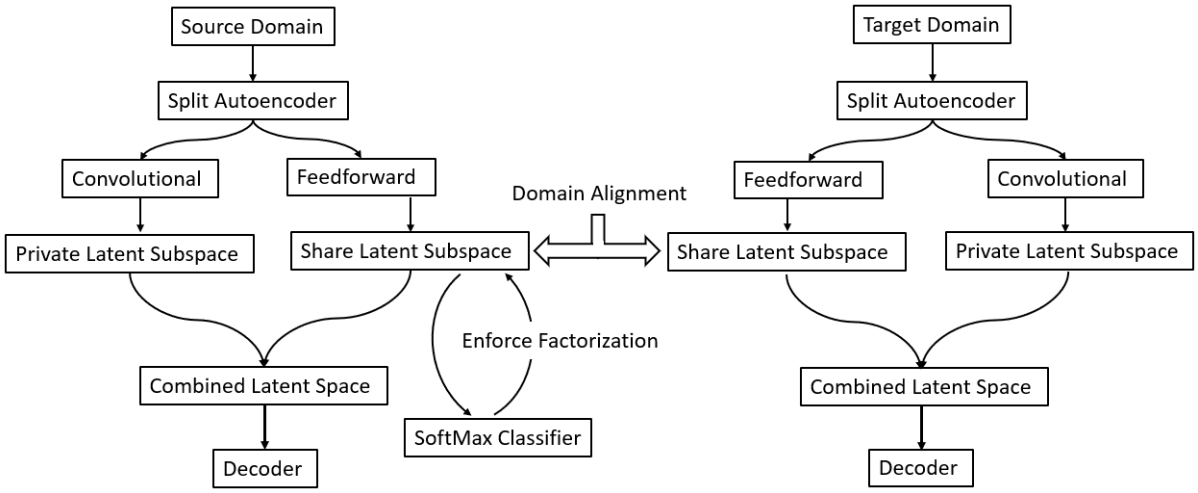

Figure 2. Overview domain adaptation strategy by joint FS-VAE design. From left to right. On the left (source domain): This branch acts on the synthetic data and subspace separation is enforced by the softmax classifier. On the right (target domain): This branch acts on the real-world data which is pre-processed to remove timing misalignment. The source domain branch is trained without the domain adaptation to obtain the initial network weights for the target domain branch. In domain adaptation training, both branches are connected to minimize the network loss function defined by Equation (15).

The goal of developing the functional subspace variational autoencoder (FS-VAE) is to identify meaningful latent space characteristics within time series data from the moving sensor by utilizing the two complementary latent subspaces as shown in Figure 2. In addition to the classification of the temporal profile as a means to detect anomalies, we also hypothesize that the long-term stability of a signal-generating mechanism captured by the sensor data can be assessed by analyzing the evolution in the size of clusters in the latent space. This is demonstrated in this paper. The output from the two subspaces can be useful in performing downstream analysis tasks, such as comprehending the progression of equipment malfunctions or system breakdowns [15].

2. Related Work

Time series sensor data gathered using a fixed sensor array can be considered analogous to imagery data, as they are both the result of time-synchronized sampling by a global clock. Convolutional variational autoencoder-based deep learning models have proven to be effective for anomaly detection and forecasting across a variety of industrial applications [16,17]. However, the underlying data model assumes that the pixels in the imagery data are independent of each other. As a result, comparing pixels becomes computationally expensive, and the process is more susceptible to noise corruption.

To migrate the time series data quality issues in the machine learning algorithms, sequence-to-sequence models, such as recurrent neural networks (RNNs) are proposed [18,19]. However, their training complexity grows linearly with the length of the sequence, which can result in prolonged computational times, especially for long sequences. Another method is called dynamic time warping (DTW) [20], which aligns sequences with varying speeds by 'warping'

the time axis. Its standard implementation, however, has a complexity of $O(N^2)$, making it computationally intensive for long sequences, despite the existence of variants designed to reduce this complexity. Both methods offer viable solutions for addressing misalignment in time series data from moving sensors, but they come with trade-offs in terms of computational complexity.

In transportation systems, substantial volumes of unlabeled data frequently arise, necessitating the employment of domain adaptation strategies to enhance model performance across diverse and novel data distributions. Domain adaptation techniques [9,10] for aligning latent spaces have been utilized in the field of machine learning. As mentioned previously, there is a lack of complete correlations among latent features across various domains. To address this issue, we extend the two-stage disentanglement method [21] and interweave domain adaptation and disentanglement without the mini-max game played in adversarial training between a generator and a discriminator network. The result is a less complex network. As illustrated in Figure 2, the input data are projected into private and shared latent subspaces using the split autoencoder as proposed in [21]. Rather than relying on adversarial training, this disentanglement is facilitated by employing a softmax classifier, thereby enhancing the efficiency of the process.

The proposed FS-VAE and domain adaptation techniques offer improvements over previous related works. Here are the main reasons why the functional data analysis (FDA) approach can provide a better data model:

- Dimensionality reduction: FDA can reduce the dimensionality of time series data by representing the data as functions with a limited number of parameters. This is particularly useful for machine learning tasks, as it reduces computational complexity and can help prevent overfitting.
- Alignment and warping: One common challenge when clustering time series data from moving sensors is that the data can have different time scales or be misaligned due to the varying speeds of the sensors. FDA offers a functional alignment technique that can help align the time series data before clustering, leading to more accurate results.
- Functional similarity: FDA allows for the comparison and clustering of time series based on their functional properties, such as shape or overall trend, rather than just pointwise similarities. This can lead to more meaningful clusters, as it focuses on the underlying structure of the time series rather than just their pointwise values.

3. FS-VAE Deep Learning Network

FS-VAE differs from conventional VAE in that the encoder input layer and the decoder output layer deal with functional data rather than vectorial data. As illustrated in Figure 1, each neural node in the input layer performs convolution operations on the input functional datasets and outputs a numerical result to deeper neural layers. The decoder works on the combined latent subspaces generated by the split encoders, which produce two sets of independent latent variables: private (unsupervised) and shared (supervised). This is also shown in Figure 2. These variables facilitate the discovery of low-dimensional latent representations of functional data. The unsupervised latent variables capture clustering correlation features in the functional datasets, while the supervised latent variables identify temporal features of functional data. To achieve these two aims, the encoder–decoder pair in the FS-VAE must reconstruct the original sequences of functional data from the union of supervised and unsupervised latent variables by optimizing an appropriately designed loss function, and the factorization of latent space is enforced by a softmax classifier loss function [22]. In the subsequent section, we will present the mathematical formulation of the FS-VAE.

3.1. Time Warping to Compensation Phase Variations

Given a set of functions $\{f_i(t)\}$ representing the functional dataset, warping functions $\{\gamma_i(t)\}$ are introduced such that

$$\tilde{f}_i(t) = f_i(\gamma_i(t)) \quad i = 1, 2, \ldots, n, \tag{1}$$

to correct phase variations. These corrections improve the validity of statistical analyses and predictions derived from the functional data by providing a more accurate reflection of the data structure. Computing the warping function in the square-root velocity function (SRVF) [13,14] space simplifies the optimization problem involved in finding the optimal alignment. In the SRVF space, defined as

$$q_i(t) = \frac{f_i'(t)}{\sqrt{|f_i'(t)|}} \quad i = 1, 2, \ldots, n, \tag{2}$$

the optimization problem becomes a standard shortest-path problem, which can be efficiently solved using dynamic programming. The distance to be minimized is the L^2 norm in Hilbert space between the template function $q_0(t)$ and warped functions:

$$d(q_0, q_i \circ \gamma_i) = \int_{\mathcal{T}} \|q_0(t) - q_i(\gamma_i(t))\|^2 dt \quad i = 1, 2, \ldots, n. \tag{3}$$

The template function is then updated from computing the average of these aligned SRVFs:

$$q_0^{\text{new}}(t) = \frac{1}{n} \sum_{i=1}^{n} q_i(\gamma_i(t)). \tag{4}$$

The template function can be initialized as the average of the functional dataset and its SVRF. After the convergence of the iterative loop formed by Equations (3) and (4), the resulting optimal warping functions $\{\gamma_i^*(t)\}$ can generate an aligned set of functions, $\{x_i(\gamma_i^*(t))\}$, which mitigate the phase variations and improve the accuracy of statistical analyses and predictions. An illustrative example of the convergence process inherent in the SRVF method can be found in Appendix A.

3.2. Explicit Functional Data Formulation

According to the notion of a multi-layer deep learning network, information flows from one layer to subsequent layer. For a particular layer, the mathematical rule relating input vector $\hat{x} \in \mathbb{R}^n$ to output vector $\hat{y} \in \mathbb{R}^m$ is

$$\hat{y} = \sigma\left(W\hat{x} + \hat{b}\right), \tag{5}$$

where m is the number of neural processing units on this layer, $W \in \mathbb{R}^{m \times n}$ is the weight matrix , $\hat{b} \in \mathbb{R}^m$ is the network bias in this layer and σ is the non-linear activation function. The network parameters are all time independent. In the case of functional time series data, $n \to \infty$ in the input layer, and $m \to \infty$ in output, and we need to replace the time-discretization coordinates with Hilbert space basis coordinates in the learning rule for these two layers.

For the functional data input layer, the learning rule for functional datasets $\{f_k(t)\}_{k=1}^{L}$ can be defined as

$$v_i = \sigma\left(\sum_{k=1}^{K} \int_{\mathcal{T}} \Psi_{i,k}(s) f_k(s) ds + b_i\right). \tag{6}$$

The time-dependent weight function $\Psi_{i,k}(t)$ will be learned by this network to extract features from the functional inputs. By using the series expansion in terms of the orthonormal basis functions $\{\varphi_1, \varphi_2, \ldots, \varphi_M\}$ with a inner product $(,)_H$, the output of the input layer is

$$v_i = \sigma\left(\sum_{j=1}^{M} \sum_{k=1}^{K} w_{j,k}^i \int_{\mathcal{T}} \varphi_j(s) f_k(s) ds + b_i\right). \tag{7}$$

The hyperparameters or the weight tensor on the 1st hidden layer to be solved are

$$W = w_{j,k}^{i}, i \in [1, K], j \in [1, M], k \in [1, K], \tag{8}$$

where M is the number of basis function to represent the functional weights, and K is the number of input functional data. The input is a matrix

$$\Psi^{j,k} = \int_{\mathcal{T}} \varphi_j(s) f_k(s) ds. \tag{9}$$

In compact form, the input vector $\hat{v}$ of the 1st hidden layer can be written as

$$\hat{v} = G_1(W \wedge \Psi + b_1), \tag{10}$$

where the $\wedge$ operator is the

$$(W \wedge \Psi)_i = w_{j,k}^{i} \Psi^{j,k}, \tag{11}$$

and the repeated indices are implicitly summed over. From a programming standpoint, Equation (11) is implemented as the i-th node, which is connected to $M \times K$ input nodes. These input nodes receive input from $\Psi^{j,k}$ as illustrated in Figure 1. The subsequent deep layers can be dense network or convolutional network as defined by a transformation

$$y_i = \sigma\left(\sum_{j=1}^{r} W_j x_{1+(i-1)s+j} + b_j\right), \tag{12}$$

with the stride s, kernel size r, weight W and bias vector b. In this context, x is the input feature from the 2nd layer in Figure 1.

For the functional data output layer,

$$\Psi^{j,k} = \sigma\left(W_{j,k}^{i} x_i + b_{i,k}\right). \tag{13}$$

Then, the output functional data can be obtained as

$$f_k(t) = \sum_{j=1}^{M} \Psi^{j,k} \varphi_j(t), k \in [1, L]. \tag{14}$$

Deep learning models for functional data can be extended in various ways [23,24]. The current approach utilizes a functional input and output layer, where the input represents the functional data, and the output provides the predictions. The deep learning layers in this model are numerical, meaning that they process numeric inputs and perform mathematical operations on them. These layers learn from the data patterns and improve the prediction accuracy over each epoch in the model training by the stochastic gradient descent optimizer [25].

In our functional data analysis of vibration signals, we parameterize the weight functions using a truncated Fourier series. Given that the frequency spectrum of our data ranges from 0 Hz to 100 Hz, we carefully select the number of basis functions to effectively capture the underlying frequency content while maintaining computational efficiency. To balance these considerations, we opt for 32 cosine and 32 sine functions, resulting in 64 basis functions. This selection results in a frequency resolution of approximately 3.125 Hz, offering a meaningful compromise between detail and model complexity. Furthermore, as the projections onto sine and cosine basis functions can yield both positive and negative values, we should choose activation functions that allow for negative values. In the present work, we used the hyperbolic tangent (tanh) as the activation function.

3.3. Latent Subspaces Domain Adaptation

Suppose that we have a training set $\mathcal{T}^s = \{(\Psi_n^s, \mathbf{Y}_n^s)\}_{n=1}^{N_s}$, which consists of labeled samples $(\Psi_n^s, \mathbf{Y}_n^s) \in$ a supervised source domain. As mentioned previously, real-world time series data from a moving sensor are usually unlabeled. We represent these target domain data as $\mathcal{T}^t = \{\Psi_n^t\}_{n=1}^{N_t}$.

FS-VAE projects the functional data into two latent subspaces as illustrated in Figure 2. The shared subspace is used to classify the temporal feature of the data. The private subspace is used to determine the clustering correlations of the data. We introduce a novel method of applying softmax classification training to facilitate the factorization of latent subspaces by the split encoders. Additionally, the reconstruction loss of the FS-VAE training can ensure private representations to capture low-dimensional projections of input data. With the labeled training dataset, the source domain branch of Figure 2 is self-sufficient in training the split encoder, softmax classifier and decoder to accomplish temporal feature classification and clustering analysis. The trained networks in source domain can be transferred to the target domain as an initial condition for the domain adaptation training.

4. Model Training

The joint FS-VAE architecture depicted in Figure 2 simultaneously performs dimensionality reduction clustering (unsupervised) and classification tasks (supervised). The total loss function of the architecture comprises two reconstruction error terms, a single coupling error term and a classification term:

$$\begin{aligned}\mathcal{L}_S = &\frac{\lambda_r}{N}\sum_{i=1}^{N}\left\|\mathbf{x}_i^s - D_s\left(E_s^c(\mathbf{x}_i^s), E_s^p(\mathbf{x}_i^s)\right)\right\|^2 + \frac{\lambda_r}{N}\sum_{i=1}^{N}\left\|\mathbf{x}_i^t - D_t\left(E_t^c(\mathbf{x}_i^t), E_t^p(\mathbf{x}_i^t)\right)\right\|^2 + \\ &\lambda_{\text{st}}\sum_{i,j\in N}\left\|E_s^c(\mathbf{x}_i^s) - E_t^c\left(\mathbf{x}_j^t\right)\right\|^2 - \frac{\lambda_c}{N}\sum_{i=1}^{N_s}\mathbf{y}_i^T \ln C\left(E_s^c\left(\mathbf{x}_i^{s0}\right)\right).\end{aligned} \tag{15}$$

The hyperparameters λ_r, λ_{st} and λ_c explicitly control this trade-off in our formulation. We set all three parameters to 1.0 for all central analyses in this manuscript.

Our model is trained by updating several components alternately using the stochastic gradient descent (SGD) algorithm. These components include the shared encoders $E_s^c(\cdot)/E_t^c(\cdot)$, the private encoders $E_s^p(\cdot)/E_t^p(\cdot)$, the decoders $D_s(\cdot)/D_t(\cdot)$, and the softmax classifier $C(\cdot)$. Our model is implemented using the PyTorch framework [26].

5. Results

5.1. Pre-processing of Target Domain Data

The real-world data used in our study came from accelerometers mounted on trains. This target domain vibration dataset records variations in the signal generation mechanism as a result of the interaction between train wheels and railway tracks. Investigating the temporal features and the long-term trends in clustering correlations can provide valuable insights for maintenance decision making and improving system reliability. To ensure the accuracy of our analysis, the data are pre-processed to eliminate timing misalignment issues according to Section 3.1. The results of our analysis of a segment of real-world data are presented in Figure 3. As illustrated in this figure, it is evident that the misalignment, or phase noise, is largely removed upon inspection. This improved data quality allows for a more accurate assessment of temporal features and long-term trends. The impact of the pre-processing step on the unsupervised learning of clustering is further investigated and compared against other standard methods. The results are tabulated in the table in Section 5.7. It can be observed that there is a consistent improvement of about 30% in the clustering performance metrics in the benchmark comparison when pre-processing is not applied. More details about the construction of the deep learning model can be found in Section 5.3.

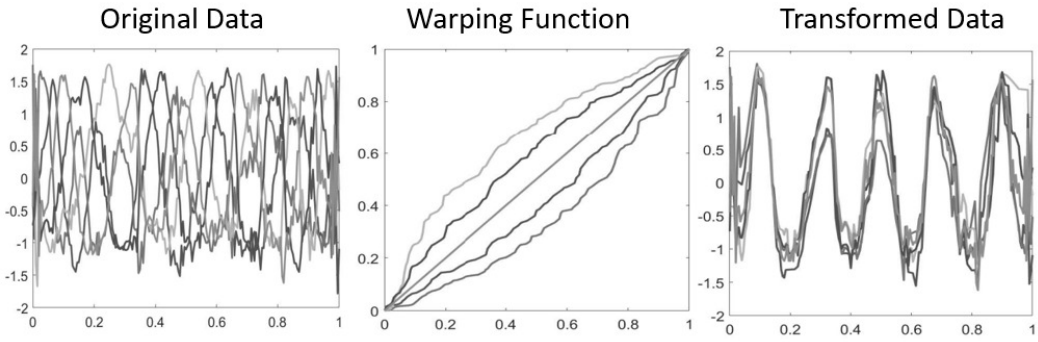

Figure 3. The demonstration of the warping function calculations used to align the real-world vibration sensor data. The original vibration data are plotted in the left figure, while the computed warping functions are displayed in the center figure. The right figure presents the aligned vibration data achieved through the warping process.

5.2. Labeled Synthetic Dataset Preparation

We evaluate our approach to temporal feature classification using simulated random vibration time series data generated by the method described in [27] for the source domain. We create 3000 samples of functional datasets, each of which contains three segments of functional data, denoted as $f^k(t)$, generated according to

$$f_k(t) = \Phi_m(t) \times S_\pi(t), \tag{16}$$
$$k \in [1, \ldots, 3000] \quad m \in [1, \ldots, 30],$$
$$\pi \in \{\text{impulsive, non-impulsive}\}$$

where $S_\pi(t)$ is the random vibration signal, and $\Phi_m(t)$ is the envelope function. The envelope function emulates the spatial feature for each functional data segment. We prepare 30 different types of envelope functions of different modulation lengths, and mix them randomly with vibration signals. The labeling of the time series data in the training dataset into impulsive signals and non-impulsive signals is determined by the spectral content on the high-frequency side. Given that the frequency spectrum of our data ranges from 0 Hz to 100 Hz, the high-frequency side is above 30 Hz, which corresponds to a pulse duration of less than 0.1 s.

According to the experimental results shown in the figure in Section 5.4, the accuracy of our technique to classify impulsive signals and non-impulsive signals remains consistent across different envelope functions, with a fluctuation of within 15%. More details are discussed in Section 5.4.

5.3. Decoder Selection and Evaluation

Our proposed split encoder FS-VAE model comprises two distinct encoders: a feed-forward network and a convolutional network. The structure of the encoders is shown in Table 1. Both encoders transform the output of the functional input layer (refer to Equation (9)), which is 192-dimensional numerical data from the overlap of functional data into different basis functions, as mentioned in Section 3.2, into separate latent spaces of dimensions 6 and 5, respectively. These outputs are combined into a joint latent space of 11 dimensions. A latent variable is sampled from this joint distribution using the reparameterization trick [7], serving as the input to the decoder for data reconstruction.

Table 1. The architecture of the encoders in the split encoder FS-VAE is as follows. The input layer, depicted in Figure 1, is labeled as "input to next layer". This layer is responsible for receiving the overlap integration between the functional input data and the basis functions. The network parameters that determine the functional weights are optimized alongside the other network parameters using backpropagation and Adam optimization.

Component	Type	Architecture
Encoder 1	Feedforward	Input: 40, Hidden Layers: 15, Output: 6
Encoder 2	Convolution	Input: 40, Conv Layers: [8, 16], Fully Connected: 6, Output: 5

The architecture of the encoders does not explicitly determine the structure of the decoder in the split encoder FS-VAE model. To identify an appropriate decoder architecture, we evaluate two candidates: a feedforward network and a convolutional network. The structures of the two decoder options are shown in Table 2. Our assessment focused on training error decay and reconstruction error, which is the Euclidean norm between the matrices in Equations (9) and (13). According to the experimental results in Figure 4, the feedforward decoder demonstrates a slower training error decay and a higher reconstruction error compared to the convolutional decoder. The convolutional decoder shows a rapid training error decay and a 100% reduction in reconstruction error, indicating more efficient learning and accurate reconstruction. Based on these results, the convolutional network is identified as the more suitable decoder for our split encoder FS-VAE model.

Table 2. The architecture of the candidate decoders in the split encoder FS-VAE differs from that of the split encoder itself. Two options are explored in this regard. The results obtained from these options are evaluated and verified using the reconstruction loss, similar to the conventional VAE approach.

Component	Type	Architecture
Decoder (Option A)	Feedforward	Input: 11, Hidden Layers: [15, 25], Output: 40
Decoder (Option B)	Convolution	Input: 11, Fully Connected: 16, Deconv Layer: [8, 4], Output: 40

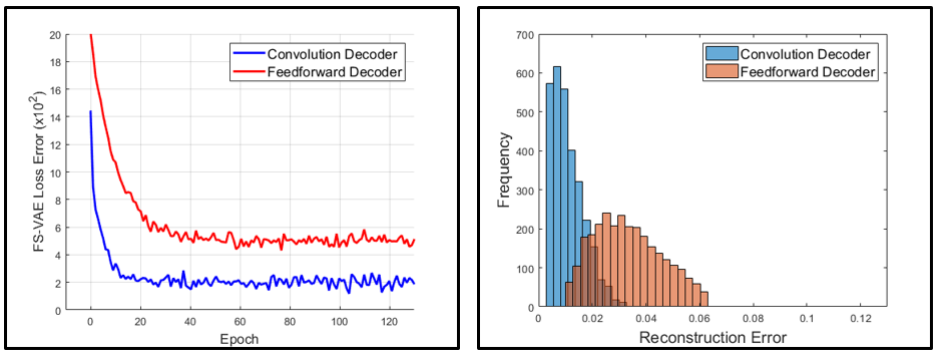

Figure 4. The reconstruction error of variational autoencoders for different decoders is analyzed. The left plot illustrates the convergence trend of the Adam optimization across epochs for each decoder option. On the right, the distribution of reconstruction errors for the validation data is displayed for each decoder option. The convolutional decoder shows better performance than the feedforward decoder.

5.4. Classification Accuracy with Synthetic Dataset

The effectiveness of the supervised learning feature of FS-VAE by the softmax classifier is investigated in this section. This multi-label classification problem is trained by minimizing the cross-entropy loss function in Equation (15). For this purpose, we consider a case with three segments ($L = 3$) as depicted in Figure 1. Each segment exhibits two behavior types: "impulsive" and "non-impulsive." The output of the softmax consists of six nodes, representing the probabilities of the two temporal features within each of the three segments. The classifier receives input from the 6-dimensional latent space generated by the feedforward encoder.

The architecture of the classifier can be found in Table 3. The effectiveness of our model is evaluated using the F1 score. Our findings, as shown in Figure 5, indicate that the F1 scores remain steady across different combinations of envelope functions and impulsive function profiles. The fluctuations in the F1-score are within 15%. The cases with relatively low F1-scores come from the impulsive function profiles that, under Fourier analysis, have similar spectral content to non-impulsive functions. This suggests the stability of coupling the softmax classifier with the feedforward encoder to encode salient features and underlying structures of data into a lower-dimensional shared latent subspace.

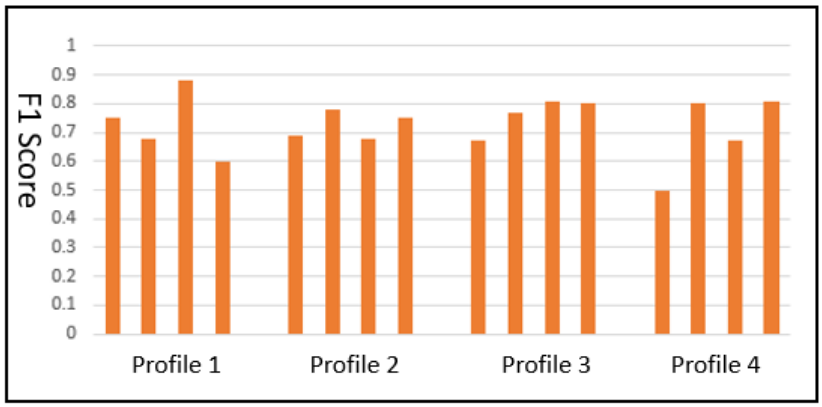

Figure 5. The F1-score was calculated for different options on the labeled synthetic data without domain adaptation. The results show four bands of different profile envelope shape functions, and within each profile, there are four variations in 'impulsive' signals selected at random.

In addition, we also performed an in-depth comparison analysis against the standard time series classification methods with vectorial data and functional data input. The results are shown in Table 4, and we observe a variety of performance patterns.

Table 3. The architecture of the softmax classifier for segment = 3 is illustrated in Figure 1. This classifier is responsible for determining the likelihood of detecting "impulsive" or "non-impulsive" signals within each segment. The input to the classifier is derived from the latent subspace of the feedforward decoder in the split encoder as depicted in Figure 2.

Layer	Size
Input	6
Hidden Layer 1	6
Hidden Layer 2	6
Output (Softmax)	6

Table 4. A comparison of the performance of of five classifiers on labeled synthetic data. The evaluation is conducted using both the vectorial/functional data model and the functional variational autoencoder (FS-VAE) approach. FS-VAE demonstrates the highest overall performance. Note that, except for SVM, all classifiers exhibit improved performance when applied to the functional data approach.

Method	**Metric**					
	Accuracy	**F1 Score**	**AUC Score**	**Precision**	**Recall**	**Specificity**
Vectorial Data						
Decision Trees	0.72	0.69	0.75	0.70	0.68	0.76
k-NN	0.68	0.66	0.71	0.67	0.65	0.73
Naive Bayes	0.69	0.67	0.72	0.68	0.66	0.74
SVM	0.78	0.76	0.82	0.77	0.74	0.82
Random Forests	0.79	0.77	0.83	0.78	0.75	0.84
MLP	0.75	0.76	0.79	0.77	0.76	0.86
Functional Data						
Decision Trees	0.77	0.75	0.80	0.76	0.74	0.81
k-NN	0.82	0.80	0.86	0.81	0.79	0.87
Naive Bayes	0.84	0.82	0.88	0.83	0.81	0.89
SVM	0.78	0.76	0.78	0.77	0.75	0.82
Random Forests	0.86	0.84	0.90	0.85	0.83	0.91
MLP	0.82	0.80	0.84	0.81	0.79	0.86
FS-VAE	0.93	0.91	0.96	0.93	0.89	0.98

For the decision trees algorithm, we find similar performance across functional and direct (discrete) approaches, with a slightly better outcome using functional data. This algorithm's AUC score is 80% for functional data and 75% for vectorial data, suggesting that decision trees algorithm algorithms may have slight benefits from the dimensionality reduction offered by the functional approach, although it can also perform satisfactorily in higher-dimensional spaces.

Support vector machine (SVM) shows a small advantage in performance with the vectorial data approach. Given that SVMs are inherently designed to handle high-dimensional data and capture complex decision boundaries, they may not significantly benefit from the functional approach's dimensionality reduction. In fact, the vectorial approach outperforms the functional approach by about 5%, with AUC scores of 82% and 78%, respectively.

In contrast, the k-nearest neighbors (kNNs), naive Bayes (NB) and multi-layer perceptron (MLP) algorithms demonstrate enhanced performance with the functional approach. These algorithms typically perform better in lower-dimensional spaces, with AUC scores of 86% for k-NN , 88% for NB and 84% for MLP with functional data, versus 71%, 72%, and 79%, respectively, with discrete data.

The versatile random forest (RF) algorithm shows a slight improvement with the functional approach compared to the direct one. Despite its capability to handle high-dimensional data and complex relationships, it achieves an AUC of 90% with functional data, compared to 83% with vectorial data. This indicates that robust and adaptable algorithms such as random forest can also benefit from the more concise data representation offered by functional methods.

Our current technique, FS-VAE, involves performing softmax classification on the split encoder variational autoencoder's latent subspace and shows remarkable performance improvements. For decision tree, FS-VAE increases the AUC to 96%, a gain of 16% over the functional approach and 21% over the vectorial approach. Even for SVM, where the

vectorial data approach performs exceptionally well, FS-VAE surpasses it with an AUC of 96%, indicating a performance gain of about 14%. For algorithms such as kNN and NB, which show significant performance improvement with the functional approach, FS-VAE drives AUCs up to around 96%, a performance gain of 10–8%, respectively. Finally, for the random forest algorithm, which shows a slight preference for the functional approach, FS-VAE pushes the AUC to 96%, a performance gain of 6%.

On average, FS-VAE improves performance by about 14.5% over the best method in the vectorial data category (random forests), and by about 7.9% over the best method in the functional data category (random forests). Thus, the FS-VAE approach, which combines functional data analysis and variational autoencoders, provides a compelling strategy that could outperform both the functional and direct approaches in time series classification tasks.

5.5. Accuracy Dependence on Segment Size

We evaluate the classifier's performance with different numbers of segments in the model, ranging from 3 to 9, with the results presented in the Figure 6. The network layer sizes are also adjusted in each case, while the number of hidden layers remains the same. Our analysis shows that a model with 6 branches or less achieves over 50% accuracy. However, with an increase in segments to 7 or more, simply adjusting the network layer sizes is insufficient to preserve the accuracy, which declines to below 50%. While there are several potential enhancements, such as adding more network layers, integrating regularization techniques, such as dropout, and fine-tuning hyperparameters, such as learning rate, batch size, or training epochs, these extensive modifications are out of the scope of the current study. For the discussions in this paper, we set the value of L to 3 when it is not explicitly mentioned.

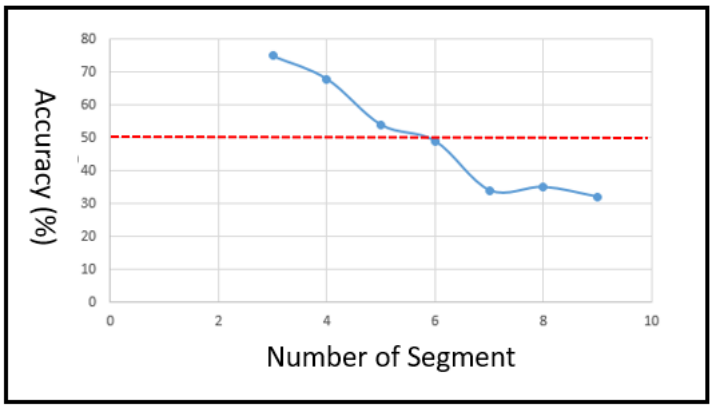

Figure 6. The softmax classification accuracy is evaluated by considering the number of segments for network architectures with different numbers of nodes in the first layer. It should be noted that the number of layers remains constant throughout the analysis.

5.6. Latent Subspaces Alignment

In our study, we implement a domain adaptation strategy using a split encoder and one decoder variational autoencoder structure for both the source and target domain branches. The split encoder produces a shared subspace and a private subspace. The shared subspace is connected to a softmax classifier to enforce the factorization of the latent space. The shared subspaces in both branches are connected by the latent space domain adaptation method, facilitating the transfer of knowledge from the source to the target domain.

The source domain is initially trained with labeled synthetic data for 100 epochs. Upon convergence, the network weights, which are achieved with a final training loss of 0.05, are used as initial values for the network weights in the target domain branch. This transfer learning trick, which provides an educated guess of the initial network parameters of the

target domain branch in Figure 2, can facilitate domain adaptation training with a modestly sized real-world dataset. Both branches are then trained together for another 50 epochs with the domain adaptation method connecting the shared subspaces, achieving a final training loss of 0.03. During the domain adaptation training process, we encounter a situation known as "latent space collapse" [28], where the diversity of latent features is suppressed to be very small. This issue is mitigated by incorporating batch normalization [29] into the training process, which helps to maintain the diversity of the latent space and improves the model's performance on the target domain data.

The result of domain adaptation training with and without batch normalization is studied with a labeled target domain dataset comprising 30 samples. The shared subspace in the trained target domain branch is connected to the softmax classifier. The classification evaluation metrics are tabulated in Table 5.

Table 5. The evaluation of domain adaptation performance metrics involves the assessment of batch normalization's impact. Domain adaptation networks are trained using both labeled synthetic and unlabeled real-world data. Subsequently, the results are compared with and without batch normalization. The classification accuracy in the target domain branch is computed by connecting the softmax classifier to the shared latent subspace in this branch, utilizing 30 labeled real-world data. A clear improvement is demonstrated in all the metrics as a result of incorporating batch normalization.

Metric	Method	
	Without Batch Normalization	**With Batch Normalization**
Accuracy	0.60	0.85
Precision	0.55	0.82
Recall	0.65	0.88
F1 Score	0.60	0.85
Mean Absolute Error	0.45	0.20
Root Mean Squared Error	0.60	0.35

In Table 5 we can see that the inclusion of batch normalization significantly improves all the metrics. The accuracy increases from 60% to 85%, precision from 55% to 82%, recall from 65% to 88%, and F1 score from 60% to 85%. The error metrics also decrease, with the mean absolute error reducing from 0.45 to 0.20 and root mean squared error from 0.60 to 0.35. These improvements demonstrate the effectiveness of the domain adaptation method, achieving an accuracy of about 85%.

5.7. Clustering in Private Latent Subspace and Prognostic Implications

This section delineates the operational principles of the FS-VAE private subspace model, specifically, its ability to extract clustering correlations from input functional data. As shown in Figure 1, the model projects the input data into separate subspaces. A portion of this projection is handled by the 1D convolutional neural network (1DCNN) decoder, which maps the data into the private subspace. This private subspace is then leveraged for clustering analysis, capturing and exploring the variations within the functional dataset. The successful factorization into subspaces is evidenced by the low reconstruction error, as generated by the total variational autoencoder (VAE). This can be visually confirmed in Figure 4, which shows a comparative representation of the original and reconstructed data, thus illustrating the effectiveness of our approach.

The neural network responsible for generating the clusters is trained concurrently with the procedures for classification and domain adaptation as previously described. This simultaneous training approach ensures that the network's learning is harmonized across these different tasks, thus leading to more cohesive and effective performance.

Our study aims to evaluate the performance of various dimensionality reduction techniques on time series data. We primarily focus on the convolutional variational auto-encoder (CVAE) [30] and principal component analysis (PCA) [31], specifically for vectorial time series data, in addition to the FS-VAE private latent subspace model. These models are

examined based on their capacity to extract meaningful clusters from the time series data. To quantify the quality of these clusters, we employ measures such as accuracy, mutual information, and purity metrics, following the standards defined in the well-established literature [32–34].

The initial stages involve the pre-processing and transformation of our time series data into a lower-dimensional representation. Here, we note a significant distinction between the techniques. CVAE and PCA, which are designed to handle discrete data, require larger network sizes—approximately 5394 and 5000 parameters, respectively. On the other hand, the FS-VAE model, designed for functional data representation, leads to a considerably smaller network size, around the order of 2200 parameters.

After transforming the data into a lower-dimensional representation with the respective techniques, we subject them to the K-means clustering algorithm as implemented in the scikit-learn library [35]. Our comparative analysis of the three dimensionality reduction techniques, considering their computational efficiency and the quality of the resultant clusters, is shown in Table 6. The accuracy in clustering performance shows improvement between 20% to 30%. Notably, there is a computational speedup of about 50% for FS-VAE compared to CVAE, and around 60% when compared to PCA. Figure 7 also shows the t-SNE plot of the clustering result of FS-VAE against PCA. By inspection, the FS-VAE model's clusters are more distinct and well separated, and the t-SNE plot provides additional evidence to support the improved clustering performance of the FS-VAE model.

We can calculate the mean intra-cluster distance (MICD) [35], which is a measure of the average distance between data points within a cluster in the private latent subspace, as an indicator of changes in the underlying signal generation process. For a given cluster C, its MICD can be computed as follows:

$$\text{MICD}(C) = \frac{1}{|C|(|C|-1)} \sum_{i \neq j} d(x_i, x_j), \tag{17}$$

where $|C|$ is the number of points in cluster $d(x_i, x_j)$ is the Euclidean distance between distinct pair of points x_i and x_j in the latent space. By tracking changes in MICD, we can detect alterations in the underlying signal generation process, which may manifest as variations in data distribution within the latent space. An increase in MICD indicates that the average distance between points within a cluster is growing. This can be linked to the signal generating process becoming more random. This randomness is often associated with a degradation in reliability.

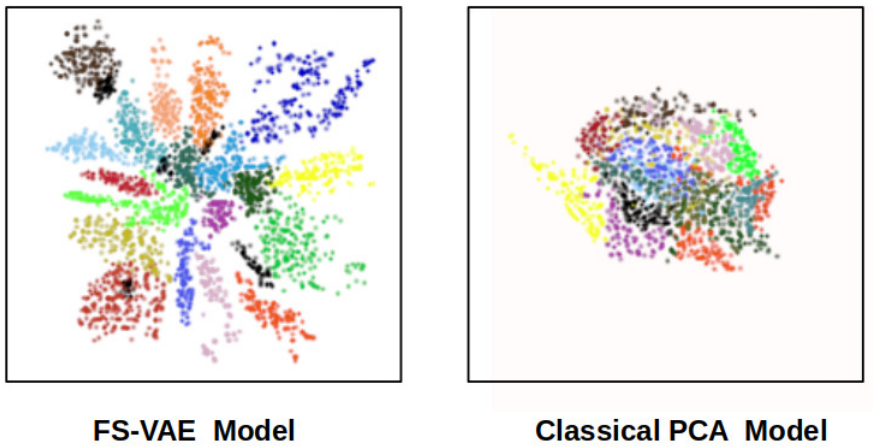

Figure 7. A comparison was made between the t-SNE projection of the clustering performance of FS-VAE on functional data and PCA on vectorial data using real-world data. The FS-VAE model shows better performance in this comparison. Additional quantitative results can be found in Table 6.

Table 6. Performance evaluated by K-means clustering on the representation extracted from functional data using different representation learning methods. The evaluated methods include convolutional variational autoencoder (VC), principal component analysis (PCA), and functional subspace variational autoencoder (FS-VAE). FS-VAE demonstrates improved performance across all metrics compared to the other methods. Additionally, the impact of not pre-processing the real-world data (as indicated by Real-world *) is examined. A clear deterioration in performance is observed when phase-variation removal is not applied.

Method	Metric			
	Accuracy	Normalized Mutual Information	Purity	Relative Speedup
Synthetic				
CVAE	0.79	0.60	0.75	1.50
PCA	0.75	0.58	0.70	1.60
FS-VAE	0.95	0.85	0.90	1.00
Real-world *				
CVAE	0.68	0.55	0.68	1.50
PCA	0.65	0.50	0.62	1.60
FS-VAE	0.82	0.65	0.80	1.00
Real-world				
CVAE	0.72	0.58	0.71	1.50
PCA	0.70	0.62	0.68	1.60
FS-VAE	0.89	0.77	0.85	1.00

Figure 8 showcases MICD plots for a series of datasets, spanning three distinct time periods, each differing by a month. In addition, it includes a magnitude-shape (MS) plot [36] for functional data analysis comparison. Both the MICD and MS plots exhibit a positive trend over the observed time period. This upward trend potentially indicates a deterioration in the robustness of the underlying signal generating process, hinting at possible degradation [37]. This consistency between the plots effectively demonstrates the utility of the private subspace in finding trends in sensor data variations. It provides valuable insights into the stability of the underlying signal generation process. This approach complements temporal feature-based classification in the shared latent subspace for anomaly detection.

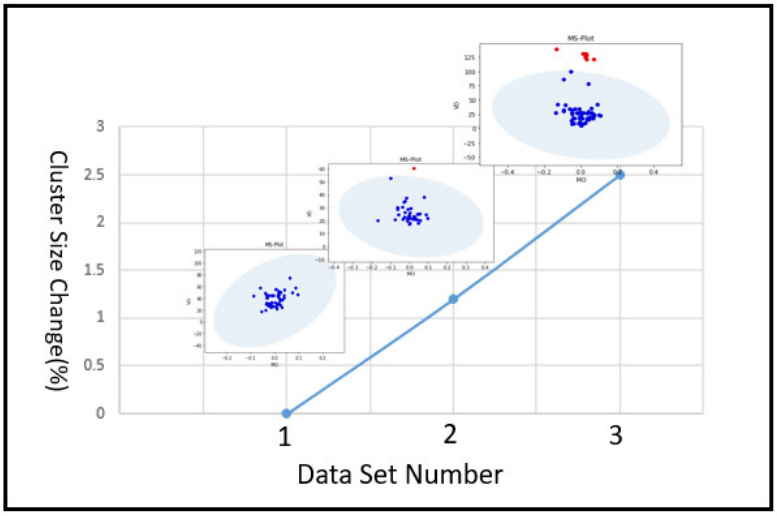

Figure 8. Mean intra-cluster size calculations are performed on three real-world datasets. The figure includes inserts displaying the output of the Magnitude-Shape Plot [36] for functional data analysis corresponding to each dataset. The observed agreement between these two approaches emphasizes the advantages of low-dimensional projection in capturing profile-based variations. This capability enables the detection of degradation in the underlying signal generation mechanism using the proposed technique.

6. Conclusions

In this paper, we leverage the benefits provided by functional data representation of time series data to develop a subspace variational autoencoder and an associated domain adaptation method to facilitate the extraction of temporal features and clustering correlations from unlabeled moving sensor data, such as the signals generated by the impact between train wheels and the railway track.

Our method shows great efficiency in extracting temporal features and clustering correlations from this unlabeled moving sensor data. We achieve a notable 30% improvement in cluster identification accuracy and a 15% improvement in classification accuracy when contrasted with traditional techniques. Furthermore, our technique also maintains a high level of domain adaptation accuracy, with a consistent quality value above 85%. The combination of improved accuracy in both clustering and domain adaptation, along with the ability to track the stability of the signal generating process, underscores the potential of our method for reliability monitoring and prognostic analysis in transportation systems.

As for future work, we plan to examine the performance of the technique proposed in this paper over conventional methods on functional datasets that exhibit non-convex shapes in latent space.

Author Contributions: Conceptualization, H.H.; Methodology, T.L. and C.-H.F.; Software, H.-T.W.; Formal analysis, T.-L.C. All authors have read and agreed to the published version of the manuscript.

Funding: Innovation and Technology Commission (ITC) of Hong Kong, via AIR@InnoHK cluster.

Data Availability Statement: The data presented in this study are available on request from the corresponding author. The data are not publicly available due to proprietary ownership.

Acknowledgments: The work presented in this article is supported by the Center for Advances in Reliability and Safety (CAiRS) admitted under AIR@InnoHK Research Cluster. The use of actual metro-railway system vibration data supplied by a CAiRS partner is also acknowledged.

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

Appendix A. Illustrating Convergence in the SRVF Method Using Two Functions

In this appendix, we illustrate the iterative procedure for aligning two functions, specifically f_1 and f_2, by employing the square-root velocity function (SRVF) [13] method. This procedure is centered around the systematic updating of the template function, computed by taking the average of progressively aligned functions. The convergence of this process, as detailed in Equations (3) and (4) in Section 3.1, is achieved when the updated and current template functions reach a predefined threshold of similarity.

Figure A1 introduces the initial functions f_1 and f_2 and the template function, which is defined as $f_3 = \frac{1}{2}(f_1 + f_2)$. The misalignment between f_1 and f_2 is clearly evident from the positions of their respective peaks. For this particular case, the iterative process of the SRVF method as defined by Equations (3) and (4) requires two iterations to achieve alignment between f_1 and f_2. The center sub-figure in Figure A1 illustrates the outcome of the first iteration, where the peaks in f_2 are aligned to those in f_1. The right sub-figure presents the results of the second iteration, reinforcing the convergence of the iteration, as the alignment remains consistent.

Figure A2 depicts the alignment of SRVF functions, q_1 and q_2, for f_1 and f_2 respectively. This visualization underscores the transformation from the original function space to the SRVF function space in which the alignment process is performed. The plot of the SRVF transformation of the template function is omitted for clarity. The central sub-figure depicts the alignment of q_1 and q_2 following the second iteration. The right sub-figure illustrates the updates in the warping function that contribute to the alignment of q_2 with q_1. The warping functions γ_p for q_1 and q_2 are shown by long-dashed and short-dashed lines, respectively. As the iterations proceed, the warping function undergoes modifications, approaching the identity function as indicated by the diagonal line in the figure. The long-dashed and short-dashed lines, which have the greatest deviations from the diagonal,

represent the warping function derived from the first iteration. The results of the second iteration exhibit a closer resemblance to the diagonal. This trend towards the identity function clearly demonstrates the convergence of the iterative process.

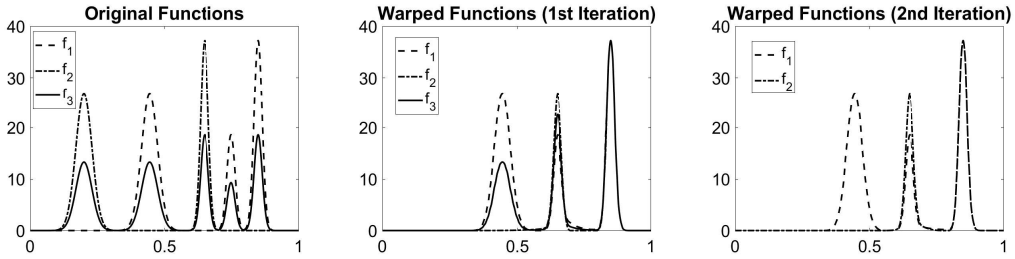

Figure A1. Illustration of function alignment using the square-root velocity function (SRVF) method, shown from left to right. The left figure depicts the original functions f_1, f_2, and the template function f_3. The central figure presents the results after the first iteration, demonstrating that f_2 has been aligned with f_1. The right figure shows the results following the second iteration. Here, f_1 and f_2 are fully aligned. The template function f_3 is omitted in this figure for clarity.

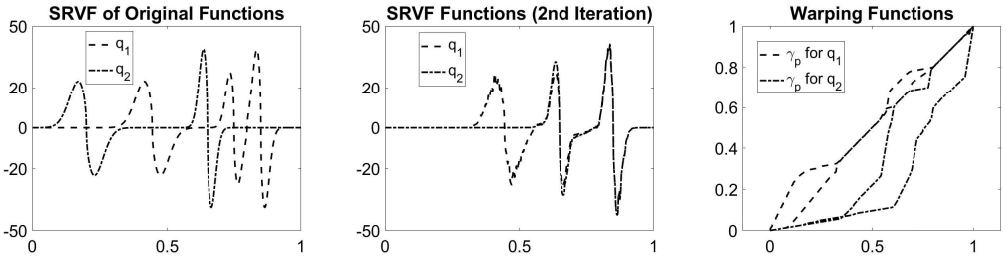

Figure A2. Illustration of the process of function alignment using the square-root velocity function (SRVF) method, shown from left to right. The left figure depicts the SRVF functions q_1 and q_2 of the original functions f_1 and f_2, respectively. The central figure presents the results after the second iteration, demonstrating that q_2 has been aligned with q_1. The right figure details the results of the warping functions for q_1 and q_2, after both the first and second iterations. The shifting of the warping functions towards the diagonal line indicates the convergent nature of the iterative computation of the warping function.

References

1. Fraga-Lamas, P.; Fernández-Caramés, T.M.; Castedo, L. Towards the Internet of Smart Trains: A Review on Industrial IoT-Connected Railways. *Sensors* **2017**, *17*, 1457. [CrossRef]
2. Pol, A.A.; Berger, V.; Germain, C.; Cerminara, G.; Pierini, M. Anomaly Detection with Conditional Variational Autoencoders. In Proceedings of the 2019 18th IEEE International Conference On Machine Learning And Applications (ICMLA), Boca Raton, FL, USA, 16–19 December 2019; pp. 1651–1657.
3. Ortega, F.; González-Prieto, Á.; Bobadilla, J.; Gutiérrez, A. Collaborative Filtering to Predict Sensor Array Values in Large IoT Networks. *Sensors* **2020**, *20*, 4628. [CrossRef]
4. Tu, D.Q.; Kayes, A.S.M.; Rahayu, W.; Nguyen, K. IoT streaming data integration from multiple sources. *Computing* **2020**, *102*, 2299–2329. [CrossRef]
5. Jan, B.; Farman, H.; Khan, M.; Talha, M.; Din, I.U. Designing a Smart Transportation System: An Internet of Things and Big Data Approach. *IEEE Wirel. Commun.* **2019**, *26*, 73–79. [CrossRef]
6. Lin, X.-X.; Lin, P.; Yeh, E.-H. Anomaly Detection/Prediction for the Internet of Things: State of the Art and the Future. *IEEE Netw.* **2021**, *35*, 212–218. [CrossRef]
7. Xie, R.; Jan, N.M.; Hao, K.; Chen, L.; Huang, B. Supervised Variational Autoencoders for Soft Sensor Modeling with Missing Data. *IEEE Trans. Ind. Inform.* **2020**, *16*, 2820–2828. [CrossRef]
8. Lee, S.; Kwak, M.K.T.; Tsui, K.-L.; Kim, S.B. Process Monitoring Using Variational Autoencoder for High-dimensional Nonlinear Processes. *Eng. Appl. Artif. Intell.* **2019**, *83*, 13–27. [CrossRef]
9. Mansour, Y.; Mohri, M.; Rostamizadeh, A. Domain adaptation with multiple sources. In *Advances in Neural Information Processing Systems*; MIT Press: Cambridge, MA, USA, 2008; Volume 21.
10. Tzeng, E.; Hoffman, J.; Saenko, K.; Darrell, T. Adversarial discriminative domain adaptation. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Honolulu, HI, USA, 21–26 July 2017; pp. 7167–7176.

11. Horvath, L.; Kokoszka, P. *Inference for Functional Data with Applications*; Springer: New York, NY, USA, 2012.
12. Aneiros, G.; Horová, I.; Hušková, M.; Vieu, P. On functional data analysis and related topics. *J. Multivar. Anal.* **2022**, *189*, 104861. [CrossRef]
13. Marron, J.S.; Ramsay, J.O.; Sangalli, L.M.; Srivastava, A. Functional data analysis of amplitude and phase variation. In *Statistical Science*; Institute of Mathematical Statistics: Beachwood, OH, USA, 2015; pp. 468–484.
14. Kurtek, S.; Srivastava, A.; Wu, W. Signal estimation under random time-warpings and nonlinear signal alignment. In *Advances in Neural Information Processing Systems*; MIT Press: Cambridge, MA, USA, 2011; Volume 24.
15. Dabou, R.T.; Kamwa, I.; Chung, C.Y.; Mugombozi, C.F. Time series-analysis based engineering of high-dimensional wide-area stability indices for machine learning. *IEEE Access* **2021**, *9*, 104927–104939. [CrossRef]
16. Santhosh, K.K.; Dogra, D.P.; Roy, P.P.; Mitra, A. Vehicular trajectory classification and traffic anomaly detection in videos using a hybrid CNN-VAE Architecture. *IEEE Trans. Intell. Transp. Syst.* **2021**, *23*, 11891–11902. [CrossRef]
17. Shieh, C.S.; Nguyen, T.T.; Horng, M.F. Detection of Unknown DDoS Attack Using Convolutional Neural Networks Featuring Geometrical Metric. *Mathematics* **2023**, *11*, 2145. [CrossRef]
18. Zheng, X.; Yu, D.; Xie, C.; Wang, Z. Outlier Detection of Crowdsourcing Trajectory Data Based on Spatial and Temporal Characterization. *Mathematics* **2023**, *11*, 620. [CrossRef]
19. Bhaskar, N.; Suchetha, M.; Philip, N.Y. Time series classification-based correlational neural network with bidirectional LSTM for automated detection of kidney disease. *IEEE Sens. J.* **2020**, *21*, 4811–4818. [CrossRef]
20. Tomasi, G.; Van Den Berg, F.; Andersson, C. Correlation optimized warping and dynamic time warping as preprocessing methods for chromatographic data. *J. Chemom. J. Chemom. Soc.* **2004**, *18*, 231–241. [CrossRef]
21. Hadad, N.; Wolf, L.; Shahar, M. A two-step disentanglement method. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Salt Lake City, UT, USA, 18–22 June 2018; pp. 772–780.
22. Bruch, S.; Wang, X.; Bendersky, M.; Najork, M. An analysis of the softmax cross entropy loss for learning-to-rank with binary relevance. In Proceedings of the 2019 ACM SIGIR International Conference on Theory of Information Retrieval, Santa Clara, CA, USA, 2–5 October 2019; pp. 75–78.
23. Rossi, F.; Conan-Guez, B. Functional multi-layer perceptron: A non-linear tool for functional data analysis. *Neural Netw.* **2005**, *18*, 45–60. [CrossRef] [PubMed]
24. Yao, J.; Mueller, J.; Wang, J.L. Deep learning for functional data analysis with adaptive basis layers. In Proceedings of the International Conference on Machine Learning, PMLR 2021, Virtual, 18–24 July 2021; pp. 11898–11908.
25. Amari, S. Backpropagation and stochastic gradient descent method. *Neurocomputing* **1993**, *5*, 185–196. [CrossRef]
26. Paszke, A.; Gross, S.; Massa, F.; Lerer, A.; Bradbury, J.; Chanan, G.; Killeen, T.; Lin, Z.; Gimelshein, N.; Antiga, L.; et al. Pytorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems*; MIT Press: Cambridge, MA, USA, 2019; Volume 32.
27. Buzzoni, M.; D'Elia, G.; Cocconcelli, M. A tool for validating and benchmarking signal processing techniques applied to machine diagnosis. *Mech. Syst. Signal Process.* **2020**, *139*, 106618. [CrossRef]
28. Feng, F.; Wang, X.; Li, R. Cross-modal retrieval with correspondence autoencoder. In Proceedings of the 22nd ACM International Conference on Multimedia, Orlando, FL, USA, 3–7 November 2014; pp. 7–16.
29. Ioffe, S.; Szegedy, C. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In Proceedings of the International Conference on Machine Learning, PMLR 2015, Lille, France, 7–9 July 2015; pp. 448–456.
30. Chadha, G.S.; Islam, I.; Schwung, A.; Ding, S.X. Deep convolutional clustering-based time series anomaly detection. *Sensors* **2021**, *21*, 5488. [CrossRef]
31. Shaw, C.T.; King, G.P. Using cluster analysis to classify time series. *Phys. D Nonlinear Phenom.* **1992**, *58*, 288–298. [CrossRef]
32. Japkowicz, N.; Shah, M. *Evaluating Learning Algorithms: A Classification Perspective*; Cambridge University Press: Cambridge, UK, 2011.
33. Cover, T.M.; Thomas, J.A. *Elements of Information Theory*; John Wiley Sons: Hoboken, NJ, USA, 2006.
34. Manning, C.D.; Raghavan, P.; Schütze, H. *Introduction to Information Retrieval*; Cambridge University Press: Cambridge, UK, 2008.
35. Pedregosa, F.; Varoquaux, G.; Gramfort, A.; Michel, V.; Thirion, B.; Grisel, O.; Blondel, M.; Prettenhofer, P.; Weiss, R.; Dubourg, V.; et al. *Scikit-Learn: Machine Learning in Python. J. Mach. Learn. Res.* **2011**, *12*, 2825–2830.
36. Fujiwara, T.; Sakamoto, N.; Nonaka, J.; Ma, K.L. A Visual Analytics Approach for Hardware System Monitoring with Streaming Functional Data Analysis. *IEEE Trans. Vis. Comput. Graph.* **2022**, *28*, 2338–2349.
37. Zhou, R.R.; Serban, N.; Gebraeel, N. Degradation modeling applied to residual lifetime prediction using functional data analysis. *Ann. Appl. Stat.* **2011**, *5*, 1586–1610. [CrossRef]