A Time-Series Data Generation Method to Predict Remaining Useful Life

Gilseung Ahn¹, Hyungseok Yun², Sun Hur² and Siyeong Lim³,*

1 Data Analytic Team 1, Hyundai Motors Company, Seoul 06797, Korea; GilseungAhn@hyundai.com
2 Department of Industrial and Management Engineering, Hanyang University, Ansan 15588, Korea; dbsgdttj3@gmail.com (H.Y.); hursun@hanyang.ac.kr (S.H.)
3 Korean Research Institute for Human Settlements, Sejong 30147, Korea
* Correspondence: limsy@krihs.re.kr

Abstract: Accurate predictions of remaining useful life (RUL) of equipment using machine learning (ML) or deep learning (DL) models that collect data until the equipment fails are crucial for maintenance scheduling. Because the data are unavailable until the equipment fails, collecting sufficient data to train a model without overfitting can be challenging. Here, we propose a method of generating time-series data for RUL models to resolve the problems posed by insufficient data. The proposed method converts every training time series into a sequence of alphabetical strings by symbolic aggregate approximation and identifies occurrence patterns in the converted sequences. The method then generates a new sequence and inversely transforms it to a new time series. Experiments with various RUL prediction datasets and ML/DL models verified that the proposed data-generation model can help avoid overfitting in RUL prediction model.

Keywords: remaining useful life prediction; data generation; symbolic aggregate approximation; run-to-failure

1. Introduction

Prognostics and health management (PHM) is an important contributor to maintaining manufacturing productivity and has rapidly evolved from corrective maintenance to condition-based maintenance (CBM). Also known as predictive or data-driven maintenance, CBM decisions are based on analysis of data gathered from equipment and sensors [1].

The established applications of CBM are monitoring, fault diagnosis, maintenance scheduling, and remaining useful life (RUL) prediction [2]. Among them, RUL prediction is the subject of growing interest because of its effectiveness in maintenance scheduling. The RUL can be defined as the difference between the current time and the time failure occurs, and RUL prediction is to predict RUL (or failure time) at the current time based on collected signals, condition, and so forth [3]. The RUL prediction techniques can be categorized into physical model-driven and data-driven techniques [4].

Zhu and Yang [5] computed the thermo-elasto-plastic stress and strain fields of turbine blade using finite element methods. They predicted the fatigue life through Basquin and Manson–Coffin formulae. They performed the thermal stress field analysis by using of Ansys software and then diagnosed thermal–mechanical stress for some nodes. Then, they predicted the fatigue life by maximum stress and mean stress. The remaining lifetime was predicted by extracting the parameter of the formula from controlled stress test data. While they introduced the influence of limited external factors such as thermal and mechanical stress on the remaining life, we consider the state as a time series and predicted the remaining life in a continuous situation probabilistically. Taheri and Taheri [6] studied the feasibility and technical design for implementing a combined heat and power system for the Mahshad Power plant. They figured out the remaining life for gas turbines on the basis of available data by approximated approach, whose prediction is derived by using...
predetermined elements or formulae. In addition, they used the operation data reflecting the current status for prediction, while our method in this paper adopts a probabilistic approach and uses obtainable data rather than deriving the close-formed formula.

Even though the physical model is easy to understand and very accurate, it is almost impossible to formulate modern industrial system. In contrast, the data-driven technique using machine learning (ML) and deep learning (DL) can formulate a very complex system with data collected from the system. For this reason, this paper focuses on data-driven RUL prediction.

Recently, ML and DL models have been frequently applied to predict RUL after training them with run-to-failure data [7]. These models learn the degradation patterns and relationships between the pattern and the RUL from the data collected until the end of the life of the components and predict the RUL of target components. In many recent studies, ML and DL models have shown superior performance for RUL problems [8–15].

Data insufficiency is a major challenge when training ML- or DL-based RUL models. Because a time-series sample is collected only after a component fails, collecting training data tends to be time-consuming and expensive. A model trained with insufficient data can be overfitted [16] and fail to accurately predict the RUL of a new component. Collecting more data can mitigate overfitting but may not be an option because of the required time and cost.

In this paper, we propose a time-series data-generation method that avoids overfitting when training an RUL model. The proposed method generates a sample on the basis of two existing samples called parents in a probabilistic manner and contributes to avoiding overfitting and increasing prediction performance. The generated samples will be added to training data for an RUL prediction model and contribute to avoid overfitting. In other words, an RUL prediction model is trained with union of original samples and generated samples. This approach is similar to SMOTE (Synthetic Minority Over-sampling Technique [17]), one of the most widely used oversampling methods that generates a minority class samples between two selected samples at random, and accordingly, the generated one may not be similar to the original samples in terms of relationship among features while contributing to the increase of classification performance such as recall and F1-score.

As far as our literature survey is concerned, ours is the first attempt to address the data insufficiency problem when training an RUL prediction model. In this paper, we propose a method to transform a time series into an alphabetical sequence and inversely transform the sequence that can be applied in time-series classification and clustering. The proposed data-generation method is not only efficient (i.e., it is inexpensive in terms of computational time) but also effective (i.e., the generated time series can help avoid overfitting).

The remainder of this paper is organized as follows. Section 2 introduces background theory and related works of RUL prediction and symbolic aggregate approximation. Section 3 proposes a time-series data-generation method for RUL prediction in detail, and Section 4 illustrates the process of the proposed method using a small example. Section 5 experimentally verifies the performance of the proposed method, and Section 6 draws conclusions and suggests future research directions.

2. Background Theory and Related Works

2.1. Remaining Useful Life Prediction

Training and usage process of the RUL prediction model proceeds as follows. First, a set \( X = \{x^{(1)}, x^{(2)}, \ldots, x^{(n)}\} \) of training time series samples \( x^{(i)}, i = 1, 2, \ldots, n \) is transformed by extracting features and the RUL such that:

\[
D(X) = \bigcup_{i=1}^{n} D(x^{(i)}),
\]

(1)
where $D(X)$ is a set of transformed time series in $X$, and $D(x^{(i)})$ is a set of pairs (feature, RUL) extracted from $x^{(i)}$:

$$D(x^{(i)}) = \left( \phi(x_{1:t}^{(i)}), y_t^{(i)} \right),$$

(2)

where $\phi$ is a vector of feature functions, $x_{1:t}^{(i)}$ is a part of the time series collected from time 1 to $t$ of sample $i$, $y_t^{(i)} = \frac{T_i - t}{T_i}$ is the RUL of sample $i$ measured at $t$, and $T_i$ is the length of $x^{(i)}$. Second, a model $f$ is trained using $D(X)$, and finally, the RUL of the new sample $n+1$ at $t$ is predicted as $f(\phi(x_{1:t}^{(n+1)}))$.

Many researchers have developed RUL prediction models based on ML or DL. For example, Ali et al. [18] introduced a root mean square entropy estimator (RMSEE), which is the entropy of the root mean square (RMS) of the $j^{th}$ windows, to capture bearing degradation and used it as a feature. They also converted an RUL prediction model into a classification problem with seven classes according to the degradation rate (that is, class 1: under 10%, class 2: 10%–25%, class 3: 25%–40%, class 4: 40%–55%, class 5: 55%–70%, class 6: 70%–85%, and class 7: 85%–100%). Finally, they used a multi-layered perceptron (MLP) combined with a simplified fuzzy adaptive resonance theory map as a prediction model. Zheng et al. [19] pointed out that a traditional regression model using features from a window is not appropriate for RUL prediction because it does not fully consider sequence information, and other sequence learning models have also flaws (e.g., hidden Markov model and recurrent neural networks do not consider long-term dependency among nodes). They proposed a deep long short-term memory (LSTM) consisting of four layers: an input layer, a multi-layer LSTM, a multi-layer perceptron (MLP), and an output layer. In their experiment, MLP, support vector regression (SVR), relevance vector regression, and a convolutional neural network were compared in terms of root mean squared error (RMSE), with the deep LSTM exhibiting the smallest RMSE for four datasets.

Table 1 summarizes previous research that developed RUL prediction models using ML and DL including convolution neural network (CNN), in terms of domain, feature, and base model. As seen Table 1, statistics such as RMS, kurtosis, and skewness are frequently used as features, and MLP, LSTM, and SVR are used primarily for models.

### Table 1. Previous studies on RUL prediction.

| Research | Domain | Feature | Base Model |
|----------|--------|---------|------------|
| [18]     | Bearing| RMSEE   | MLP        |
| [19]     | General| Original| LSTM       |
| [20]     | Bearing| Original| CNN        |
| [21]     | Rotating machine | RMS, kurtosis, etc. | MLP |
| [22]     | Bearing| Spectral kurtosis | SVR |
| [23]     | Data   | RMS, peak, kurtosis, crest factor, etc. | SVR |
| [24]     | Bearing| RMS and kurtosis | LSTM |

General: not considering specific industry domain; original: not using specific feature functions but using raw values.

### 2.2. Symbolic Aggregate Approximation

Pattern extraction is very important for time series analysis such as classification and clustering, but it is hard to extract patterns directly from time series due to huge search space [25]. Thus, representation methods such as symbolic aggregate approximation (SAX), discrete cosine transform (DCT), and discrete wavelet transform (DWT) are usually used to represent time series as sequence before pattern extraction. In this paper, each time series is discretized as alphabet sequence using SAX, patterns are extracted from the sequences, and a new sequence is generated considering the extracted pattern.

SAX can convert a time series $x$ into an alphabetical sequence $S$ for efficient time-series data mining in the following manner [26]. First, the $t^{th}$ element $x^t$ of $x$ is normalized as $\tilde{x}^t = \frac{x^t - \mu}{\sigma}$, where $\mu$ and $\sigma$ denote the mean and standard deviation of $x$, respectively.
Second, $\hat{x}^d$ is split into $w$ windows and $r = (r_1, r_2, \ldots, r_w)$, where $r_j, 1 \leq j \leq w$ is a representative value such as the mean of the values in the $j$th window, is calculated. One can introduce the standard deviation as the alternative representative value. Third, break points $\beta_k$ for $k = 1, 2, \ldots, l$ are computed, where $l$ is the number of alphabetical strings defined by the user, satisfying $\Pr(\beta_k \leq r_j < \beta_{k+1}) = 1/l$. Finally, an alphabet is assigned to each window on the basis of the break points. That is, if $\beta_k \leq r_j < \beta_{k+1}$, then the $k$th alphabet string is assigned to the $j$th window.

Figure 1 illustrates a SAX application process when the time series is assumed to be normalized. Time series is split into $w = 6$ windows, and mean values $r_1, r_2, \ldots, r_6$ in each window are calculated. Three ($l = 3$) alphabetical strings $A$, $B$, and $C$ are introduced, and then three break points $\beta_{\mu,1}, \beta_{\mu,2}, \beta_{\mu,3}$ for means are obtained. Then, the alphabet $A$ is assigned to those means less than $\beta_{\mu,1}$, $C$ to those bigger than $\beta_{\mu,2}$, and $B$ to those between $\beta_{\mu,1}$ and $\beta_{\mu,2}$. Then, we obtain a sequence $S = C - B - A - C - B - C$ of alphabetical strings that is converted from the time series.

![Figure 1. Symbolic aggregate approximation.](image)

SAX has been used to extract features from time series for various tasks such as classification and clustering. For example, Georgoulas et al. [27] extracted alphabetical features to represent the vibration of bearings and used them to detect bearing faults with various classifiers. Park and Jung [28] proposed a method to reveal rules from multivariate time series. It transforms a time series to an alphabetical sequence through SAX and identifies frequent patterns from the sequences using association rule mining. Notaristefano et al. [29] made groups of electrical load pattern by reducing data size using SAX.

3. Proposed Data Generation Method

This section explains the proposed three-phase data-generation method: preprocessing, generating an alphabetical sequence, and generating time-series values. In the first phase, every time-series sample is transformed into a pair of vectors, one of window means and another of window standard deviations, and then each vector is transformed into an alphabetical sequence. In the second phase, two arbitrarily selected pairs of alphabetical sequences form a new sequence pair, with a pattern similar to those of the originally selected sequences. In the third phase, time-series values for each window are generated from the generated pair. Table 2 presents the mathematical notations used in this paper.
Table 2. Mathematical notations used in this paper.

| Notation | Meaning |
|----------|---------|
| $x^{(i)}$ | Time series sample $i$ ($i = 1, 2, \cdots, n$), $x^{(i)} = (x_1^{(i)}, x_2^{(i)}, \cdots, x_{T_i}^{(i)})$, where $x_t^{(i)}$ denotes value measured at $t$, and $T_i$ denotes length of $x^{(i)}$ (i.e., whole life of $x^{(i)}$). |
| $y_t^{(i)}$ | RUL of $x^{(i)}$ at $t$, where $y_t^{(i)} = \frac{T_i - t}{T_i}$. |
| $\Phi$ | Vector of feature functions to extract features from $x^{(i)}$ to train an RUL prediction model. |
| $D(x^{(i)})$ | Dataset generated by transforming $x^{(i)}$ for RUL prediction, where the bottom 30% and top 30% are trimmed for stability, that is, $D(x^{(i)}) = \{ (\Phi(x_{1,t}^{(i)}), y_t) \mid T_i \times 0.3 \leq t \leq [T_i \times 0.7] \}$. |
| $f$ | Supervised model for RUL prediction. |
| $z(k)$ | Generated time-series $k$ ($k = 1, 2, \cdots, m$), $z(k) = (z_1^{(k)}, z_2^{(k)}, \cdots, z_{T_k}^{(k)})$ where $T_k$ denotes length of $z^{(k)}$ (i.e., whole life of $z^{(k)}$). |
| $\mu_i, s_i$ | Mean and standard deviation of $x^{(i)}$. |
| $\hat{x}^{(i)}$ | Normalized $x^{(i)}$ with mean $\mu_i$ and standard deviation $s_i$. |
| $w$ | Number of windows. |
| $\mu^{(i)}$ | Window mean vector $i$, $\mu^{(i)} = (\mu_1^{(i)}, \mu_2^{(i)}, \cdots, \mu_w^{(i)})$, where $\mu_i^{(i)}$ denotes mean of $x_i^{(i)}$ in the $j$th window. |
| $\sigma^{(i)}$ | Window standard deviation vector $i$, $\sigma^{(i)} = (\sigma_1^{(i)}, \sigma_2^{(i)}, \cdots, \sigma_w^{(i)})$, where $\sigma_j^{(i)}$ denotes standard deviation of $\hat{x}_j^{(i)}$ in the $j$th window. |
| $A_{\mu}$ | Alphabetical set for $\mu^{(i)}$, $A_{\mu} = \{ A_{\mu,1}, A_{\mu,2}, \cdots, A_{\mu,|A_{\mu}|} \}$. |
| $A_{\sigma}$ | Alphabetical set for $\sigma^{(i)}$, $A_{\sigma} = \{ A_{\sigma,1}, A_{\sigma,2}, \cdots, A_{\sigma,|A_{\sigma}|} \}$. |
| $\beta_{\mu,q}$ | The $q^{th}$ ($q = 1, \cdots, |A_{\mu}|$) break point for $\mu_i^{(i)}$. |
| $\beta_{\sigma,q}$ | The $q^{th}$ ($q = 1, \cdots, |A_{\sigma}|$) break point for $\sigma_i^{(i)}$. |
| $\mathcal{M}^{(i)}$ | Alphabetical sequence of window mean vector $i$, $\mathcal{M}^{(i)} = (\mathcal{M}_1^{(i)}, \mathcal{M}_2^{(i)}, \cdots, \mathcal{M}_w^{(i)})$, where $\mathcal{M}_j^{(i)}$ denotes alphabet for $\mu_j^{(i)}$. |
| $\mathcal{S}^{(i)}$ | Alphabetical sequence of window standard deviation vector $i$, $\mathcal{S}^{(i)} = (\mathcal{S}_1^{(i)}, \mathcal{S}_2^{(i)}, \cdots, \mathcal{S}_w^{(i)})$, where $\mathcal{S}_j^{(i)}$ denotes alphabet for $\sigma_j^{(i)}$. |

3.1. Preprocessing

The objective of the preprocessing phase is to express $x^{(i)}$ as a tuple $(\mathcal{M}^{(i)}, \mathcal{S}^{(i)})$, where $\mathcal{M}^{(i)}$ and $\mathcal{S}^{(i)}$ are window mean vector $i$ and window standard deviation vector $i$, respectively. The preprocessing phase consists of four steps: $z$-normalization, segmentation, calculation of break points, and conversion into an alphabetical sequence, as illustrated in Figure 2.

In the first step, $x^{(i)}$ for $i = 1, 2, \cdots, n$ is normalized to $\hat{x}^{(i)}$ with its mean $\mu_i$ and standard deviation $s_i$ as:

$$
\hat{x}_t^{(i)} = \frac{x_t^{(i)} - \mu_i^{(i)}}{s_i^{(i)}}.
$$

(3)

In the second step, $\hat{x}^{(i)}$ for $i = 1, 2, \cdots, n$ is split into $w$ windows, where $w$ is the number of windows set by the user, and the mean and standard deviation of each window are calculated by:
Processes 2021, 9, 1115

\[ \mu_j^{(i)} = \frac{\sum_{i=j \times n_{ij} + 1}^{(j+1) \times n_{ij}} \tilde{x}_j^{(i)}}{n_{ij}}, \]

\[ \sigma_j^{(i)} = \sqrt{\frac{\sum_{i=j \times n_{ij} + 1}^{(j+1) \times n_{ij}} (\tilde{x}_j^{(i)} - \mu_j^{(i)})^2}{n_{ij} - 1}}, \]

where \( n_{ij} \) is the number of elements in the \( j \)th \((j = 1, 2, \cdots, w)\) window of \( \tilde{x}^{(i)} \), which equals \( T_j/w \) when \( j \neq w \), and \( \tilde{x}^{(i)} \) is expressed as a pair of vectors, one of window mean vector \( \mu^{(i)} = (\mu_1^{(i)}, \mu_2^{(i)}, \cdots, \mu_w^{(i)}) \), and another of window standard deviation vector \( \sigma^{(i)} = (\sigma_1^{(i)}, \sigma_2^{(i)}, \cdots, \sigma_w^{(i)}) \).

In the third step, break points \( \beta_\mu \) and \( \beta_\sigma \) for \( \mu_j^{(i)} \) and \( \sigma_j^{(i)} \) \((i = 1, \cdots, n, j = 1, \cdots, w)\), respectively, are obtained according to the size of the set of the alphabetical strings, \(|A|\), which is also the user parameter. As explained in Section 2.2, the break points \( \beta_\mu \) and \( \beta_\sigma \) are used as criterion to convert mean and standard deviation of each window into alphabets. \( \beta_\mu \) and \( \beta_\sigma \) are calculated and used not for individual samples but for all samples to consider the sample’s scale when generating new sample. The break point is obtained from \( \Pr(\beta_q \leq x < \beta_{q+1}) = \frac{q}{|A|} \), which implies that each interval \([\beta_q, \beta_{q+1}]\), \( q = 1, 2, \cdots, |A| - 1 \), contains the same number of values, and \( \beta_{|A|} \) is therefore \( \frac{|A|}{|A|} \times 100\% \).

In the fourth step, \( \mu^{(i)} \) and \( \sigma^{(i)} \) are expressed as alphabetical sequences \( M^{(i)} \) and \( S^{(i)} \), respectively, as follows:

\[
M_j^{(i)} = \begin{cases} 
A_{\mu, 1}, & \beta_{\mu, 0} < \mu_j^{(i)} \leq \beta_{\mu, 1} \\
A_{\mu, 2}, & \beta_{\mu, 1} < \mu_j^{(i)} \leq \beta_{\mu, 2} \\
\vdots & \vdots \\
A_{\mu, |A_{\mu}|}, & \beta_{\mu, |A_{\mu}|-1} < \mu_j^{(i)} \leq \beta_{\mu, |A_{\mu}|} 
\end{cases}
\]
\[ S_j^{(i)} = \begin{cases} 
    \mathcal{A}_v,1, & \beta_v,0 < c_j^{(i)} \leq \beta_v,1 \\
    \mathcal{A}_v,2, & \beta_v,1 < c_j^{(i)} \leq \beta_v,2 \\
    \vdots & \vdots \\
    \mathcal{A}_v,|\mathcal{A}_v|, & \beta_v,|\mathcal{A}_v| - 1 < c_j^{(i)} \leq \beta_v,|\mathcal{A}_v| 
\end{cases}, \]  

(7)

where \( \beta_v,0 \) and \( \beta_v,0 \) are negative infinity, and \( \mathcal{A}_{\mu,i} \) and \( \mathcal{A}_{\nu,j} \) indicate the \( i \)-th predefined alphabet (e.g., \( \mathcal{A}_{\mu,1} = \mathcal{A}, \mathcal{A}_{\nu,A} = \) for the window mean and standard deviation, respectively.

The first phase is summarized in Algorithm A1 in the Appendix A.

3.2. Generating Alphabetical Sequences

The second phase is to generate artificial sequences of alphabets for \( \mathcal{M}^{(k)} \) and \( \mathcal{S}^{(k)} \) on the basis of two randomly selected parental samples \( \left( \mathcal{M}^{(a)}, \mathcal{S}^{(a)} \right) \) and \( \left( \mathcal{M}^{(b)}, \mathcal{S}^{(b)} \right) \). Figure 3 illustrates an example of the generating process of alphabetical sequences \( \mathcal{M}^{(k)} \) based on \( \mathcal{M}^{(a)} = \text{ABCB} \) and \( \mathcal{M}^{(b)} = \text{BBB} \). Note that generating process of \( \mathcal{S}^{(k)} \) is the same as the process of \( \mathcal{M}^{(k)} \).

![Figure 3. Generating process of alphabetical sequences.](image)

In this figure, an edge from alphabet A to B denotes that alphabet A impacts on generating alphabet B.

Sequence \( \mathcal{M}^{(k)} \) for the window mean is generated by sequentially selecting an element from either \( \mathcal{M}_j^{(a)} \) and \( \mathcal{M}_j^{(b)} \) on the basis of the probabilities \( \text{Pr} \left( \mathcal{M}_j^{(a)} | \mathcal{M}_{j-1}^{(k)} \right) \) and \( \text{Pr} \left( \mathcal{M}_j^{(b)} | \mathcal{M}_{j-1}^{(k)} \right) \), where \( \mathcal{M}_j^{(k)} = \left( \mathcal{M}_1^{(k)}, \mathcal{M}_2^{(k)}, \ldots, \mathcal{M}_j^{(k)} \right) \) is a partial sequence of \( \mathcal{M}^{(k)} \). Sequence \( \mathcal{S}^{(k)} \) for the window standard deviation is similarly generated on the basis of \( \text{Pr} \left( \mathcal{S}_j^{(a)} | \mathcal{S}_{j-1}^{(k)} \right) \) and \( \text{Pr} \left( \mathcal{S}_j^{(b)} | \mathcal{S}_{j-1}^{(k)} \right) \). \( \mathcal{M}^{(k)} \) and \( \mathcal{S}^{(k)} \) are assumed to be independent of each other for simplicity.

Probabilities \( \text{Pr} \left( \mathcal{M}_1^{(a)} \right) \) and \( \text{Pr} \left( \mathcal{M}_1^{(b)} \right) \), which are used to select the first element \( \mathcal{M}_1^{(k)} \), can be calculated by:

\[
\text{Pr} \left( \mathcal{M}_1^{(a)} \right) = \frac{\sum_{i=1}^{n} I \left( \mathcal{M}_1^{(i)} = \mathcal{M}_1^{(a)} \right) + \alpha}{n + \alpha \times |\mathcal{A}_\mu|},
\]

(8)

\[
\text{Pr} \left( \mathcal{M}_1^{(b)} \right) = \frac{\sum_{i=1}^{n} I \left( \mathcal{M}_1^{(i)} = \mathcal{M}_1^{(b)} \right) + \alpha}{n + \alpha \times |\mathcal{A}_\mu|},
\]

(9)

where \( I \left( \text{condition} \right) \) is an indicator function, which returns 1 if \text{condition} is satisfied, and 0, otherwise, and \( \alpha \geq 0 \) is a Laplace smoothing parameter to prevent \( \text{Pr} \left( \mathcal{M}_1^{(a)} \right) \) and
\( \Pr(M_1^{(b)}) \) from becoming either zero or one. That is, \( \Pr(M_1^{(a)}) \) and \( \Pr(M_1^{(b)}) \) are calculated as smoothed ratio of \( M_1^{(a)} \) and \( M_1^{(b)} \) among the first alphabets of mean sequences, respectively.

Because \( M_1^{(a)} \) and \( M_1^{(b)} \) are selected with probabilities in (8) and (9), respectively, these probabilities should be normalized by dividing each by their sum as presented in Equations (10) and (11):

\[
\tilde{\Pr}(M_1^{(a)}) = \frac{\Pr(M_1^{(a)})}{\Pr(M_1^{(a)}) + \Pr(M_1^{(b)})},
\]
\[
\tilde{\Pr}(M_1^{(b)}) = \frac{\Pr(M_1^{(b)})}{\Pr(M_1^{(a)}) + \Pr(M_1^{(b)})}
\]

Probabilities, \( \Pr(M_1^{(a)}|M_1^{(k)}_{1:j-1}) \) and \( \Pr(M_1^{(b)}|M_1^{(k)}_{1:j-1}) \), used to select the \( j \)th (2 \( \leq j \leq w \) element \( M_1^{(k)} \)), can be calculated using the Markovian assumption:

\[
\Pr(M_1^{(a)}|M_1^{(k)}_{1:j-1}) = \Pr(M_1^{(a)}|M_1^{(k)}_{j-1}),
\]
\[
\Pr(M_1^{(b)}|M_1^{(k)}_{1:j-1}) = \Pr(M_1^{(b)}|M_1^{(k)}_{j-1})
\]

where \( \Pr(M_1^{(a)}|M_1^{(k)}_{j-1}) \) and \( \Pr(M_1^{(b)}|M_1^{(k)}_{j-1}) \) are calculated by:

\[
\Pr(M_1^{(a)}|M_1^{(k)}_{1:j-1}) = \frac{\sum_{i=1}^{n} \sum_{b=\min(w,j-1)}^{\max(w,j-1)} I((M_1^{(a)} = M_1^{(k)}_{j-1}) \text{ and } (M_1^{(b)} = M_1^{(i)}_{j-1}))}{\sum_{i=1}^{n} \sum_{b=\min(w,j-1)}^{\max(w,j-1)} I(M_1^{(b)} = M_1^{(k)}_{j-1})},
\]
\[
\Pr(M_1^{(b)}|M_1^{(k)}_{1:j-1}) = \frac{\sum_{i=1}^{n} \sum_{b=\min(w,j-1)}^{\max(w,j-1)} I((M_1^{(a)} = M_1^{(k)}_{j-1}) \text{ and } (M_1^{(b)} = M_1^{(i)}_{j-1}))}{\sum_{i=1}^{n} \sum_{b=\min(w,j-1)}^{\max(w,j-1)} I(M_1^{(b)} = M_1^{(k)}_{j-1})}
\]

In Equations (14) and (15), \( L \) is a parameter to restrict the search space to determine the number of alphabets that match \( M_1^{(k)}_{1:j-1}, M_1^{(a)}, M_1^{(b)} \) in \( M_1^{(i)} \) for all values of \( i \). The parent sample is randomly selected, and to produce the next sample, we adopt the Markov process for randomness and variability. In the Markov assumption, we can add more variability by properly setting the value of \( L \) representing the size of the search space (Equations (14) and (15)). These probabilities should be normalized as follows:

\[
\tilde{\Pr}(M_1^{(a)}|M_1^{(k)}_{1:j-1}) = \frac{\Pr(M_1^{(a)}|M_1^{(k)}_{1:j-1})}{\Pr(M_1^{(a)}|M_1^{(k)}_{1:j-1}) + \Pr(M_1^{(b)}|M_1^{(k)}_{1:j-1})},
\]
\[
\tilde{\Pr}(M_1^{(b)}|M_1^{(k)}_{1:j-1}) = \frac{\Pr(M_1^{(b)}|M_1^{(k)}_{1:j-1})}{\Pr(M_1^{(a)}|M_1^{(k)}_{1:j-1}) + \Pr(M_1^{(b)}|M_1^{(k)}_{1:j-1})}
\]

An algorithm to generate alphabetical sequences of an artificial sample’s mean and standard deviation is presented as Algorithm A2. It is based on sampling from categorical distribution to select either \( a \) or \( b \). For example, the first element of \( M_1^{(k)} \) follows the categorical distribution \( C(M_1^{(a)}, M_1^{(b)}|\tilde{\Pr}(M_1^{(a)}), \tilde{\Pr}(M_1^{(b)})) \), implying that \( M_1^{(k)} \) is selected from \( M_1^{(a)} \) and \( M_1^{(b)} \) with probabilities \( \tilde{\Pr}(M_1^{(a)}) \) and \( \tilde{\Pr}(M_1^{(b)}) \).
3.3. Generating Time-Series Values

In this phase, time-series values $z_j^{(k)}$ in window $j$ are generated from $M_j^{(k)}$ and $S_j^{(k)}$ for $j = 1, 2, \ldots, w$. Let $M_j^{(k)}$ be $A_{\mu, r_1}$ implying that $\beta_{\mu, r_1 - 1} < \mu_j^{(k)} \leq \beta_{\mu, r_1}$, and $S_j^{(k)}$ be $A_{\sigma, r_2}$ implying that $\beta_{\sigma, r_2 - 1} < \sigma_j^{(k)} \leq \beta_{\sigma, r_2}$. We assume that $z_j^{(k)}$ follows a normal distribution, with the mean $\mu_j^{(k)}$ and standard deviation $\sigma_j^{(k)}$, and $\mu_j^{(k)}$ and $\sigma_j^{(k)}$ uniformly distributed in $[\beta_{\mu, r_1 - 1}, \beta_{\mu, r_1}]$ and $[\beta_{\sigma, r_2 - 1}, \beta_{\sigma, r_2}]$, respectively, when $r_1$ and $r_2$ are not 1. When $r_1$ is 1, we set $\mu_j^{(k)} = \beta_{\mu, r_1}$, and $\sigma_j^{(k)} = \beta_{\sigma, r_2}$ when $r_2$ is 1. We also assume that the length of $z_j^{(k)}$ follows a uniform distribution in $[\min(\frac{T_s}{w}, \frac{T_t}{w}), \max(\frac{T_s}{w}, \frac{T_t}{w})]$, where $a$ and $b$ are indices of the parents of $k$, and $\frac{T_s}{w}$ and $\frac{T_t}{w}$ denote the number of elements in each window of $x^{(a)}$ and $x^{(b)}$, respectively.

After a $z_j^{(k)}$ for every value of $j$ is generated, it should be inversely transformed to $z_j^{(k)}$ using its mean $\bar{z}^{(k)}$ and standard deviation $s^{(k)}$. We set them as a weighted average of $\bar{x}^{(a)}$ and $\bar{x}^{(b)}$, and $s^{(a)}$ and $s^{(b)}$, respectively, as presented Equations (18) and (19):

$$\bar{z}^{(k)} = r \times \bar{x}^{(a)} + (1 - r) \times \bar{x}^{(b)}, \quad (18)$$

$$s^{(k)} = r \times s^{(a)} + (1 - r) \times s^{(b)}, \quad (19)$$

where $0 < r < 1$ is randomly chosen. That is, Equation (18) means $\bar{z}^{(k)}$ is randomly selected between $\bar{x}^{(a)}$ and $\bar{x}^{(b)}$, and Equation (19) means $s^{(k)}$ is randomly selected between $s^{(a)}$ and $s^{(b)}$.

4. Numerical Example

This section describes an example of how a new time series $z^{(k)}$ is generated using the proposed method. Table 3 shows example dataset $X$, which consists of five samples. Each sample is a time series collected from start to failure of a part by a sensor. In each sample, there are 18 numeric data, implying that life of each corresponding part is 18. Please note that the proposed method can be applied to the data that includes samples with different lengths.

| Index | Data | Mean | Standard Deviation |
|-------|------|------|-------------------|
| 1     | (10, 10, 11, 11, 12, 13, 13, 14, 13, 13, 13, 14, 15, 14, 15, 16, 16) | 13.17 | 1.77 |
| 2     | (10, 9, 9, 9, 10, 9, 10, 10, 10, 10, 9, 8, 9, 9, 9, 8, 7, 7, 7) | 8.83 | 1.01 |
| 3     | (11, 11, 12, 13, 12, 12, 12, 11, 11, 10, 9, 8, 9, 10, 10, 10, 10) | 10.56 | 1.34 |
| 4     | (9, 8, 9, 9, 8, 9, 9, 10, 11, 11, 11, 10, 10, 10, 9, 8) | 9.39 | 0.95 |
| 5     | (11, 11, 11, 12, 12, 12, 12, 12, 12, 12, 11, 12, 13, 13, 14, 15, 15) | 12.33 | 1.25 |

Phase 1. Preprocessing

(1) z-normalization

Each sample in Table 3 is normalized according to its mean and standard deviation as follows.

$$\bar{z}^{(1)} = (-1.79, -1.79, -1.22, -1.22, -0.66, -0.09, -0.09, 0.47, -0.09, -0.09, -0.09, 0.47, 1.03, 0.47, 0.47, 1.03, 1.6, 1.6),$$

$$\bar{z}^{(2)} = (1.15, 0.16, 0.16, 1.15, 0.16, 1.15, 1.15, 1.15, 0.16, 0.16, -0.82, 0.16, 0.16, -0.82, -1.81, -1.81, -1.81),$$
\[ \tilde{x}^{(3)} = (0.33, 0.33, 1.08, 1.82, 1.08, 1.08, 0.33, 0.33, 0.33, -0.41, -1.16, -1.9, -1.16, -1.16, -0.41, -1.16, -0.41), \]
\[ \tilde{x}^{(4)} = (-0.41, -1.46, -0.41, -0.41, -0.41, -1.46, -0.41, -0.41, -0.41, 0.64, 1.69, 1.69, 1.69, 0.64, 0.64, 0.64, -0.41, -1.46), \]
\[ \tilde{x}^{(5)} = (-1.07, -1.07, -1.07, -0.27, -0.27, -0.27, -0.27, 0.53, -0.27, -1.07, -1.07, -0.27, -0.27, 1.34, 2.14, 2.14). \]

(2) Segmentation

Each normalized sample is split into \( w \) (\( w = 6 \)) windows, and the mean and standard deviation of each window are calculated. For example, the first window of \( \tilde{x}^{(2)} \) is \((1.15, 0.16, 0.16)\) and its mean and standard deviation \( \mu^{(2)} \) and \( \sigma^{(2)} \) are 0.49 and 0.47, respectively.

Values for \( \mu^{(i)} \) and \( \sigma^{(i)} \) for all \( i \) and \( j \) are calculated as follows.

\[ \mu^{(1)} = (-1.60, -0.66, 0.10, 0.10, 0.66, 1.41), \sigma^{(1)} = (0.27, 0.46, 0.26, 0.26, 0.26, 0.27), \]
\[ \mu^{(2)} = (0.49, 0.49, 1.15, -0.17, -0.17, -1.81), \sigma^{(2)} = (0.47, 0.47, 0.00, 0.46, 0.46, 0.00), \]
\[ \mu^{(3)} = (0.58, 1.33, 0.58, -0.41, -1.41, -0.66), \sigma^{(3)} = (0.35, 0.35, 0.35, 0.61, 0.35, 0.35), \]
\[ \mu^{(4)} = (-0.76, -0.76, -0.41, 1.34, 0.99, -0.41), \sigma^{(4)} = (0.49, 0.49, 0.00, 0.49, 0.49, 0.86), \]
\[ \mu^{(5)} = (-1.07, -0.27, -0.00, -0.80, 0.26, 1.87), \sigma^{(5)} = (0.00, 0.00, 0.38, 0.38, 0.38, 0.38). \]

(3) Calculation of break points

We set \( |A_{\mu}| = |A_{\sigma}| = 3 \), and break points \( \beta_{\mu, j} \) and \( \beta_{\sigma, j} \) are those that divide the values of all values of \( \mu^{(i)} \) and \( \sigma^{(i)} \), respectively, into three equal parts. For example, \( \beta_{\mu, 1} = -0.41 \) is 1/3 quantile of \((-1.60, -0.66, 0.10, 0.10, 0.66, 1.41)\), \( \beta_{\mu, 2} = -0.17 \) is 1/3 quantile of \((-0.17, -1.81, 0.58, 1.33, 0.58, -0.41)\), \( \beta_{\mu, 3} = -0.66 \) is 1/3 quantile of \((-0.66, -0.76, -0.41, 1.34, 0.99, -0.41)\). In this manner, all break points can be calculated by:

\[ \beta_{\mu} = (\beta_{\mu, 1}, \beta_{\mu, 2}, \beta_{\mu, 3}) = (-0.41, 0.49, 1.87), \]
\[ \beta_{\sigma} = (\beta_{\sigma, 1}, \beta_{\sigma, 2}, \beta_{\sigma, 3}) = (0.32, 0.46, 0.86). \]

(4) Conversion into an alphabetical sequence

On the basis of \( \beta_{\mu} \) and \( \beta_{\sigma} \), \( \mu^{(i)} \) and \( \sigma^{(i)} \) are converted to an alphabetical sequence \( \mathcal{M}^{(i)} \) and \( \mathcal{S}^{(i)} \) for all \( i \), respectively. For example, \( \mu^{(1)} = (-1.60) \) is converted as alphabet \( \AA \) because \( \beta_{\mu, 1} = -0.41 < -1.60 < \beta_{\mu, 1} \), and \( \mu^{(3)} = 0.10 \) is converted as alphabet \( \BB \) because \( \beta_{\mu, 1} < 0.10 < \beta_{\mu, 2} \). Thus, \( \mu^{(1)} = (-1.60, -0.66, 0.10, 0.10, 0.66, 1.41) \) is converted as \( \AA \BB \CC \BB \CC \CC \AA \CC \BB \AA \BB \AA \BB \AA \BB \AA \BB \). \( \mathcal{M}^{(i)} \) and \( \mathcal{S}^{(i)} \) for all \( i \) are obtained as follows.

\[ \mathcal{M}^{(1)} = \AA \BB \CC \CC \BB \AA, \]
\[ \mathcal{M}^{(2)} = \CC \CC \BB \AA, \]
\[ \mathcal{M}^{(3)} = \CC \CC \BB \AA, \]
\[ \mathcal{M}^{(4)} = \AA \BB \CC \BB \AA, \]
\[ \mathcal{M}^{(5)} = \BB \AA \BB \CC \AA, \]
\[ \mathcal{S}^{(1)} = \AA \BB \CC \CC \AA, \]
\[ \mathcal{S}^{(2)} = \CC \CC \BB \AA, \]
\[ \mathcal{S}^{(3)} = \CC \CC \BB \AA, \]
\[ \mathcal{S}^{(4)} = \AA \BB \CC \BB \AA, \]
\[ \mathcal{S}^{(5)} = \BB \AA \BB \CC \AA. \]

Phase 2. Generating an alphabetical sequence

Suppose sample 1 and 3 are randomly selected as parents. The first mean alphabetical strings of sample 1 and 3 are \( \AA \) and \( \CC \), and the first standard deviation alphabetical strings are \( \DD \) and \( \DD \). Thus, \( \Pr(\AA), \Pr(\CC), \Pr(\DD), \) and \( \Pr(\DD) \) should be calculated and normalized using Equations (6)–(9).

\[ \Pr(\AA) = \frac{3}{5}, \Pr(\CC) = \frac{2}{5}, \Pr(\AA) = \frac{3}{5}, \Pr(\CC) = \frac{2}{5} = \frac{2}{5} \]
\[ \Pr(\DD) = \frac{2}{5}, \Pr(\DD) = \frac{1}{5}. \]

\( \mathcal{M}_{1}^{(k)} \) is sampling from \( C(\AA, \CC, \DD) \), and \( \mathcal{S}_{1}^{(k)} \) is sampling from \( C(\DD, \DD, \DD) \), and as a result, \( \AA \) and \( \CC \) are selected.
The second mean alphabets of sample 1 and 3 are $\Lambda$ and $\mathbb{C}$, and the second standard deviation alphabets are $\Lambda$ and $\mathbb{C}$. Therefore, $\Pr(\Lambda|\Lambda)$, $\Pr(\mathbb{C}|\Lambda)$, $\Pr(\mathbb{D})$, and $\Pr(\mathbb{Z})$ should be calculated and normalized using Equations (10)–(15). For convenience, we set $L$ to 1, and accordingly, $M_{1,3}^{(i)}$ and $S_{1,3}^{(i)}$ for all $i$ were used for the calculation.

$$
\Pr(\Lambda|\Lambda) = \frac{2}{3}, \quad \Pr(\mathbb{C}|\Lambda) = \frac{0}{3}, \quad \Pr(\Lambda|\mathbb{C}) = \frac{\frac{2}{3} + 0}{2} = 1, \quad \Pr(\mathbb{C}|\mathbb{C}) = \frac{0}{\frac{2}{3} + 0} = 0
$$

$$
\Pr(\mathbb{Z}) = \frac{1}{3}, \quad \Pr(\mathbb{D}) = \frac{1}{3}, \quad \Pr(\mathbb{Z}) = \frac{\frac{1}{3} + \frac{1}{2}}{2} = \frac{1}{2}, \quad \Pr(\mathbb{D}) = \frac{\frac{1}{3} + \frac{1}{2}}{2} = \frac{1}{2}.
$$

$M^{(k)}_{1}$ is sampling from $C(\Lambda, \mathbb{C}|1, 0)$, and $S^{(k)}_{1,2}$ is sampling from $C\left(\frac{1}{2}, \frac{1}{2}\right)$, and as a result, $\Lambda$ and $\mathbb{C}$ are selected. This process repeats until $j$ becomes $w = 6$.

From this phase, we obtain $M^{(k)} = \Lambda\Lambda\mathbb{B}\mathbb{C}\mathbb{C}$ and $S^{(k)} = \mathbb{D}$.

**Phase 3. Generating time-series values**

In this phase, $z^{(k)}$ are generated from $M^{(k)}_{j}$ and $S^{(k)}_{j}$ for $j = 1, 2, \cdots, 6$, and we obtain $z^{(k)} = (-0.6, -0.11, 0.16, -0.23, -1.32, -0.12, -0.04, 0.44, 0.66, 0.64, 1.23, 0.39, 0.88, 0.9, 1.86, 0.45, -0.2, 1.09)$.

The generation process for the first window ($j = 1$) is as follows. Since $M^{(k)}_{1} = \Lambda$ and $S^{(k)}_{1} = \mathbb{D}$, $U\left[\min\left(\frac{T_1}{10}, \frac{T_2}{10}\right), \max\left(\frac{T_3}{10}, \frac{T_4}{10}\right)\right] = U[3, 3] = 3$ samples are generated, where each sample follows a normal distribution with a mean $\beta_{\mu, 1} = -0.41$ and standard deviation $\beta_{\sigma, 1} = 0.32$, because $M^{(k)}_{j}$ and $S^{(k)}_{j}$ are the first alphabets, and we use the constant mean and standard deviation. As a result, we obtain $\bar{z}^{(k)}_1 = (-0.60, -0.11, 0.16)$. As an another example, for the fourth window ($j = 4$), $M^{(k)}_4 = \mathbb{C}$ and $S^{(k)}_4 = \mathbb{D}$, and three samples are generated if each sample follows a normal distribution with a mean in $U[\beta_{\mu, 1} = -0.41, \beta_{\mu, 2} = 0.49]$ and standard deviation $\sigma$ in $U[\beta_{\sigma, 1} = 0.32, \beta_{\sigma, 3} = 0.46]$. As a result, we obtain $\bar{z}^{(k)}_4 = (0.88, 0.90, 1.86)$.

Finally, $z^{(k)}$ is obtained by inversely transforming $\bar{z}^{(k)}_j$ with $\bar{z}^{(k)}_j = 0.65 \times z^{(1)}_j + (1 - 0.65) \times z^{(3)}_j = 12.25$ and $s^{(k)}_j = 0.82 \times x^{(1)}_j + (1 - 0.18) \times x^{(3)}_j = 1.42$, where $r = 0.65$ and 0.18 are randomly chosen weights.

$z^{(k)} = (11.4, 12.1, 12.48, 11.93, 10.38, 12.08, 12.2, 12.88, 13.19, 13.16, 14.00, 12.81, 13.5, 13.53, 14.89, 12.89, 11.97, 13.80)$.

Figure 4 shows the generated sample and its parents (sample 1 and 3) of the example. Dashed and dotted lines denote samples 1 and 3 in Table 3, respectively, and solid line denotes the generated sample when the parents are the sample 1 and 3. Y-axis denotes the sensor value and X-axis denotes time, and thus the horizontal length of a line indicates the whole life. As explained before, the length (i.e., whole life) of every sample in X is 18, and thus the length of the generated sample is also 18. To be more accurate, the length of the generated sample follows uniform distribution with [minimum length of parents, maximum length of parents].

As seen in this graph, the generated sample follows a similar pattern to the ones of sample 1 and 3, which implies it contains the characteristics of exiting samples. However, at the same time, the generated sample should not be too close to the exiting samples in order to ensure the variability. The proposed method selects two parent samples at random, and all alphabet sequences are created on the basis of the Equations (12)–(17), which ensure enough randomness and variability of the generated samples when, e.g., selecting time series size for each alphabet, selecting the first alphabet, and so forth.
Processes 2021, 9, 1115 12 of 18

Figure 4. Generated sample example.

5. Experiment and Results

In this section, we describe an experiment to verify that the samples generated by the proposed method contribute to training an RUL model without overfitting. Two RUL prediction models were compared in terms of mean absolute percent error (MAPE), one with an original dataset X and the other with a dataset X ∪ Z, where Z is an artificially generated dataset. Section 5.1 explains the procedure of the experiment, Section 5.2 introduces the datasets and hyperparameters used in the experiment, and Section 5.3 shows the results.

5.1. Procedure

First, an original sample $i$, $x^{(i)}$ is reserved for the test, and the others (i.e., $X^{(-i)} \equiv X - \{x^{(i)}\}$) are used for training. Second, an RUL prediction model $f_1$ is trained with $X^{(-i)}$, to which the transformation for RUL prediction is applied, $\bigcup_{x \in X^{(-i)}} D(x)$. Third, the MAPE of the model for $D(x)$, $MAPE_{1,i}$ is calculated. Fourth, we repeat $Q = 100$ times to generate $Z = \{z^{(k)} | k = 1, 2, \ldots, n - 1\}$ using the proposed algorithm for $X^{(-i)}$ under hyperparameters $w, L, \mu_A, \mu_C$; train $f_2$ with $\bigcup_{x \in X^{(-i)} \cup Z} D(x)$; and then calculate $MAPE_{2,i}^{(q)}$. Finally, $MAPE_{1,i}$ and the mean of $\{MAPE_{2,i}^{(q)}\}$ are compared. This procedure is repeated for all possible values of $i, w, L, |A_u|, |A_v|$, and the models.

The specific procedure is described in Algorithm A3 and the flowchart to illustrate to calculate $MAPE_{1,i}$ and $MAPE_{2,i}$ is presented in Figure 5.

In Step 6 and Step 7 of this algorithm, MAPE is calculated by:

$$MAPE = \frac{1}{T_i \times 0.7 - T_i \times 0.3} \times \sum_{t=0.7}^{T_i \times 0.3} \left| \frac{y_t^{(i)} - f_1 \left( \Phi \left( x_t^{(i)} \right) \right)}{y_t^{(i)}} \right|$$ (20)

This figure illustrates an example of the procedure to calculate $MAPE_{1,4}$ and $MAPE_{2,4}$. The specific process illustrated in this figure is as follows:

1. $x^{(1)}, x^{(2)}, x^{(3)}$, and $x^{(4)}$ are transformed into $D \left( x^{(1)} \right), D \left( x^{(2)} \right), D \left( x^{(3)} \right)$, and $D \left( x^{(4)} \right)$ by applying feature functions, respectively.
2. RUL prediction model, $f_1$, is trained with $D \left( x^{(1)} \right) \cup D \left( x^{(2)} \right) \cup D \left( x^{(3)} \right)$.
3. $D \left( x^{(4)} \right)$ is used to validate $f_1$. That is, $\hat{y}_t^{(4)} = f_1 \left( \Phi \left( x_t^{(4)} \right) \right)$ for all $t$ is obtained and the prediction results are used to calculate $MAPE_{1,4}$. 

---

Reference:

Figure 4. Generated sample example.
(4) Three new samples, $z^{(1)}$, $z^{(2)}$, and $z^{(3)}$, are generated by means of the proposed method.

(5) $z^{(1)}$, $z^{(2)}$, and $z^{(3)}$ are also transformed into $D(z^{(1)})$, $D(z^{(2)})$, and $D(z^{(3)})$.

(6) RUL prediction model, $f_2$ is trained with $D(x^{(1)}) \cup D(x^{(2)}) \cup D(x^{(3)}) \cup D(z^{(1)}) \cup D(z^{(2)}) \cup D(z^{(3)})$.

(7) $D(x^{(4)})$ is used to validate $f_2$. That is, $\hat{y}_t^{(4)} = f_2(\Phi(x_{1:t}^{(4)}))$ for all $t$ is obtained, and the prediction results are used to calculate $MAPE_{2,4}$.

---

**Table 4.** Used datasets.

| Dataset                | Feature    | Number of Samples | Mean Length | Reference |
|------------------------|------------|-------------------|-------------|-----------|
| Battery #1             | Capacity   | 4                 | 159.00      | [30]      |
| Battery #6             | Capacity   | 4                 | 90.75       |           |
| FEMTO Bearing Set #1   | Vibration signal | 4         | 44,154,880.00 | [31] |

All three datasets are well known and have been widely used in the literature for the purpose of verifying the performance of the developed machine learning methods. Each sample in the first and second dataset is a time series of the capacity of lithium-ion battery until it is dead. Discharge was carried out at 24 °C, and each battery was regarded as dead when its capacity was at 30%. Each sample of the third dataset was a signal from vibration sensor attached to bearing. The operating condition of the bearing was 1800 rpm and 4000N, and sampling frequency of the sensor was 25.6 kHz.

Hyperparameters for each experiment are given in the following Table 5.
Table 5. Hyper parameters for each experiment.

| Dataset  | Model                  | $w$ | $L$ | $|\mathcal{A}_u|$ | $|\mathcal{A}_v|$ | $\phi$ |
|----------|------------------------|-----|-----|-------------------|-------------------|-------|
| Battery  | MLP (5, 5) LSTM (10, 5, 32, 50) SVR (1, 0.1, rbf) | 50  | 3   | 3                 | 3                 |       |
|          |                        |     |     |                   |                   |       |
| Bearing  | MLP (10, 10) LSTM (20, 10, 32, 50) SVR (1, 0.1, rbf) | 10000 | 6   | 4                 | 4                 |       |

Battery: Battery #1 and #6; bearing: FEMTO Bearing Set #1; MLP($h_1$, $h_2$): multi-layered perceptron with two hidden layers with $h_1$ and $h_2$ nodes, respectively; LSTM($h$, $s$, $b$, $e$): long short-term memory with $h$ neurons, $s$ timestamps, $b$ batch size, and $e$ epochs; SVR($C$, $\epsilon$, $\kappa$): support vector regression with regularization parameter $C$, epsilon $\epsilon$, and kernel function $\kappa$.  

5.3. Results

Figures 6–8 show the experiment results with battery #1, battery #6, and bearing datasets. In the figures, blue bars denote the MAPEs of the model trained with original samples except for the test sample, and the orange bars denote the MAPEs of a model trained with original samples and generated samples.

From the results presented in Figures 6–8, we found the following. First, when we included the generated samples for training, except for the LSTM with battery #1 as shown in Figure 6, the MAPEs were smaller than those of cases with only original samples. In the case of MLP with bearing, shown in Figure 8, 20.6% of MAPE was decreased, which was the largest improvement. This shows the proposed method to artificially generate the training samples is effective and could be used to improve the performance of models. Second, MAPE of the model trained using original samples without test sample could be very high. For example, MAPEs of MLP and SVR for the bearing dataset were 35.82% and 37.21%, respectively. This may have been because the features of some test samples (i.e., cumulative root mean square and kurtosis) are quite different from those of the other samples. In other words, the relationships between feature vectors and label (i.e., RUL) are markedly different from each other. In this case, the proposed method effectively decreased the MAPE, as in the case of the MLP for the bearing dataset. Third, the proposed method
showed bigger MAPE when using LSTM for battery #1 (Figure 6), contrary to the other results. In essence, LSTM considered previous feature values (i.e., cumulative RMS and kurtosis at time \( t - 1 \)) to predict the current label (i.e., RUL at \( t \)), but the proposed method did not consider the relationship between two consecutive values. Instead, it took the relationship between two consecutive windows into the consideration, and the values within a window were aggregated to single value, either mean or standard deviation. We think this sometimes may worsen the performance of the model when using more samples for training, which, in turn, results in the larger value of MAPE. However, we obtained smaller MAPEs in battery #6 and bearing, as shown in Figures 7 and 8.

![Figure 7. Experiment result for battery #6.](image)

![Figure 8. Experiment result for bearing.](image)

From the experiment, we verified that the proposed method can solve the data insufficiency problem that is common for RUL prediction and often leads to overfitting. In other words, the RUL prediction model trained with original samples and generated samples is more generalized than the one trained with original samples only.
6. Conclusions

Due to time and cost, it is often difficult to collect sufficient run-to-failure data to train ML- and DL-based RUL models. The data insufficiency problem can result in overfitting and undermining of a model’s performance. In this paper, we proposed a time-series data-generation method that identifies patterns from alphabetical sequences converted from original time-series samples using SAX and generates new sequence on the basis of these patterns. Finally, it generates time-series values from each alphabet in the generated sequence. In an experiment using three benchmark datasets, we found that the samples generated by the proposed method effectively increased the performance of the RUL prediction model.

Future efforts to improve the proposed method should take into the account the relationship between consecutive values when generating time-series values. In addition, the proposed method was designed for a univariate time series and may be not appropriate for multivariate time series, which are common in datasets used for RUL predictions. The proposed method should therefore be expanded to consider multivariate time series. Finally, the proposed method has many parameters such as the number of windows, alphabets, and generated samples, which would impact the prediction performance of RUL prediction model. Thus, in the future research, sensitivity analysis of the parameters should be conducted and the method to choose the proper parameter values should be developed.

Author Contributions: Conceptualization, G.A., S.H. and S.L.; methodology, G.A.; software, G.A. and H.Y.; data curation, H.Y.; original draft preparation, G.A. and H.Y.; review and editing, G.A., S.H. and S.L.; funding acquisition, S.H. All authors have read and agreed to the published version of the manuscript.

Funding: This work was supported by the National Research Foundation of Korea (NRF) grant funded by the Korean government ((MSIT)2019R1A2C1088255).

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

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

Appendix A

Algorithm A1. Preprocessing phase.

| Step | Description |
|------|-------------|
| **Input** | $x^{(i)}$ for $i = 1, 2, \cdots, n$, $w$, $A_{\mu}$, $A_{\sigma}$. |
| **Step 0** | Initialize $i$ as 1. |
| **Step 1** | Normalize $x^{(i)}$ as $\tilde{x}^{(i)}$ with its mean and standard deviation. |
| **Step 2** | Split $\tilde{x}^{(i)}$ into $w$ windows, and convert $\tilde{x}^{(i)}$ into $\mu^{(i)}$ and $\sigma^{(i)}$ by calculating the mean and standard deviation of each window. |
| **Step 3** | Find break points $\beta_{\mu}$ and $\beta_{\sigma}$ according to $|A_{\mu}|$ and $|A_{\sigma}|$. |
| **Step 4** | Convert $\mu^{(i)}$ and $\sigma^{(i)}$ into $M^{(i)}$ and $S^{(i)}$, respectively. |
| **Step 5** | If $i$ equals to $n$, terminate the algorithm. Otherwise, increase $i$ by 1 and go to Step 1. |

| **Output** | $\{ (M^{(i)}, S^{(i)}) | i = 1, 2, \cdots, n \}$. |
Algorithm A2. Alphabetical sequence generation.

Input \( (\mathcal{M}^{(a)}, \mathcal{S}^{(a)}), (\mathcal{M}^{(b)}, \mathcal{S}^{(b)}) \).

**Step 1** Calculate \( \tilde{\Pr}(\mathcal{M}_1^{(a)}), \tilde{\Pr}(\mathcal{M}_1^{(b)}), \tilde{\Pr}(\mathcal{S}_1^{(a)}), \) and \( \tilde{\Pr}(\mathcal{S}_1^{(b)}) \).

**Step 2** Sampling \( \mathcal{M}_1^{(k)} \sim C(\mathcal{M}_1^{(a)}, \mathcal{M}_1^{(b)}|\tilde{\Pr}(\mathcal{M}_1^{(a)}), \tilde{\Pr}(\mathcal{M}_1^{(b)})) \).

**Step 3** Sampling \( \mathcal{S}_1^{(k)} \sim C(\mathcal{S}_1^{(a)}, \mathcal{S}_1^{(b)}|\tilde{\Pr}(\mathcal{S}_1^{(a)}), \tilde{\Pr}(\mathcal{S}_1^{(b)})) \).

**Step 4** \( \mathcal{M}_1^{(k)} = \mu_1 \) and \( \mathcal{S}_1^{(k)} = \sigma_1 \).

**Step 5** Initialize \( j = 2 \).

**Step 6** Calculate \( \tilde{\Pr}(\mathcal{M}_j^{(a)}|\mathcal{M}_{j-1}^{(a)}), \tilde{\Pr}(\mathcal{M}_j^{(b)}|\mathcal{M}_{j-1}^{(b)}), \tilde{\Pr}(\mathcal{S}_j^{(a)}|\mathcal{S}_{j-1}^{(a)}), \) and \( \tilde{\Pr}(\mathcal{S}_j^{(b)}|\mathcal{S}_{j-1}^{(b)}) \).

**Step 7** Sampling \( \mathcal{M}_j^{(k)} \sim C(\mathcal{M}_1^{(a)}, \mathcal{M}_1^{(b)}|\tilde{\Pr}(\mathcal{M}_1^{(a)}), \tilde{\Pr}(\mathcal{M}_1^{(b)})) \).

**Step 8** Sampling \( \mathcal{S}_j^{(k)} \sim C(\mathcal{S}_1^{(a)}, \mathcal{S}_1^{(b)}|\tilde{\Pr}(\mathcal{S}_1^{(a)}), \tilde{\Pr}(\mathcal{S}_1^{(b)})) \).

**Step 9** If \( j \) equals to \( w \), terminate the algorithm. Otherwise, increase \( j \) by 1 and go to **Step 5**.

Output \( (\mathcal{M}^{(k)}, \mathcal{S}^{(k)}) \).

Algorithm A3. Procedure of the experiment.

Input \( \phi, f, X, w, L, |A_p|, |A_r|, Q \).

**Step 1** Transform \( x^{(i)} \) to \( D(x^{(i)}) = \left\{ (\phi(x_{1}^{(i)}), y_{1}^{(i)}), T_{1} \times 0.3 \leq t \leq T_{1} \times 0.7 \right\} \) for all \( i \).

**Step 2** Initialize \( i = 1 \).

**Step 3** Train \( f_1 \) with \( \bigcup_{x \in X^{(-i)}} D(x) \).

**Step 4** Calculate MAPE, \( MAPE_{1,i} \), of \( f_1 \) for \( y^{(i)} \), using Equation (20).

**Step 5** Initialize \( q = 1 \).

**Step 6** Generate a set of new time series samples, \( Z = \left\{ z^{(k)} \right\}_{k = 1, 2, \ldots, n-1} \) by applying the proposed method to \( X - \{ x^{(i)} \} \) with parameters \( w, L, |A_p| \) and \( |A_r| \).

**Step 7** Transform \( z^{(k)} \) to \( D(z^{(k)}) = \left\{ (\phi(z_{1}^{(k)}), y_{1}^{(k)}), T_{k} \times 0.3 \leq t \leq T_{k} \times 0.7 \right\} \) for all \( k \).

**Step 8** Train \( f_2 \) with \( \bigcup_{x \in X^{(-i)} \cup Z} D(x) \).

**Step 9** Calculate MAPE, \( MAPE_{2,i}^{(q)} \), of \( f_2 \) for \( y^{(i)} \), using Equation (20).

**Step 10** If \( q \) does not equal to \( Q \), increase \( q \) by 1 and go to **Step 6**. Otherwise, calculate mean of \( MAPE_{2,i}^{(q)} \) for all \( q \).

**Step 11** If \( i \) equals to \( n \), terminate this algorithm. Otherwise, increase \( i \) by 1 and go to **Step 3**.

Output \( (MAPE_{1,i}, MAPE_{2,i}) \) for \( i = 1, 2, \ldots, n \).
29. Notaristefano, A.; Chicco, G.; Piglione, F. Data size reduction with symbolic aggregate approximation for electrical load pattern grouping. *IET Gener. Transm. Distrib.* **2013**, *7*, 108–117. [CrossRef]

30. Saha, B.; Goebel, K.; Battery Data Set. NASA Ames Prognostics Data Repository. Available online: [http://ti.arc.nasa.gov/project/prognostic-data-repository](http://ti.arc.nasa.gov/project/prognostic-data-repository) (accessed on 16 May 2020).

31. Nectoux, P.; Gouriveau, R.; Medjaher, K.; Ramasso, E.; Chebel-Morello, B.; Zerhouni, N.; Varnier, C. PRONOSTIA: An experimental platform for bearings accelerated degradation tests. In Proceedings of the IEEE International Conference on Prognostics and Health Management, Denver, CO, USA, 18–21 June 2012.