


Article

Long-Term Prediction of Crack Growth Using Deep Recurrent Neural Networks and Nonlinear Regression: A Comparison Study

Salahuddin Muhammad Iqbal ¹, Jun-Ryeol Park ², Kyu-Il Jung ³ , Jun-Seoung Lee ⁴ and Dae-Ki Kang ^{1,*} ¹ Department of Computer Engineering, Dongseo University, Busan 47011, Korea² Buzzni AI Lab., Seoul 08788, Korea³ JCMEEK, Seoul 07591, Korea⁴ Infrancis Research Lab., Seoul 06640, Korea

* Correspondence: dkkang@dongseo.ac.kr; Tel.: +82-51-320-1724

Abstract: Cracks in a building can potentially result in financial and life losses. Thus, it is essential to predict when the crack growth is reaching a certain threshold, to prevent possible disaster. However, long-term prediction of the crack growth in newly built facilities or existing facilities with recently installed sensors is challenging because only the short-term crack sensor data are usually available in the aforementioned facilities. In contrast, we need to obtain equivalently long or longer crack sensor data to make an accurate long-term prediction. Against this background, this research aims to make a reasonable long-term estimation of crack growth within facilities that have crack sensor data with limited length. We show that deep recurrent neural networks such as LSTM suffer when the prediction's interval is longer than the observed data points. We also observe a limitation of simple linear regression if there are abrupt changes in a dataset. We conclude that segmented nonlinear regression is suitable for this problem because of its advantage in splitting the data series into multiple segments, with the premise that there are sudden transitions in data.

Keywords: deep recurrent neural networks; LSTM; Seq2Seq LSTM; segmented nonlinear regression; long-term prediction; crack growth



Citation: Iqbal, S.M.; Park, J.-R.; Jung, K.-I.; Lee, J.-S.; Kang, D.-K.

Long-Term Prediction of Crack Growth Using Deep Recurrent Neural Networks and Nonlinear Regression: A Comparison Study. *Appl. Sci.* **2022**, *12*, 10514. <https://doi.org/10.3390/app122010514>

Academic Editor: Christian W. Dawson

Received: 18 July 2022

Accepted: 13 October 2022

Published: 18 October 2022

Publisher's Note: MDPI stays neutral with regard to jurisdictional claims in published maps and institutional affiliations.



Copyright: © 2022 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<https://creativecommons.org/licenses/by/4.0/>).

1. Introduction

Infrastructure (e.g., tunnel, road, system communication, and power plant) is essential in social and economic activities in contemporary civilization. Investment in infrastructure development has given an crucial impact on economic improvement. Private and public sectors can utilize the infrastructure to increase the efficiency of goods transportation and services. Therefore, it is necessary to prevent those losses before a fatal disaster happens.

Specifically, in this paper, we consider the problem of detecting crack growth in a building structure. Exponential crack growth in a building structure (e.g., concrete and steel frame) could make the structure unstable, thereby damaging the whole structure. However, generally speaking, predicting such property in the short term is not a viable option. Moreover, decision-makers typically consider a long-term projection to formulate their policies. Hence, predicting long-term crack growth is essential in preventing infrastructure disturbance.

We approach this problem with segmented nonlinear regression as an estimator of long-term prediction for the crack growth in infrastructure. We perform diverse experiments to evaluate the effectiveness of segmented nonlinear regression. We then compare the result with three other models: long short-term memory (LSTM), Sequence-to-Sequence (Seq2Seq) LSTM, and simple linear regression. As a result, we have found that our implementation of segmented regression is comparable to the Seq2Seq LSTM model. Moreover, our segmented regression model outperformed the LSTM model and the linear regression model.

2. Materials and Methods

2.1. Crack Sensor Data

To evaluate our approach, we aggregate crack sensor data from various locations in South Korea. We consider two kinds of infrastructure for this study: an underground facility and a bridge. We collect the sensor data within different time frames and with different sampling rates. The data used in this research had been provided by Infrancis Co., Ltd. in Seoul, Korea, a software company working with KT Corporation (formerly Korea Telecom, Korea's largest telecommunications company). Due to confidential reasons, we do not disclose the types and the positions of the sensors.

In this paper, we report the data in millimeter measurements with varying sample rates. Due to the continuous form of the data, we perform several time-series related data preprocessing methods, described in Section 3.1.

2.2. Recurrent Neural Network (RNN)

Recurrent Neural Network (RNN) [1] is a feed-forward neural network with the addition of feedback connections. The output of a unit in an RNN model at the current timestep is supplied back to the unit of the RNN model as an additional input for the subsequent step. This architecture innately enables the RNN model to learn temporal dependence in the data. For this reason, we consider RNN as one of our approaches to perform long-term prediction in this research.

In this section, we briefly discuss the Jordan Network [2], one of the widely recognized classical RNN architectures. Let each step t , with input x_t , hidden state h_t , and the output y_t , then the architecture and operation of the Jordan Network can be expressed as follows:

$$h_t = \tanh(W_h x_t + U_y y_{t-1} + b_h) \quad (1)$$

$$y_t = \tanh(W_y x_t + U_h h_{t-1} + b_y) \quad (2)$$

where W_h and U_h are hidden state parameter matrices, W_y and U_y are the output parameter matrices, b_h is parameter vectors, and \tanh is the hyperbolic tangent activation function for non-linearity. The gradients of the above units (Equations (1) and (2)) are computed using back-propagation through time (BPTT) algorithm [3]. BPTT unfolds RNN's hidden units backward over time, and back-propagates the gradients of the hidden units at the corresponding time step. The parameters of the hidden units are updated according to the gradients in the respective time steps.

Several studies have reported that RNN has a problem with a long-term dependency issue [4,5]. One of the main reasons is vanishing and exploding gradient problems [5], which can happen when a gradient-based machine learning algorithm tries to fit its model using data with long-term dependencies. Long short-term memory (LSTM) addressed this issue by introducing gates to control the flow of information [6]. In this work, we consider LSTM and Sequence-to-Sequence (Seq2Seq) LSTM architecture as our long-term estimator candidate.

2.2.1. Long-Short Term Memory (LSTM)

Long-Short Term Memory (LSTM) determines the quantity of information needed to be preserved and forgotten with the gates mechanism. A basic LSTM unit consists of a cell state c_t , a forget gate f_t , an input gate i_t , and an output gate o_t . LSTM uses a cell state c_t as a memory that stores information from all computations. To prevent the storage of unnecessary information in a long sequence of inputs, a forget gate f_t discards a fragment of it. Using an input gate i_t , LSTM can select inputs that maximize outputs. The purpose of an output o_t gate is to decide which part of the cell state c_t is relevant to the output. Finally, a hidden state h_t of a basic LSTM is an element-wise product of o_t and the hyperbolic

tangent of c_t . For every time-step t , the compact forms of the equations for the forward pass of an LSTM unit, based on the descriptions above, are described as follows:

$$i_t = \sigma(W_i x_t + U_i h_{t-1} + b_i) \tag{3}$$

$$f_t = \sigma(W_f x_t + U_f h_{t-1} + b_f) \tag{4}$$

$$o_t = \sigma(W_o x_t + U_o h_{t-1} + b_o) \tag{5}$$

$$c_t = f_t \odot c_{t-1} + i_t \odot \tanh(W_c x_t + U_c h_{t-1} + b_c) \tag{6}$$

$$h_t = o_t \odot \tanh(c_t) \tag{7}$$

where $x_t \in \mathbb{R}^d$ is the input; $W \in \mathbb{R}^{h \times d}$, $U \in \mathbb{R}^{h \times h}$, and $b \in \mathbb{R}^h$ are weight matrices and bias vector parameters respective to each function; σ and \tanh are sigmoid and hyperbolic tangent functions, respectively.

2.2.2. Sequence-to-Sequence LSTM

Sequence-to-Sequence (Seq2Seq) LSTM is an architecture that uses two sections of LSTM that act as an encoder and a decoder [7]. This architecture has been implemented in many natural language processing (NLP) tasks (e.g., machine translation [8] and summarization [9]). There are two parts in Seq2Seq LSTM: an encoder and a decoder. The encoder reads an input sequence and extracts the necessary information. Then, the decoder produces an output prediction based on information from the hidden state vector of the encoder. Lastly, the decoder, which is trained with the teacher forcing method [10], uses the actual output o_t as an input in the next time step x_{t+1} . Figure 1 illustrates the architecture of Seq2Seq LSTM.

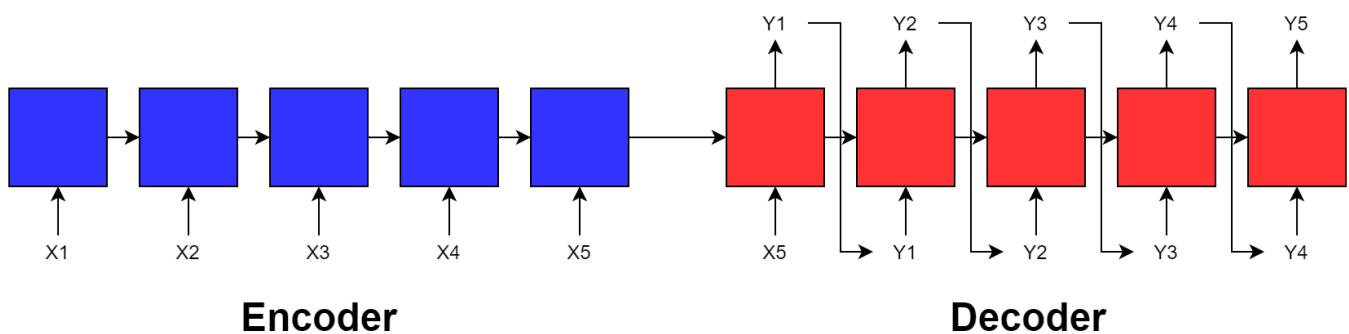


Figure 1. Diagram of Seq2Seq LSTM.

2.3. Regression Analysis

In this section, we briefly describe two methods of regression analysis that we implement in this study to find the relationship of changes in crack sensor values over time.

2.3.1. Linear Regression

Linear regression is one of the regression analysis methods for mapping the dependent variable y with the independent variable x , such that:

$$y_n = \alpha x_n + \beta + \epsilon_n \tag{8}$$

where α and β are the intercept and bias parameters, respectively; ϵ is residual, and n is the number of observations. Various methods, such as Ordinary Least Square (OLS) or Gradient Descent, can be used to estimate two parameters by minimizing the appropriate loss function. A widely used loss function is a mean square error (MSE) function, which minimizes the sum of residuals between the true label and the model output (y).

2.3.2. Segmented Nonlinear Regression

Segmented nonlinear regression—also known as piecewise regression—is a regression analysis method that partitions the independent variable data into segments, and fits each segment using linear regression methods. Dynamic programming (DP) [11] and greedy merging [12–15] are the existing approaches for finding the breakpoint locations of the input data and fit line equations for each segment. Although DP [11] has shown promising results, it is not practical for implementation on massive data points, due to its quadratic running time, $O(n^2)$ [13,14].

In this paper, here we implement a greedy merging approach by Acharya et al. [14] that has achieved efficient computation with the tradeoff of a slight decrease in precision. Algorithm 1 is the pseudocode for the entire process of segmented nonlinear regression with greedy merging. The algorithm initially makes a partition of n number of data points into n segments with a length of 1. Then, it performs the least square method to calculate the loss of a linear function for merging neighboring pairs of segments. Acharya et al. have defined the error of merging two adjacent segments S_{2c-1} and S_{2c} as:

$$\epsilon_c = \|y_c - X_c \alpha_c\|_2^2 - s^2 |S_{2c-1} \cup S_{2c}| \quad (9)$$

where c is the index for the two segments, y_c is the true value of the data points of the two segments, X_c are the input data points of the two segments, α_c is the weight of least square fit, $X_c \alpha_c$ is the prediction, and S is the set of all segments. In simple terms, the merging error is the mean-squared error of the pair of segments subtracted by variance s^2 times the length of the pair of segments. Consequently, Algorithm 1 merges the pairs when the pairs are not one of the τ -largest errors, where τ is a hyperparameter. The merging processes continue until they met the target number of segments T .

Algorithm 1: Segmented Nonlinear Regression

Function GreedyMerging(s, T, τ, X_n, y_n):

```

/* Perform partition of  $n$  data points into segments of length 1
*/
S ← {{1}, {2}, ..., {n}}
i ← 0
/* Merge the segments in a greedy way with  $T$  as the target number
of segments. */
while |Si| > T do
  Let  $m_i$  be the current number of segments.
  for  $c \in \{1, 2, \dots, \frac{m_i}{2}\}$  do
    /* Calculate the least squares LS fit. */
     $\alpha_c \leftarrow LS(X, y, S_{2c-1} \cup S_{2c})$ 
    /* Calculate the merging errors */
     $\epsilon_c = \|y_c - X_c \alpha_c\|_2^2 - s^2 |S_{2c-1} \cup S_{2c}|$ 
  end
  Let  $A$  be the set of indices  $c$  with the  $\tau$ -largest errors  $\epsilon_c$ .
  Let  $B$  be the set of the other indices.
  /* Keep the segments unmerged with large merging errors. */
   $S_{i+1} \leftarrow \bigcup_{c \in A} \{S_{2c-1}, S_{2c}\}$ 
  /* Merge the other segments. */
   $S_{i+1} \leftarrow S_{i+1} \cup \{S_{2c-1} \cup S_{2c} | c \in B\}$ 
  i ← i + 1
end
return S // the least squares fit to the data on every segment in
Si

```

3. Experiment Details

3.1. Data Preprocessing

In the experiment to evaluate our approach, we have gathered data from four crack sensors in two different facilities. We label the data from the first position of the first facility as Crack A1, and the data from the second position of the first facility as Crack A2. Similarly, we put labels to the data from the second facility at respective positions as Crack B1 and Crack B2. We perform data collection for each crack sensor at a distinct time frame. Specifically, we have collected Crack A1 and Crack A2 data from 25 February 2019 to 31 August 2019, and from 1 June 2019 to 31 July 2019, respectively. As for Crack B1 and Crack B2, we have accumulated them from 21 June 2019 to 31 August 2019 and from 13 June 2019 to 31 August 2019, respectively.

We discover two issues from the sensor data at all facilities. The first issue is that the sampling rates are not consistent, and the second issue is that there are many missing values. We conjecture that these problems happen due to either transmission error or device glitch. We perform downsampling to establish a consistent sampling rate, which decreases the original sampling rate into hourly intervals and computes the mean value in each hour. To address the missing values, we apply linear interpolation between two data points where the gap happens. Figure 2 shows the results of preprocessed data with the interpolation of Crack A1 data. Overall, we preprocess the data into 4513 data points for Crack A1, 1465 data points for Crack A2, 1729 data points for Crack B1, and 1907 data points for Crack B2.

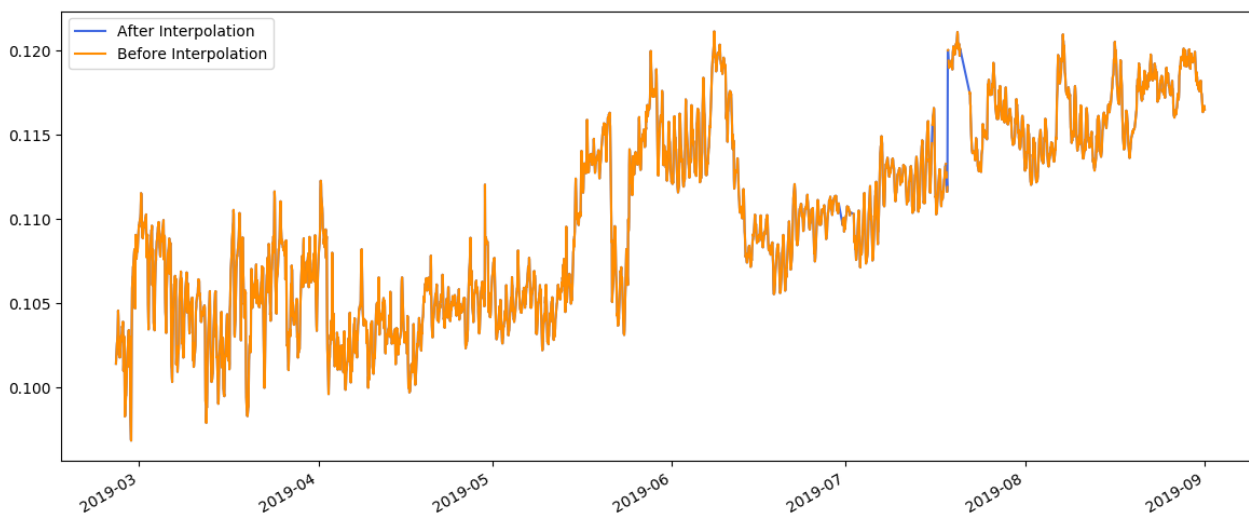


Figure 2. Preprocessed data: the blue line shows the data points after linear interpolation filled the gap of missing values in the Crack A1 data.

Since we are dealing with time-series data, we split them such that the first 90% sequences are a training set and the last 10% are a test set. Figure 3 illustrates the general data splitting strategy used in this paper. We will detail this strategy in the next section.



Figure 3. Illustration of training and test data split of time-series data.

3.2. Sliding Window

Since LSTM and Seq2Seq LSTM need a fixed sequence of data as an input to train the model and to make inferences from the model, we applied data reshaping with a fixed-length sliding window. The fixed-length window slides through the preprocessed data that eventually formed several batches of fixed sequence data. Then, we feed these batches into both LSTM models and Seq2Seq LSTM models as inputs for training. Subsequently, we perform batch training that enables fast and efficient computation. Additionally, the batches with the sliding window ensure that the model can capture the lagged dependencies in the whole sequence of data.

As an illustration, let the sequence of toy data ($x_1 \dots x_8$) and the fixed-length sliding window with a length of 6 in LSTM operate as follows. The fixed-length window in the first batch covers from x_1 to x_6 . Then, the window slides to the next sequence comprising x_2 to x_7 . Afterward, the window proceeds to the next sequence containing from x_3 to x_8 . Lastly, we set the last data point in each batch as the ground truth that we compare with the output of LSTM.

In our experiment, we choose 100 as the length of the sliding window in the training setup for LSTM, and select the 100th data point as the ground truth. We illustrate this sliding window mechanism in Figure 4. As aforementioned, we labeled the last data point in each batch as ground truth to calculate the gradient.

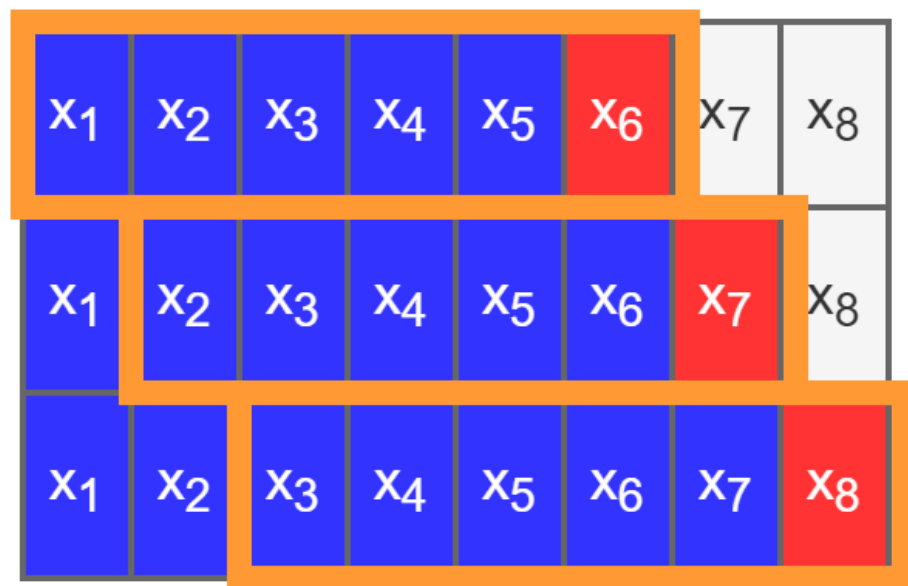


Figure 4. Illustration of the sliding window operation for LSTM on toy data. Blue boxes denoted as input data for LSTM, red boxes denoted as ground truth that needed to be predicted using LSTM, and orange rectangles are the sliding windows.

As for Seq2Seq LSTM, we perform a fixed-length sliding window operation that is similar to what has previously been described, but with a slight change. We divide the sequence sampled by the sliding window in half, which produces two new sequences. We select the first sequence as input for the encoder, and the rest as the ground truth to compare with the decoder's prediction. We move the fixed-length window to stride as long as the length of the ground truth to prevent the network from peeking the future for every batch. In the experiment, we use the sliding window with a length of 256 and divide it into two sequences, where the length of each sequence is 128. Figure 5 illustrates the sliding window process in Seq2Seq LSTM on the toy data.

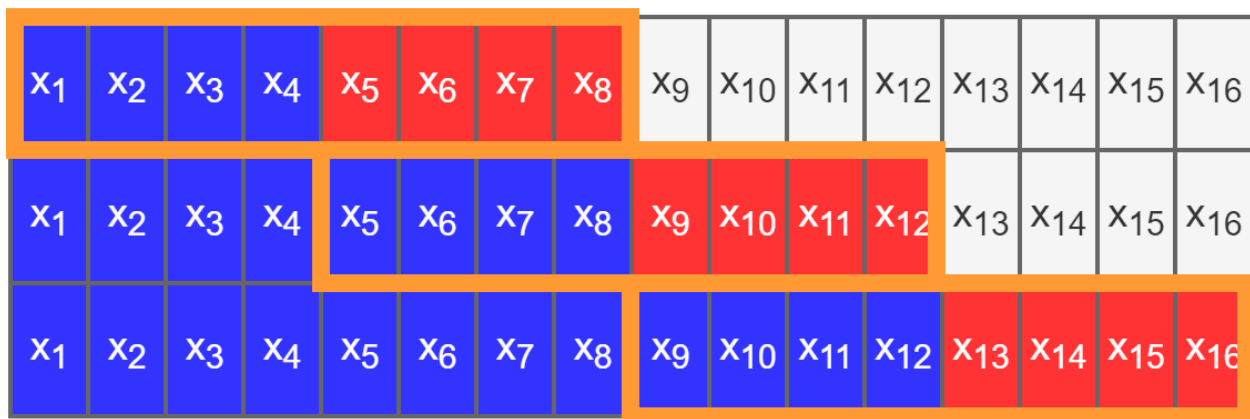


Figure 5. Illustration of the sliding window operation for Seq2Seq LSTM on toy data. Blue boxes denoted as the input data for encoder, red boxes denoted as the ground truth to be predicted by decoder, orange rectangles denoted as the sliding windows.

Note that we do not perform data reshaping for the linear regression method and the segmented nonlinear regression method we have applied, because we let them process one data point at one time step.

3.3. Model Hyperparameters

To make a fair comparison between the LSTM and Seq2Seq LSTM model, we train both models with similar hyperparameters. Both models consist of two stacked LSTMs, 64 hidden units, and a fully connected layer as an output layer. We train both models with 10,000 epochs and choose the optimally performing model based on loss value. We adopt the root mean square error (RMSE) estimator as a loss function for the LSTM and Seq2Seq LSTM models. We set the initial learning rate of LSTM and Seq2Seq LSTM as 0.0001.

For regression analysis, we train linear regression and segmented nonlinear regression with OLS to estimate the parameters. We carry out hyperparameter searching for segmented nonlinear regression. As a result, we discover that the model with hyperparameters T of 8 and τ of 4 has shown the highest performance. We also set s hyperparameter as 0 in segmented nonlinear regression after we discover that it can introduce too high a variance in the experiment results. Since we handle time series data, we make use of the last segment's parameter in the segmented nonlinear regression model to perform inference, because the test set is the next continuity of the training set.

3.4. Hardware Systems

For fair comparison, we perform a series of data preprocessing and experiments in the same hardware system. We use Intel Xeon E5-2620, which has eight cores and 16 threads in a central processor. To handle a huge dataset, we utilize 64 GB of RAM. Finally, we use Nvidia 1080Ti 11 GB as a GPU to accelerate the training and inference of the LSTM and Seq2Seq LSTM models.

3.5. Evaluations

We choose the root mean square error (RMSE) as our quantitative evaluation measure for models' predictions in the test set. The model with a lower RMSE score indicates the best-performed model. We also analyze the long-term prediction capability of all models with 10-year growth predictions on the Crack A2 data.

4. Results

4.1. Long Short-Term Memory (LSTM)

Figure 6a shows that the LSTM model's predictions are constant, and that the discrepancies between the ground truth are visible. The same figure shows the model produced a constant value of 0.078 throughout the end of the Crack A2 test set, while the ground truth varies with the highest value of 0.076. We observe the model performed poorly, with similar behavior in the remaining models. As a result, the LSTM models achieved the worst RMSE score compared with other models. We believe this outcome is because the model failed to capture the lagged dependencies of the training data.

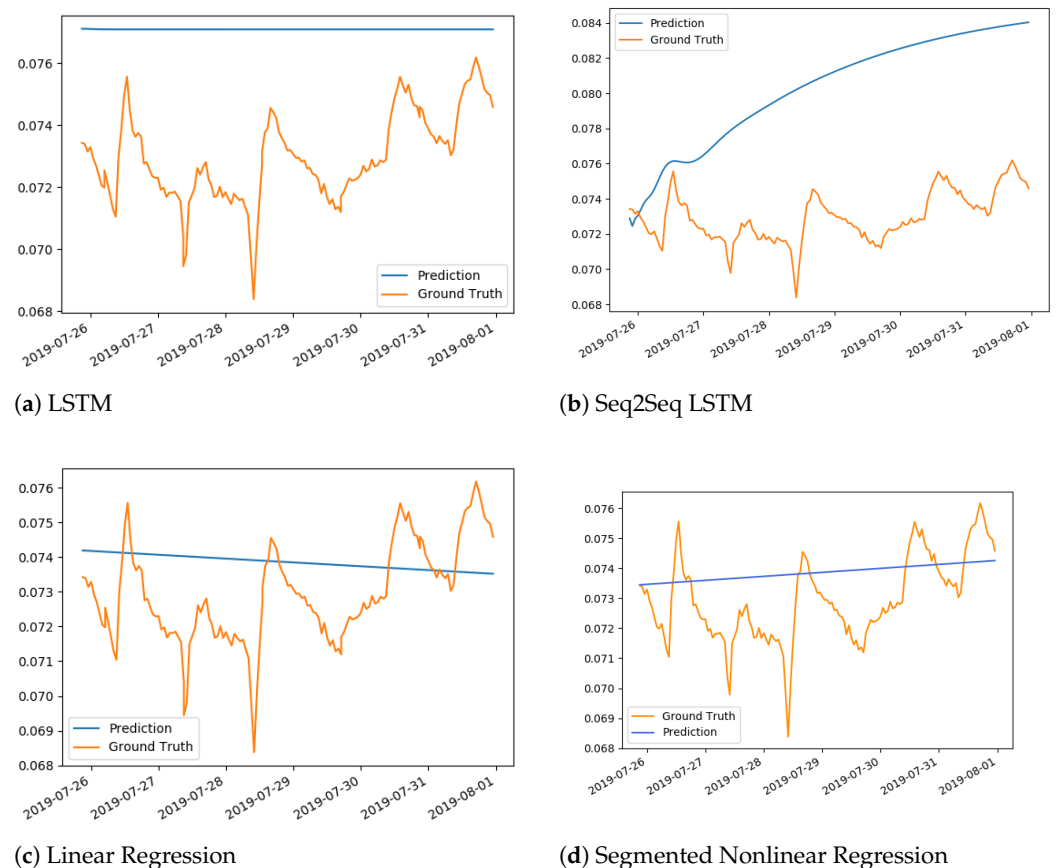


Figure 6. Evaluation of each model on Crack A2 test set.

4.2. Seq2Seq LSTM

In Figure 6b, we observe the Seq2Seq LSTM model successfully emulated the Crack A2 test set from date 26 July 2019 until 27 July 2019. However, the model overshoots for the remaining test set with a margin of 0.024 at the last data point. We suspect the model showed such a behavior because it could not effectively learn temporal dependencies on the data that exhibited abrupt changes. Although the model exhibited overshooting, it achieved the best RMSE score among the models. Table 1 shows that the Seq2Seq LSTM model outperforms the LSTM, linear regression, and segmented nonlinear regression models for all the test sets.

4.3. Linear Regression

We observe the performance of the linear regression model is relatively effective on Crack A1 and Crack A2 test set. However, its predictions are inferior to other algorithms on the Crack B1 and Crack B2 test sets, as shown in Table 1. We assume that the model could not make a compelling prediction owing to a sudden change in the train set, as is observed from Figure 7d. In the figure, we discover the linear regression model's predictions were

disparate than the ground truth on the Crack B1 train set. Therefore, the linear regression model produced an unsatisfactory result on the Crack B1 test set with an RMSE score of 0.12429; in the meantime, the Seq2Seq LSTM model obtained 0.00253.

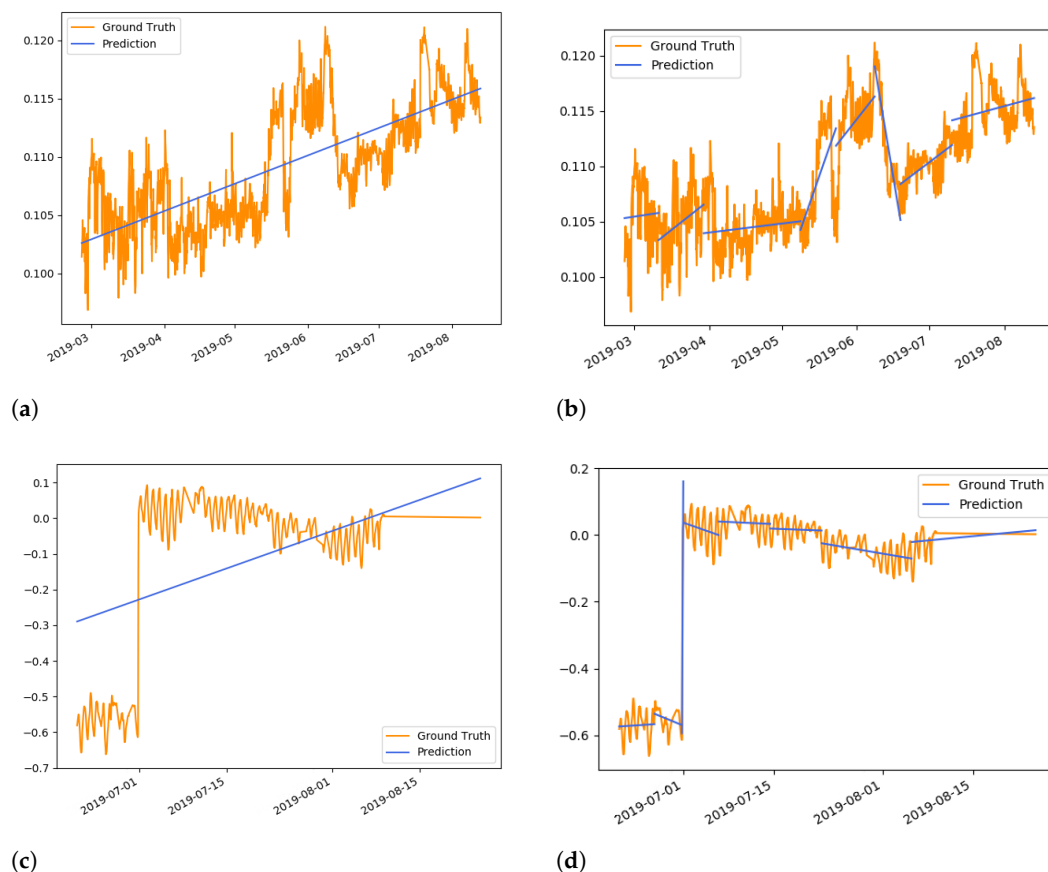


Figure 7. Prediction of linear regression and segmented nonlinear regression on Crack A1 and Crack B1 train set. (a) Linear Regression on Crack A1 train set. (b) Segmented Nonlinear Regression on Crack A1 train set. (c) Linear Regression on Crack B1 train set. (d) Segmented Nonlinear Regression on Crack B1 train set.

4.4. Segmented Nonlinear Regression

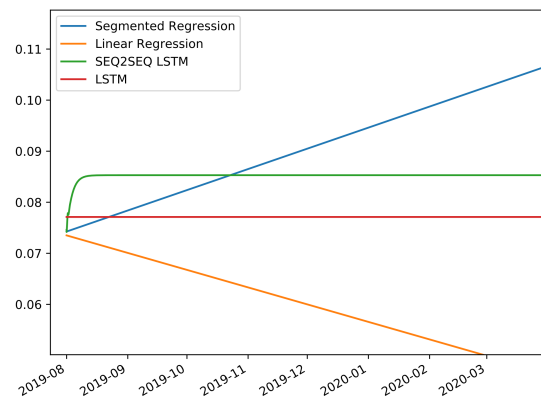
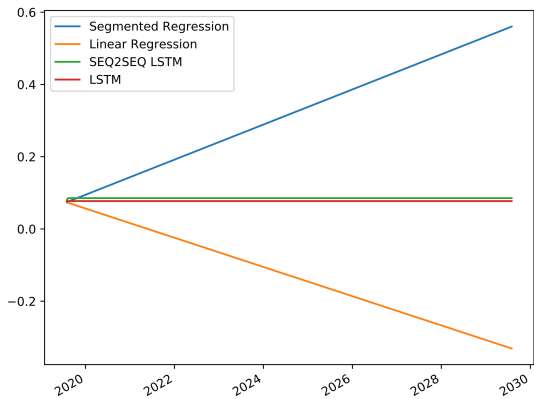
We discover the segmented nonlinear regression model considerably outperformed the linear regression model and the LSTM model based on the RMSE score. The model also rivals the Seq2Seq LSTM model’s result on the Crack A2 test set. From Table 1, we find the segmented nonlinear regression model significantly surpassed the linear regression model on any test set. Figure 7d shows that the model can approximate broken segments effectively, despite abrupt changes on the Crack B1 train set.

Table 1. Performance comparison of all the models on the test sets. Scores in the table are from RMSE metric. A low score means high performance. The highest performances are in bold.

Prediction Model	Crack A1	Crack A2	Crack B1	Crack B2
LSTM	0.00834	0.00439	0.07238	0.20018
Seq2Seq LSTM	0.00064	0.00166	0.00253	0.00235
Linear Regression	0.00169	0.00178	0.12429	0.90953
Segmented Nonlinear Regression	0.00167	0.00158	0.02490	0.02920

4.5. Long-Term Prediction Analysis

In addition to quantitative evaluations based on the RMSE metric, we analyze the long-term prediction capability of each model. We conduct experiments in which individual models produce two kinds of prediction: (1) 10 years and (2) eight months on each sensor. We present the prediction results on Crack A2 in Figure 8, where each model obtains similar performance in terms of the RMSE metric.



(a) Ten-year prediction of all models on Crack A2.

(b) Eight month prediction of all models on Crack A2.

Figure 8. Long-term prediction of all models on Crack A2. (a) Ten-year prediction of each model on Crack A2 from 1 August 2019 to 1 August 2029, and (b) displays eight-month prediction of each model on Crack A2 from 1 August 2019 to 1 April 2020.

We discover that the outputs of the LSTM model and the Seq2Seq LSTM model are indistinguishable, as both models output constant values after a few steps. Figure 8b shows that the Seq2Seq LSTM model produced its predictions in a logarithmic fashion in less than a month until it generated fixed outputs for the rest of its prediction. We observe that the LSTM model performed similarly, with an exception in which the model produced static outputs from the beginning. As a result, both models failed to emulate the crack’s growth in the long term, as in our experiment for 10-year prediction. Figure 8a shows the results mentioned earlier on Crack A2 data. Furthermore, the training process has required a tremendous amount of time on both models. Similarly, the inference process for the 10-year prediction took a massive span of durations for the LSTM model and the Seq2Seq LSTM models, as we report in Table 2. For these reasons, we consider that the LSTM model and the Seq2Seq LSTM model may not be suitable for long-term predictions.

Table 2. Average time consumption of the training process, and 10-year inference for each model on every sensor. We report the duration for both cases in h:mm:ss format, where h is hours, mm is minutes, and ss is seconds. The fastest time is in bold.

Prediction Model	Training Duration	Inference Duration
LSTM	1:18:27	8:23:57
Seq2Seq LSTM	1:15:34	1:25:11
Linear Regression	0:00:07	0:01:03
Segmented Nonlinear Regression	0:00:11	0:01:06

Contrary to the LSTM model and the Seq2Seq LSTM model, the linear regression requires a small amount of training and inference time. On average, the model only took a minute and three seconds to produce 10-year predictions. Although the linear regression model achieved the fastest training and inference time among the models, we discover the linear regression model produced negative increment outputs on 10-year and eight-month predictions, as shown in Figure 8a and Figure 8b, respectively. We conjecture that the decremental happened as a result of the inaccurate prediction. Figure 6c supports our

assumption where it shows that the model produced linearly decreasing values, while the ground truth abruptly increases. Thus, we consider that the linear regression model may be inadequate for long-term prediction as well.

We observe that the segmented nonlinear regression model generated incremental values, while the other models produced either constant values or decremental outputs in Figure 8a,b. Moreover, Table 1 shows that the model produced predictions that had comparable performance to the Seq2Seq LSTM model on each sensor, despite several abrupt changes in presence. Above all, the segmented nonlinear regression model required a short period of training and inference processes that are comparable to those of the linear regression model. Considering the factors as mentioned earlier, it is reasonable to conclude that the segmented nonlinear regression model is an acceptable application for the long-term prediction of crack growth when there are multiple abrupt changes in the data.

5. Conclusions and Future Work

In this study, we exhaustively explore several methods for the long-term prediction of crack growth in two separate facilities that are practically in use. We evaluate the performance of LSTM, Seq2Seq LSTM, linear regression, and segmented nonlinear regression models on the test set, with RMSE as our evaluation metric. We also analyze 10-year and eight-month predictions of the models on each crack sensor. Our experiment results show that the LSTM model has the worst performance among the models in terms of RMSE score. On the other hand, the Seq2Seq LSTM model achieves the highest performance in the same test sets. While the linear regression model is comparable to the segmented nonlinear regression model on the Crack A1 and Crack A2 test set, the segmented nonlinear regression model obtains more satisfactory results when there are abrupt changes apparent in the data, such as Crack B1 and Crack B2. The long-term prediction results show that the LSTM model and Seq2Seq LSTM model are ineligible as the predictor, due to a failure to emulate the crack growth, where both models give a constant output. Additionally, both models require a tremendous amount of time for training and inference. Finally, our analysis of long-term prediction points to segmented nonlinear regression being the ideal candidate as the predictor, which is computationally efficient and exhibits comparably high performance.

In future work, we would like to consider other research directions for more robust prediction. The first one is to incorporate temperature as the independent variable of prediction, which we assume causes rapid expansion and shrinkage of the materials that lead to the growth of the crack. The next one to regard in the regression analysis are the external shocks that cause abrupt changes in crack sensor reading. Finally, we plan to extend our machine learning and deep learning-based research to accommodate the influence on the stress distribution in the cracked region using the geometry of the structure and loading cases [16–19].

Author Contributions: Conceptualization, D.-K.K. and S.M.I.; methodology, D.-K.K.; software, S.M.I.; validation, D.-K.K., J.-R.P., K.-I.J. and J.-S.L.; formal analysis, S.M.I. and D.-K.K.; investigation, J.-R.P.; resources, J.-R.P.; data curation, J.-R.P.; writing—original draft preparation, S.M.I.; writing—review and editing, D.-K.K.; visualization, S.M.I.; supervision, D.-K.K.; project administration, D.-K.K., K.-I.J., and J.-S.L.; funding acquisition, D.-K.K., K.-I.J. and J.-S.L. All authors have read and agreed to the published version of the manuscript.

Funding: This research was supported by the Basic Science Research Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Science and ICT (NRF-2022R1A2C2012243).

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: The data that support the findings of this study are available from Infranics Co., Ltd., but restrictions apply to the availability of these data, which were used under license for the current study, and so are not publicly available. Data are however available from the authors upon reasonable request and with permission of Infranics Co., Ltd.

Acknowledgments: The authors wish to thank members of the Dongseo University Machine Learning/Deep Learning Research Lab., members of Infranics Research Lab. at Infranics Co., Ltd., and anonymous referees for their helpful comments on earlier drafts of this paper.

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

Abbreviations

The following abbreviations are used in this manuscript:

RNN	Recurrent Neural Network
LSTM	Long Short-Term Memory
Seq2Seq LSTM	Sequence-to-Sequence Long Short-Term Memory
RMSE	Root Mean Square Error
MSE	Mean Square Error

References

- Rumelhart, D.E.; Hinton, G.E.; Williams, R.J. Learning Internal Representations by Error Propagation. In *Parallel Distributed Processing: Explorations in the Microstructure of Cognition, Volume 1: Foundations*; MIT Press: Cambridge, MA, USA, 1986; pp. 318–362.
- Jordan, M.I. Attractor Dynamics and Parallelism in a Connectionist Sequential Machine. In *Artificial Neural Networks: Concept Learning*; IEEE Press: New York, NY, USA, 1990; pp. 112–127.
- Mozer, M.C. A Focused Backpropagation Algorithm for Temporal Pattern Recognition. In *Backpropagation: Theory, Architectures, and Applications*; L. Erlbaum Associates Inc.: Mahwah, NJ, USA, 1995; pp. 137–169.
- Bengio, Y.; Simard, P.; Frasconi, P. Learning Long-Term Dependencies with Gradient Descent is Difficult. *Trans. Neur. Netw.* **1994**, *5*, 157–166. [[CrossRef](#)] [[PubMed](#)]
- Hochreiter, S. Gradient flow in recurrent nets: The difficulty of learning long-term dependencies. In *A Field Guide to Dynamical Recurrent Neural Networks*; IEEE Press: New York, NY, USA, 2001; pp. 237–244.
- Hochreiter, S.; Schmidhuber, J. Long Short-Term Memory. *Neural Comput.* **1997**, *9*, 1735–1780. [[CrossRef](#)] [[PubMed](#)]
- Sutskever, I.; Vinyals, O.; Le, Q.V. Sequence to Sequence Learning with Neural Networks. In Proceedings of the 27th International Conference on Neural Information Processing Systems, NIPS’14, Montreal, QC, Canada, 8–13 December 2014; MIT Press: Cambridge, MA, USA, 2014; Volume 2, pp. 3104–3112.
- Cho, K.; van Merriënboer, B.; Gulcehre, C.; Bahdanau, D.; Bougares, F.; Schwenk, H.; Bengio, Y. Learning Phrase Representations using RNN Encoder–Decoder for Statistical Machine Translation. In Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), Doha, Qatar, 25–29 October 2014; pp. 1724–1734. [[CrossRef](#)]
- Nallapati, R.; Zhou, B.; dos Santos, C.; Gulçehre, C.; Xiang, B. Abstractive Text Summarization using Sequence-to-sequence RNNs and Beyond. In Proceedings of the 20th SIGNLL Conference on Computational Natural Language Learning, Berlin, Germany, 11–12 August 2016; pp. 280–290. [[CrossRef](#)]
- Goodfellow, I.J.; Bengio, Y.; Courville, A. *Deep Learning*; MIT Press: Cambridge, MA, USA, 2016.
- Bellman, R. On the Approximation of Curves by Line Segments Using Dynamic Programming. *Commun. ACM* **1961**, *4*, 284. [[CrossRef](#)]
- Kappel, S. Piecewise Regression: When One Line Simply Isn’t Enough. 2017. Available online: <https://www.datadoghq.com/blog/engineering/piecewise-regression/> (accessed on 17 July 2022)
- Haiminen, N.; Gionis, A.; Laasonen, K. Algorithms for unimodal segmentation with applications to unimodality detection. *Knowl. Inf. Syst.* **2008**, *14*, 39–57. [[CrossRef](#)]
- Acharya, J.; Diakonikolas, I.; Li, J.; Schmidt, L. Fast Algorithms for Segmented Regression. In Proceedings of the 33rd International Conference on Machine Learning, New York, NY, USA, 19–24 June 2016; Balcan, M.F., Weinberger, K.Q., Eds.; PMLR: New York, NY, USA, 2016; Volume 48, pp. 2878–2886.
- Muggeo, V.M.R. Estimating regression models with unknown break-points. *Stat. Med.* **2003**, *22*, 3055–3071. [[CrossRef](#)] [[PubMed](#)]
- Ravi-Chandar, K. 2.05—Dynamic Fracture. In *Comprehensive Structural Integrity*; Milne, I., Ritchie, R.O., Karihaloo, B., Eds.; Pergamon: Oxford, UK, 2003; pp. 285–361. [[CrossRef](#)]
- Webster, G.A. 5.05—Creep Crack Growth. In *Comprehensive Structural Integrity*; Milne, I., Ritchie, R.O., Karihaloo, B., Eds.; Pergamon: Oxford, UK, 2003; pp. 241–271. [[CrossRef](#)]

18. Riesch-Oppermann, H. Fracture Mechanics: Probabilistic Approaches. In *Encyclopedia of Materials: Science and Technology*; Buschow, K.H.J., Cahn, R.W., Flemings, M.C., Ilshner, B., Kramer, E.J., Mahajan, S., Veyssi re, P., Eds.; Elsevier: Oxford, UK, 2006; pp. 1–6. [[CrossRef](#)]
19. Megson, T.H.G. Chapter 15—Fatigue. In *Aircraft Structures for Engineering Students*, 7th ed.; Megson, T.H.G., Ed.; Aerospace Engineering; Butterworth-Heinemann: Oxford, UK, 2022; pp. 485–508. [[CrossRef](#)]