water

MDPI

Article

Stress Estimation of Concrete Dams in Service Based on Deformation Data Using SIE–APSO–CNN–LSTM

Liang Tao [1,2], Dongjian Zheng [1,2], Xin Wu [1,2,*], Xingqiao Chen [1,2], Yongtao Liu [1,2,3], Zhuoyan Chen [1,2] and Haifeng Jiang [1,2]

[1] State Key Laboratory of Hydrology-Water Resources and Hydraulic Engineering, Hohai University, Nanjing 210098, China
[2] College of Water Conservancy and Hydropower Engineering, Hohai University, Nanjing 210098, China
[3] Department of Civil and Architectural Engineering, Aarhus University, Aarhus 8000, Denmark
* Correspondence: wuxin@hhu.edu.cn

Abstract: The stress behavior of key parts of concrete dams is related to the safe operation of the dam. However, the stress sensors in concrete are susceptible to aging and failure with increasing service life. Estimating the structural stress under sensor failure or data loss scenarios for concrete dams in service is essential and complex. This study presents a stress estimation method driven by the observation data. Firstly, a one-to-one correspondence exists between dam deformation reflecting the load effect and structural stress. Estimating the structural stress by simulating load effects with dam deformation is more convenient when it is hard to simulate complex load effects directly. Therefore, based on the observed data before stress sensor failure, the spatial–temporal relationship between structure stress and multi-point deformations of a concrete dam is developed using convolutional neural networks (CNN) and long short-term memory (LSTM). An improved particle swarm optimization algorithm combined with swarm information entropy (SIE–APSO) is proposed simultaneously for tuning the network's hyperparameter and accelerating the convergence. Finally, the stress estimation of the target part of the concrete dam in service is obtained. The case shows that it is valid and feasible. The RMSE decreased by approximately 21–58%, MAPE decreased by 19–58%, and ARV decreased by 22–94% compared with the load-stress relationship model.

Keywords: concrete dam; stress estimate; data spatial–temporal association; CNN–LSTM; improved particle swarm optimization

Citation: Tao, L.; Zheng, D.; Wu, X.; Chen, X.; Liu, Y.; Chen, Z.; Jiang, H. Stress Estimation of Concrete Dams in Service Based on Deformation Data Using SIE–APSO–CNN–LSTM. *Water* **2023**, *15*, 59. https://doi.org/10.3390/w15010059

Academic Editor(s): Miguel Á. Toledo, Rafael Morán

Received: 30 November 2022
Revised: 17 December 2022
Accepted: 21 December 2022
Published: 24 December 2022

1. Introduction

Structural health monitoring (SHM) systems utilize numerous sensors installed on dams to acquire timely and continuous data on the state of structures. Stress sensors are commonly embedded in key parts of the dam to obtain structural strength information. Hence, stress monitoring is important for dam safety assessment. However, stress sensors are susceptible to aging and failure due to slow processes such as corrosion and fatigue. Maintenance is also costly, since they are embedded in concrete. The research on stress estimating for scenarios such as sensor failure and data loss is essential in dam health monitoring.

Many studies have been conducted for the above scenarios. Li et al. [1] searched for valid samples with the highest correlation with the default samples using the K-nearest neighbor (KNN) algorithm. They took the average value of the relevant samples as the target values. Zhang et al. [2] proposed a method for recovering missing data from long-term measurements that considers the full-life monitoring of structures. Their method analyzes the correlations of multiple strain sensors installed in a steel structure of a stadium building and restores the missing data by interpolating their relationships. Wang et al. [3]

proposed a missing value complementation method based on kernel-independent component analysis (KICA) and relevant vector machine (RVM) models. They performed a nonlinear transformation of the correlated measurement points of the target measurement points to extract the independent components. The independent components and measured values of the target measurement points are then used as input and output for model training and applied to a concrete dam. Some scholars have also developed estimation models based on: the relationship between multiple factors of environmental loads (water pressure, temperature, and time factors) and structural responses [4], such as the conventional hydrostatic–seasonal–time (HST) statistical model [5]; a machine learning (ML)-based model, such as support vector regression (SVR) [6]; random forests (RF), based on random decision-making [7]; Gaussian process regression (GPR), based on the probability distribution [8]; boosted regression trees (BRT) combined with regression trees and an enhancement method [9]; recurrent neural network (RNN) and its improved algorithms, such as long-short term memory (LSTM) neural networks [10] and gated recurrent unit (GRU) neural networks [11]; and temporal convolution network (TCN) [12].

The above studies estimate target values mainly (i) from the data of proximity time; (ii) from the data of adjacent measurement points of the same sensor type; or (iii) based on the relationship among the load factors and structure responses. However, the first type of method is only for data interpolation problems, which are not applicable for future values estimation after sensor damage. The second type of method does not involve the physical mechanism of the structure. It only estimates by interpolation, which is less reliable in the case of limited data from the same sensor type in the vicinity. The third type of method contains the load and structural response information. Nevertheless, it is tough to simulate complex load effects using load factors directly. Moreover, the estimation ability of this method is limited in dealing with situations beyond the experienced conditions.

The monitoring data of various sensor types depict the operation information of the dam system from different aspects, and there is a one-to-one correspondence among them. At the same time, dam deformation data are the direct externalization of the load effect, and they have the advantage of convenient collection and maintenance compared with stress data. These premises open up an intriguing avenue of research: estimating structural stress by simulating the load effects with observed deformation data. Specifically, the spatial–temporal association among multiple relevant deformation sensors and target stress sensors can be utilized to support stress estimation at the target part of concrete dams. The spatial–temporal associations among the measured sequences of multiple monitoring points are generally difficult to express with explicit functions. The recently developed deep learning (DL) models have provided an actionable solution to extract information from various kinds of data [13,14]. The convolutional neural network (CNN), inspired by the visual system, includes network attributes such as convolution, pooling operations, and shared weights, enabling it to capture data features from aspatial perspective [15]. The recurrent neural network (RNN) has a cyclic connection to the architecture, meaning it can update the current state based on past states and current input data [16]. Unfortunately, RNN is unable to connect the relevant data when the input gap is large. Long short-term memory (LSTM), derived from RNN, aims to handle the problem of long-term dependencies by providing gate functions for RNN [17]. It has been widely adopted in research areas concerned with sequential data because of its the powerful learning capacity [18–20]. Therefore, LSTM networks are introduced for processing the time series data in this work.

Different deep learning (DL) architectures are suitable for different datasets [21]. Therefore, the hyperparameters must be specified to fit a DL model to the target task. The literature [22] suggests that the DL models will benefit from the tuning process since they often have many hyperparameters requiring selection. Hyperparameter tuning is an optimization problem. Particle swarm optimization (PSO) is a reasonable choice due to its simple structure and inexpensive computational cost [23]. However, this algorithm is prone to fall into the local optimum since the particles are only guided by the individual

and group optimal solutions during the iterative process [24]. Improvement to alleviate this problem is needed.

Based on the advantages of the DL technique mentioned above, the CNN and LSTM networks are employed sequentially to extract the spatial–temporal association between the historical monitoring data of the failure stress sensor and multiple relevant deformation sensors. An improved PSO algorithm combined with swarm information entropy (SIE–APSO) is also proposed for tuning the model hyperparameter during the training process. Then, the estimation value at the location of the failed stress sensor can be derived by feeding the trained learning model the existing deformation monitoring data. In addition, the effectiveness of combining the estimation model with the deformation data presented in this study is evaluated through comparison with HST, neural network (NN) with single hidden layer, and SVR established on the load–stress relationships. Some other algorithms are also introduced for the performance comparison of SIE–APSO. Furthermore, the estimation performance that depends on the model input configuration, that is, the length of the input sample, is examined.

The rest of this paper is organized as below: the overall architecture of the proposed learning model for stress estimation is detailed in Section 2, and then we give a brief introduction of the methodology involved; Section 3 presents the actual project and its monitoring system mentioned in this study, followed by the data configuration process for model development; model architecture and estimation results are reported in Section 4; Section 5 discusses the performance comparison and further analysis of the model; finally, the conclusions and limitations are summarized in Section 6.

2. Methodology

The modeling framework of the presented stress estimation model of concrete dams is illustrated in Figure 1, and it contains four processes:

1. Data configuration. Observations from dam deformation and stress sensors are collected to provide learning data for the estimation model. The supposed failure time of the target stress sensor is considered to be the splitting point between the training and testing periods. The data samples are then configured via the sliding-window approach.
2. Model development. Taking the deformation matrix as the model input and extracting the spatial features among different deformation monitoring points via CNN each day produces the output sequence containing the temporal attributes. Take this sequence as the input of the single-layered LSTM to obtain the temporal representation of the sequence. Finally, set a fully connected (FC) layer to get the model output and take it as the value of concrete stress.
3. Tuning and training. The training and validation subset are obtained via the time series cross-validation (CV) technique performed on the training data. Subsequently, taking the average accuracy of the validation set of each fold as the fitness function, the hyperparameter tuning of the estimation model is implemented via the proposed SIE–APSO.
4. Evaluation. The estimation performance of the trained model is evaluated by feeding it the configured out-of-training data. Furthermore, the root mean square error (RMSE), average absolute percentage error (MAPE), and average relative variance (ARV) are utilized for model evaluation.

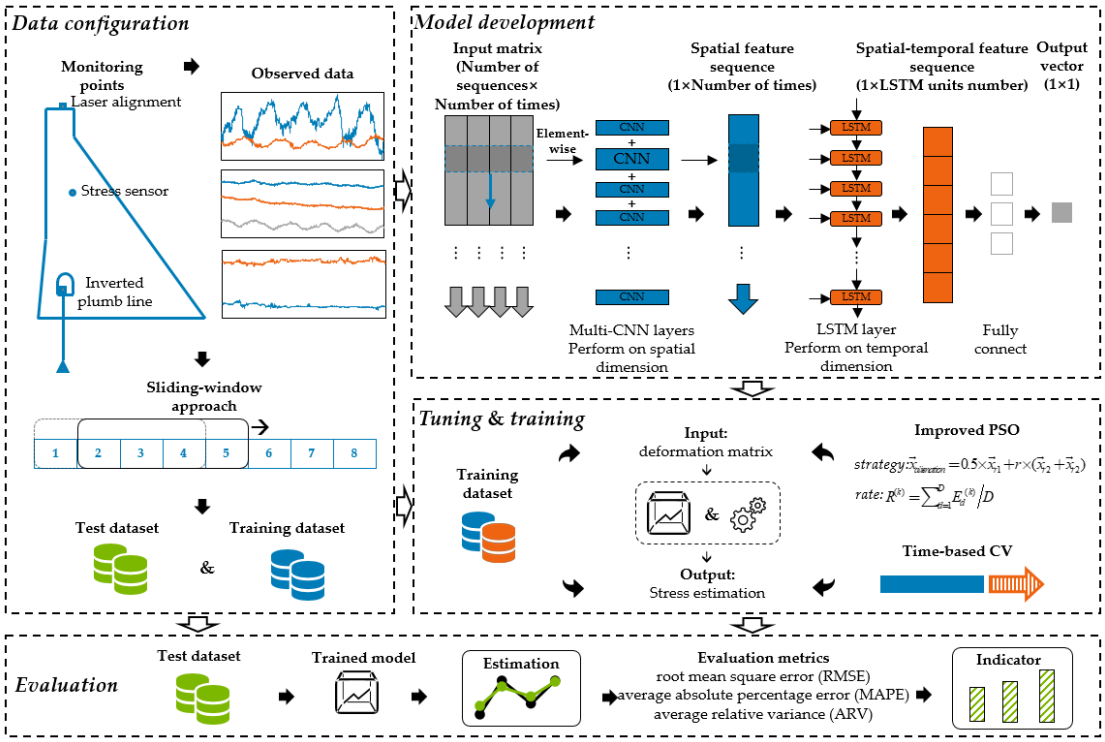


Figure 1. The process of the presented estimation model.

2.1. The CNN–LSTM Estimation Model

The stress estimation model of concrete dams utilizes data from multiple deformation points to predict the stress response at the site of the failed stress sensor. Expressly, assume $\boldsymbol{I}$ is a two-dimensional matrix ($n \times m$) composed of the deformation measured values, where n and m are the number of deformation sequences and the number of monitoring times and represent the spatial and temporal dimensions of the data, respectively. Then, n CNN layers are adopted in parallel to perform convolution processing on the data of the first dimension (spatial dimension) of the matrix $\boldsymbol{I}$. That is, m deformation data from the same monitoring time participate in the calculation of a convolutional layer. A spatial feature sequence is generated by extracting the spatial association between multiple deformation monitoring points each time. Notably, this sequence contains temporal attributes. The operation of the LSTM is performed subsequently on the temporal dimensions of this sequence and then obtains the spatial–temporal features of the data. The generated spatial–temporal features are connected to target stress values through a single-layer network for estimation. The architectures of CNN and LSTM are briefly introduced as follows.

2.1.1. Convolutional Neural Network

CNN is a feedforward neural network that has the characteristics of autonomously extracting data features and excellent classification capabilities [25]. The convolution operation is implemented through the convolutional layer, which means the network can conveniently process the data with a net structure. The two-dimensional discrete convolution formula is as follows:

$$S(i,j) = (I * K)(i,j) = \sum_{m}\sum_{n} I(i+m, j+n)K(m,n) \quad (1)$$

where S is the feature mapping; I is the two-dimensional input; K is the convolution kernel; m,n are the element index of K and less than the number of rows and

columns of the convolution kernel, respectively; i, j are the element index of S and less than the number of rows and columns of the feature mapping, respectively.

A basic CNN architecture consists of input, convolutional, pooling, fully connected, and output layers, as shown in Figure 2. Among them, the convolutional layer and the pooling layer can be stacked repeatedly according to the complexity of the net structure data. During this process, the features of net structure data are extracted through convolution operations in the convolutional layer. Then, the dimension of the features is reduced in the pooling layer to improve the calculation accuracy, effectively avoiding overfitting and reducing the computational load. Finally, the classification or regression results of the data are obtained by the well-trained model according to its features. The feature extractor, composed of the convolutional layer and the pooling layer, is automatically optimized during the network training process without additional intervention.

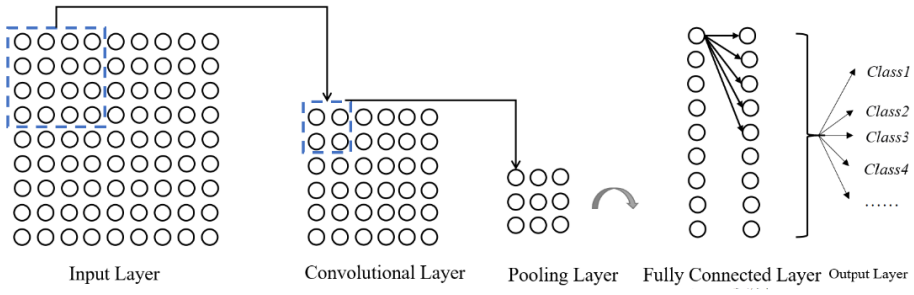


Figure 2. Basic structure of the Convolutional Neural Network.

Both the convolution and pooling operations of CNN are linear. Therefore, some improvements, such as activation functions and regularization, are frequently introduced to enhance the model's performance. There are activation functions for non-linearities; regularization is introduced via normalizations and dropout to counteract the exploding gradient problem and prevent overfitting.

2.1.2. Long Short-Term Memory Network

LSTM is an improved architecture of Recurrent Neural Networks (RNNs), which introduce three control units of input, output, and forget gates to intervene in the cell state [26]. The input gate and the output gate can regulate the input and output vectors of the unit, respectively; the forget gate will evaluate whether the memory cell's state conforms to the rules to perform the state's retaining or discarding operation. These gate structures can keep track of arbitrary long-term dependencies from time series data, making them more insensitive to gap length than RNNs. The function of the cell state is roughly the same as the hidden layer vector in the traditional neural network. The basic architecture of the LSTM unit is depicted in Figure 3.

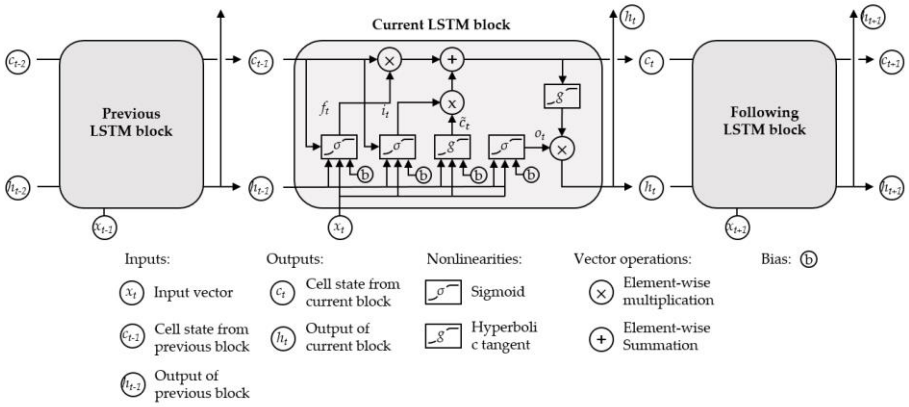


Figure 3. The architecture of the LSTM.

Given, an input vector $\boldsymbol{x}_t \in \mathbb{R}^d$, the output vector $\boldsymbol{h}_t$ of the single hidden layer LSTM unit can be expressed as in the following formulas:

$$\boldsymbol{i}_t = \sigma(\boldsymbol{W}^{(i)}\boldsymbol{x}_t + \boldsymbol{U}^{(i)}\boldsymbol{h}_{t-1} + \boldsymbol{b}^{(i)}) \tag{2}$$

$$\boldsymbol{f}_t = \sigma(\boldsymbol{W}^{(f)}\boldsymbol{x}_t + \boldsymbol{U}^{(f)}\boldsymbol{h}_{t-1} + \boldsymbol{b}^{(f)}) \tag{3}$$

$$\tilde{\boldsymbol{c}}_t = \tanh(\boldsymbol{W}^{(\tilde{c})}\boldsymbol{x}_t + \boldsymbol{U}^{(\tilde{c})}\boldsymbol{h}_{t-1} + \boldsymbol{b}^{(\tilde{c})}) \tag{4}$$

$$\boldsymbol{c}_t = \boldsymbol{D}_t^{(i)}\tilde{\boldsymbol{c}}_t + \boldsymbol{D}_t^{(f)}\boldsymbol{c}_{t-1} \tag{5}$$

$$\boldsymbol{o}_t = \sigma(\boldsymbol{W}^{(o)}\boldsymbol{x}_t + \boldsymbol{U}^{(o)}\boldsymbol{h}_{t-1} + \boldsymbol{b}^{(o)}) \tag{6}$$

$$\boldsymbol{h}_t = \boldsymbol{D}_t^{(o)} \tanh(\boldsymbol{c}_t) \tag{7}$$

where $\boldsymbol{i}_t \in (0,1)^h$, $\boldsymbol{f}_t \in (0,1)^h$ and $\boldsymbol{o}_t \in (0,1)^h$ are the activation vector of the input, forget, output gate, respectively; $\boldsymbol{h}_t \in (-1,1)^h$ is the hidden state vector or output vector; $\tilde{\boldsymbol{c}}_t \in (-1,1)^h$ and $\boldsymbol{c}_t \in \mathbb{R}^h$ are the cell input activation vector and cell state vector, respectively; $\boldsymbol{W}^{(i)} \in \mathbb{R}^{h\times d}$ and $\boldsymbol{U}^{(i)} \in \mathbb{R}^{h\times h}$ are the weight matrices, and $\boldsymbol{b}^{(i)} \in \mathbb{R}^h$ is the bias vector of the network parameters that need to be learned during training; the superscripts d and h refer to the number of input features and number of hidden units, respectively; $\boldsymbol{D}_t^{(i)} = \mathrm{diag}(\boldsymbol{i}_t)$, $\boldsymbol{D}_t^{(f)} = \mathrm{diag}(\boldsymbol{f}_t)$ and $\boldsymbol{D}_t^{(o)} = \mathrm{diag}(\boldsymbol{o}_t)$ where $\mathrm{diag}(\cdot)$ is the diagonal operator; the sigmoid function $\sigma(\cdot)$ and hyperbolic tangent function $\tanh(\cdot)$ are applied element-wise to the vector.

2.2. Improvement of the Particle Swarm Optimization

Particle swarm optimization (PSO) is a heuristic technique inspired by cooperative predation behavior among individuals in a bird swarm. Detailed and rigorous descriptions can be found in [27]. Assume $\vec{v}_i^{(k)} \in \mathbb{R}^D$ and $\vec{x}_i^{(k)} \in \mathbb{R}^D$ are the velocity and position (candidate solution) of the i-th particle in the k-th generation, respectively, and D is the dimension of the position, which in this paper is the number of the optimized hyperparameters in the DL model. The calculation formula is then [27]:

$$\vec{v}_i^{(k+1)} \leftarrow \omega^{(k)} \cdot \vec{v}_i^{(k)} + \varphi_p r_p \cdot (\vec{p}_i^{(k)} - \vec{x}_i^{(k)}) + \varphi_g r_g \cdot (\vec{p}_g^{(k)} - \vec{x}_i^{(k)}) \tag{8}$$

$$\vec{x}_i^{(k+1)} \leftarrow \vec{x}_i^{(k)} + \vec{v}_i^{(k+1)} \tag{9}$$

where $\vec{p}_i^{(k)} \in \mathbb{R}^d$ and $\vec{p}_g^{(k)} \in \mathbb{R}^d$ is the historical best position of the i -th particle and all particles, respectively; ω is the inertia weight of the particle; φ_p and φ_g are the cognitive coefficient and social coefficient. The algorithm gives the particle three behavior patterns according to the formula above:

- inheritance of its inertia;
- the particle's cognitive behavior, which represents the thinking of the particle itself;
- the population's social sharing behavior represents the information sharing and cooperation between particles.

The above formula shows that the particles will approach the individual or group's optimal position from the initial stage of iteration. As a consequence, particles are susceptible to falling into local optimum. In fact, the swarm system will be disturbed by environmental changes, and the swarm objective has yet to be specific. Some birds will move in different directions from others. This phenomenon, called alienation here, is especially evident in the early stages of swarm development. Simulating this behavior is an exciting avenue in dealing with the problem of getting stuck in the local optimum. Therefore, we consider adopting particles' alienation to simulate this behavior, which produces the reference and possibility to search in other directions. This process was implemented by combing three random dissimilar particles in this study. Suppose $\vec{x}_{alienation}$ represents the position of the alienation particle; the alienation strategy proposed is then:

$$\vec{x}_{alienation} = 0.5 \times \vec{x}_{r1} + r \times (\vec{x}_{r2} + \vec{x}_{r2}) \tag{10}$$

where $\vec{x}_{r1}, \vec{x}_{r1}, \vec{x}_{r1}$ are the position vectors of the three particles, and $r_1 \neq r_2 \neq r_3$; r is a random number from 0 to 1. Notably, the bounds are still working, although the expectations of the position vector still fall within the set range.

Subsequently, we proposed an approach for evaluating the swarm alienation rate considering multi-dimensional entropy. This process makes the swarm adaptively and probabilistically accept particle alienation behavior. Suppose $x_{i,j}^{(k)}$ is the j -th dimension value of $x_i^{(k)}$. The alienation rate is calculated as follows.

$$R^{(k)} = \sum_{d=1}^{D} E_d^{(k)} \Big/ D \tag{11}$$

$$E_d^{(k)} = -\sum_{i=1}^{N} p(x_{i,d}^{(k)}) \ln p(x_{i,d}^{(k)}) \Big/ \ln N \tag{12}$$

where the alienation rate $R \in (0,1)$; $E_d^{(k)}$ is the entropy of the swarm in the d -thdimension; $p(x_{i,d}^{(k)})$ is the distribution probability of particles when the d -th dimension interval is divided equally according the number of particles; and N is the number of particles.

Moreover, a greedy strategy is used for current particles to update the particle position:

$$\vec{x} = \begin{cases} \vec{x} = \vec{x}_{alienation}, & if\ fitness(\vec{x}_{alienation}) < fitness(\vec{x}_{original}) \\ \vec{x} = \vec{x}_{original}, & otherwise \end{cases} \tag{13}$$

The pseudo code of the proposed swarm information entropy-based alienation particle swarm optimization (SIE–APSO) is presented in Figure 4.

Initialize the particle swarm P_i *(i = 1, 2, …, n)*
Initialize parameters ω, φ_p, φ_g
Calculate the fitness of each particle
while *(k<Max number of iterations)*
  calculate the alienation rate of the current particle swarm by equations (11),(12)
  for *each particle*
    calculate the original position of the current particle by equations (8),(9)
    if *random state is lower than the alienation rate*
      calculate the alienation position of the current particle by equation (10)
    update the position $\vec{x}$ *of the current particle by eqation (13)*
  end for
  k=k+1
end while
return $\vec{x}$

Figure 4. Pseudo code of the SIE–APSO algorithm.

Improvement in the above can achieve the following effectiveness compared to the standard PSO:

- Reduce the probability of falling into a local optimum. With the preform of iteration, the diversity of particles will continue to decrease, and it is easy to fall into the local optimum. Simulation of particle alienation behavior can reduce the probability of this phenomenon.
- Balance the global and local search capabilities of the algorithm. The particles are uniformly distributed in the early iteration with a larger swarm information entropy. At this stage, the higher alienation rate effectively prevents the particles from moving directly to the current optimal solution, giving it a better global exploration ability. While in the later stage, the particles are gradually concentrated. The decreasing of the alienation rate provides better local development of the particles.

2.3. Time Series Cross-Validation Technique

The cross-validation (CV) technique is used for robustly choosing the best ML model. It is achieved by tuning hyperparameters on training subsets and evaluating them on the complementary subset of the data. Unlike other data types, temporal dependencies exist in time series data, making the order of the data vital in time series-related problems. Hence, we must withhold all data about events that occur chronologically after the events used for fitting the model [28]. We therefore adopt the time series CV technique on a rolling basis for the hyperparameter tuning of the DL model [29]. Figure 5 gives a depiction of this approach where the fold is 3. This process starts with splitting the data set into a training and a testing set. The testing set is preserved for evaluating the cross-validated model. The training set is split temporally into k folds containing the training subset and validation subset in time series CV. The data after the validation subset will never be used for model training in each fold. Then the validation error obtained on all folds is averaged for evaluating the model under the specific hyperparameter combination.

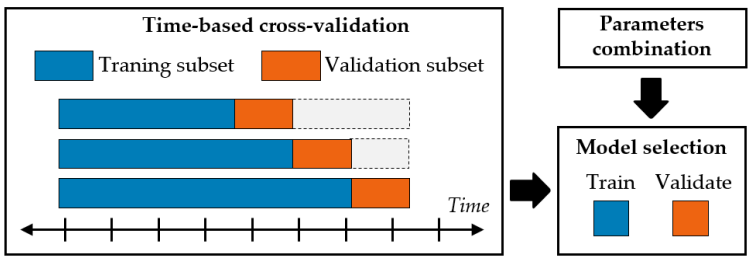


Figure 5. The time-series cross-validation technique on a rolling basis. The fold is three.

2.4. Evaluation Metrics

The root mean square error (RMSE) and average absolute percentage error (MAPE) are frequently used as evaluation indicators of the regression model, computed as Equations (14)–(15).

$$\mathrm{RMSE} = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(y_o^{(i)} - y_p^{(i)})^2} \tag{14}$$

$$\mathrm{MAPE} = \frac{1}{N}\sum_{i=1}^{N}\left|\frac{y_o^i - y_p^i}{y_o^i}\right| \tag{15}$$

where $y_o^{(i)}$ is the i-th observed value; $y_p^{(i)}$ is the i-th predicted value; N is the number of data. Given that RMSE is measured in the same units as the target variable, MAPE is measured in the relative percentage of error; these provide a practical accuracy evaluation. Moreover, the average relative variance (ARV) adopts as a measure of accuracy in this study [30]:

$$\mathrm{ARV} = \frac{\sum_{i=1}^{N}(y_o^{(i)} - y_p^{(i)})^2}{\sum_{i=1}^{N}(y_o^{(i)} - \bar{y}_o)^2} = \frac{MSE}{\sigma^2} \tag{16}$$

where $\bar{y}_o$ $\bar{y}_o$ is the mean of the observed data. Given that ARV represents the ratio between the mean squared error and the variance, it considers both the magnitude and the deviation of the target variable. Furthermore, a model with ARV = 1 is as accurate an estimate as the mean of the observed.

3. Case Study

3.1. Project Description

An engineering project in southwestern China contains the water retaining structure, flood discharge, energy dissipation structure, and diversion (tail) water power generation system, as shown in Figure 6. This dam is a roller-compacted concrete (RCC) gravity dam composed of the left and right retaining dam section, the left and right middle-hole dam section, and the discharge dam section with a 5-hole at the river bed. The dam crest elevation is 1334.0 m, and the maximum dam height is 168.0 m, with a normal storage level of 1330.0 m and a total storage capacity of 760 million m^3. The dam is located in a high mountain and valley area, with steep slopes and a relatively complex geological structure. The overall slopes of the terrain on the bank sides are relatively neat and have an asymmetrical "V" shape in the valleys. The monitoring object of the dam monitoring system mainly includes:

- environment variables;
- horizontal and vertical displacement of the dam;
- the inclination of the dam foundation and dam body;
- the stress and strain of the dam concrete.

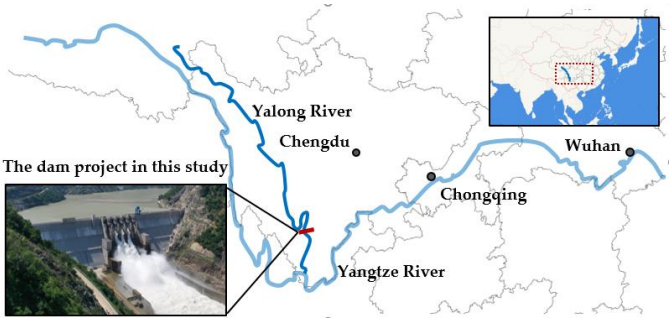


Figure 6. The geographical location and picture of the dam in this study.

Considering the representativeness of monitoring points and the completeness of the sequence, we selected the measured values of two deformation monitoring points and one stress monitoring point of the 9# dam section for model development. Figure 7 gives the cross-section of the 9# dam section and the layout of the monitoring points. The height and the foundation elevation of this dam section are 148.0 m and 1186.0 m, respectively.

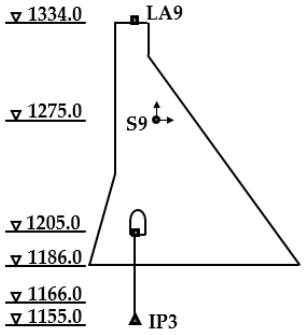
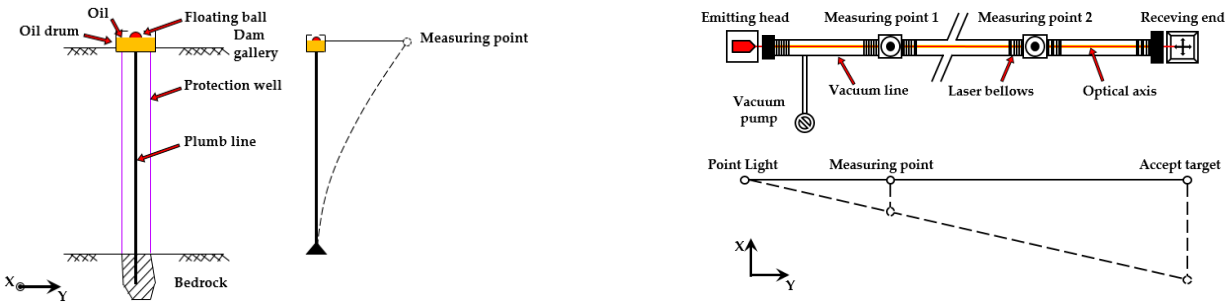


Figure 7. The layout of the monitoring project for 9# section of the dam.

The monitoring points LA9 and IP3 are installed at the 1334 m elevation of the crest and 1205 m elevation of the dam foundation corridor of the 9# dam section, respectively, to monitor the displacement along and perpendicular to the river. LA9 is monitored by vacuum laser alignment (LA), and IP3 is monitored by inverted plumb lines (IP). Figure 8 depicts the working principle of the vacuum laser and inverted plumb line for dam deformation monitoring. LA systems are commonly used to monitor relative movement from the dam crest to the bank. The light spot at the receiving end changes with the movement of the zone plate on the measuring point. Therefore, the relative displacement of the measuring point can be calculated via the positional relationship among the transmitting end, the zone plate (measuring point), and the receiving end. At the same time, the absolute displacement of the vacuum laser measuring point can be obtained by superimposing the absolute displacement of the endpoint. IP systems are generally used to monitor relative movement from the dam's rock foundation to the dam's gallery level. The IPs include the floating ball group, the plumb line, and the anchoring point. The buoyancy of the floating ball keeps the vertical line in a vertical state. By regularly monitoring the movement of the floating ball, the displacement of the inspection gallery relative to the vertical line or the bottom of the borehole (assumed to be a fixed point of horizontal displacement) can be obtained. In addition, a stress sensor S9 was installed at the 1275 m elevation to measure the concrete stress of vertical and downstream direction in this area, which is the principal direction of interest for the stress monitoring of gravity dams.



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

Figure 8. Instrument schematic of the dam deformation monitoring system (X-direction, Y-direction represent the direction along the river and the direction perpendicular to the river, respectively). (**a**) Inverted plumb line, (**b**) Vacuum laser.

3.2. Data Configuration

The observed value curves of the selected monitoring points are presented in Figure 9. For model validation, this paper assumed that the stress sensor S9 in dam section 9# was damaged at a specific time on 11 April 2015. In other words, the monitoring data of the 9# dam section from 7 November 2014 to 4/ November 2015 (containing 275 days of continuous data) and 4 December 2015 to 7 December 2015 (containing 90 days of continuous data) are selected for model training and testing, respectively. Then, the estimation model was trained with an input matrix size of 10 × 4 (4 sets of deformation monitoring values in 10 consecutive days of monitoring points LA9 and IP3) and output size of 1 × 1 (one of the two direction stress values on the 11th day of sensor S9). So the training set size is 275 − 10 = 265, and the test set size is 90.

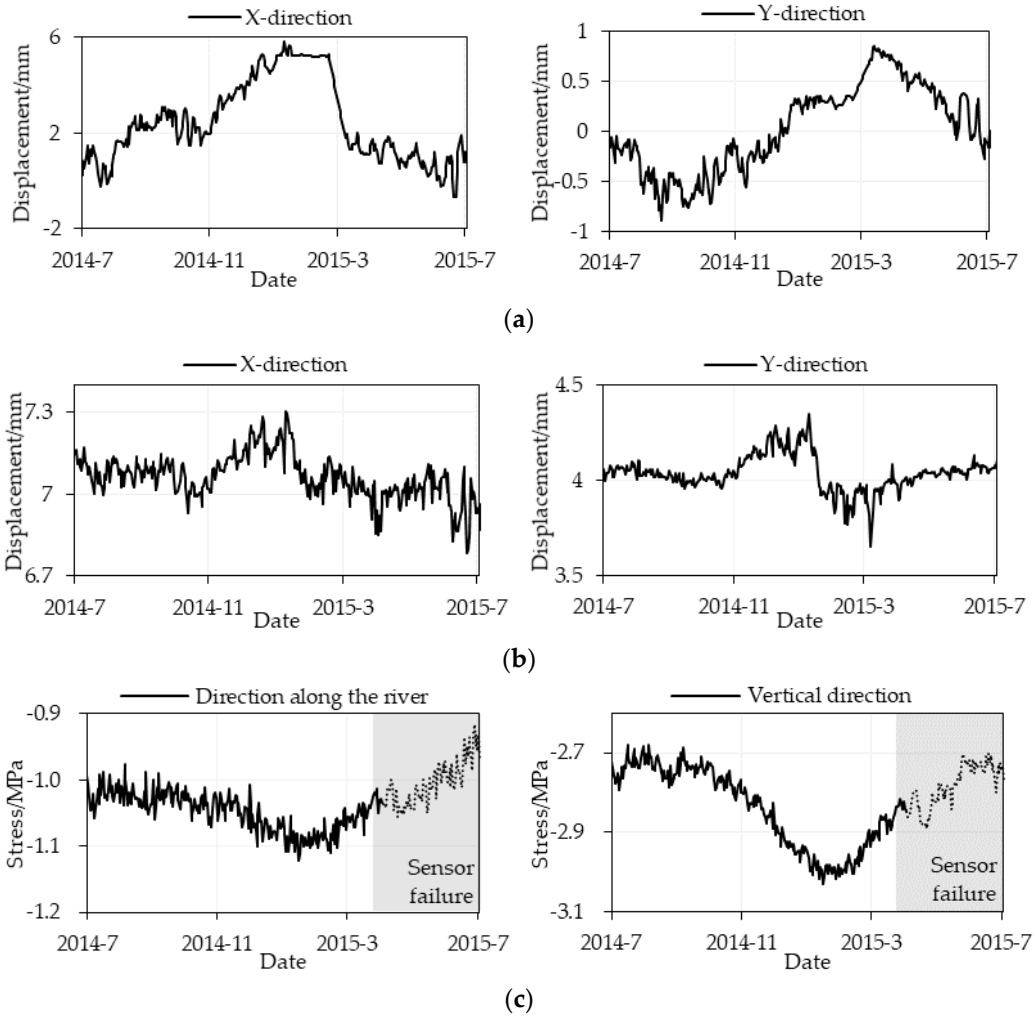


Figure 9. Time curves of the monitoring data (X-direction, Y-direction represent the direction along the river and the direction perpendicular to the river, respectively). The shaded area is the sensor failure period. (**a**) Vacuum laser alignment monitoring point LA9, (**b**) Inverted plumb line monitoring point IP3, and (**c**) Stress sensor S9.

4. Results

4.1. Hyperparameter Tuning Result of the Model

Lesser layers and a concise structure can maintain crucial features in the training process of the DL model with a small sample size [31]. Considering the amount of data involved, we set the domains of the model hyperparameters to small values to maintain a concise structure. Moreover, the training set was divided into five folds containing subsets

for the time series CV. The hyperparameter combinations of the presented model tuned via SIE–APSO and time series CV technique are reported in Table 1. Notably, the CNN layer is implemented on each day's data. Hence, the number of CNN filters is a vector of size 10 for the model with ten consecutive days' worth of data of the input matrix. See the Appendix for more details about the hyperparameter results.

Table 1. The hyperparameters of the presented model (take the vertical direction as an example).

Layers	Parameters	Domain (Upper Limit, Lower Limit)	Optimization Value
CNN	filter number of each layer	(2, 64)	(27, 35, …, 23)
LSTM	unit number	(2, 32)	11
	dropout rate	(0, 0.5)	0.187
FC	unit number	(2, 32)	15
Optimizers	algorithms	-	Adagrad
	learning rate	(10^{-8}, 10^{5})	0.300
-	batch size	(32, 256)	76

CNN: convolutional neural network; LSTM: long short-term memory network; FC: fully connected layer.

4.2. Estimation Stress Results Evaluation

The estimated curve of the proposed learning model for the normal stress of vertical and downstream directions is presented in Figure 10. The shaded part indicates the failure period of the target stress sensor. Figure 11 depicts the error-epoch charts of training and testing sets. The evaluation results of the training and testing periods are presented in Table 2. The results show that the estimation model maintains a good estimate of accuracy in both the training and testing periods. Specifically, the estimation performance of the vertical direction does not show degradation in the testing period, while the RMSE increased by around 40% of the direction along the river. We suspect the reason is that the stress observations on the direction along the river were subjected to greater perturbations than those in the vertical direction, as shown in the time curves. This perturbation may be caused by observation errors. Nevertheless, the estimated curves still appropriately display the approximate variation of stresses without any overfitting phenomena.

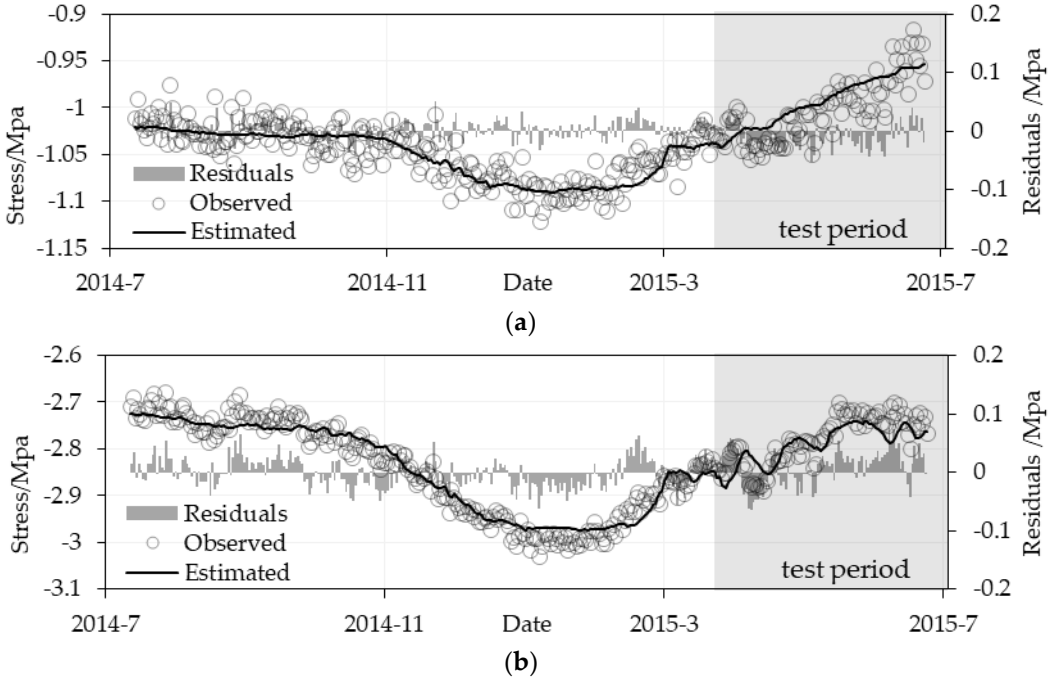


Figure 10. Predictions of the proposed estimation model (lines) versus observed data (circles). The residuals between them are shown in the bar chart. Shaded part is the testing period. (**a**) Direction along the river. (**b**) Vertical direction.

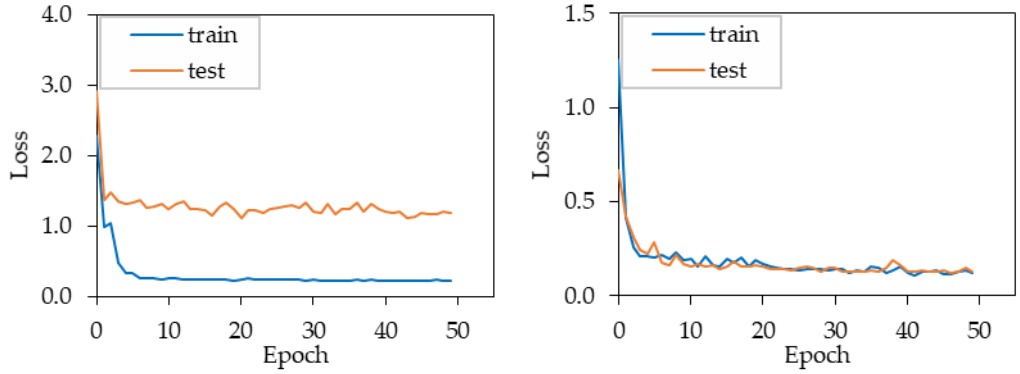


Figure 11. Error-epoch charts of training and testing sets of the proposed estimation model. (**a**) Direction along the river. (**b**) Vertical direction.

Table 2. Evaluation results of the proposed model.

Index	RMSE		MAPE		ARV	
Direction	Training Period	Testing Period	Training Period	Testing Period	Training Period	Testing Period
Direction along the river	0.016	0.021	1.195%	1.678%	0.319	0.378
Vertical direction	0.024	0.032	0.678%	0.973%	0.055	0.387

RMSE: root mean square error; MAPE: average absolute percentage error; ARV: average relative variance.

5. Discussion

5.1. Estimation Performance Comparison

In this section, three conventional models established based on the load-stress relationship—HST, SVR, and NN—are adopted for the performance comparison of the estimation models proposed. The model factors of SVR and NN are the same as HST. Detailed and rigorous descriptions can be found in [30]. The hyperparameters of SVR and NN have been tuned as in the proposed model. Considering the limited monitoring points and the method reliability, this work does not adopt the method estimated from the data of adjacent points of the same sensor type as a comparison. Due to space constraints, Figure 12 gives the estimation results of different models in the Z-direction only. The performance evaluation results of each model are shown in Figure 13. The following can be deduced from the comparison results.

HST and NN fitting performances are barely satisfactory during the training period, while the difference comes in the prediction performance compared with the proposed model. The curve of the HST model exhibits significant cyclical variation throughout, resulting in an overall more considerable estimated value than the observed value in the testing period. In addition, the curves of the HST and NN models both show two significant fluctuations during the testing period. The analysis suggests that the water level change, which is not experienced during the training period, as in Figure 12(d), results in significant fluctuations in the model estimates at the corresponding moments, but the observed fluctuations are relatively stable. SVR models perform poorly in both the training and testing periods. For the performance gain, from the evaluation index perspective, the RMSE of the two-directional stress estimated from the proposed model has decreased by approximately 0.016(21%)–0.044(58%); compared with three comparison models, the

MAPE decreased by 0.443%(19%)–1.355%(58%), and the ARV decreased by 0.470(22%)–1.303(94%).

We speculate that the reasons are as follows. The input factors of the above models were frequently unable to describe the dam's load-stress relationship comprehensively. The estimation results of these models have a greater probability of exaggerated fluctuations facing an unexperienced load (water level fluctuation), which is not the case for the actual structure. In comparison, dam deformation and stress are symbiotic responses of the structure, and the relationship between the two is more constrained by the system's intrinsic state. Therefore, describing the association among these responses is more favorable for estimation than the relationship between structural stress and external loads.

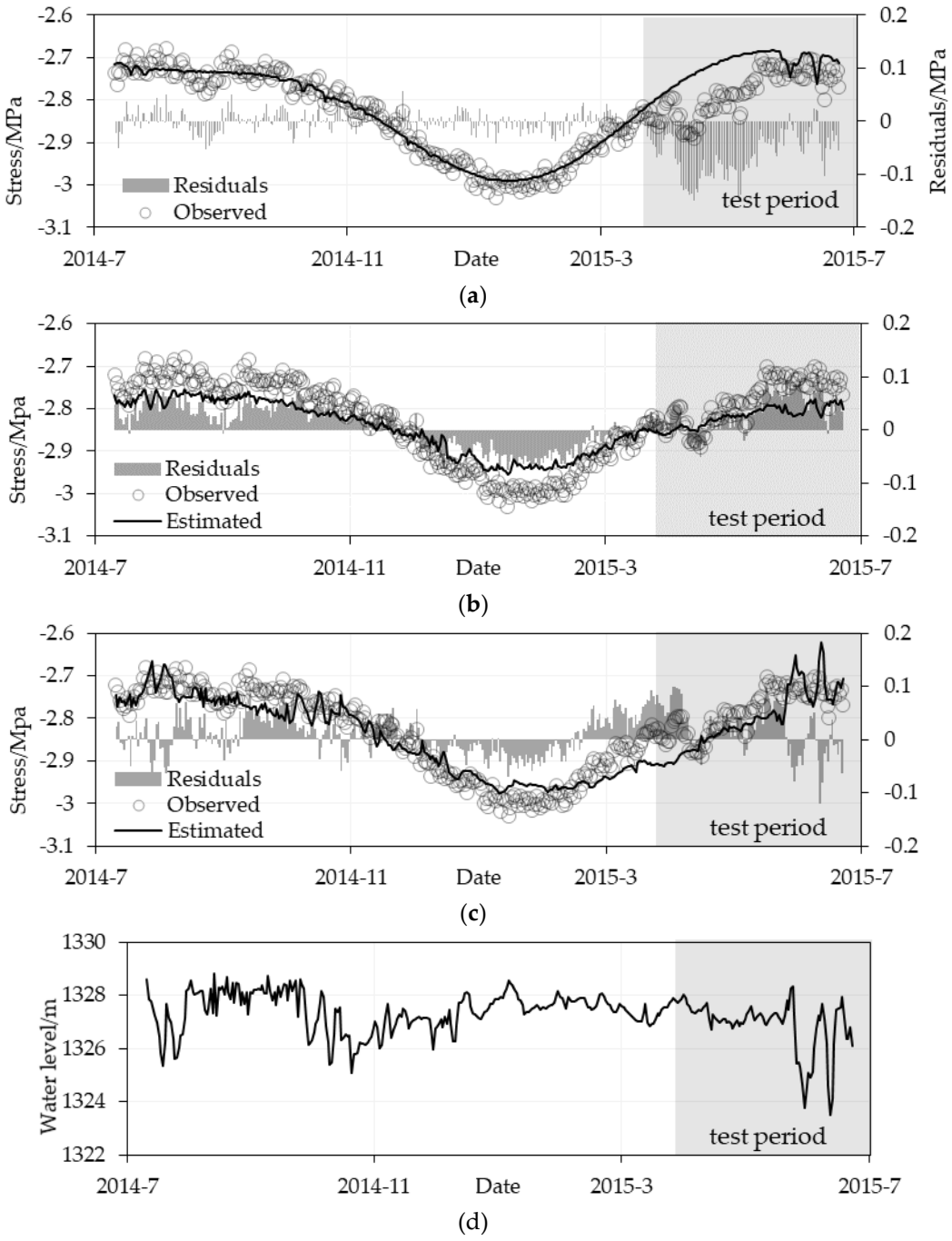
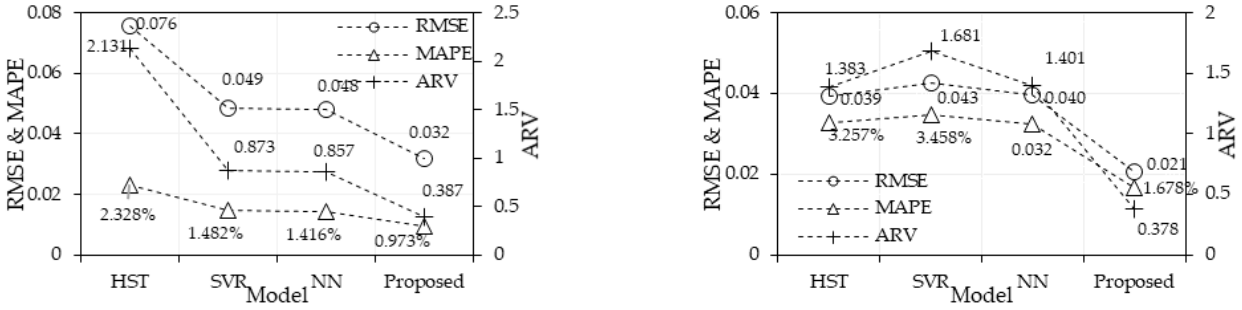


Figure 12. Time curve of each estimation model and water level. Shaded part is the testing period. (**a**) HST. (**b**) SVR. (**c**) NN. (**d**) Time curve of water level. HST: hydrostatic-seasonal-time statistical model; SVR: support vector regression; NN: neural network with single hidden layer.



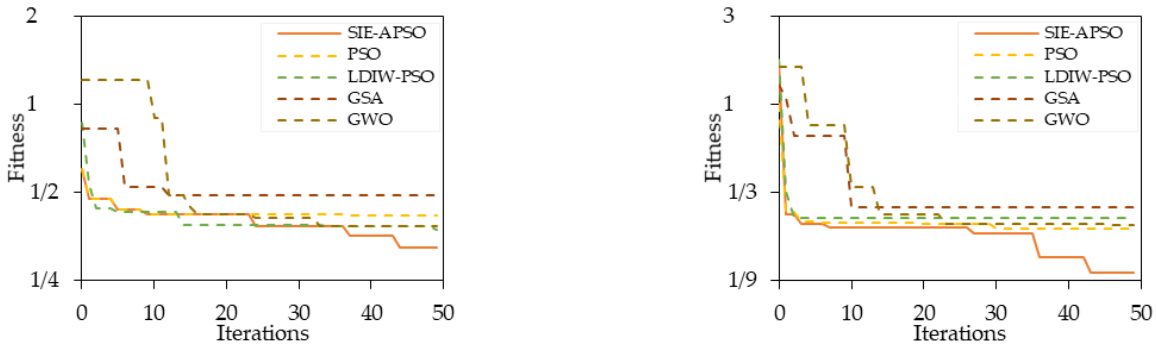
(**a**) (**b**)

Figure 13. Evaluation metrics for each model. The results correspond to the testing period. (**a**) Direction along the river, (**b**) Vertical direction.

The comparison analysis demonstrates that the proposed estimation model combining deformation data performs better in dealing with the stress estimation under sensor failure and data loss scenarios, and especially in dealing with dam responses under unexperienced conditions.

5.2. Evaluation of Tuning Algorithms SIE–APSO

In this section, we analyze the performance of the SIE–APSO proposed in this paper via other algorithms for algorithm verification. The SIE–APSO is compared to the standard PSO [32], linear descent inertia weight strategy PSO (LDIW–PSO) [33], the standard grey wolf optimizer (GWO) [34] as SI (swarm intelligence)-based technique, and the gravitational search algorithm (GSA) [35] as a physics-based algorithm. Note that all algorithms were run on Intel (R) Core (TM) i5-8400 with 16 G memory, and the software environment is python 3.7. The population number of each algorithm was set to 20; the maximum number of iterations was 50; and the maximum particle velocity was set to 10% of the search domain. The convergence curve of each tuning algorithm is reported in Figure 14. Regarding the dataset in this work, the convergence curve shows that the GSA and GWO algorithms converge slowly and have a poor optimization result. In contrast, the PSO and its improved algorithms show a faster convergence process. Moreover, the SIE–APSO algorithm can jump out of the local solution and achieve more satisfactory results when the PSO and LDIW–PSO algorithms reach the convergence bottleneck.



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

Figure 14. Performance comparison of SIE–APSO (swarm information entropy-based alienation particle swarm optimization), PSO (particle swarm optimization), LDIW–PSO (linear descent inertia weight strategy particle swarm optimization), GSA (gravitational search algorithm), and GWO (standard grey wolf optimizer). Convergence curve of each tuning algorithm. (**a**) Direction along the river, (**b**) Vertical direction.

In order to evaluate the algorithms' efficiency, we further give the time cost of each algorithm in the same experimental environment in Table 3. We are informed from this

result that the performance improvement brought by SIE–PSO inevitably results in a slight increase of the time cost of 19.8% and 23.9% in two directions. However, the cost is still within an acceptable range compared to GWO and GSA, and the its performance is improved by 38.0% and 71.3% in two directions compared to others. Consequently, the proposed SIE–APSO algorithm is preferred for hyperparameter tuning of the estimation model, considering the algorithm's performance and efficiency.

Table 3. Time cost of optimization (unit: second).

Direction	SIE–APSO	PSO	LDIW–PSO	GSA	GWO
Direction along the river	1283.1	1074.7	1473.2	6050.2	1887.6
Vertical direction	1755.4	1236.7	963.8	6738.8	2493.7
Average	1519.2	1155.7	1218.5	6394.5	2190.7

SIE–APSO: swarm information entropy-based alienation particle swarm optimization; PSO: particle swarm optimization; LDIW–PSO: linear descent inertia weight strategy particle swarm optimization; GSA: gravitational search algorithm; GWO: standard grey wolf optimizer.

5.3. Effect Analysis of the Input Sample Length

In order to further analyze the effect of the input sample length on the estimated model, 4 sample sets with different input lengths from 10 to 80 are used for evaluation. The evaluation indexes of the model estimation results with different input lengths are given in Figure 15. The model error tends to fluctuate and increase with the more considerable input length, as shown in the trend line marked by the dashed line in this figure. The analysis results indicate that the length of the input samples affects the model estimation accuracy, and too lengthy periods will be more unfavorable.

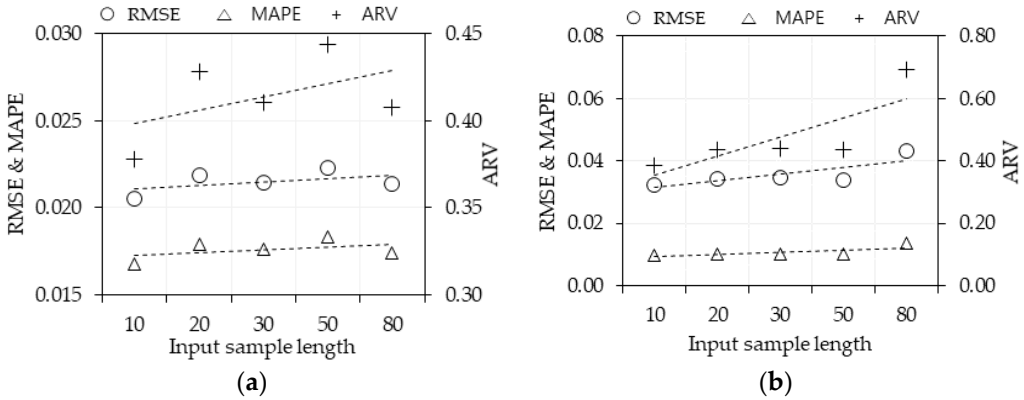


Figure 15. Estimation results of proposed model with different input sample lengths. The dashed line is the fitted trend line. (**a**) Direction along the river, (**b**) Vertical direction.

6. Summary and Conclusions

A learning model based on CNN and LSTM was developed for response estimation of concrete dams under stress sensor failure and data loss scenarios. This model presented the estimation result based on the spatial–temporal association of data between the target stress sensor and multiple deformation sensors. Simultaneously, an improved PSO algorithm for hyperparameter tuning was studied. The presented estimation model was validated on the monitoring data of an actual engineering project. The main conclusions are as follows:

1. By comparing three conventional models established with load-stress relationships, this work verified the feasibility of the proposed estimation model based on the data's spatial–temporal association among the multiple monitoring points of dam deformation and stress.

2. The proposed tuning algorithm SIE–APSO, which maintains higher computational accuracy and stability without losing efficiency compared with the standard PSO, is presented and has been tested on the target dataset.
3. The estimation method provides reliable data supplements for the strength evaluation of concrete dams under the scenario of sensor failure and data loss, especially dealing with the dam responses under unexperienced conditions.

However, like all data-driven estimation models currently, the association consistency among structural stress and deformation and the estimation accuracy may be partially affected by the integrity of the concrete. Therefore, limitations of this method exist. The period in this work is one year, and the application scenario is specified to a short-term range where the structure is not subject to significant disturbances, such as damage. This means that estimating over a more extended period needs to consider changes in the structural state. In addition, fewer data samples are commonly insufficient for tuning and training deep network models.

Future research can combine inversion and forward calculation for timely updating of the model to achieve longer-term estimation for the above limitation. Some investigations may also benefit from the advancement of algorithms in the deep learning (DL) field, such as combining transfer learning with numerical models to supplement data samples for real structures.

Author Contributions: Conceptualization, L.T., D.Z. and X.W.; methodology, L.T. and X.W.; software, L.T., X.W., X.C., Y.L., Z.C. and H.J.; validation, L.T., D.Z., X.W. and Z.C.; formal analysis, L.T. and H.J.; investigation, L.T. and X.C.; resources, D.Z., L.T. and X.W.; data curation, L.T., X.W. and X.C.; writing—original draft preparation, L.T.; writing—review and editing, L.T., X.W. and Y.L.; visualization, L.T., X.W. and X.C.; supervision, D.Z. and Y.L.; project administration, D.Z.; funding acquisition, D.Z. and L.T. All authors have read and agreed to the published version of the manuscript.

Funding: This research has been funded by the National Natural Science Foundation of China [Grant No. 52179128]; Postgraduate Research & Practice Innovation Program of Jiangsu Province [Grant No. 2015B33214].

Data Availability Statement: The data that support the findings of this study are available from the corresponding author, [Xin Wu], upon reasonable request.

Conflicts of Interest: The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Appendix A

Table A1. The hyperparameters of the proposed model (direction along the river).

Layers	Parameters	Domain [upper limit, lower limit]	Optimization Value
CNN	filter number of each layer	(2, 64)	(13, 19, 10, 9, 16, 15, 17, 14, 15, 12)
LSTM	unit number	(2, 32)	19
	dropout rate	(0, 0.5)	0.335
MLP	unit number	(2, 32)	23
Optimizers	algorithms	-	Adagrad
	learning rate	(10^{-5}, 10^{-1})	0.100
-	batch size	(32, 256)	165

Table A2. The hyperparameters of the proposed model (vertical direction).

Layers	Parameters	Domain [Upper Limit, Lower Limit]	Optimization Value

CNN	filter number of each layer	(2, 64)	(27, 35, 21, 12, 24, 33, 26, 31, 20, 23)
LSTM	unit number	(2, 32)	11
	dropout rate	(0, 0.5)	0.187
MLP	unit number	(2, 32)	15
Optimizers	algorithms	-	Adagrad
	learning rate	(10^{-5}, 10^{-1})	0.300
-	batch size	(32, 256)	76

Appendix B

The following is the code of the CNN–LSTM model implemented based on Keras 2.6.0.

```
def buildModel(param):
    # param: the network hyperparameter
    # output dimension
    output_dim = 1
    # cnn
    cnn_input = layers.Input(shape = (sample_length, feature_size_cnn, 1))
    # split layer
    split_cnn_input = layers.Lambda(lambda x: tf.split(x, sample_length, axis = 1))(cnn_input)
    # single cnn layer
    for i in range(sample_length):
        locals()['cnn_spf' + str(i + 1)] = layers.Conv2D(filters = int(param[i]), kernel_size = cnn_filters_size, padding = padding_method)(split_cnn_input[i])
        locals()['cnn_spf' + str(i + 1)] = layers.Flatten()(locals()['cnn_spf' + str(i + 1)])
        locals()['cnn_spf' + str(i + 1)] = layers.Dense(1)(locals()['cnn_spf' + str(i + 1)])
    # concatenate layer
    cnn_spf_all = locals()['cnn_spf1']
    for i in range(sample_length-1):
        cnn_spf_all = layers.concatenate([cnn_spf_all, locals()['cnn_spf' + str(i + 2)]], axis=-1)
    cnn_spf_all = tf.expand_dims(cnn_spf_all, axis = −1)
    # single layer LSTM_spf
    lstm_spf_tpf = layers.LSTM(units=int(param[sample_length]), activation = 'softsign', dropout = param[sample_length + 1])(cnn_spf_all)
    # Flatten
    f = layers.Dense(param[sample_length + 2])(lstm_spf_tpf)
    # output
    out = layers.Dense(output_dim)(f)
    # model compile
    model = Model([cnn_input], out)
    optimizer = optimizers. Adagrad(learning_rate = 10. ** param[sample_length + 3])
    model.compile(loss='mse', optimizer=optimizer)
    return model
```

References

1. Li, M.-L.; Wang, Z.-M.; Ma, N. A Novel Prediction Model for the Missing Data of Environmental Measurement. *J. Sichuan Univ.* **2003**, *4*, 736–739. Available online https://www.doc88.com/p-0993274892105.html (accessed on 29 November 2022)
2. Zhang, Z.; Luo, Y. Restoring method for missing data of spatial structural stress monitoring based on correlation. *Mech. Syst. Signal Process.* **2017**, *91*, 266–277. https://doi.org/10.1016/j.ymssp.2017.01.018.

3. Wang, J.; Yang, J.; Cheng, L. An interpolation method based on KICA-RVM for missing monitoring data of dam. *J. Water Resour. Water Eng.* **2017**, 28, 197–201. Available online http://szyysgcxb.alljournals.ac.cn/szyysgcxb/ch/reader/view_abstract.aspx?doi=10.11705/j.issn.1672-643X.2017.01.35 (accessed on 29 November 2022)
4. Li, B.; Yang, J.; Hu, D. Dam monitoring data analysis methods: A literature review. *Struct. Control Health Monit.* **2020**, 27, e2501. https://doi.org/10.1002/stc.2501.
5. Mata, J.; Tavares de Castro, A.; Sá da Costa, J. Constructing statistical models for arch dam deformation. *Struct. Control Health Monit.* **2014,** *21*, 423–437. https://doi.org/10.1002/stc.1575.
6. Su, H.; Li, X.; Yang, B.; Wen, Z. Wavelet support vector machine-based prediction model of dam deformation. *Mech. Syst. Signal Process.* **2018**, *110*, 412–427. https://doi.org/10.1016/j.ymssp.2018.03.022.
7. Belmokre, A.; Mihoubi, M.K.; Santillán, D. Analysis of Dam Behavior by Statistical Models: Application of the Random Forest Approach. *KSCE J. Civ. Eng.* **2019**, *23*, 4800–4811. https://doi.org/10.1007/s12205-019-0339-0.
8. Lin, C.; Li, T.; Chen, S.; Liu, X.; Lin, C.; Liang, S. Gaussian process regression-based forecasting model of dam deformation. *Neural Comput. Appl.* **2019**, *31*, 8503–8518. https://doi.org/10.1007/s00521-019-04375-7.
9. Salazar, F.; Toledo, M..; González, J.M.; Oñate, E. Early detection of anomalies in dam performance: A methodology based on boosted regression trees. *Struct. Control Health Monit.* **2017**, *24*, e2012. https://doi.org/10.1002/stc.2012.
10. Liu, W.; Pan, J.; Ren, Y.; Wu, Z.; Wang, J. Coupling prediction model for long-term displacements of arch dams based on long short-term memory network. *Struct. Control Health Monit.* **2020**, *27*, e2548. https://doi.org/10.1002/stc.2548.
11. Zhang, J.; Cao, X.; Xie, J.; Kou, P. An Improved Long Short-Term Memory Model for Dam Displacement Prediction. *Math. Probl. Eng.* **2019**, *2019*, 6792189. https://doi.org/10.1155/2019/6792189.
12. Wu, X.; Zheng, D.-J.; Liu, Y.-T.; Chen, Z.-Y.; Chen, X.-Q. Temporal convolution network-based time frequency domain integrated model of multiple arch dam deformation and quantification of the load impact. *Struct. Control Health Monit.* **2022**, *29*, e3090. https://doi.org/10.1002/stc.3090.
13. Rawat, W.; Wang, Z. Deep Convolutional Neural Networks for Image Classification: A Comprehensive Review. *Neural Comput.* **2017**, *29*, 2352–2449. https://doi.org/10.1162/neco_a_00990.
14. Khan, S.; Yairi, T. A review on the application of deep learning in system health management. *Mech. Syst. Signal Process.* **2018**, 107, 241–265. https://doi.org/10.1016/j.ymssp.2017.11.024.
15. Long, J.; Shelhamer, E.; Darrell, T. Fully convolutional networks for semantic segmentation. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Boston, MA, USA, 7–12 June 2015; pp. 3431–3440. https://doi.org/10.1109/CVPR.2015.7298965.
16. Werbos, P.J. Generalization of backpropagation with application to a recurrent gas market model. *Neural Netw.* **1988**, *1*, 339–356. https://doi.org/10.1016/0893-6080(88)90007-x.
17. Hochreiter, S.; Schmidhuber, J. Long Short-Term Memory. *Neural Comput.* **1997**, *9*, 1735–1780. https://doi.org/10.1162/neco.1997.9.8.1735.
18. Yu, Y.; Si, X.; Hu, C.; Zhang, J. A Review of Recurrent Neural Networks: LSTM Cells and Network Architectures. *Neural Comput.* **2019**, *31*, 1235–1270. https://doi.org/10.1162/neco_a_01199.
19. Zhu, G.; Zhang, L.; Shen, P.; Song, J. Multimodal Gesture Recognition Using 3-D Convolution and Convolutional LSTM. *IEEE Access* **2017**, *5*, 4517–4524. https://doi.org/10.1109/access.2017.2684186.
20. Yang, Y.; Dong, J.; Sun, X.; Lima, E.; Mu, Q.; Wang, X. A CFCC-LSTM Model for Sea Surface Temperature Prediction. *IEEE Geosci. Remote Sens. Lett.* **2017**, *15*, 207–211. https://doi.org/10.1109/lgrs.2017.2780843.
21. Zöller, M.-A.; Huber, M.F. Benchmark and Survey of Automated Machine Learning Frameworks. *J. Artif. Intell. Res.* **2021**, *70*, 409–472. https://doi.org/10.1613/jair.1.11854.
22. Yang, L.; Shami, A. On hyperparameter optimization of machine learning algorithms: Theory and practice. *Neurocomputing* **2020**, 415, 295–316. https://doi.org/10.1016/j.neucom.2020.07.061.
23. Bonyadi, M.R.; Michalewicz, Z. Particle swarm optimization for single objective continuous space problems: A review. *Evol. Comput.* **2017**, *25*, 1–54. https://doi.org/10.1162/EVCO_r_00180.
24. Zhan, Z.-H.; Zhang, J.; Li, Y.; Shi, Y.-H. Orthogonal Learning Particle Swarm Optimization. *IEEE Trans. Evol. Comput.* **2010**, *15*, 832–847. https://doi.org/10.1109/tevc.2010.2052054.
25. Gu, J.; Wang, Z.; Kuen, J.; Ma, L.; Shahroudy, A.; Shuai, B.; Liu, T.; Wang, X.; Wang, G.; Cai, J.; et al. Recent advances in convolutional neural networks. *Pattern Recognit.* **2018**, *77*, 354–377. https://doi.org/10.1016/j.patcog.2017.10.013.
26. Cho, K.; Van Merrienboer, B.; Bahdanau, D.; Bengio, Y. On the properties of neural machine translation: Encoder-decoder approaches. In Proceedings of the Eighth Workshop on Syntax, Semantics and Structure in Statistical Translation (SSST-8), Doha, Qatar, 25 October 2014; pp. 103–111. https://doi.org/10.3115/v1/w14-4012.
27. Bratton, D.; Kennedy, J. Defining a Standard for Particle Swarm Optimization. In Proceedings of the 2007 IEEE Swarm Intelligence Symposium, Honolulu, HI, USA, 1–5 April 2007; pp. 120–127. https://doi.org/10.1109/SIS.2007.368035.
28. Tashman, L.J. Out-of-sample tests of forecasting accuracy: An analysis and review. *Int. J. Forecast.* **2000**, *16*, 437–450. https://doi.org/10.1016/s0169-2070(00)00065-0.

29. Bergmeir, C.; Benítez, J.M. On the use of cross-validation for time series predictor evaluation. *Inf. Sci.* **2012**, *191*, 192–213. https://doi.org/10.1016/j.ins.2011.12.028.
30. Salazar, F.; Toledo, M.A.; Oñate, E.; Morán, R. An empirical comparison of machine learning techniques for dam behaviour modelling. *Struct. Saf.* **2015**, *56*, 9–17. https://doi.org/10.1016/j.strusafe.2015.05.001.
31. Smith, L.N.; Topin, N. Deep convolutional neural network design patterns. *arXiv* **2016**, https://doi.org/10.48550/arXiv.1611.00847.
32. Kennedy, J.; Eberhart, R. Particle Swarm Optimization. In Proceedings of the ICNN'95—International Conference on Neural Networks, Perth, Australia, 27 November–1 December 1995; Volume 4, pp. 1942–1948. https://doi.org/10.1109/ICNN.1995.488968.
33. Shi, Y.; Eberhart, R. A modified particle swarm optimizer. In Proceedings of the 1998 IEEE International Conference on Evolutionary Computation Proceedings, Anchorage, AK, USA, 4–9 May 1998; pp. 69–73. https://doi.org/10.1109/ICEC.1998.699146.
34. Mirjalili, S.; Mirjalili, S.M.; Lewis, A. Grey Wolf Optimizer. *Adv. Eng. Softw.* **2014**, *69*, 46–61. https://doi.org/10.1016/j.advengsoft.2013.12.007.
35. Rashedi, E.; Nezamabadi-Pour, H.; Saryazdi, S. GSA: A gravitational search algorithm. *Inf. Sci.* **2009**, *179*, 2232–2248. https://doi.org/10.1016/j.ins.2009.03.004.