Supplementary Information

Computational intelligence for soft strain sensor sustainability

Authors: Jiali Li[1,‡], Haitao Yang[2,3,‡*], Lanjing Wang[2,‡], Runxiang Li[4], Chong Sun[5], Fuhui Zhou[6], Qiulei Liu[2], Wei Wang[2], Jinghan Li[7], Chengzhi Hu[1], Xiaodong Chen[8,9*], Xiaonan Wang[10*]

Affiliations:

[1] Key Laboratory of Environmental Aquatic Chemistry, State Key Laboratory of Regional Environment and Sustainability, Research Center for Eco-Environmental Sciences, Chinese Academy of Sciences, Beijing, China.

[2]Institute of Flexible Electronics (IFE) & Frontiers Science Center for Flexible Electronics, Northwestern Polytechnical University, Xi'an, Shaanxi, China.

[3]Henan Institute of Flexible Electronics (HIFE), Zhengzhou, Henan, China.

[4]Department of Chemical and Biomolecular Engineering, National University of Singapore, Singapore, Singapore.

[5]School of Astronautics, Northwestern Polytechnical University, Xi'an, China.

[6]College of Artificial Intelligence, Nanjing University of Astronautics and Aeronautics, Nanjing, China.

[7]School of Environmental Science and Engineering, Suzhou University of Science and Technology, Suzhou, China.

[8]Innovative Center for Flexible Devices (iFLEX), Max Planck-NTU Joint Laboratory for Artificial Senses, School of Materials Science and Engineering, Nanyang Technological University, Singapore, Singapore.

[9]Institute for Digital Molecular Analytics and Science (IDMxS), Nanyang Technological University, Singapore, Singapore.

[10]Department of Chemical Engineering, Tsinghua University, Beijing, China.

[‡]These authors contribute equally to this work.

*Email: iamhtyang@nwpu.edu.cn (H. Yang), chenxd@ntu.edu.sg (X. Chen), wangxiaonan@tsinghua.edu.cn (X. Wang).

This PDF file includes:

Supplementary Notes 1 to **21**

Supplementary Figs. 1 to **38**

Supplementary Tables 1 to **6**

Supplementary Movies 1 to **2**

Supplementary Fig. 22	Comparison of predicted and actual applied strains by black-box sequential models in the last 100 cycles of the cycling attenuation task based on sensor resistance feedback.
Supplementary Fig. 23	Predictive performance of the LSTM model on the cycling attenuation issue.
Supplementary Fig. 24	Predictive performance of the GRU model on the cycling attenuation issue.
Supplementary Fig. 25	Predictive performance of the 1DCNN model on the cycling attenuation issue.
Supplementary Fig. 26	Predictive performance of the Transformer model on the cycling attenuation issue.
Supplementary Fig. 27	Predictive performance of the D_Former model with lower parameter counts on the cycling attenuation issue.
Supplementary Fig. 28	Resistance-Time curves of strain sensors in Batch 5.
Supplementary Fig. 29	Predictive performance of different models for the cycling attenuation during the first 2000 cycles of Batch 5.
Supplementary Fig. 30	Predictive performance of the LSTM model on the batch inconsistency issue (Batch 5).
Supplementary Fig. 31	Predictive performance of the GRU model on the batch inconsistency issue (Batch 5).
Supplementary Fig. 32	Predictive performance of the 1DCNN model on the batch inconsistency issue (Batch 5).
Supplementary Fig. 33	Predictive performance of the Transformer model on the batch inconsistency issue (Batch 5).
Supplementary Fig. 34	Predictive performance of the D_Former model on the batch inconsistency issue (Batch 5).
Supplementary Fig. 35	Sequential motion analysis of quadrupedal soft robot at different time points.
Supplementary Fig. 36	Predictive performance of the D_Former model on dexterous robot hand application (train on Batch 2, 3, predict on Batch 1).

Supplementary Note 1. Overview of sensor fabricated for each issue and applications.

Due to the limited shelf life of the strain sensors, separate batches were fabricated for various issues and applications. Strain-resistance data from each sensor was extracted and used to train models for different issues.

Nonlinearity issue: three different carbon waste-based strain sensors were developed. These sensors varied in their composition with carbon black waste percentages of 2wt%, 4wt% and 6wt%, and they are named Batch 1st, Batch 2nd and Batch 3rd, respectively.

Hysteresis issue: the hysteresis issue utilized a single sensor containing 6% carbon black waste**,** HysteresisSensor 1.

Cycling attenuation issue: the cycling attenuation issue utilized a single sensor containing 6% carbon black waste, CyclingAttenuationSensor 1.

Batch inconsistency issue: for the batch inconsistency issue, five batches of sensors were produced, all containing 6% carbon black waste. They were all from different production lots, and all of them showed a decrease in durability over time. For the batch inconsistency problem, the first four batches of sensors, i.e., Batch 1, Batch 2, Batch 3, and Batch 4, were used for model training. The remaining Batch 5 is used for model testing.

Flexible robot arm applications

For the application of flexible robot arm end-effector position recognition, we prepared four batches of sensors using a 6% carbon black waste ratio, and four sensors (Sensor①, Sensor②, Sensor③, Sensor④) in each batch were installed on both sides of the two key joints of the robotic arm in batches (see **Fig. 7a** for the installation locations), to collect the position signals of the flexible robotic arm movement.

Data from the first three batches of sensors (Batch1Sensors; Batch2Sensors; Batch3Sensors) were used for model training. The fourth batch (Batch4Sensors) was used for testing.

Soft quadruped robot applications

For the Soft Quadruped Robot Applications, we prepared three batches of sensors using a 6% carbon black waste ratio, with four sensors in each batch. One batch of sensors, referred to as "BatchXSensors," is installed on the four limbs of the soft quadruped robot during each movement (the specific installation locations are shown in **Figure 7f**).

The motion signals collected by the first two batches of sensors (Batch1Sensors and Batch2Sensors) were utilized for model training, while the third batch (Batch3Sensors) was reserved for testing.

Dexterous robot hand applications

For the dexterous robot hand application, we also fabricated three batches of sensors, one sensor per batch, all of which were prepared using 6% carbon black waste. The three sensors were positioned on the three robotic fingers (see **Supplementary Fig. 38a** for the exact mounting locations). Two of these sensors (i.e., Batch 1 and Batch 2) were used for model training. The third sensor (i.e., Batch 3) was employed for testing.

Supplementary Note 2. Life cycle assessment methodology.

The goal of the Life Cycle Assessment (LCA) experiment is to compare the environmental impacts of the fabrications of four different strain sensors: MXene-based strain sensor, silver nanowires-based strain sensor, graphene-based strain sensor and carbon waste-based (CW) strain sensor.

This LCA leverages the data pool from the 2013 Fifth Assessment Report by the Intergovernmental Panel on Climate Change (IPCC) to quantify the carbon emissions throughout the production stages of these sensors. The IPCC's report, known for its examination of climate science, provides reliable and updated emission factors, ensuring the LCA calculations are rooted in the latest scientific consensus. The LCA study primarily emphasizes the preparation steps in producing various sensors, assessing environmental impacts based on factors like material usage, waste, and equipment energy consumption at each phase. The LCA of the four different strain

sensors is calculated based on their fabrication procedures[1–3] as illustrated in **Supplementary Figs. 2-5**. A step-by-step explanation of the calculation is cataloged in **Supplementary Tables 1-4**.

The carbon emissions calculated from LCA of four different strain sensors are further analyzed. The critical steps that contribute to carbon emissions during sensor preparation include electricity energy, substrate materials, active materials and consumables.

Fig. 2c and Supplementary Figs. 6a-c presents the Pie chart analysis of carbon footprints for CW and nanomaterial sensors, respectively. In **Fig. 2c**, with the recycled CW materials and simple fabricate procedure, 98% carbon footprints of CW sensor come from its elastic substrate (i.e., Ecoflex). As comparison in **Supplementary Fig. 6a**, the carbon footprints of MXene sensor consisted of many parts, and elastic substrate (i.e., Ecoflex) only contributed 5.5% of total CO_2 emission. Besides, as shown in **Supplementary Figs. 6b** and **6c**, the elastic materials in silver nanowires-based and graphene-based sensors taken up 25% and 28.6% CO_2 emissions, respectively. Because elastic material is a necessary component of soft sensor, a low proportion of CO_2 emission from elastic material means a large total CO_2 emission and low eco-friendliness level, and vice versa. From this point of view, among various nanomaterial sensors, CW sensor represents an ideal option to fabricate eco-friendly soft sensors.

Supplementary Note 3. Training of white-box (sigmoid function) model for nonlinearity issue.

In the context of a carbon waste-based strain sensor, the strain values, when plotted against resistance, exhibit a nonlinear S-shaped curve. Thus, a four-parameter sigmoid function (**Equation 3** in main content) was chosen as the white box model due to its inherent S-shaped curve characteristics and its simple mathematical expression. Next, 10 cycles (around 15,000 data points) of data, with strain values serves as x and resistance as y, was used for training to fit the four learnable parameters. 'Dogbox' optimization algorithm from SciPy library (further discussion in **Supplementary Note 4**) was employed to optimize the learnable parameters that fits the training data well.

The method mentioned here were applied in three different sensors with carbon black concentrations of 2%, 4%, and 6%, corresponding to Batch 1st, Batch 2nd and Batch 3rd (see

Supplementary Note 1). For each sensor, data in 10 cycles were taken as training data, and then 30 cycles of data (see **Supplementary Fig. 7**) were used for testing the optimized function.

Supplementary Note 4. Dogbox approach for nonlinear curves

The Dogbox algorithm falls under the category of trust-region algorithms. Trust-region algorithms are iterative methods for minimizing a function over a region that is expanded or contracted at each iteration based on the optimization progress. This particular algorithm operates within a defined 'trust-region', essentially a search space wherein the algorithm anticipates the minimum of the objective function to be located.

In a more detailed perspective, the Dogbox algorithm iteratively adjusts the parameters within a bounded region, striving to minimize the Residual Sum of Squared (RSS) error between the predicted and actual outcomes, as shown in **Supplementary Fig. 8**.

Where RSS is a measure used to quantify the discrepancy between the observed data and the values predicted by the model. Given a dataset with n observations, where y_i represents the actual value of i^{th}observation and $\hat{y}_i$ denotes the predicted value by model, it is computed as **Equation S1**.

$$RSS = \sum_{i=1}^{n} (y_i - \hat{y}_i)^2 \tag{S1}$$

This will adjust the model with each iteration to better align with the empirical data points. During each iteration, the algorithm performs a step that either approaches the boundary of the trust region or stays within the trust region, depending upon the problem's landscape. This is accomplished by solving a quadratic approximation of the objective function within a trust region (i.e., a region around the current estimate of the solution). This approach ensures a steady optimization, preventing abrupt jumps in parameter space that might result in less-than-ideal solutions.

Supplementary Note 5. Details of models used in nonlinearity issue.

In the process of addressing the nonlinearity issue, three computational approaches are explored: Sigmoid function curve fitting (white-box), deep neural network (DNN) (black-box), and support vector regression (SVR) (black-box). Only the details of DNN, SVR models are presented in this note, as the detailed discussion of the white box optimization is in main context and **Supplementary Figs. 9-10** and **Supplementary Table 5**. Detailed implementations of these models in Python are described in **Supplementary Note 18**.

Deep Neural Network

Deep Neural Network (DNN) is a type of artificial neural network that consists of multiple layers of neurons, typically organized into an input layer, several hidden layers, and an output layer. These hidden layers enabled the network to learn hierarchical representations of the data, with each layer capturing increasingly abstract features. The primary advantage of DNNs was their ability to model non-linear relationships in the data, allowing them to perform exceptionally well in tasks such as classification, regression, and pattern recognition.

The hidden layers in DNN model used activation functions, like ReLU (Rectified Linear Unit), to introduce non-linearity, enabling the network to learn complex mappings between input and output. In this study's setup, the DNN model was designed to perform regression tasks using input features to predict resistance values. The model followed a common architecture of having fully connected layers (also known as dense layers) between the input and output layers. Each hidden layer consisted of a number of neurons that interacted through weighted connections, and the network was trained using backpropagation to minimize the error between predicted and actual values.

Support Vector Regression

SVR is a subtype of support vector machine (SVM) devised specifically for regression tasks. Unlike traditional classification tasks, the goal of SVR is to find a regression function that approximates continuous target values, aiming to minimize prediction errors while ensuring the smoothness of the regression function. SVR introduces a tolerance range (epsilon), allowing a

certain margin of error. Only when the difference between the predicted value and the true value exceeds this range is the error penalized. SVR uses kernel methods to map data to higher-dimensional spaces to handle nonlinear relationships. The optimization objective of SVR is to find the best regression function by minimizing the loss function and regularization term, thereby balancing error minimization and model complexity control. The advantage of SVR lies in its ability to effectively handle noisy data and provide strong modeling capabilities when facing complex nonlinear regression problems.

While both SVR and DNN can model non-linear relationships, they do not provide an easily interpretable transformation of non-linear curves to linear ones. The black-box nature of these models is not able to do a straightforward transformation of the underlying relationships they capture. This highlights the advantage of the implemented white-box model optimization strategy, which offers a simpler and more efficient solution to address the issue s of non-linearity.

Supplementary Note 6. Details of data driven black-box models for sequential data.

Four data-driven models, Long Short-Term Memory (LSTM), Gated Recurrent Unit (GRU), 1-Dimensional Convolutional Neural Network (1DCNN), and Transformer were established to address the hysteresis issue. Each of these models offers unique capabilities in processing sequential data, thereby standing as potential candidates for solving the hysteresis problem.

Long Short-Term Memory

The LSTM architecture used in this study can be seen in **Supplementary Fig. 12**. As a type of recurrent neural network (RNN), LSTM function by identifying patterns in sequential data. LSTM integrate memory units and gates (i.e., input, forgetting, and output gates) that allow the network to learn and remember long sequences, avoiding the gradient vanishing problem. In the context of the issues in strain sensors, LSTM can learn and remember the variations in the resistance-strain curves over extended sequences, enabling it to predict current strain values based on a series of resistance values.

Gated Recurrent Unit

The architecture of GRU used in this study is shown in **Supplementary Fig. 13**. GRU is another type of RNN, but simplifies the LSTM architecture by merging the forgetting gate and the input gate into a single "update gate". Thus, GRU has fewer parameters than LSTM, which makes it faster to train. In this study, GRU works similarly as LSTM to learn the patterns over the sequences and predict the current strain value.

1D-Convolutional Neural Network

The architecture of 1DCNN used in this study is shown in **Supplementary Fig. 14**. Standard CNNs focus on 2D or 3D image data for image processing, whereas 1D convolutional neural networks (1DCNN) are tailored for sequences. 1DCNN use a one-dimensional convolutional filter that slides over the sequence to recognize local patterns or features in sequences to predict the current strain value.

Transformer

Transformer is primarily known for its application in Natural Language Processing tasks, especially when it is used to handle long sequences of data. This makes it a suitable architecture for the tasks in both hysteresis and durability. As shown in **Supplementary Fig. 15**, the primary mechanism that empowers the Transformer is the self-attention mechanism. For a given input sequence, the self-attention mechanism computes a weighted sum of all the elements in the sequence for each element in the sequence. These weights are determined by the compatibility of the elements. The higher the compatibility between two elements, the more one element will attend to the other. The self-attention mechanism is capable of handling long sequences as it considers relationships between all pairs of elements in a sequence. This global view ensures that even distant elements can influence the output, allowing the Transformer to capture long-range dependencies, which is often essential for understanding the context in sequences. In this research, most tasks involve sequence-to-value predictions. Instead of the decoder from the original Transformer, a

prediction head consisting of successive feed-forward layers is utilized to produce desired outputs like the strain values.

Supplementary Note 7. Training of black-box sequential models for hysteresis issue.

To address the two distinct resistance-strain curves that occur during the stretching and recovery phases of strain sensors (the hysteresis issue), it is essential to utilize both past and current resistance values to predict the current strain value. In this study, four data-driven models were implemented: LSTM, GRU, 1DCNN, and Transformer. The training process used 10 cycles of data from HysteresisSensor 1, which consisted of a total of 183,200 data points, specifically addressing the hysteresis issue. The data was split in a 9:1 ratio, with 90% used for training and 10% for validation. The models were then tested on data from HysteresisSensor 1 over 5 cycles. The performance of the models is shown in **Supplementary Fig. 17a-c** (LSTM, 1DCNN, GRU) and **Figure 4e** (Transformer).

As discussed in the main content, the models need to intake a sequence of resistance information. By tuning the sequence length as a hyperparameter on the validation data, the optimal sequence lengths for the models are determined. The optimal sequence lengths are 35 for both the LSTM and GRU, 30 for the 1DCNN, and a much longer sequence length of 200 for the Transformer model. Further details on the other hyperparameters such as batch size, hidden size and learning rate are stated in **Supplementary Note 19**.

Supplementary Note 8. Mechanism of SHapley Additive exPlanations (SHAP).

SHAP (SHapley Additive exPlanations) is a method for explaining machine learning model predictions. SHAP uses a concept from cooperative game theory called the Shapley value. In a cooperative game, as shown in **Supplementary Fig. 16a**, players collaborate to achieve a certain reward. The Shapley value helps distribute this reward among the players based on their individual contributions. Consider players A, B and C who join in a certain sequence to play a game and earn a reward. When all of them have joined the game, based on their different skills points in a specific game (e.g., 8, 3 and 3 for players A, B, and C.), they can achieve a 100 reward. When they join in

sequence, A, B, C their marginal contributions to the reward are 35, 35 and 30, respectively. However, if the sequence changes, the marginal contributions might change too. When the sequence is changed from player A, B, C to player A, C, B, if players B and C have a similar skill set, the rewards are still 35, 70, and 100 after player A, C, B joins the game, respectively. Then the players' respective marginal reward is changed (player A, B, C = 35, 30, 35). To accurately determine each player's contribution, one needs to calculate the marginal reward for every possible sequence of players joining the game. The SHAP value for a player is calculated as the average of their marginal across all possible sequences, as shown in **Equation S2**. Features in a machine learning model can be thought of as players in a game.

$$\Phi_i = \frac{M_i}{N} \tag{S2}$$

The SHAP value of a player is represented as Φ_i, where i indicates the specific player. It is calculated as the ratio of the marginal reward sum of this specific player across all possible sequences (M_i) to the total number of possible sequences (N).

Back to the context of SHAP in sensor issues, as shown in **Supplementary Fig. 16b**, consider a sequence of resistance values from time $t-n$ to t as input. This sequence can be likened to players in a cooperative game. Just like in a cooperative game, where players join forces to maximize a reward, here, resistance values collaborate to yield the accurate strain prediction. The SHAP value can then be used to measure the importance of each resistance value within this sequence in predicting the strain. For each resistance value, SHAP computes its marginal contribution by assessing the difference in the predicted strain when the resistance value is present versus when it's absent from the sequence. Given the motion's directional information embedded within the sequence, the SHAP value for a resistance can be computed as **Equation S3**. SHAP offers a transparent mechanism to show how these data-driven models interpret resistance sequences, providing insights on choosing the right model to solve the sensor issues.

$$\Phi_r = \frac{M_r}{N_w} \tag{S3}$$

The SHAP value of a resistance value is denoted as Φ_r, where r represents the resistance value under consideration. It is calculated as the ratio of the total marginal rewards contributed by

this specific resistance value across all possible sequences within the window (M_r) to the total number of possible sequences within the window (N_w).

Supplementary Note 9. Discussion of SHAP values for each model in hysteresis issue.

As described in **Supplementary Note 8**, SHAP values are used to evaluate the contribution of various feature arrangements iteratively. The higher the SHAP value, the greater the correlation between the feature and the model's output. A non-zero SHAP value indicates that the feature has played a role in the model's prediction.

Figure 4f and **Supplementary Figs. 18a-b** present the SHAP values of four models under the hysteresis issue, providing insights into how data from different positions influence strain value predictions across models. The resistance data from each time period was sequentially used as features, and the computed SHAP values were normalized to facilitate a more intuitive analysis of the data's contribution to the final prediction. A non-zero SHAP value indicates that the corresponding data point has a certain degree of influence on the prediction. As shown in **Figure 4f (i)** and **Supplementary Figs. 18a** and **18b**, the Transformer, LSTM, and GRU models exhibit a global perspective when processing input sequences, with most data points contributing to the predictions. In contrast, the 1DCNN model (**Figure 4f (ii)**) shows a clear difference, as it excels in capturing local features but is relatively less effective at capturing global information. This characteristic explains the performance differences among the models in strain value prediction.

Supplementary Note 10. Training of black-box sequential models for cycling attenuation issue.

In this cycling attenuation issue, we employed data-driven models using a total of 20,000 cycles of data collected from sensors for training and testing the models. To effectively train and validate the models, the data set was divided in a 2:8 ratio into training and testing sets. Specifically, the resistance data from the first 4,000 cycles were used as the training set to train the models. Subsequently, the models were tested on the remaining 16,000 cycles to assess their predictive

capabilities over a longer time series. This approach aims to enhance the accuracy and reliability of the models in predicting material durability.

Supplementary Note 11. Mechanism of casual convolutional operations and rotary embedding in D_Former.

Casual convolutional operations

Time series data is inherently characterized by different sub-sequence local patterns, which can exhibit substantial variations over time. The canonical Transformer model, renowned for its prowess in processing sequential data, may not optimally exploit these local nuances. This limitation stems from the Transformer's inherent design, which computes similarities across an entire sequence without prioritizing the proximate data points, a mechanism termed as self-attention (**Supplementary Fig. 19a**).

To rectify this shortcoming, convolutional self-attention mechanism is used in the proposed model, D_Former, to augment the Transformer's capacity to discern and leverage local patterns within the data. By employing a 1-D variable convolution kernel, the input data undergoes a transformation. This convolutional kernel, defined by its kernel size 'k', ensures that a cluster of adjacent data points are considered, thereby enhancing the model's sensitivity to local contexts during similarity computations (**Supplementary Fig. 19b**). This method can improve the accuracy of predictions by ensuring that the model considers local patterns in the data.

Rotary embedding

The sequential nature of time-series data accentuates the significance of positional information, given the crucial role of temporal dependencies. The Rotary Position Embedding (RoPE) is designed to capture the relative position (its position in relation to other data points). This is achieved by applying a rotation operation to the embeddings of the tokens, where tokens represent individual elements or data points in the sequence.

The integration of relative position embedding is executed: the embedding vector is rotated by an angle that is a multiple of its positional index. This rotation encapsulates the essence of RoPE

(**Supplementary Fig. 20**). By rotating data points on a two-dimensional plane, the degree of rotation is dependent on the data point's position within the sequence, thereby embedding the positional information. Mathematically, RoPE employs a multiplication-based approach, integrating relative positional information through the application of a rotation matrix, as represented in **Equation S4**, which is a 2-dimensional scenario:

$$f_{\{q,k\}}(x_m, m) = \begin{pmatrix} \cos m\theta & -\sin m\theta \\ \sin m\theta & \cos m\theta \end{pmatrix} \begin{pmatrix} W_{\{q,k\}}^{(11)} & W_{\{q,k\}}^{(12)} \\ W_{\{q,k\}}^{(21)} & W_{\{q,k\}}^{(22)} \end{pmatrix} \begin{pmatrix} x_m^{(1)} \\ x_m^{(2)} \end{pmatrix} \quad \text{(S4)}$$

By rotating the embeddings based on their positional indices, the transformed Q, K, and V matrices now contain not just raw data information but also the relative positional data. In the context of our sensor material analysis, it means that when the model tries to understand the decay behavior of a sensor material at a time, it isn't just looking at the resistance value of that time. It also considers how this value stands in relation to previous or subsequent time, providing a richer understanding of the data.

Supplementary Note 12. Training of D_Former model for cycling attenuation issue.

Acknowledging the limitations in existing models - LSTM, GRU, 1DCNN, Transformer (the designs of these models are similar to **Supplementary Note 6**, just that different training data and features compared to models from hysteresis issue), the D_Former model is introduced. The casual convolutional operations and rotary positional embedding (**Supplementary Note 11**) is added into the traditional Transformer architecture to create the D_Former. By tuning the hyperparameter of ‘d_model’ and ‘hidden_layer’ in D_Former, the training approach mentioned in main content is explored.

After making predictions for a singular sensor, D_Former model has better accuracy and learning efficiency. The performances for all models are shown in **Supplementary Figs. 21-26**.

Supplementary Note 13. Dimensionality reduction and visualization of sensor data using uniform manifold approximation and projection(UMAP).

Uniform Manifold Approximation and Projection (UMAP) is a nonlinear dimensionality reduction technique designed to visualize high-dimensional data while preserving both local and global structures of the dataset. It works by constructing a high-dimensional graph of data points and optimizing a low-dimensional graph to be as structurally similar as possible.

In the batch inconsistency task, this study applies UMAP technology for dimensionality reduction and visualization of sensor data. Processing the data from 20,000 cycles across four sensors requires significant computational resources and may lead to convergence issues, preventing the effective capture of the underlying structure of the data. Therefore, this study only selects the first 1800 cycles of data from each sensor for dimensionality reduction.

The process begins by loading the raw data files from the four sensors, with each file containing 18 columns of resistance data (each column representing 100 cycles of data). The 'load_batch_data' function is used to read the CSV files in batches and store the resistance data in NumPy arrays. Subsequently, the data from all four sensors is merged into a single large dataset for further processing.

The data is then passed to UMAP for dimensionality reduction. Initially, a nearest-neighbor graph is constructed to calculate the k nearest neighbors for each data point, capturing the similarities between the data points. UMAP models the local geometric structure, preserving local relationships among data points, and uses similarity weights to represent the relative positions of data points in high-dimensional space. The core operation of UMAP is optimizing the cross-entropy loss function to minimize the discrepancy in the similarity distributions of data points between the high-dimensional and low-dimensional spaces, ensuring that the structure of the data is preserved as much as possible after dimensionality reduction.

This process combines both local and global structural information, and uses graph-based optimization methods, adjusting the positions of data points in the low-dimensional space through stochastic gradient descent (SGD). During this process, the parameters 'n_neighbors' and 'min_dist' are set to control the neighborhood size and the distance between data points, helping to ensure that the local structure of the original data is preserved after reduction.

After dimensionality reduction, the data is mapped to a two-dimensional space, compressing the original high-dimensional data into two dimensions. The 2D coordinates of each batch are saved into separate CSV files for subsequent analysis and visualization.

Supplementary Note 14. Training of models for batch inconsistency issue.

Five models (LSTM, GRU, 1DCNN, Transformer and D_Former) were trained to solve this Out-of-distribution (OOD) task. As shown in **Figure 6a**, all the cycles from sensor Batch 1, Batch 2, Batch 3, Batch 4 are combined and used for training. Apart from the two features (resistance and cycles) used in the previous cycling attenuation task, another input feature – relative resistance is added. Relative resistance, denoted as is calculated using the **Equation S5**, where is the relative resistance at strain, is the initial resistance at the very beginning in cycle 0, is the resistance under strain and represents the applied strain. The calculated along with resistance values and cycles are fed into the models as features for training.

$$S_\varepsilon = \frac{R_\varepsilon - R_0}{R_0} \tag{S5}$$

All five models were evaluated on the testing data (**Supplementary Fig. 28**), which was constructed from data collected by Batch 5 over 20,000 cycles (i.e., the test dataset for Batch 5 contains 20,000 cycles). As shown in **Supplementary Figs. 29-34** and the **Fig. 6** in main content, while LSTM, GRU, 1DCNN, and Transformer models struggle to adapt effectively across different distributions, the D_Former model stood out with its high prediction accuracy. It adeptly navigated the shifts between training and testing distributions, delivering accurate predictions despite batch inconsistencies.

Supplementary Note 15. Carbon waste-based strain sensor on flexible robot arm.

This study employs a prototype of a tendon-driven flexible robotic arm, which consists of eight joints and a pair of grippers. Each joint is driven by two tendons powered by servomotors. Due to its high degree of freedom, hyper-redundant structure, and strong nonlinear and coupled characteristics, the flexible robotic arm poses significant issues for traditional perception and

positioning methods, making precise end-effector pose estimation difficult. The goal of this study is to achieve precise perception and localization of the robotic arm's end position through sensor data acquisition.

Strain sensors were installed on both sides of the movement direction of each joint of the flexible robotic arm, with two sensors per joint to capture motion. These sensors provide real-time monitoring of each joint's movement state and strain variation, particularly as the strain signals from the two sensors exhibit opposite trends when the joint moves in different directions. This differentiated signal provides the perception system with rich motion information. However, it was observed during the experiments that there are inconsistencies among sensors from different manufacturing batches, which affects the stability and reliability of the sensor data, thereby increasing the issue of learning across different batches of sensor data.

To address these issues, this study utilized sensor data from three different batches to train the model (Batch1Sensors; Batch2Sensors; Batch3Sensors). The model was then tested using sensor data from a previously unseen fourth batch (Batch4Sensors). This approach aimed to validate the model's generalization capability when dealing with new batch data, ensuring that the system could adapt to batch differences that might be encountered in real-world applications. The experimental results demonstrate that the proposed method effectively handled batch discrepancies and successfully achieved accurate predictions for the fourth batch of data (**Supporting Movie 1**). The specific prediction results, presented in **Fig. 7d**, highlight the model's stability and adaptability under complex conditions.

Supplementary Note 16. Carbon waste-based strain sensor on soft quadruped robot.

In this study, the manufacturing process of the quadrupedal soft robot is relatively simple, primarily involving silicone casting, wax mold fabrication, and mold assembly. First, Computer-Aided Design (CAD) files are downloaded and used for 3D printing. These files, created using CAD software, contain detailed design data that define the shape and structure of the object to be printed. After 3D printing, the molds are coated with wax to facilitate the demolding process. Then, Smooth-Sil silicone is poured into the molds and allowed to cure. The resulting parts are used to create wax molds, which are ultimately used to produce the wax components needed for the

quadrupedal soft robot. The wax components are vented and then silicone is cast on both sides to form the final robot parts. After curing, demolding, wax removal, and final bonding are performed to obtain a functional, mobile soft robot. The entire process requires silicone, wax, 3D-printed molds, adhesives, and corresponding tools to achieve a pneumatic-controlled walking soft robot.

The quadrupedal soft robot features two hollow internal chambers. When one of these chambers is pressurized, it causes the robot's legs and sides to deform and bend, creating a walking motion. The robot moves by balancing itself on two diagonally opposite legs while moving the other pair forward to achieve locomotion. By attaching four strain sensors from the same batch to the robot's limbs, the motion state of the robot can be effectively perceived during its movement (**Supplementary Fig. 35**).

However, there are certain performance differences among sensors from different manufacturing batches, which present significant issues for motion perception in the soft robot. In experiments involving multiple batches of sensor data, differences in sensor sensitivity, response time, and accuracy can directly affect the precise perception and positioning of the soft robot's movement. Batch inconsistencies lead to data discrepancies, making it difficult for the robot's perception system to generalize well across different sensor batches, thereby impacting the robot's motion control and stability.

To address these issues, this study used sensor data from two batches for model training: Batch1Sensors; Batch2Sensors. The trained model was then used to predict the motion of the soft robot with sensors from a third batch (Batch3Sensors), as shown in **Figure 7g**. The experimental results indicate that, although there are physical differences between sensors from different batches, effective motion prediction for new batch sensors can still be achieved through comprehensive data training and model optimization, thereby demonstrating the model's generalization capability and adaptability in the face of batch inconsistencies (**Supporting Movie 2**).

Supplementary Note 17. Carbon waste-based strain sensor on dexterous robot hand.

A new batch of carbon waste-based strain sensor was prepared and applied to a dexterous robot hand (i.e., Batch 1, Batch 2, Batch 3). These three sensors were attached to three robot fingers. As the arm moved at a steady angular velocity, the change in angle caused a non-uniform shift in

resistance. This scenario is also considered as a batch inconsistency problem, as the data are out-of-distribution. To address this, two different experiments were designed for this application:

1. Multi-batch sensor training: Data from two batch sensors (Batch 2 and Batch 3) were combined for training and tested on the third (Batch 1). This approach leverages more training data to improve model generalization and performance. Results are shown in **Supplementary Figs. 36**.

2. Single-batch sensor training: The D_Former model was trained using data from a single batch (Batch 2) and tested on another batch (Batch 1). This experiment evaluates the model's performance with limited training data and shows that reducing training data only slightly impacts prediction accuracy. Results are shown in **Supplementary Fig. 37**.

Supplementary Note 18. Implementation of sigmoid curve fitting in python for nonlinearity issue.

Three main steps were implemented in Python to fit the sigmoid curve to the half-cycle stretching sensor data. In this study, Batch 1 data was used as the experimental dataset to compare the performance of different models. In the sigmoid fitting method, the data was first retrieved in CSV format from Batch 1 and converted into a NumPy array. The implementation involved the following steps:

1. Sigmoid Function Definition: The sigmoid function was defined and fitted according to **Equation 3** (in the main text).

2. Curve Fitting: Using the ‘curve_fit’ function from ‘scipy.optimize’, the ‘dogbox’ method was selected to ensure optimal fitting of the sigmoid curve to the experimental data.

3. Visualization: A scatter plot was generated to compare the actual data points with the fitted sigmoid curve, providing a clear visualization of the fitting accuracy.

Additionally, details regarding the training methods for DNN and SVR models can be found on the project's GitHub repository. The open-source code for implementing curve fitting in Python is also available on GitHub at https://github.com/jiali1025/Computational-intelligence-for-soft-strain-sensor-sustainability/tree/main/nonlinearity.

Supplementary Note 19. Implementation of black-box sequential models in python for hysteresis issue.

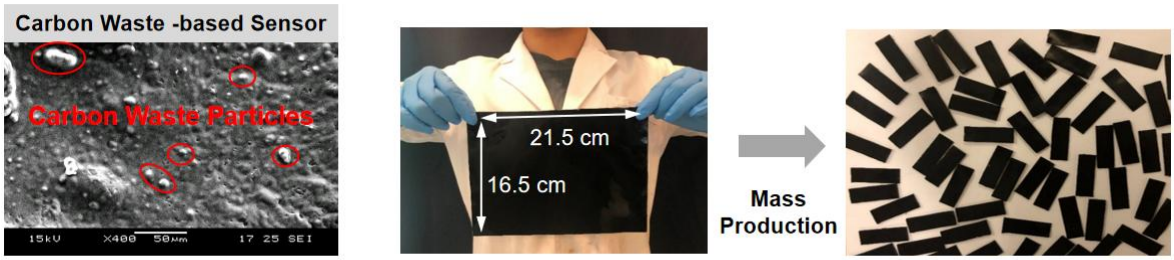
Five main steps are implemented in Python to solve the hysteresis problem. Here only GRU model is explained as an example. The first step was to introduce the Strain_Dataset class. This class is designed to load and preprocess data from a CSV file, converting it into sequences that are compatible with the GRU model. The second step was to implement the architecture of our prediction model using the RNN definition. This architecture incorporates a GRU backbone and two fully connected layers. The third step involved training the model through the train() function. It periodically saves iterations of the model when a better performance is noted on the validation set. In the fourth step, two functions, namely dev() and test(), were utilized. While dev() evaluates the model against a validation dataset, test() evaluates its performance on a separate test dataset. Lastly, there were setup undertaken to define hyperparameters and to create a directory saving iterations of the model. Only GRU model is explained here, the other three models implemented in Python can be found in GitHub (https://github.com/jiali1025/Computational-intelligence-for-soft-strain-sensor-sustainability/tree/main/hysteresis).

Supplementary Note 20. Implementation of models in python for cycling attenuation issue.

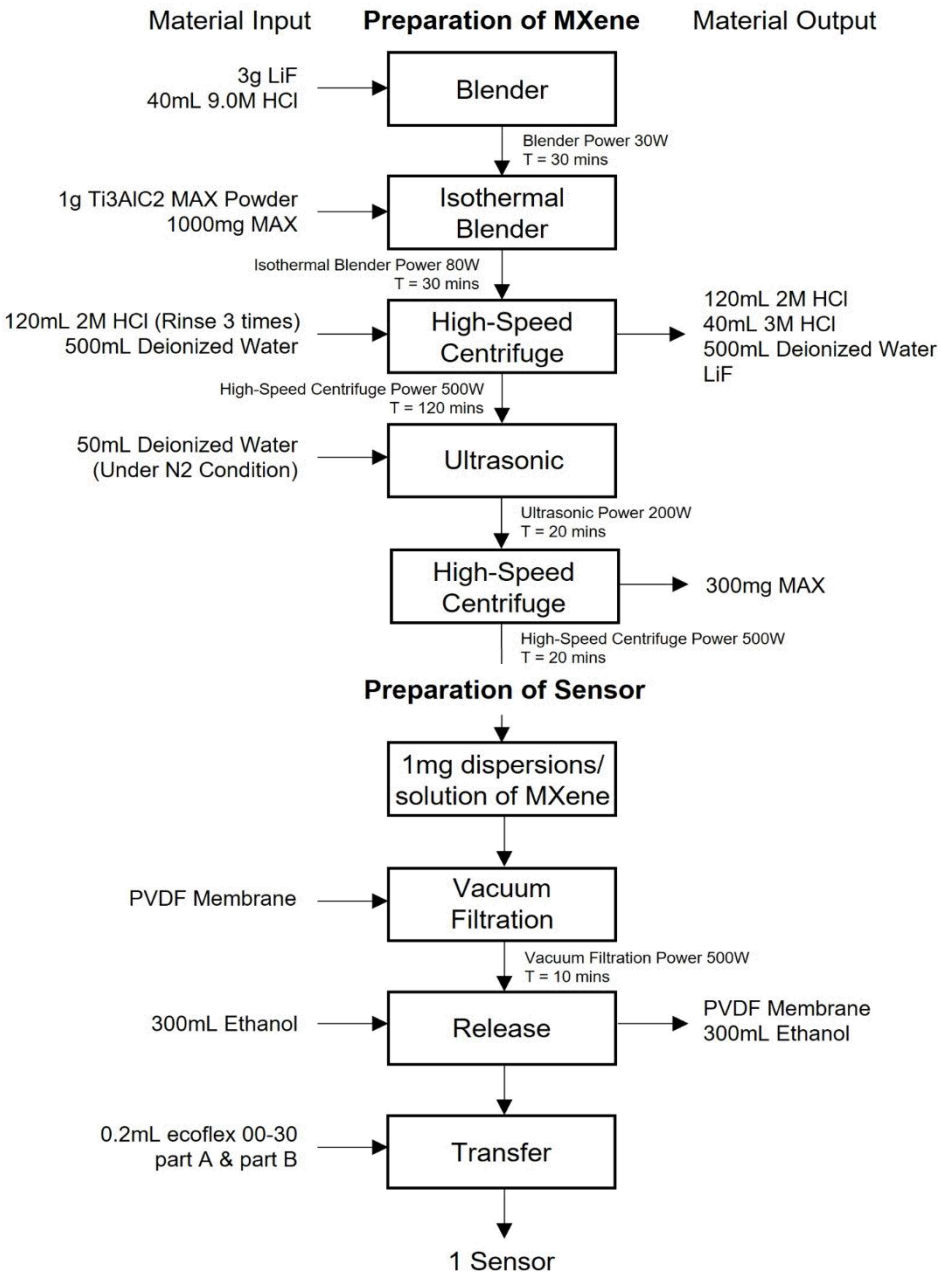
Five different ML models were implemented in Python to address the cycling attenuation problem. The implementation for the four models (LSTM, GRU, 1DCNN, Transformer) and training were similar to **Supplementary Note 19**. While the D_Former is implemented differently. The architecture of D_Former (**Supplementary Note 11**) is done in the code with additional casual convolution layer (causal_convolution_layer.context_embedding) and RoFormer encoder (modeling_roformer.RoFormerEncoder(config)). Two functions, dev() for validation set evaluation, and test() for test dataset evaluation, are employed to gauge the performance of the trained model. Defining model hyperparameters and creating a directory structure that holds model iterations were necessary steps. (https://github.com/jiali1025/Computational-intelligence-for-soft-strain-sensor-sustainability/tree/main/cycling%20attenuation).

Supplementary Note 21. Implementation of models in python for batch inconsistency issue.

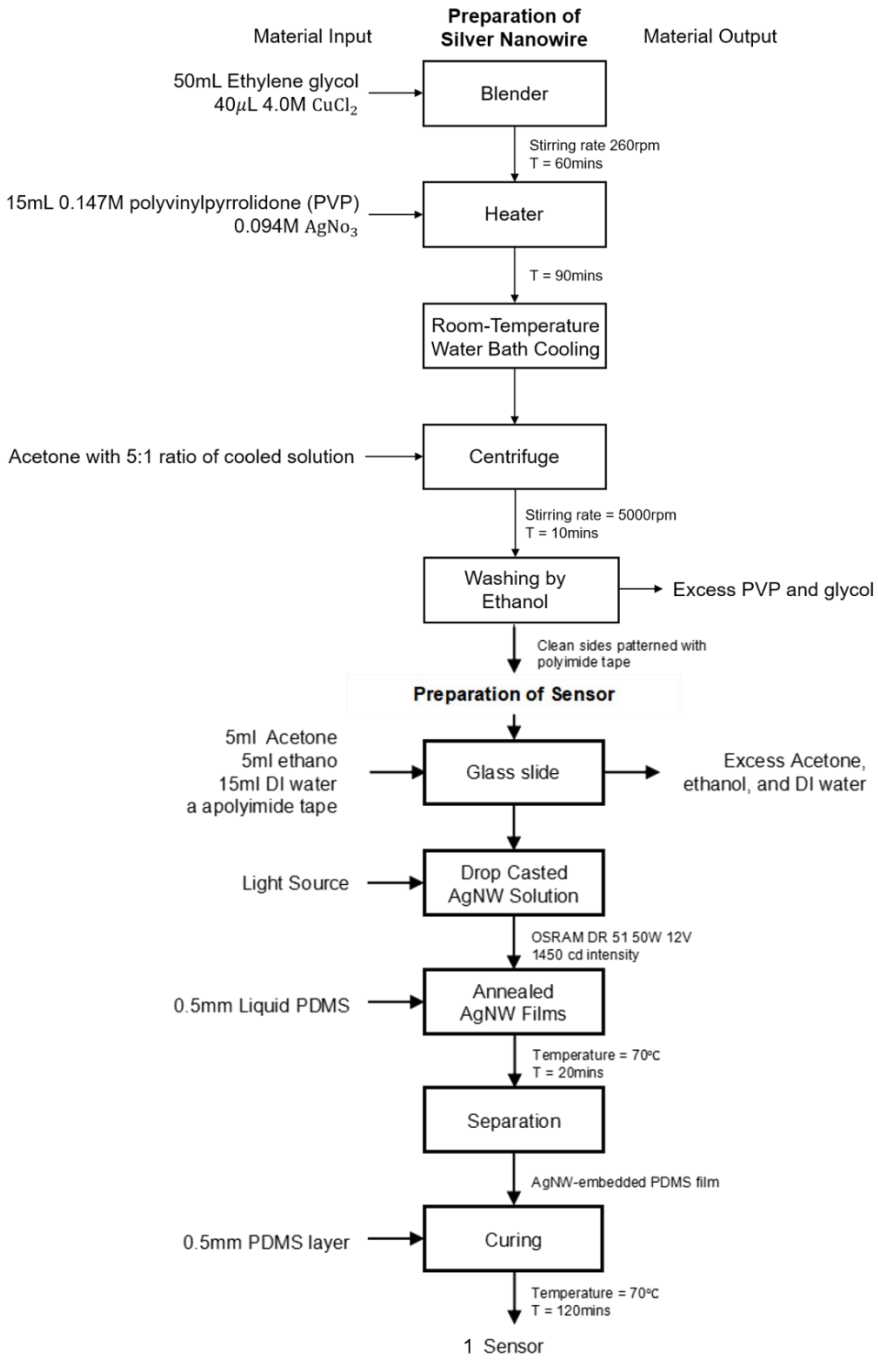
Five models are implemented in Python to solve the batch inconsistency issue. The architectures and implementations of these models are similar to those described in **Supplementary Note 20**, with additional of utilizing multiple batch sensor data to improve generalization across different batches. Furthermore, a 30-fold data sparsification technique was applied in this study to reduce data density while maintaining essential features for robust model training and evaluation. (https://github.com/jiali1025/Computational-intelligence-for-soft-strain-sensor-sustainability/tree/main/batch%20inconsistency).



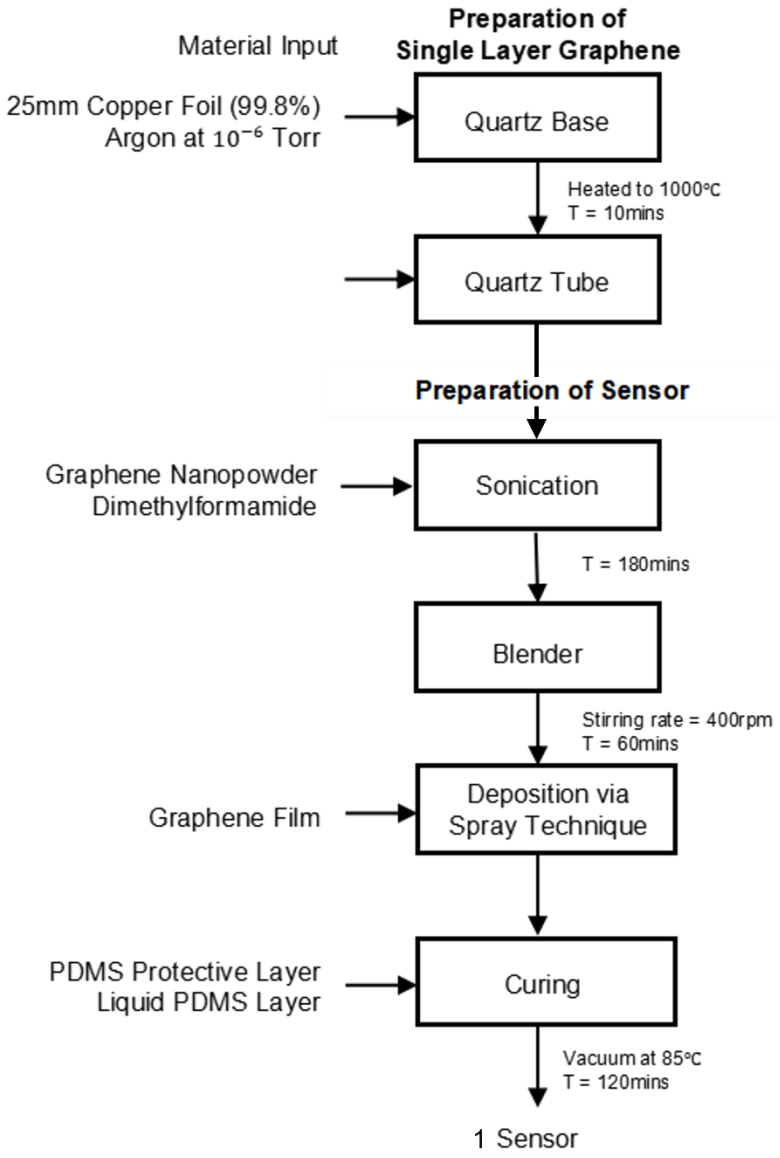
Supplementary Fig. 1. SEM image and demonstration of scalable production for carbon waste-based strain sensors.



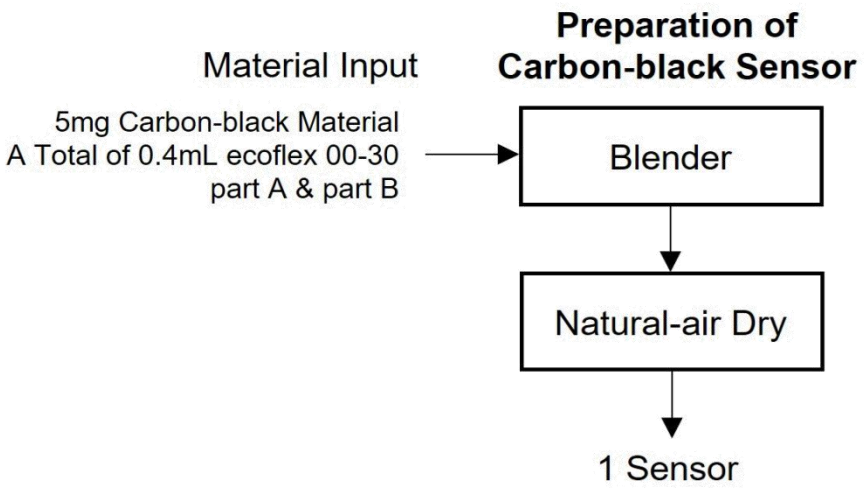
Supplementary Fig. 2. Sensor fabrication steps using MXene materials. Step 1: Preparation of MXene through chemical etching of MAX powder, washing, sonication, and centrifugation to obtain MXene suspension. Step 2: Fabrication of the MXene strain sensor by filtering 1 mg of MXene solution onto a PVDF membrane, followed by ethanol-assisted membrane separation and transfer onto a silica gel elastomer formed from ecoflex 00-30, resulting in the final MXene-based strain sensor[1].



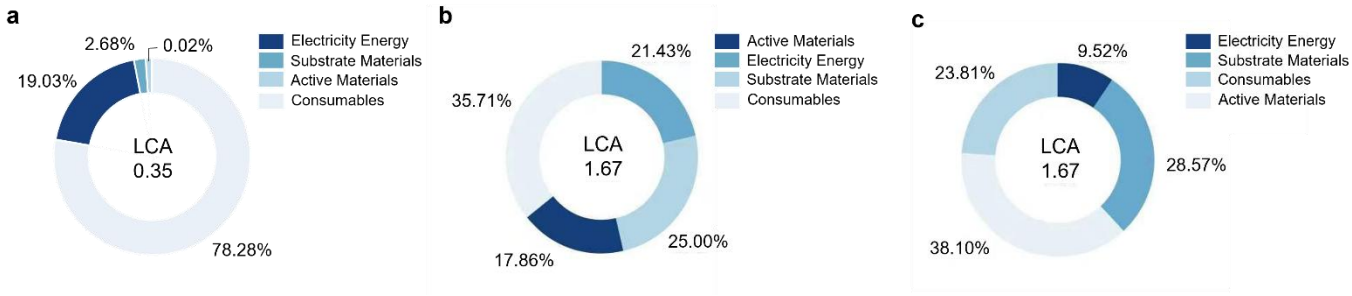
Supplementary Fig. 3. Sensor fabrication steps using silver nanowires materials. Step 1: Synthesis of silver nanowires through heating ethylene glycol, adding 4M, 0.147M PVP, and 0.094M, followed by quenching, acetone washing, and ethanol purification. Step 2: Fabrication of the AgNW sensor by drop-casting AgNW solution onto polyimide-patterned slides, light-assisted drying, and annealing at 200°C, and embedding in a PDMS layer. Copper wires are attached with silver paste, and a second PDMS layer is applied for sealing and curing[2].



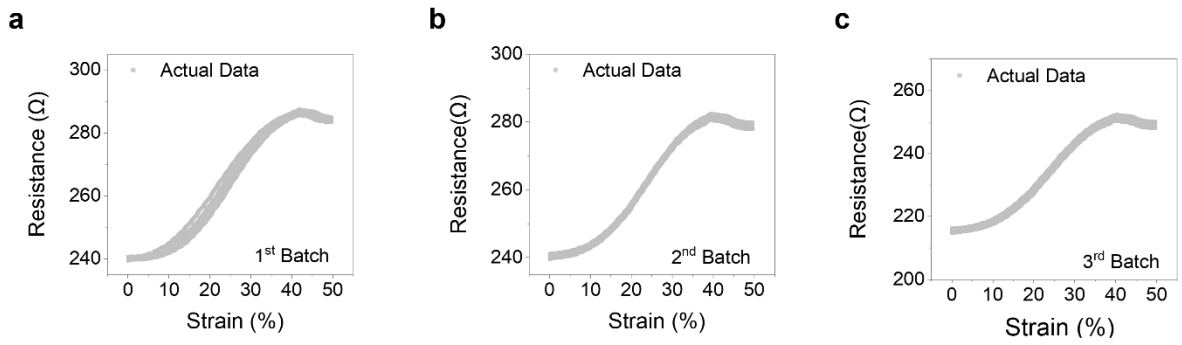
Supplementary Fig. 4. Sensor fabrication steps using graphene materials. Step 1: Single-layer graphene (SLG) is prepared using a chemical vapor deposition (CVD) system by heating a copper foil to 1000°C under vacuum, with methane and hydrogen gases facilitating graphene growth. Step 2: Graphene nano powder is mixed with dimethylformamide, sonicated, and spray-deposited onto an 85°C substrate to form the graphene film. A PDMS protective layer is then added, and the sensor is completed by curing under vacuum at 85°C for 2 hours[3].



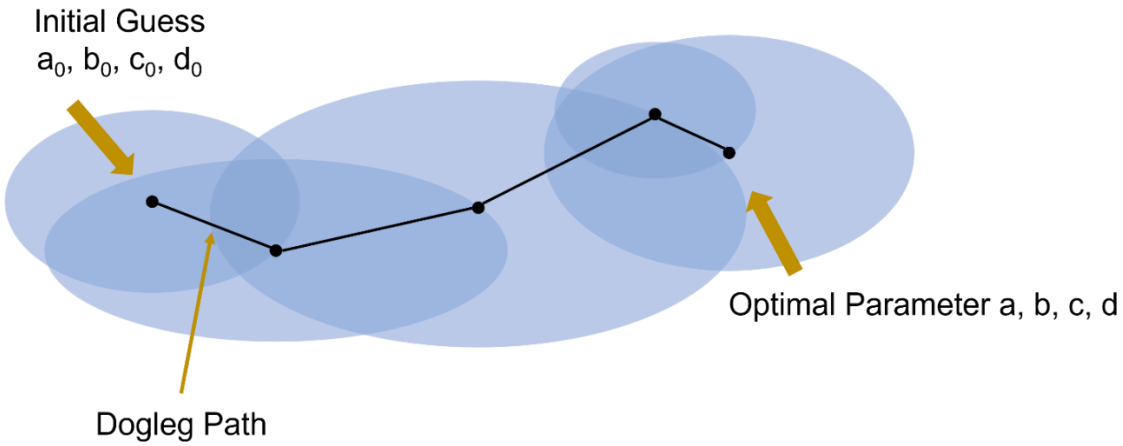
Supplementary Fig. 5. Sensor fabrication steps using carbon waste (CW) materials. The process involves mixing 0.4 mL of ecoflex 00-30 (0.2 mL each of part A and B) with 10 mg of recycled industrial waste carbon black using a blender. The mixture is then left to naturally air dry for 12 hours, resulting in the final carbon waste-based strain sensor.



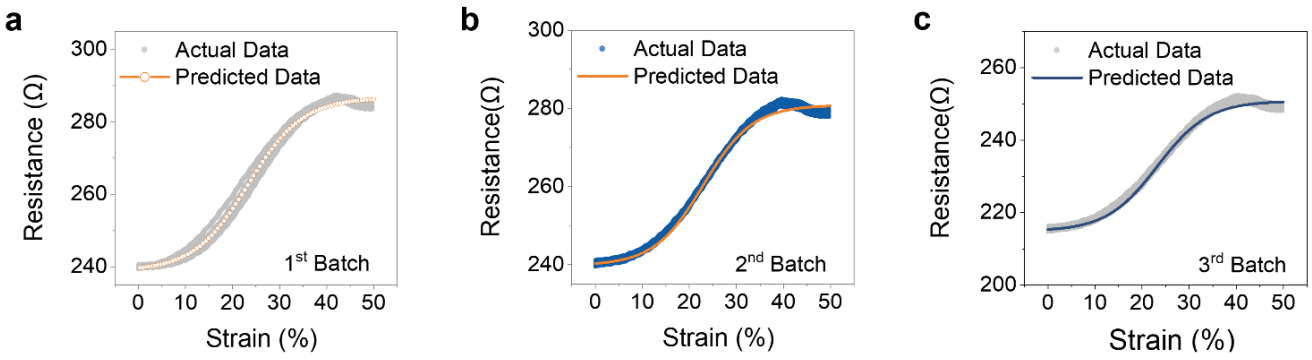
Supplementary Fig. 6. Pie charts showing LCA carbon emissions for various sensor fabrications. (a) MXene-based strain sensor, **(b)** silver nanowires-based strain sensor, and **(c)** graphene-based strain sensor. Detailed calculation processes can be found in **Supplementary Tables 1-4**.



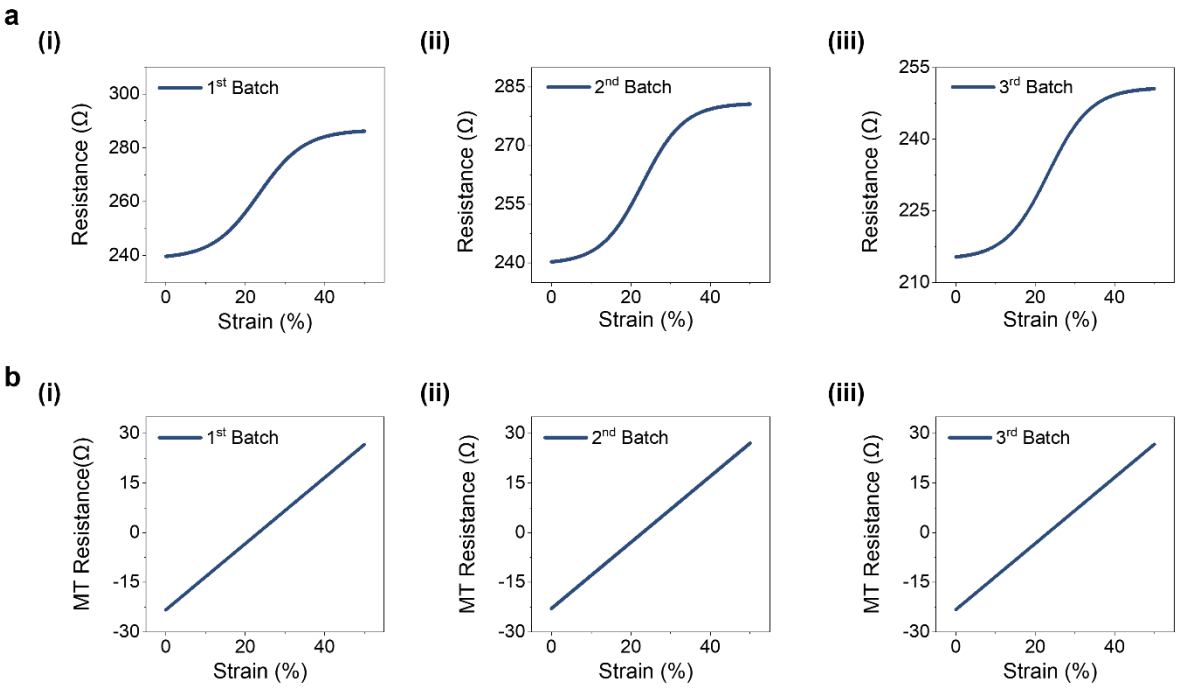
Supplementary Fig. 7. Nonlinear Response Characteristics of CW Sensors with Different CW Loadings. (a) 2 wt.% CW Loading. **(b)** 4 wt.% CW Loading. **(c)** 6 wt.% CW Loading.



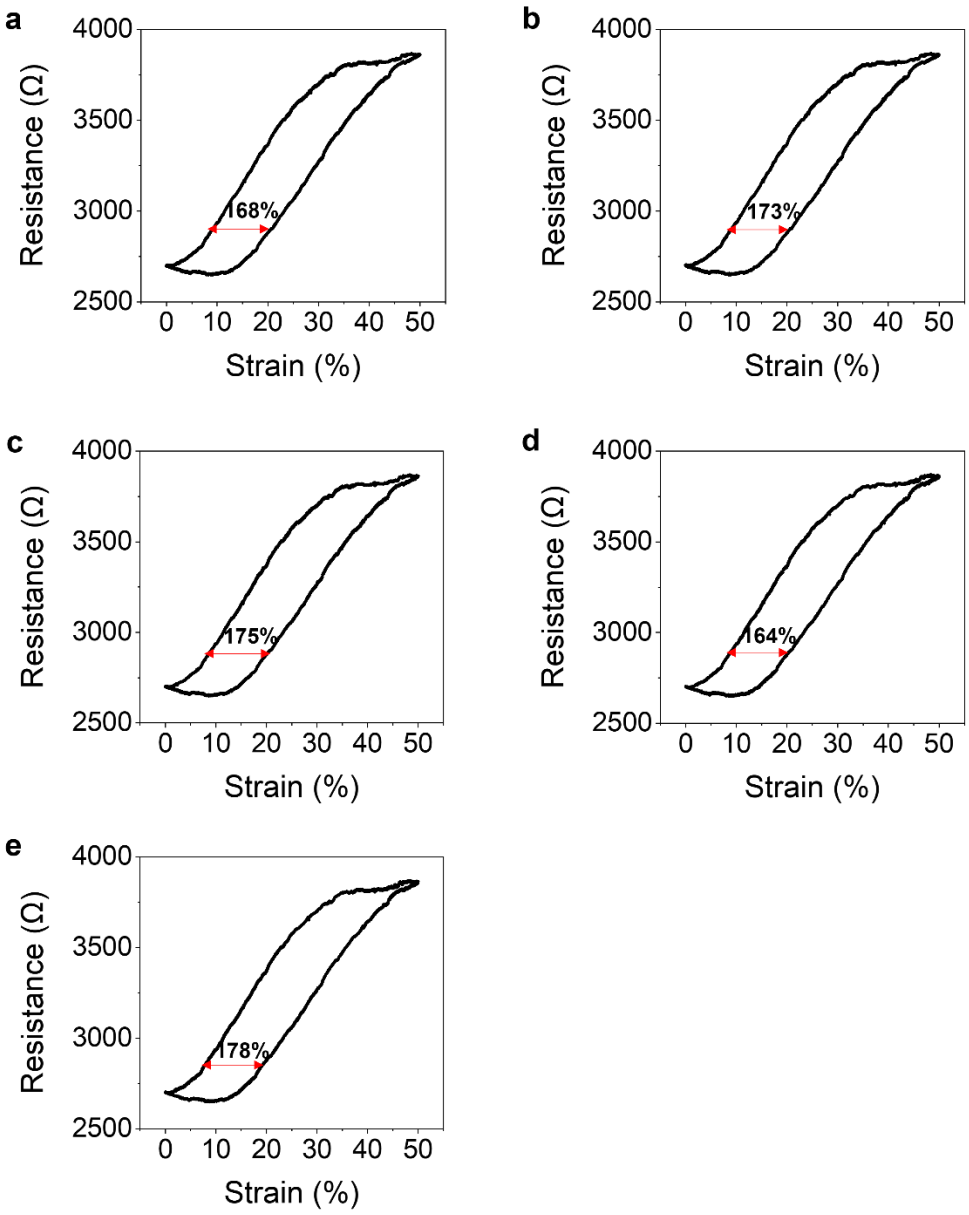
Supplementary Fig. 8. Schematic representation of the 'Dogbox' trust-region algorithm optimizing nonlinear curves. The algorithm controls the step size of each iteration by setting a trust region, ensuring that each iteration finds a valid and reliable solution. The size of the trust region is dynamically adjusted during the iterations to balance the precision of the local search and the convergence speed of the algorithm. A detailed discussion is provided in **Supplementary Note 4**.



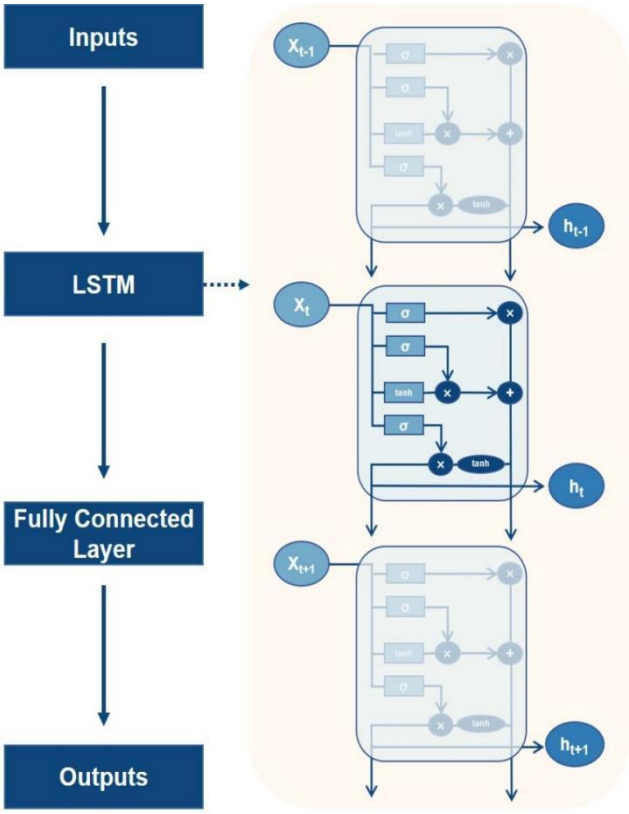
Supplementary Fig. 9. Comparison of sigmoid function curve fitting results for different sensor batches. **(a)** Visualization of predicted and true sensing curves for the 1st batch 30-cycle test set. **(b)** Visualization of predicted and true sensing curves for the 2nd batch 30-cycle test set. **(c)** Visualization of predicted and true sensing curves for the 3rd batch 30-cycle test set.



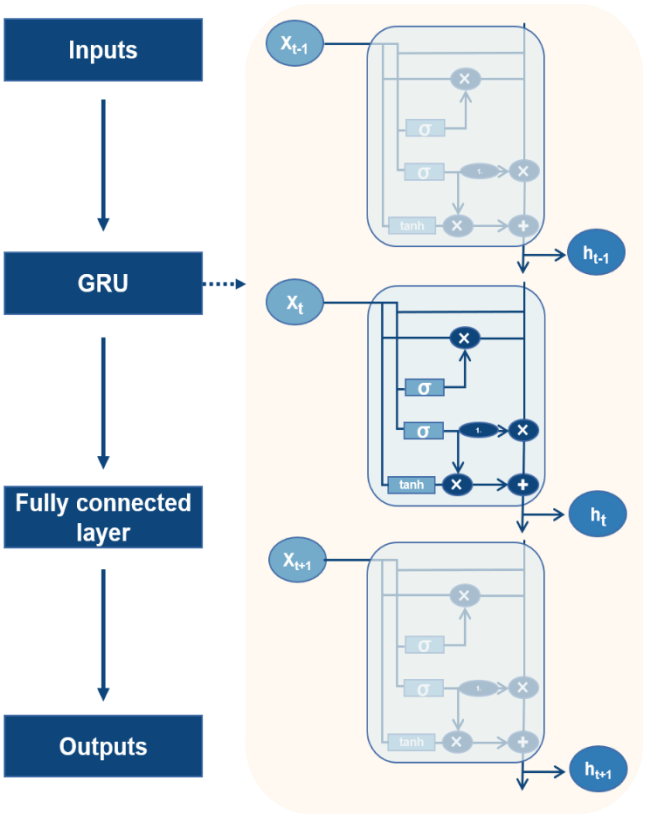
Supplementary Fig. 10. Curve fitting and linear transformation. (a) Fitted curves using the sigmoid function for 2 wt.% (i), 4 wt.% (ii), and 6 wt.% (iii) CW particle loadings. **(b)** Linear transformation of sensor responses using the sigmoid function under 2 wt.% (i), 4 wt.% (ii), and 6 wt.% (iii) CW particle loadings.



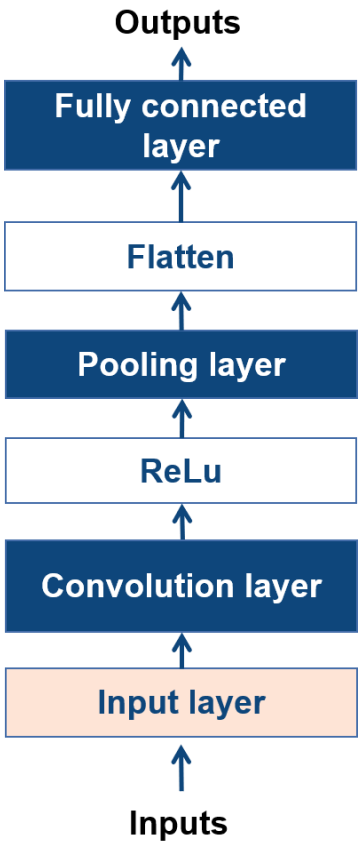
Supplementary Fig. 11. Hysteresis behavior across five cycles for HysteresisSensor 1. (a)-(e) Hysteresis curves of the continuous 5 cycles for the CW sensor, with the relative differences of the resistance-based strain values calculated. Detailed calculation methods are provided in the **Methods.**



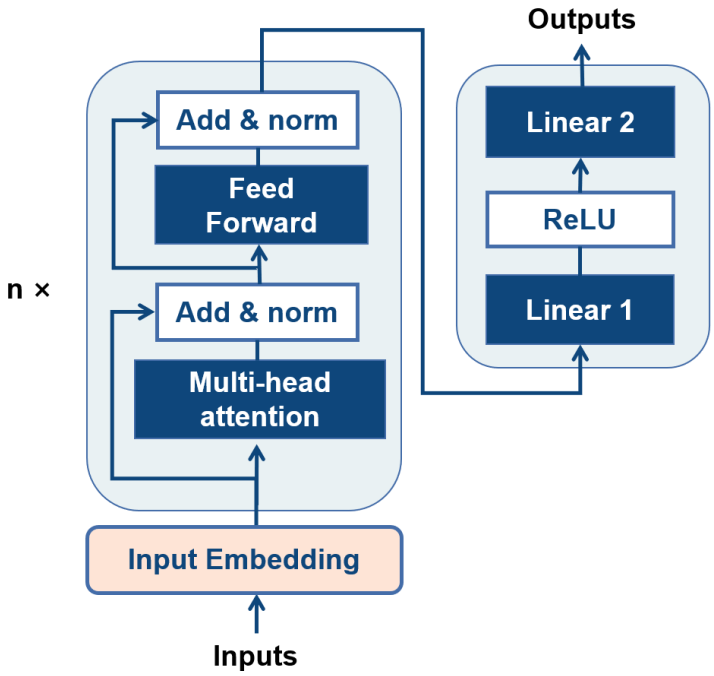
Supplementary Fig. 12. Architecture overviews of LSTM black-box sequential model. The figure illustrates the structure of the LSTM network, which includes memory units and gates (input, forget, and output) to capture sequential dependencies. This architecture allows the model to process and retain information from past data, making it capable of predicting strain values based on a series of resistance measurements in strain sensor applications (see detailed description in **Supplementary Note 6**).



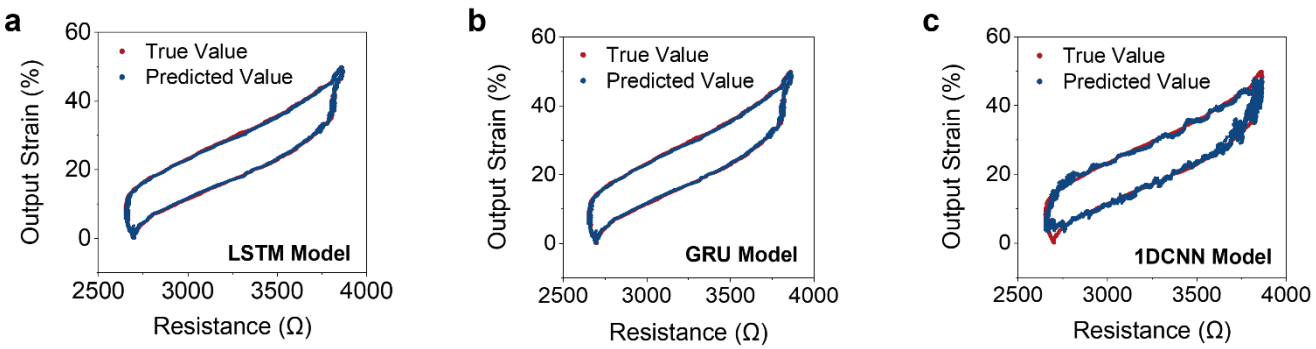
Supplementary Fig. 13. Architecture overviews of GRU black-box sequential model. The figure illustrates the structure of the GRU network, which uses gating mechanisms to control the flow of information through the network. GRUs combine the input and forget gates into a single update gate, enabling them to capture long-range dependencies in sequential data. This architecture is particularly effective in predicting strain values from a series of resistance measurements in strain sensor applications (see detailed description in **Supplementary Note 6**).



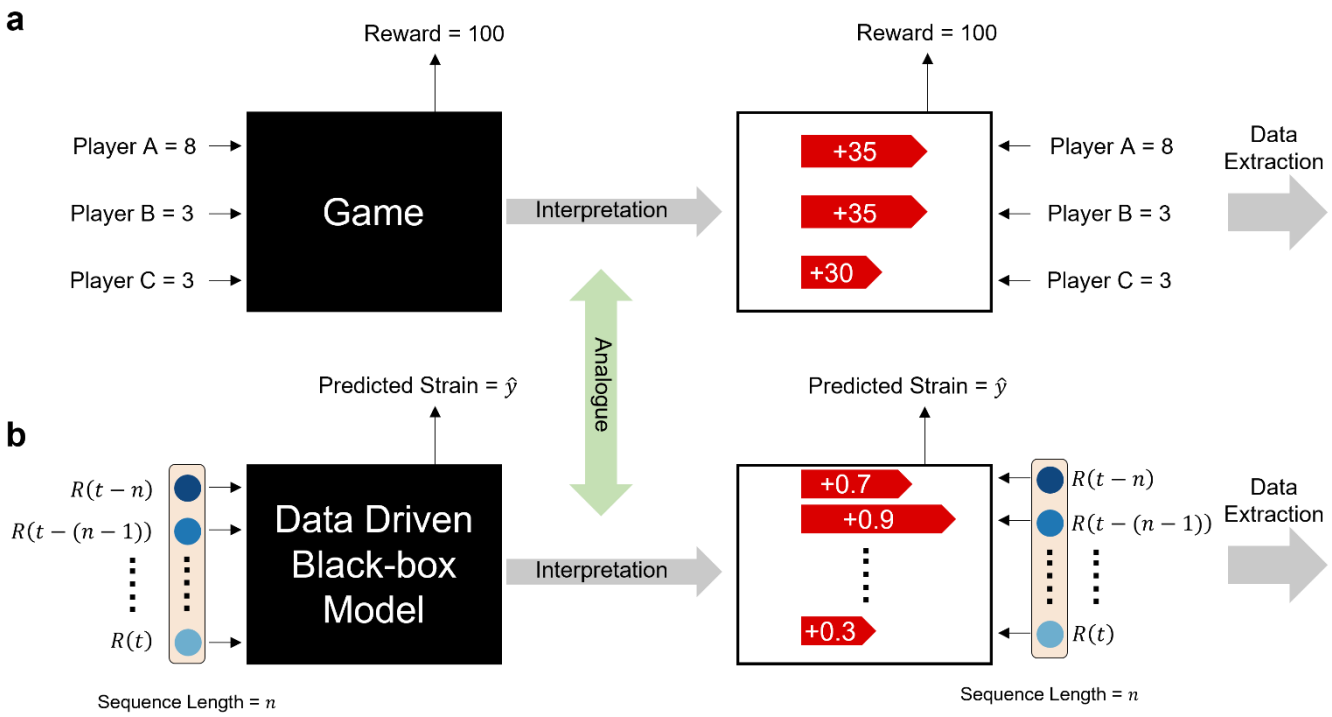
Supplementary Fig. 14. Architecture overviews of 1DCNN black-box sequential model. The figure illustrates the sequential structure of the 1DCNN, starting with the input layer, followed by a convolutional layer, activation function (ReLU), pooling layer, flattening layer, and ending with a fully connected layer. This architecture is designed to extract hierarchical features from sequential resistance data, enabling the model to predict strain values in strain sensor applications (see detailed description in **Supplementary Note 6**).



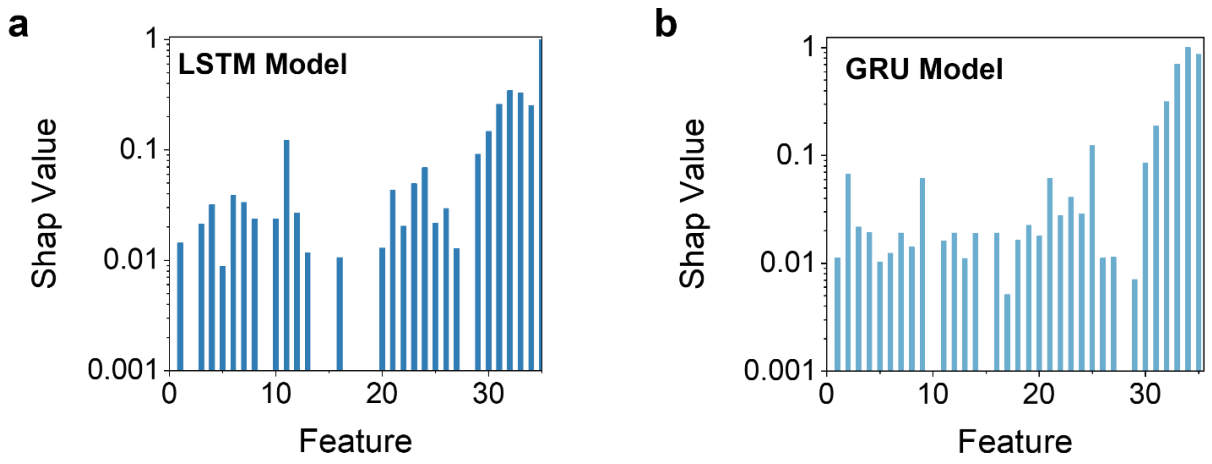
Supplementary Fig. 15. Architecture overviews of Transformer black-box sequential model. The figure illustrates the structure of the Transformer model, including three main components: the input embedding layer, the encoder, and the decoder. This architecture leverages self-attention mechanisms to capture relationships between different elements in sequential data, enabling it to effectively process resistance-strain data and predict strain values in strain sensor applications (see detailed description in **Supplementary Note 6**).



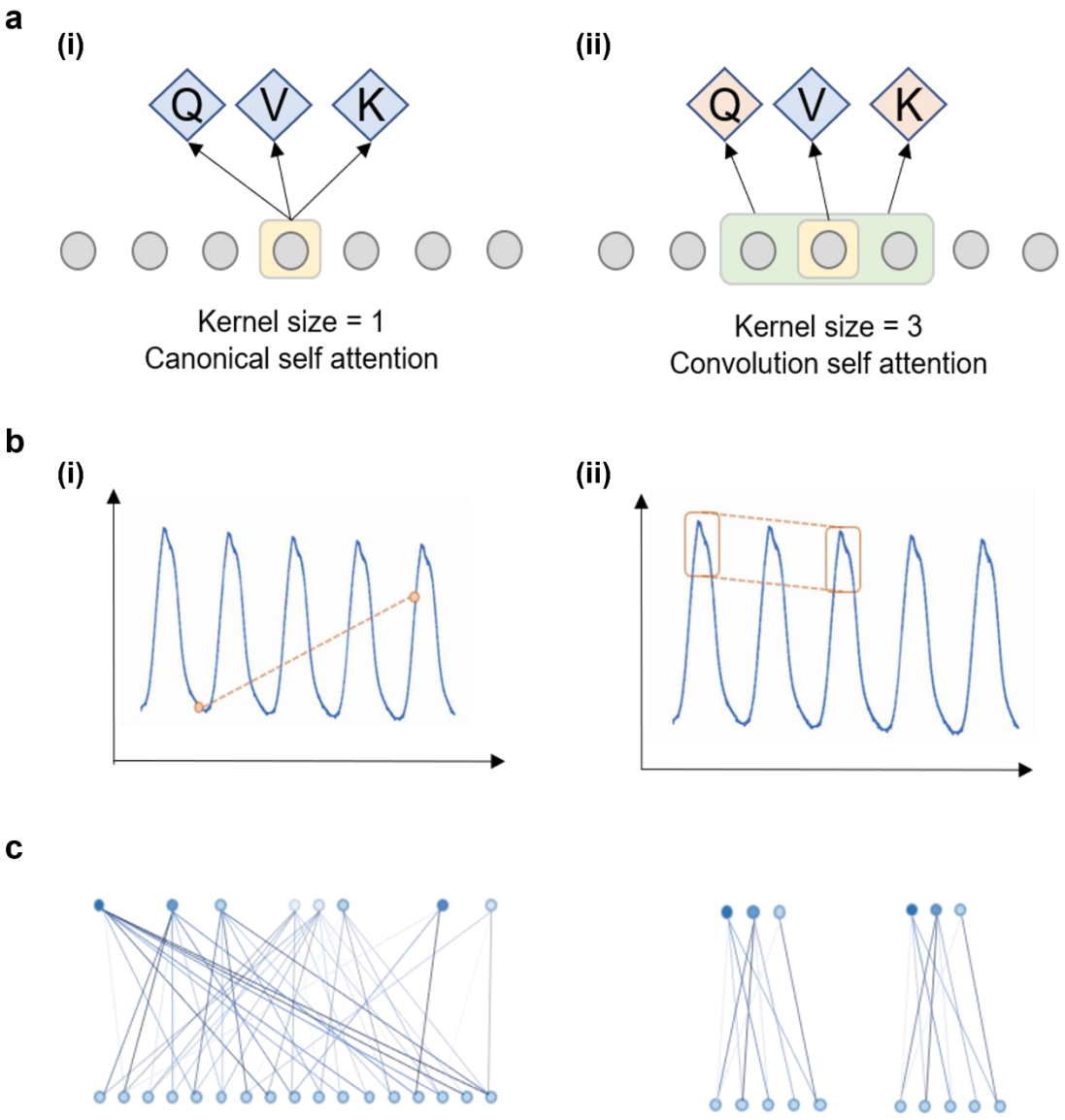
Supplementary Fig. 16. Model predictions for hysteresis issue. (a) Visualization of predicted and true sensing curves for LSTM model. **(b)** Visualization of predicted and true sensing curves for GRU model. **(c)** Visualization of predicted and true sensing curves for 1DCNN model.



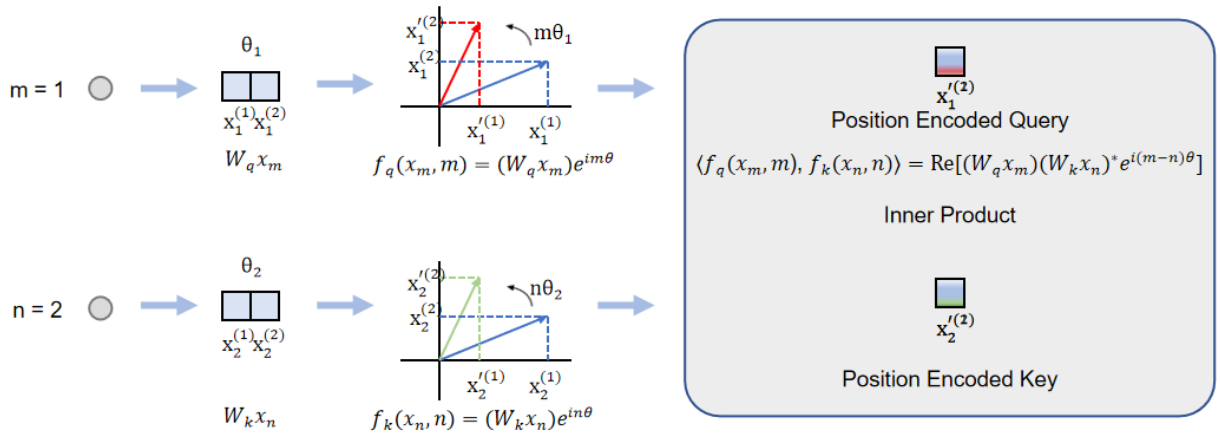
Supplementary Fig. 17. Schematic figure for SHAP mechanism. (a) Illustration of the SHAP value in a cooperative game, where players join in different sequences to contribute to the total reward. The Shapley value is calculated as the average of marginal contributions across all possible sequences. **(b)** Application of the SHAP mechanism in sensor issues, where a sequence of resistance values from $t-n$ to t is treated as input. Each resistance value's SHAP value represents its importance in predicting the strain, calculated based on its marginal contribution when included or excluded from the sequence. Detailed information can be found in **Supplementary Note 8**.



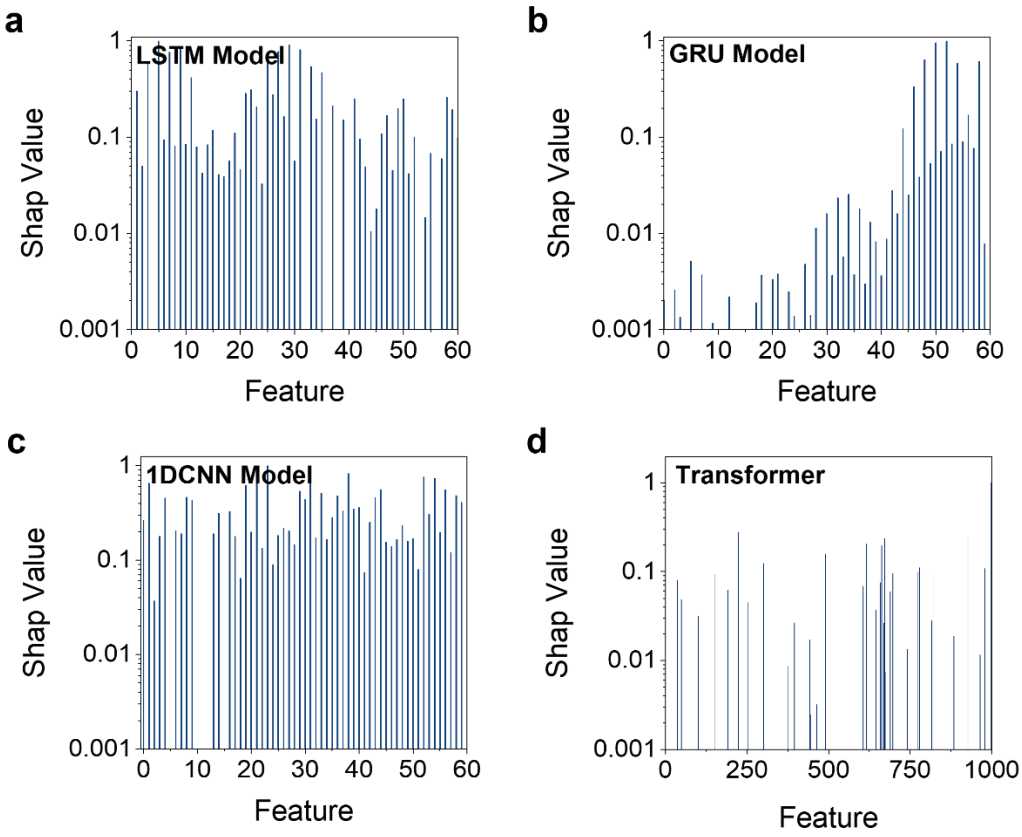
Supplementary Fig. 18. SHAP analysis for hysteresis issue. (a) SHAP values for the LSTM model and **(b)** SHAP values for the GRU model. These visualizations highlight the contributions of resistance values at different positions within the input sequence. Detailed analysis can be found in **Supplementary Note 9**.



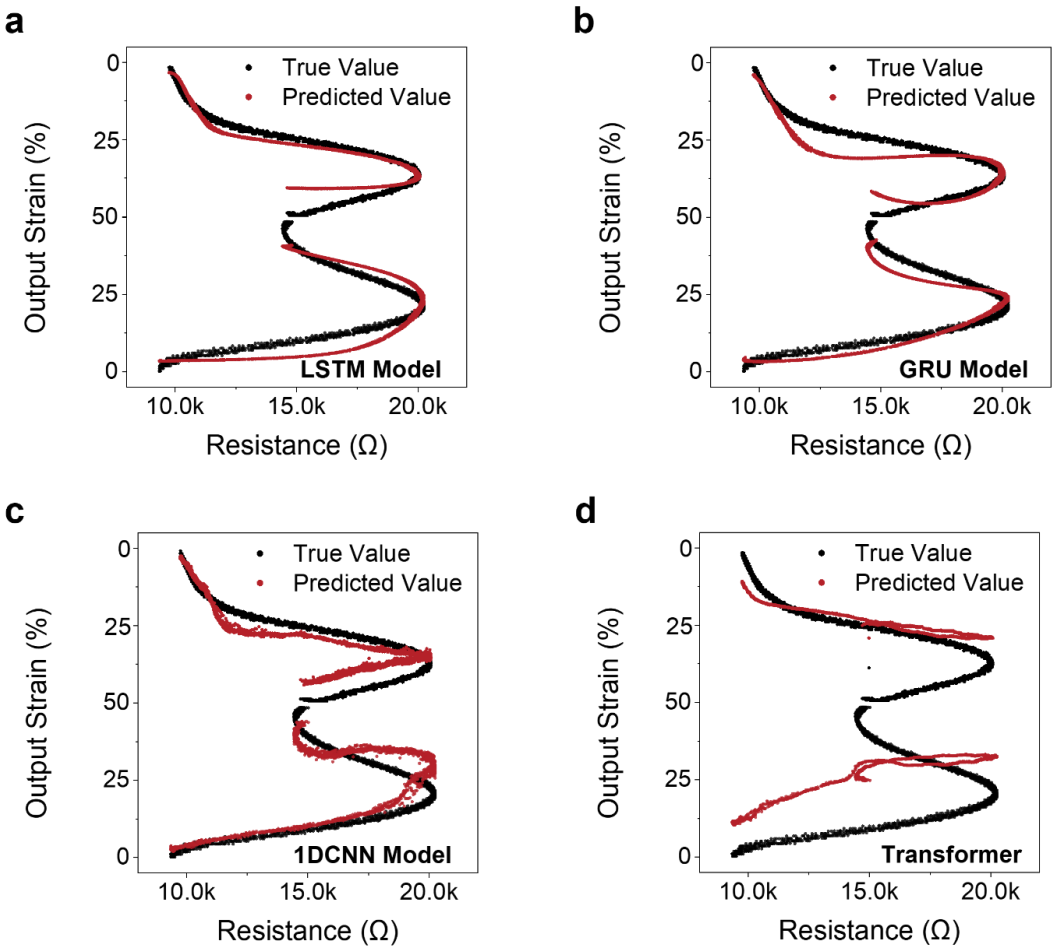
Supplementary Fig. 19. Comparison of canonical and convolutional self-attention in time series data. (a) Structural comparison of self-attention mechanisms, showing the differences between canonical self-attention (i) and convolutional self-attention (ii). **(b)** Illustrations of sensitivity to local context: canonical self-attention (i) focuses more on global information, while convolutional self-attention (ii) more effectively captures local context. **(c)** Visualization of attention distribution differences, where convolutional self-attention demonstrates higher effectiveness in capturing local patterns compared to canonical self-attention. Detailed information can be found in **Supplementary Note 11**.



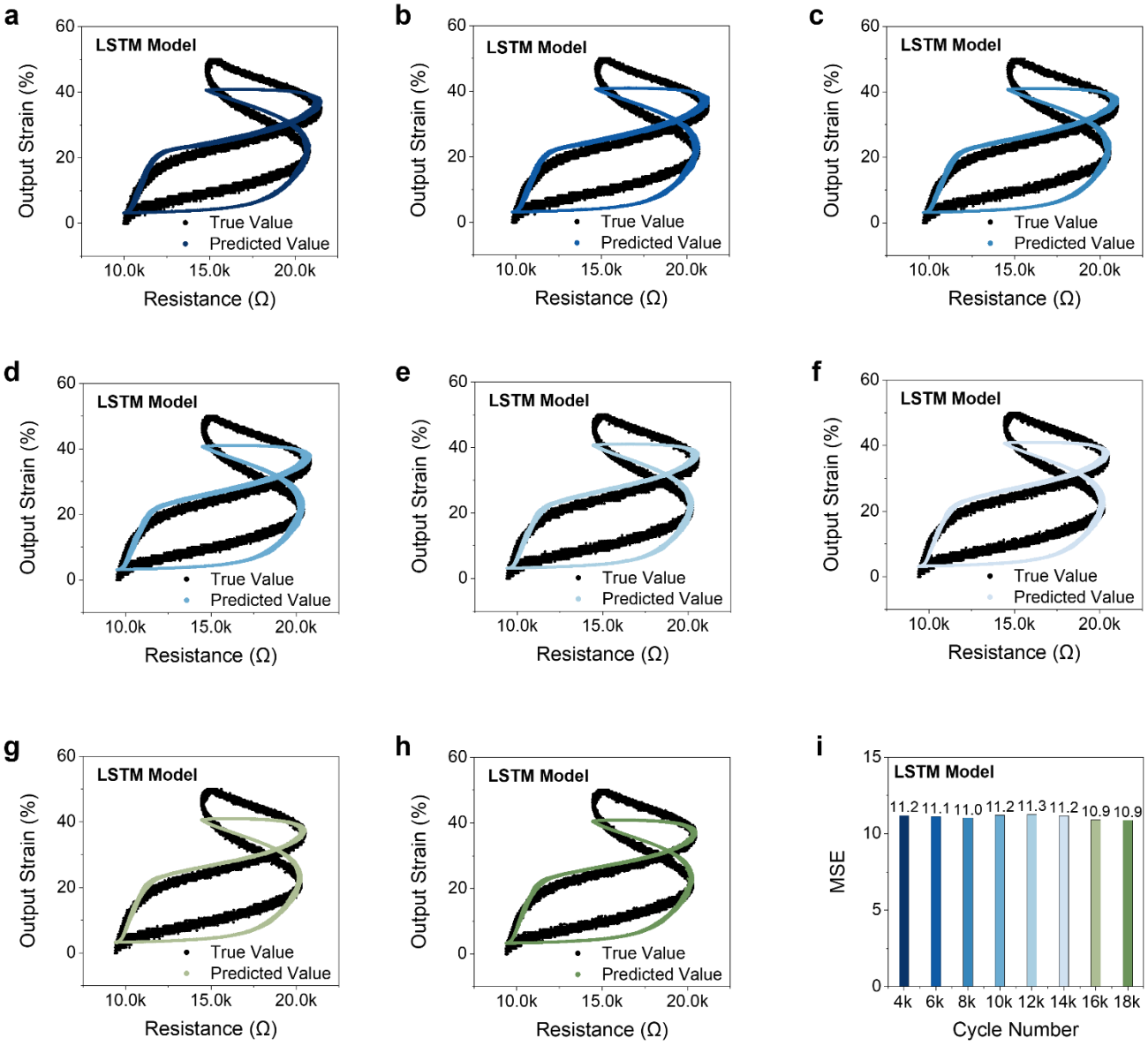
Supplementary Fig. 20. Schematic figure of the Rotary Position Embedding (RoPE) in time-series data. RoPE integrates relative positional information into embeddings through rotational transformations based on positional indices. The transformed Q, K, and V matrices not only encode the raw data but also represent relative positional relationships, enabling the model to more effectively capture temporal dependencies and sequential relationships. Detailed information can be found in **Supplementary Note 11**.



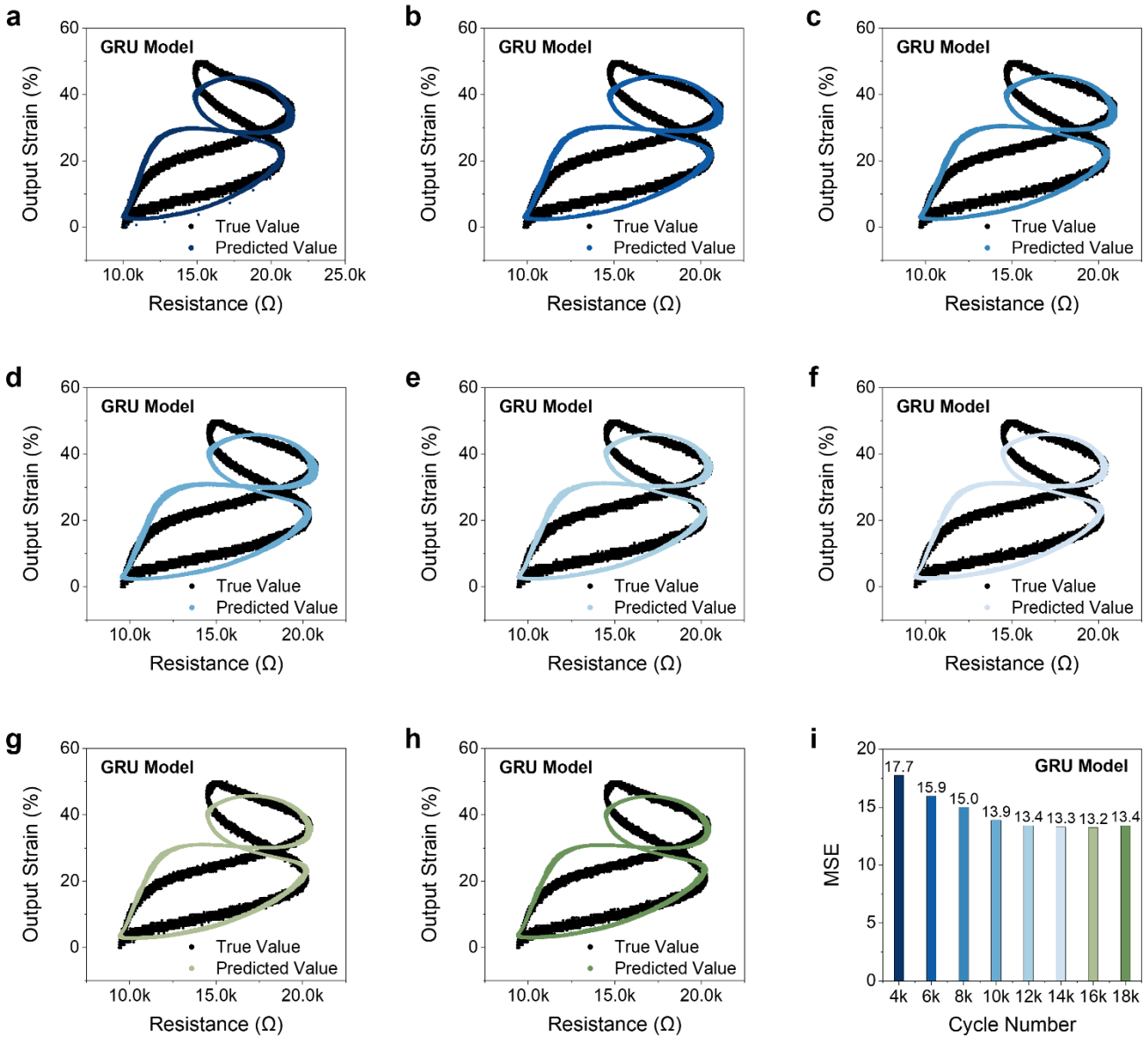
Supplementary Fig. 21. SHAP analysis of black-box sequential models for cycling attenuation problem. (a) SHAP values for the LSTM model; **(b)** SHAP values for the GRU model; **(c)** SHAP values for the 1DCNN model; **(d)** SHAP values for the Transformer model. These visualizations highlight the contributions of resistance values at different positions within the input sequence to the model predictions, providing a deeper understanding of model behavior.



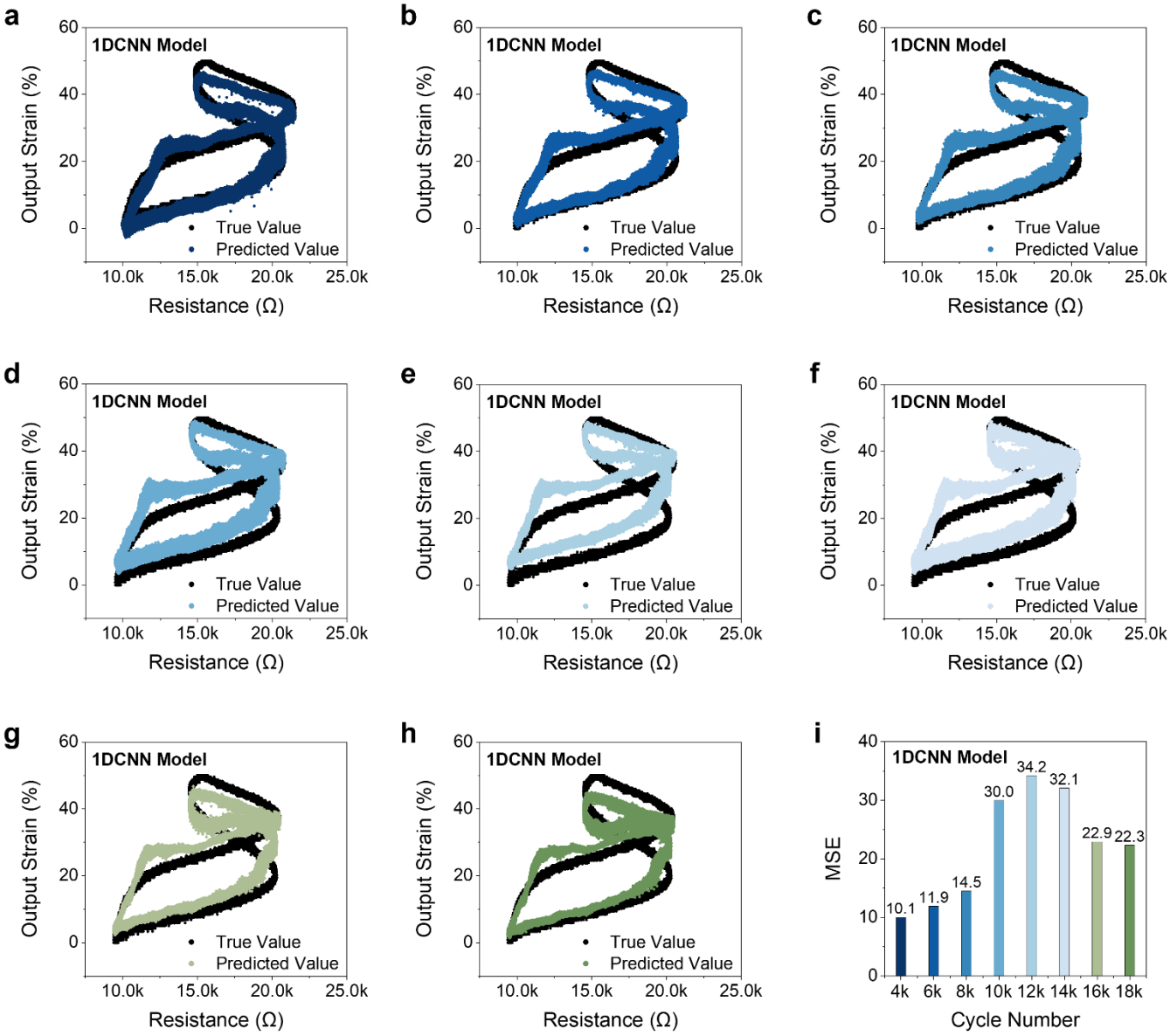
Supplementary Fig. 22. Comparison of predicted and actual applied strains by black-box sequential models in the last 100 cycles of the cycling attenuation task based on sensor resistance feedback. (a) LSTM, **(b)** GRU, **(c)** 1DCNN, and **(d)** Transformer.



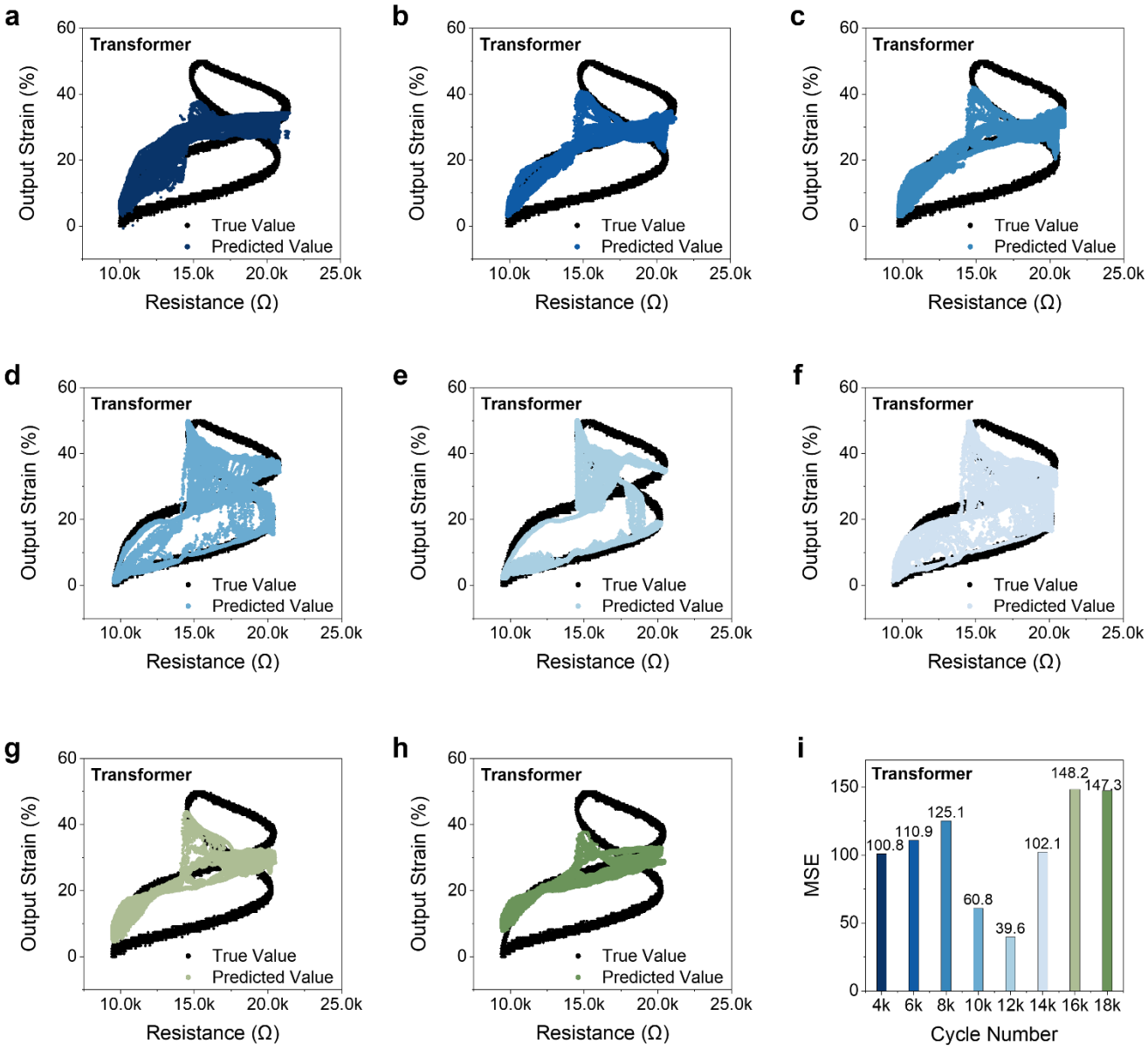
Supplementary Fig. 23. Predictive performance of the LSTM model on the cycling attenuation issue. (a)-(h) Visualization of LSTM predictions every 2k cycles starting from the 4000th cycle. **(i)** Overall LSTM predictions from the 4000th to the 20,000th cycle, with each column representing the MSE values for every 2k cycles.



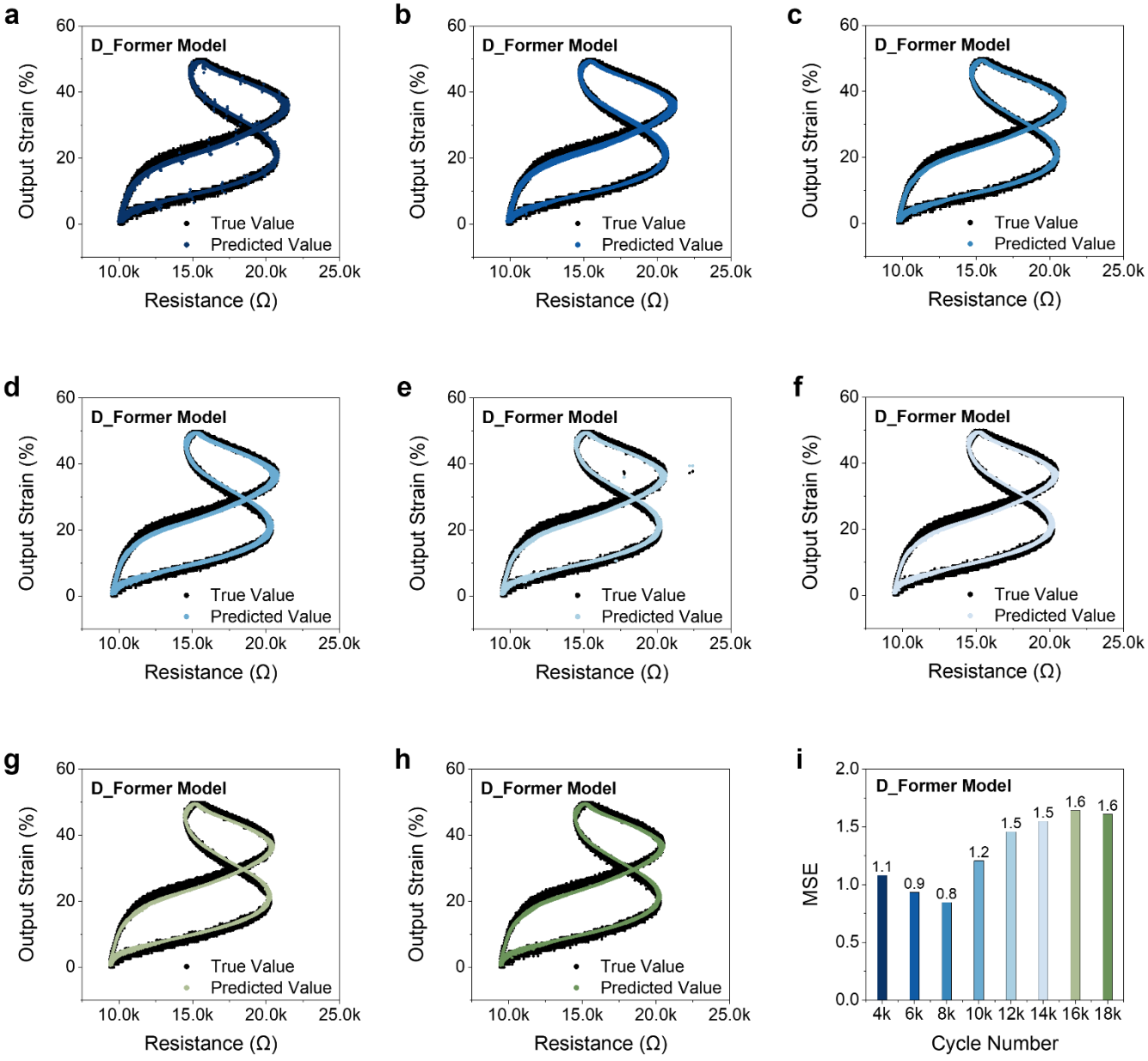
Supplementary Fig. 24. Predictive performance of the GRU model on the cycling attenuation issue. (a)-(h) Visualization of GRU predictions every 2k cycles starting from the 4000th cycle. **(i)** Overall GRU predictions from the 4000th to the 20,000th cycle, with each column representing the MSE values for every 2k cycles.



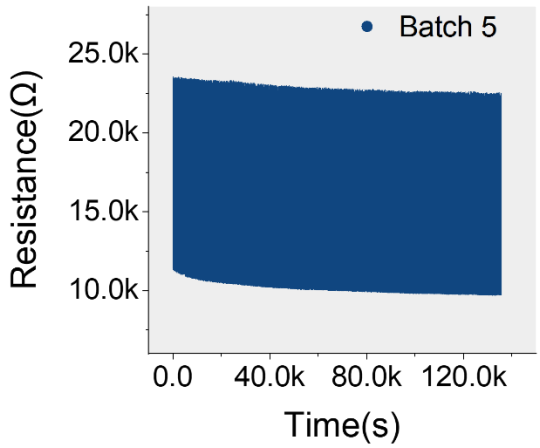
Supplementary Fig. 25. Predictive performance of the 1DCNN model on the cycling attenuation issue. (a)-(h) Visualization of 1DCNN predictions every 2k cycles starting from the 4000th cycle. **(i)** Overall 1DCNN predictions from the 4000th to the 20,000th cycle, with each column representing the MSE values for every 2k cycles.



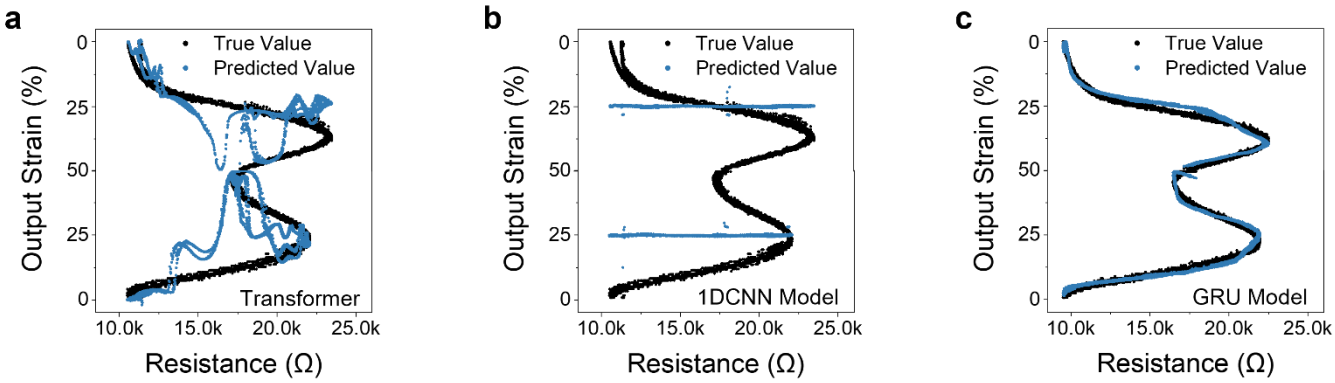
Supplementary Fig. 26. Predictive performance of the Transformer model on the cycling attenuation issue. (a)-(h) Visualization of Transformer predictions every 2k cycles starting from the 4000th cycle. **(i)** Overall Transformer predictions from the 4000th to the 20,000th cycle, with each column representing the MSE values for every 2k cycles.



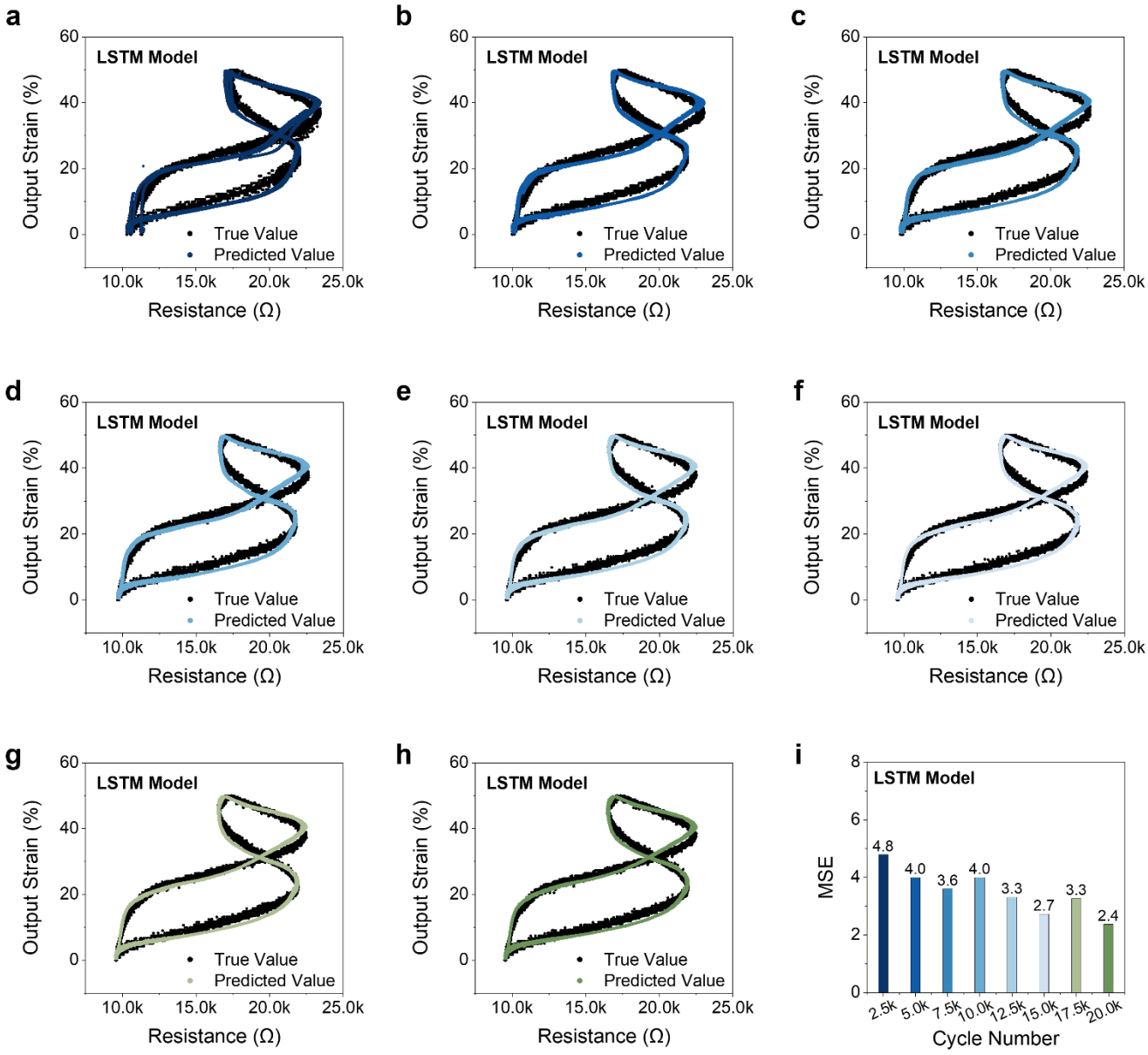
Supplementary Fig. 27. Predictive performance of the D_Former model on the cycling attenuation issue. (a)-(h) Visualization of D_Former predictions every 2k cycles starting from the 4000th cycle. **(i)** Overall D_Former predictions from the 4000th to the 20,000th cycle, with each column representing the MSE values for every 2k cycles.



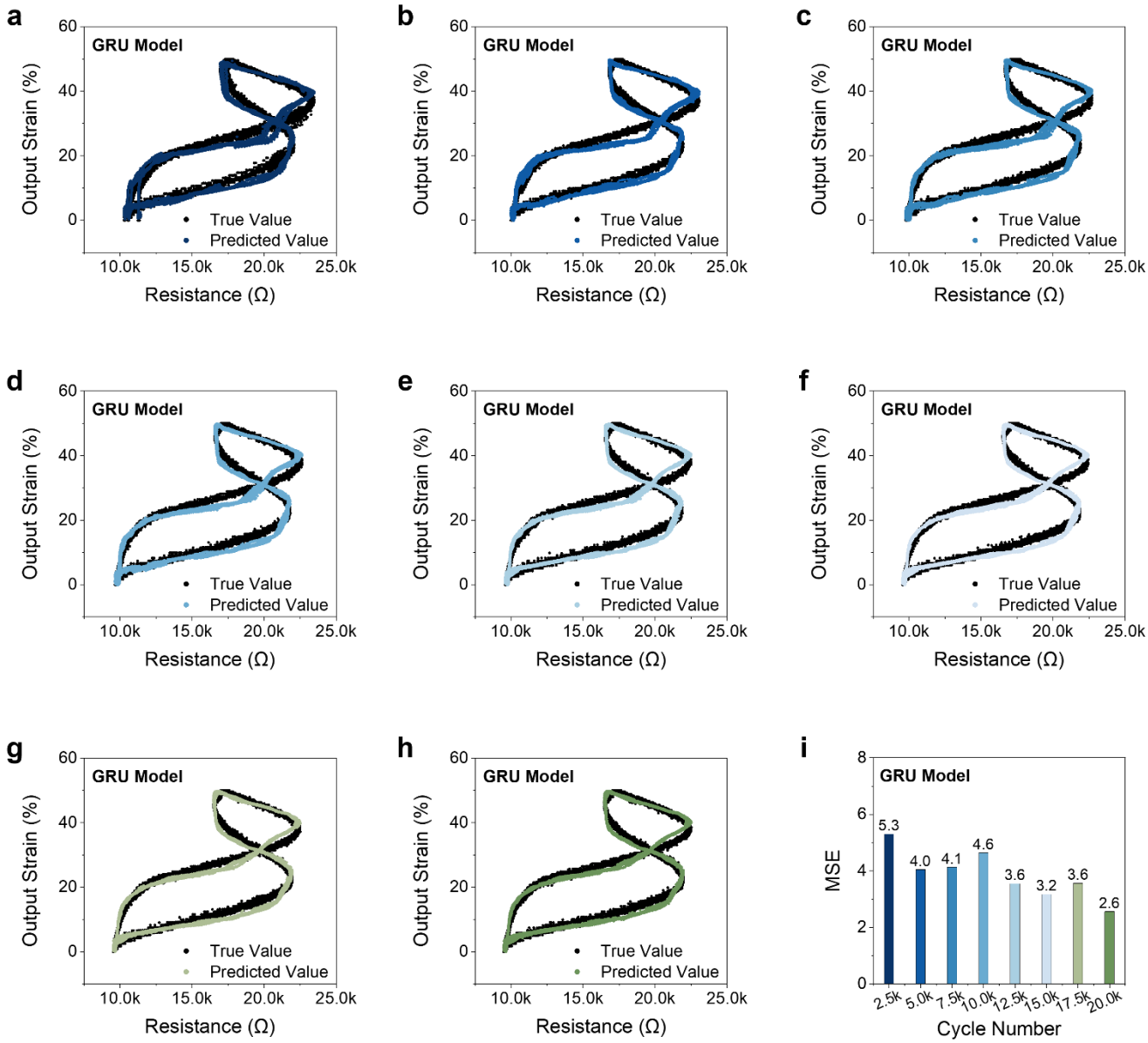
Supplementary Fig. 28. Resistance-Time curves of strain sensors in Batch 5. The resistance variations over time during testing, compared to **Fig. 6a**, highlight the cycling attenuation and inconsistencies across different batches of fabricated sensors.



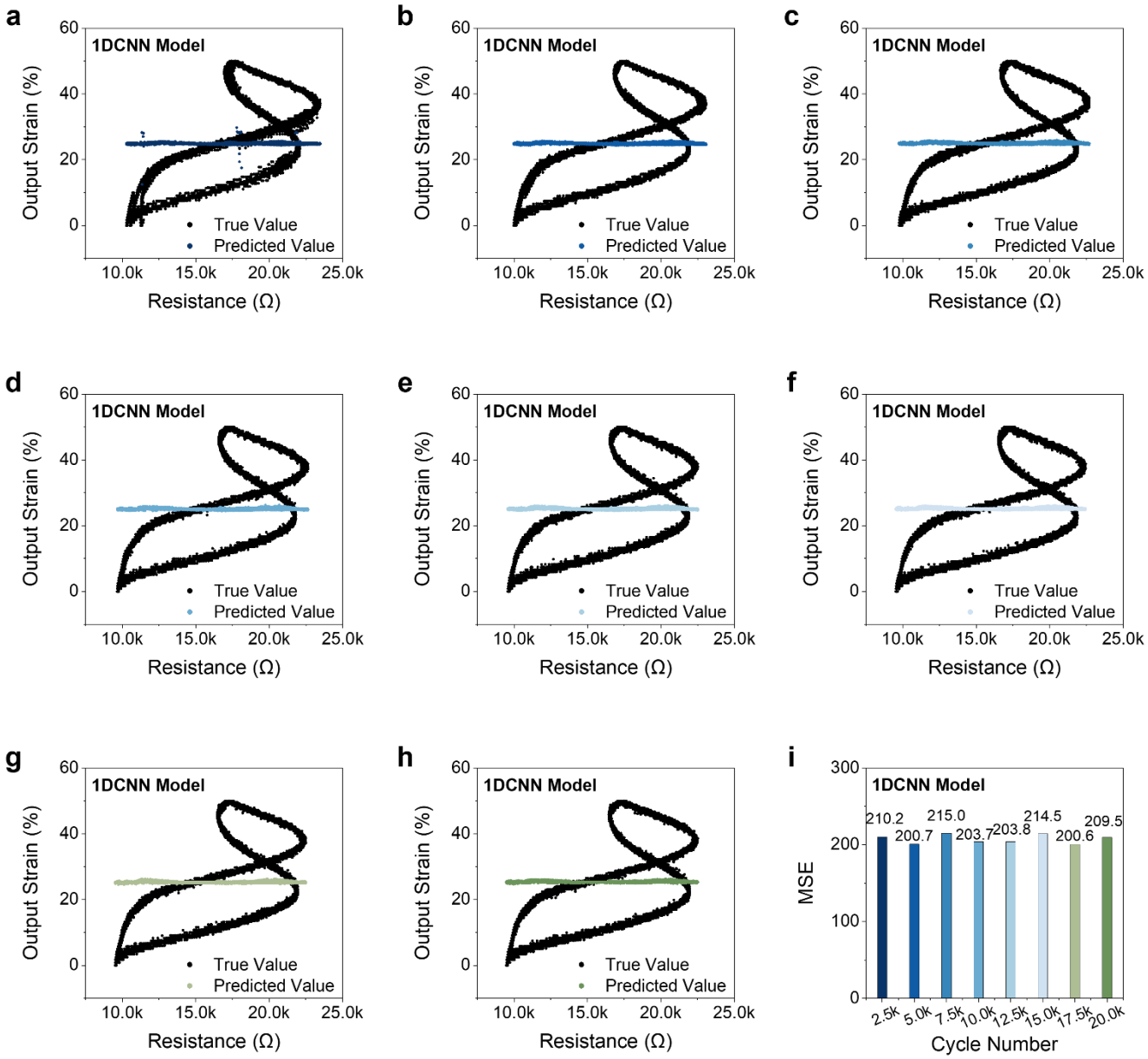
Supplementary Fig. 29. Predictive performance of different models for the cycling attenuation during the first 2000 cycles of Batch 5. (a) Visualization of the predicted and actual applied strains by the Transformer model. **(b)** Visualization of the predicted and actual applied strains by the 1DCNN model. **(c)** Visualization of the predicted and actual applied strains by the GRU model.



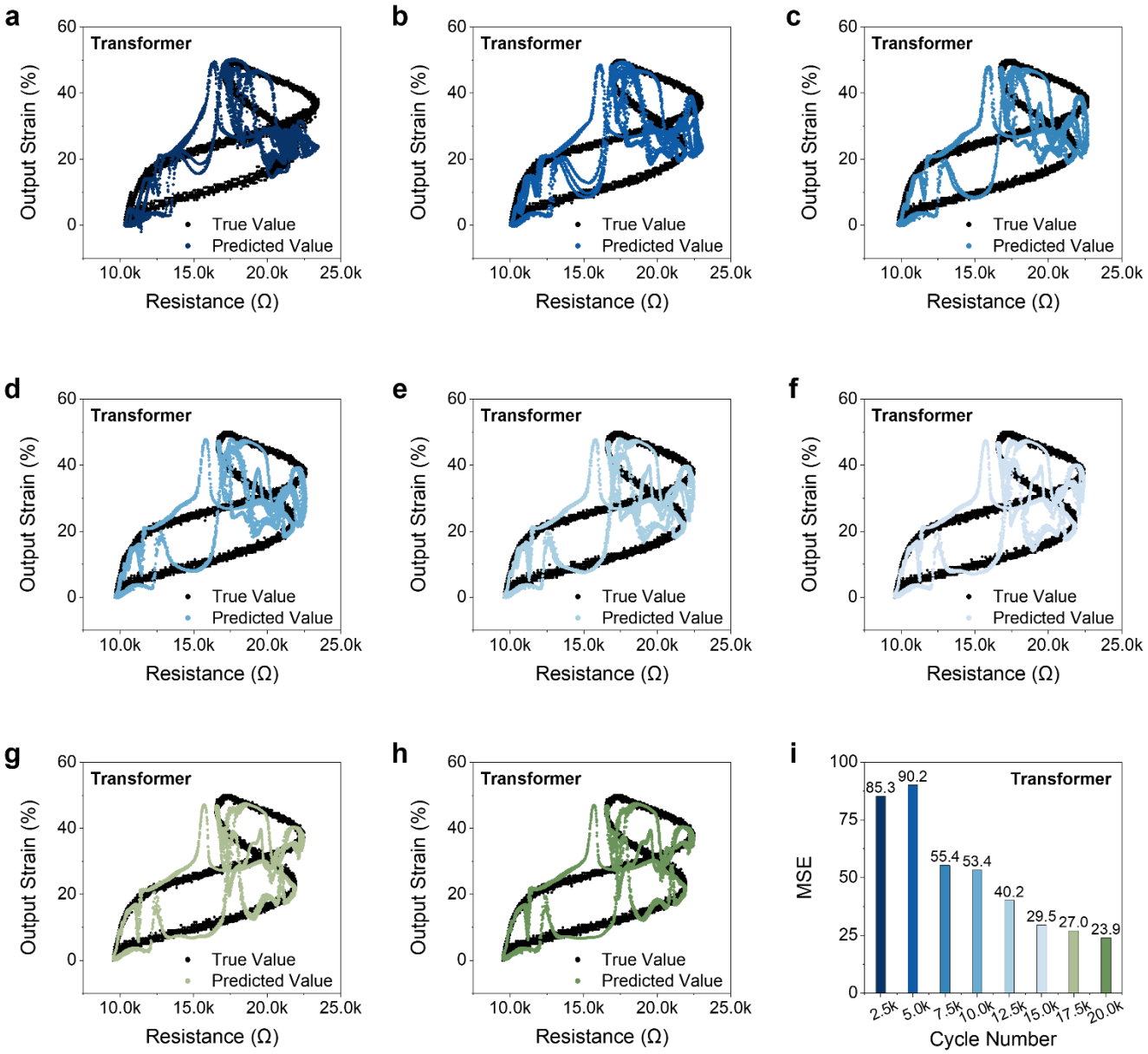
Supplementary Fig. 30. Predictive performance of the LSTM model on the batch inconsistency issue (Batch 5). (a)-(h) Visualizations of LSTM predictions at every 2.5k cycles, starting from the 0th cycle. **(i)** Overall LSTM predictions from the 0th to the 20,000th cycle, with each column representing the MSE values for every 2.5k cycles.



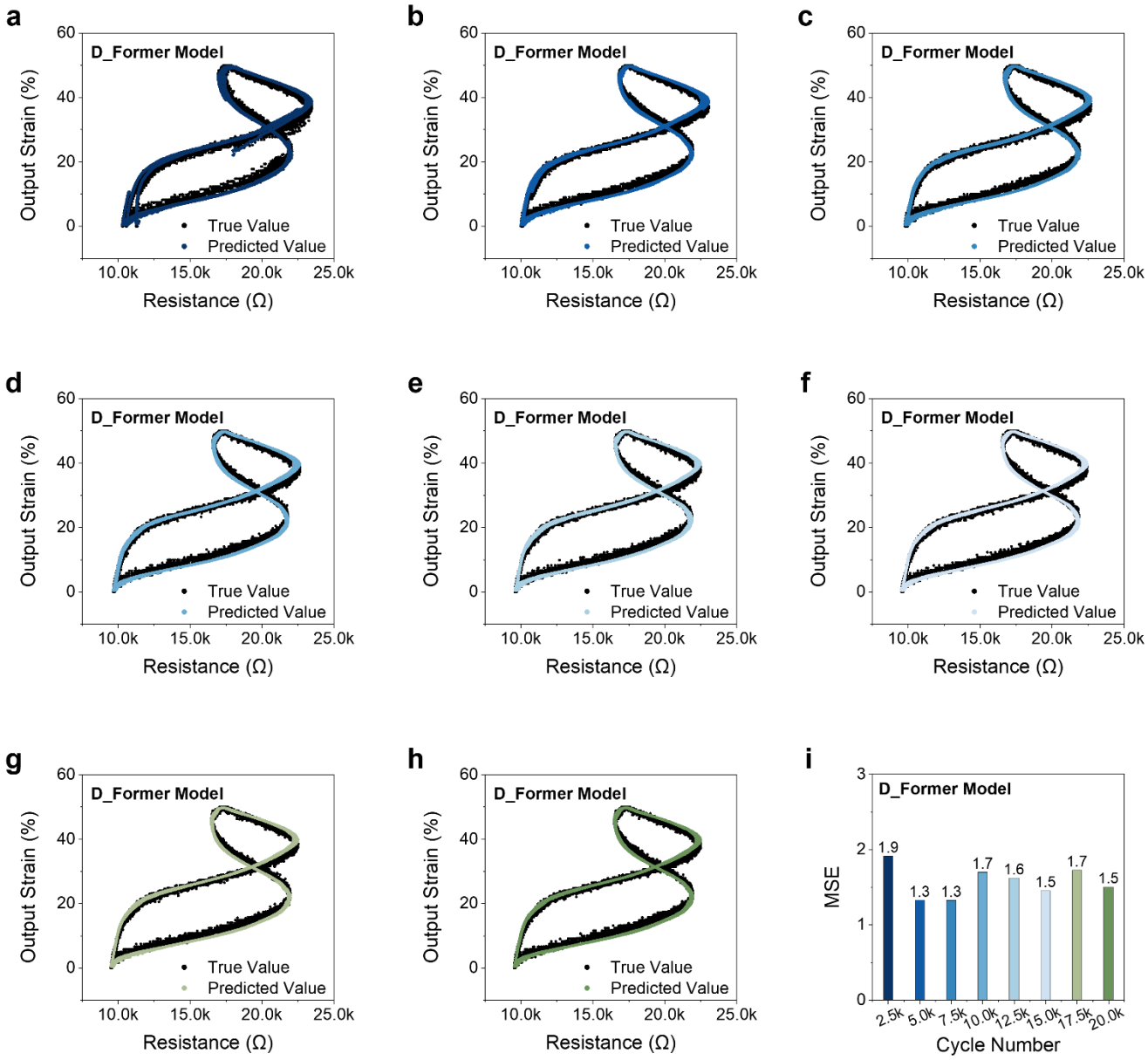
Supplementary Fig. 31. Predictive performance of the GRU model on the batch inconsistency issue (Batch 5). (a)-(h) Visualizations of GRU predictions at every 2.5k cycles, starting from the 0^{th} cycle. **(i)** Overall GRU predictions from the 0^{th} to the $20{,}000^{th}$ cycle, with each column representing the MSE values for every 2.5k cycles.



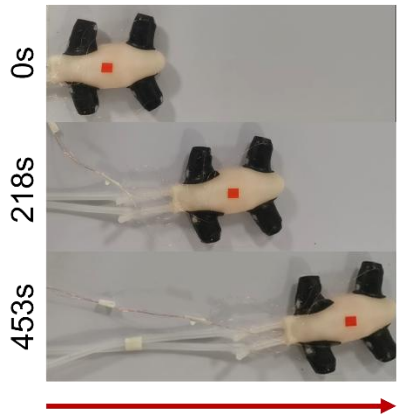
Supplementary Fig. 32. Predictive performance of the 1DCNN model on the batch inconsistency issue (Batch 5). (a)-(h) Visualizations of 1DCNN predictions at every 2.5k cycles, starting from the 0th cycle. **(i)** Overall 1DCNN predictions from the 0th to the 20,000th cycle, with each column representing the MSE values for every 2.5k cycles.



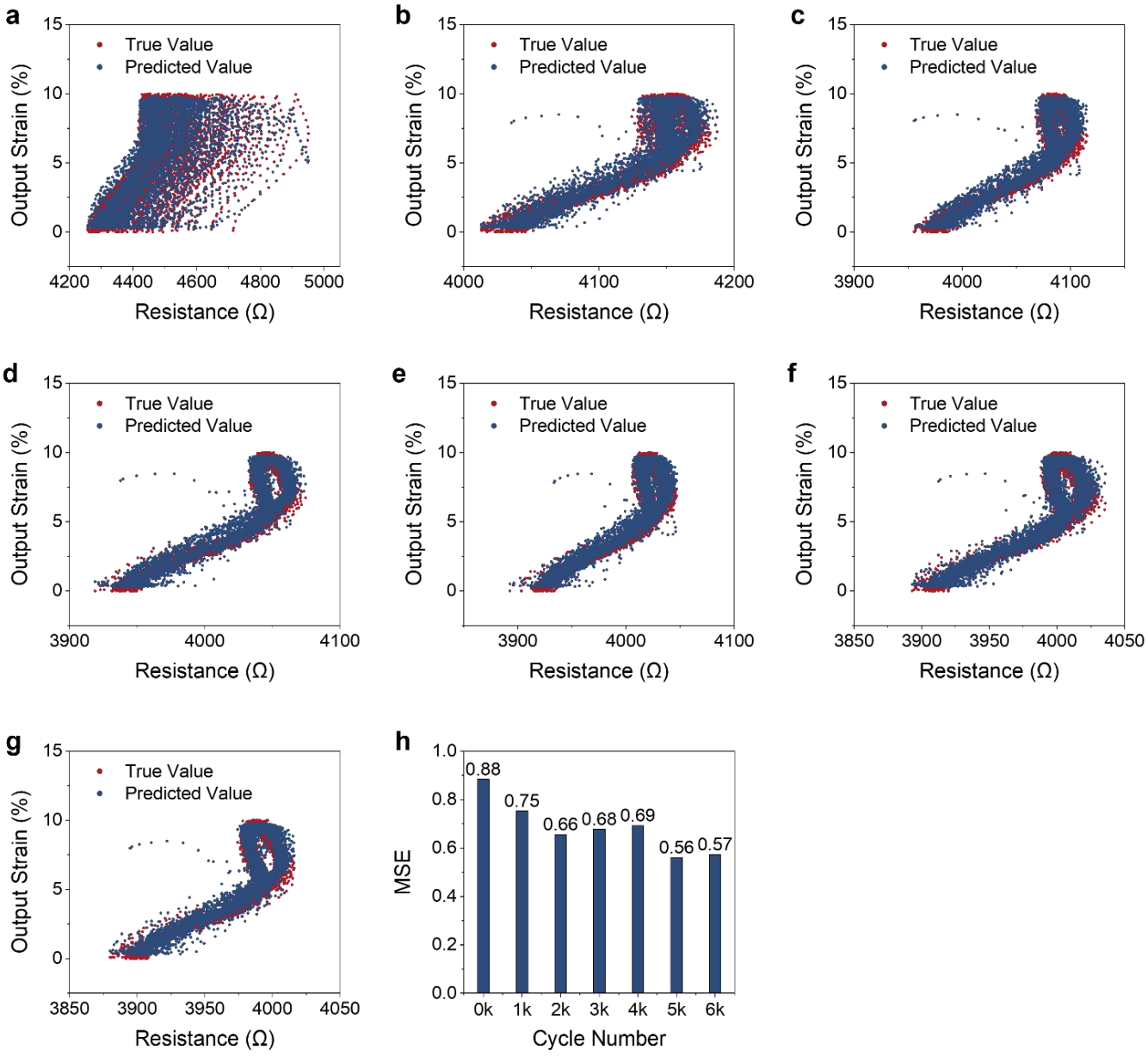
Supplementary Fig. 33. Predictive performance of the Transformer model on the batch inconsistency issue (Batch 5). (a)-(h) Visualizations of Transformer predictions at every 2.5k cycles, starting from the 0th cycle. **(i)** Overall Transformer predictions from the 0th to the 20,000th cycle, with each column representing the MSE values for every 2.5k cycles.



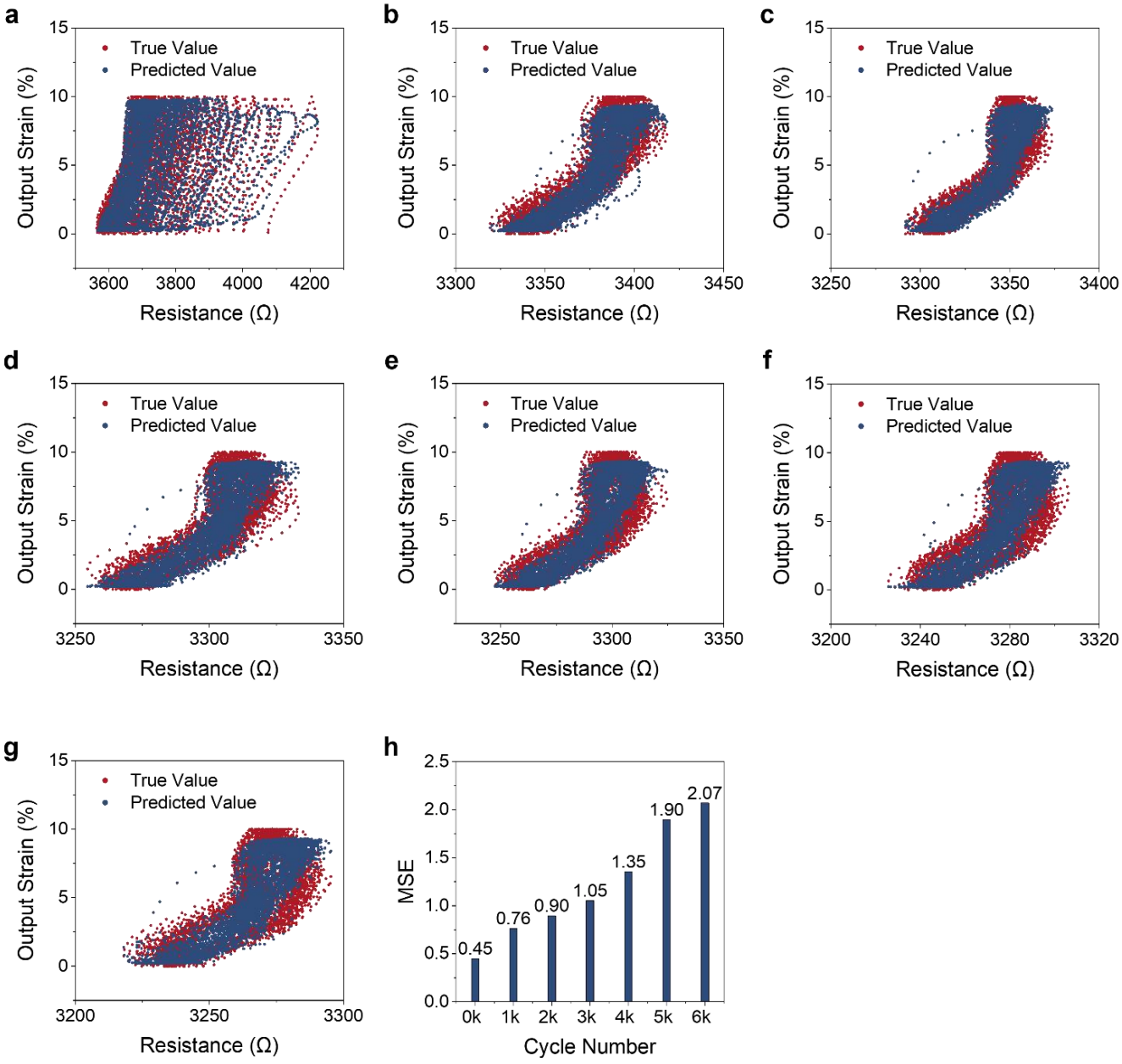
Supplementary Fig. 34. Predictive performance of the D_Former model on the batch inconsistency issue (Batch 5). (a)-(h) Visualizations of D_Former predictions at every 2.5k cycles, starting from the 0^{th} cycle. **(i)** Overall D_Former predictions from the 0^{th} to the $20{,}000^{th}$ cycle, with each column representing the MSE values for every 2.5k cycles.



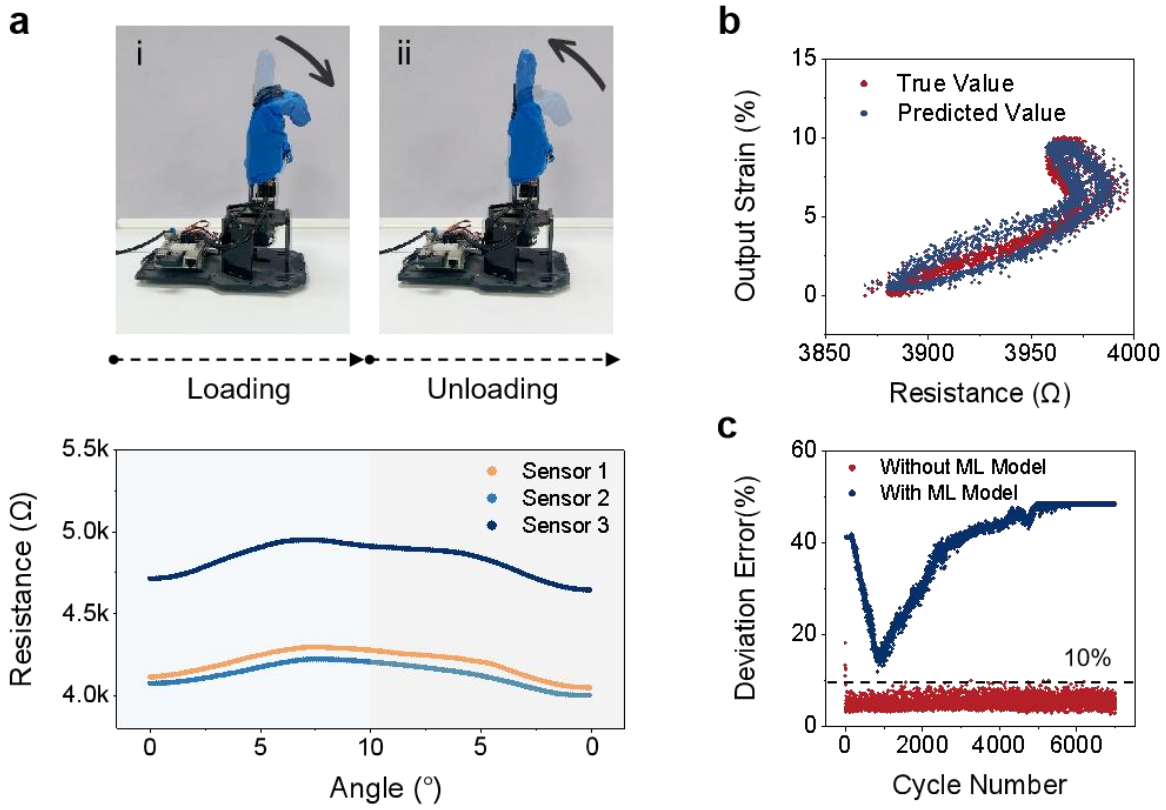
Supplementary Fig. 35. Sequential motion analysis of quadrupedal soft robot at different time points. The motion is monitored using four strain sensors attached to the robot's limbs, effectively perceiving its movement state.



Supplementary Fig. 36. Predictive performance of the D_Former model on dexterous robot hand application (train on Batch 2, 3, predict on Batch 1). (a)-(g) Visualizations of D_Former predictions at every 100 cycles, starting from the 0th cycle. **(h)** Overall D_Former predictions from the 0th to the 6,000th cycle, with each column representing the MSE values for every 100 cycles.



Supplementary Fig. 37. Predictive performance of the D_Former model on dexterous robot hand application (train on Batch 3, predict on Batch 2). (a)-(g) Visualizations of D_Former predictions at every 100 cycles, starting from the 0th cycle. **(h)** Overall D_Former predictions from the 0th to the 6,000th cycle, with each column representing the MSE values for every 100 cycles.



Supplementary Fig. 38. Analysis of dexterous robotic hand motion states and model prediction results based on strain sensors. (a) Fist and release states of the dexterous robotic hand, along with the response of strain sensors on three joints during this motion. (b) Model prediction of the joint angle using training data from two joints, predicting the motion of the third joint. (c) Comparison of deviation errors during grasping tasks with and without ML model assistance.

Supplementary Table 1. LCA calculations for MXene-based sensor preparation process.

Flows	Type of substance input	Unit	Amount of substance input(kg)	Type of substance exported	Amount of substance exported (kg)	Activity name	Reference product name	Geography	IPCC 2013 GWP20a (kg)	Total carbon emissions
Blender	LiF	kg	0.003	/	0	lithium fluoride production	lithium fluoride	RoW	15.56223	0.04669
	9M Hcl	kg	0.01314	/	0	hydrochloric acid production, from the reaction of hydrogen with chlorine	hydrochloric acid, without water, in 30% solution state	RoW	1.68634	0.02216
	electricity consumption	kWh	0.015	/	0	heat and power co-generation, natural gas, conventional power plant, 100MW electrical	electricity, high voltage	RoW	0.72560	0.01088
Isothermal blender	Ti_3AlC_2 Max power	kg	0.001	/	0	aluminium alloy production, Metallic Matrix Composite	aluminium alloy, metal matrix composite	RoW	23.26089	0.02326
	electricity consumption	kWh	1.96	/	0	heat and power co-generation, natural gas, conventional power plant, 100MW electrical	electricity, high voltage	RoW	0.72560	1.42218
High-speed centrifuge	2M Hcl	kg	0.00876	/	0	hydrochloric acid production, from the reaction of hydrogen with chlorine	hydrochloric acid, without water, in 30% solution state	RoW	1.68634	0.01477
	deionized water	kg	0.5	2M Hcl、9M Hcl、deionized water、LiF	0.5249	water production, deionised	water, deionised	RoW	0.00060	0.00030
	electricity consumption	kWh	1	/	0	heat and power co-generation, natural gas, conventional power plant, 100MW electrical	electricity, high voltage	RoW	0.72560	0.72560393
Ultrasonic	deionized water	kg	0.05	deionized water	0.005	water production, deionised	water, deionised	RoW	0.00060	3.0161E-05
	electricity consumption	kWh	0.0667	/	0	heat and power co-generation, natural gas, conventional power plant, 100MW electrical	electricity, high voltage	RoW	0.72560	0.04840

High-speed centrifuge	/	/	0	Ti_3AlC_2 Max power	0.0003	aluminium alloy production, Metallic Matrix Composite	aluminium alloy, metal matrix composite	RoW	23.26089	0.00698
	electricity consumption	kWh	0.0667	/	0	heat and power co-generation, natural gas, conventional power plant, 100MW electrical	electricity, high voltage	RoW	0.72560	0.04840
Preparation of sensor	Average usage rate calculated at 50%									0.00675
Vaccum filtration	pvdf membrane	kg	0.000204037	/	0	polyvinylfluoride, film production	polyvinylfluoride, film	RoW	31.42296	0.00641
	electricity consumption	kWh	0.0833	/	0	heat and power co-generation, natural gas, conventional power plant, 100MW electrical	electricity, high voltage	RoW	0.72560	0.06044
Delamination	Ethanol	kg	0.2367	Ethanol (Cannot be re-used)	0.2367	ethanol production from sugar beet molasses	ethanol, without water, in 95% solution state, from fermentation	RoW	1.13427	0.26848
	/	/	0	pvdf membrane(Cannot be re-used)	0.000204037	polyvinylfluoride, film production	polyvinylfluoride, film	RoW	31.42296	0.00641
Transfer	ecoflex 00-30 partA&B	kg	0.0052	/	0	silicone product production	silicone product	RoW	3.91818	0.02037
Total	0.35150									

Remarks: the calculation of the liquid material quality is calculated by multiplying the dosage by the density; pvdf film is calculated by multiplying the volume by the bulk weight of 1.8g/cm^3; due to the failure to find a special calculation of Ti_3Al_2C, but based on aluminium alloys and ceramic materials production process and raw materials are similar, so here the use of aluminium alloy production, the Metallic Matrix Composite.

Supplementary Table 2. LCA calculations for CW-based sensor preparation process.

Flows	Type of substance input	Unit	Amount of substance input(kg)	Type of substance exported	Amount of substance exported(kg)	Activity name	Reference product name	Geography	IPCC 2013 GWP20a (kg)	Total carbon emissions
Blender	ecoflex 00-30 partA&B	kg	0.0052	/	0	activated silica production	activated silica	GLO	1.80911	0.00941
	carbon black	kg	0.000005	/	0	carbon black production	carbon black	GLO	1.95422	9.7711E-06
	electricity consumption	kWh	0.00025	/	0	heat and power co-generation, natural gas, conventional power plant, 100MW electrical	electricity, high voltage	RoW	0.72560	0.00018
Air-dried	/	/	/	/	0	/	/	/	/	/
Total	0.009598546									

Supplementary Table 3. LCA calculations for silver nanowires-based sensor preparation process.

Flows	Type of substance input	Unit	Amount of substance input(kg)	Type of substance exported	Amount of substance exported(kg)	Activity name	Reference product name	Geography	IPCC 2013 GWP20a (kg)	Total carbon emissions
Heated magnetic stirrer	Ethylene glycol	kg	0.05	/	0	ethylene glycol production	ethylene glycol	RoW	2.5440	0.1270
	electricity consumption	kWh	0.1	/	0	heat and power co-generation, natural gas, conventional power plant, 100MW electrical	electricity, high voltage	RoW	0.7256	0.0730
Heater	$CuCl_2$	kg	0.0013544	/	0	iron(II) chloride production	iron(II) chloride	GLO	0.2466	0.0003
	electricity consumption	kWh	0.05	/	0	heat and power co-generation, natural gas, conventional power plant, 100MW electrical	electricity, high voltage	RoW	0.7256	0.0363
Reactor	Polyvinylpyrrolidone (PVP)	kg	0.0000172	/	0	/	/	/	/	0
	$AgNO_3$	kg	0.01598	/	0	potassium nitrate production	potassium nitrate	RoW	2.1628	0.0346
Heater	electricity consumption	kWh	0.3	/	0	heat and power co-generation, natural gas, conventional power plant, 100MW electrical	electricity, high voltage	RoW	0.7256	0.2177
Reactor	acetone	kg	0.3136	/	0	acetone production, liquid	acetone, liquid	RoW	3.2704	1.0256
Centrifuge	electricity consumption	kWh	0.083333	/	0	heat and power co-generation, natural gas, conventional power plant, 100MW electrical	electricity, high voltage	RoW	0.7256	0.0605
Washing	ethanol	kg	0.9468	ethanol	0.2367	ethanol production from sugar beet molasses	ethanol, without water, in 95% solution state, from fermentation	RoW	1.1343	1.0739
Preparation of sensor										
Cleaning glass slide	acetone	kg	0.00392	acetone	0.00392	acetone production, liquid	acetone, liquid	RoW	3.2704	0.0128
	ethanol	kg	0.003945	ethanol	0.003945	ethanol production from sugar beet molasses	ethanol, without water, in 95%	RoW	1.1343	0.0045

							solution state, from fermentation			
	DI water	kg	0.015	DI water	0.015	water production, deionised	water, deionised	RoW	0.0006	9.048E-06
	polyimide tape	kg	0.09	/	0	glass fibre reinforced plastic production, polyamide, injection moulded	glass fibre reinforced plastic, polyamide, injection moulded	RoW	10.926	0.9833
Glass slide	AgNW solution	kg	/							0.01766
Lamp light	electricity consumption	kWh	0.01667	/	0	heat and power co-generation, natural gas, conventional power plant, 100MW electrical	electricity, high voltage	RoW	0.7256	0.0121
Glass slide	/	kg	0	polyimide tape	0.09	/	/	/	/	0
Heater	electricity consumption	kWh	0.066667	/	0	heat and power co-generation, natural gas, conventional power plant, 100MW electrical	electricity, high voltage	RoW	0.7256	0.0484
Glass slide	liquid PDMS	kg	0.00003	/	0	polydimethylsiloxane production	polydimethylsiloxane	GLO	21.6533	0.0007
Curing	electricity consumption	kWh	0.11667	/	0	heat and power co-generation, natural gas, conventional power plant, 100MW electrical	electricity, high voltage	RoW	0.7256	0.0847
Glass slide	liquid PDMS	kg	0.00003	/	0	polydimethylsiloxane production	polydimethylsiloxane	GLO	21.6533	0.0007
Curing	electricity consumption	kWh	0.7	/	0	heat and power co-generation, natural gas, conventional power plant, 100MW electrical	electricity, high voltage	RoW	0.7256	0.5079
Total	1.672640631									

Note: The mass of the liquid material is calculated by multiplying the dosage by the density, polyimide tapes are replaced by polyamide, and polyvinylpyrrolidone is ignored due to its low dosage.

Supplementary Table 4. LCA calculations for graphene-based sensor preparation process.

Flows	Type of substance input	Unit	Amount of substance input(kg)	Type of substance exported	Amount of substance exported (kg)	Activity name	Reference product name	Geography	IPCC 2013 GWP20a (kg)	Total carbon emissions
Quartz holder	copper foil	kg	0.0067	/	0	copper collector foil production, for Li-ion battery	copper collector foil, for Li-ion battery	GLO	9.4776	0.0635
Pumping unit	electricity consumption	kWh	0.045	/	0	heat and power co-generation, natural gas, conventional power plant, 100MW electrical	electricity, high voltage	RoW	0.7256	0.03265
Heater	electricity consumption	kWh	0.5	/	0	heat and power co-generation, natural gas, conventional power plant, 100MW electrical	electricity, high voltage	RoW	0.7256	0.3628
Quartz tube	methane	kg	0.000287	/	0	biomethane production, low pressure, vehicle grade	biomethane, low pressure, vehicle grade	RoW	2.1906	0.00063
	hydrogen	kg	0.000009	/	0	hydrogen production, gaseous, petroleum refinery operation	hydrogen, gaseous	RoW	1.8158	1.63E-05
Preparation of sensor										0.4596
Sonication	Dimethylformamide (DMF)	kg	0.000944	/	0	N,N-dimethylformamide production	N,N-dimethylformamide	RoW	3.8735	0.0037
	electricity consumption	kWh	1.5	/	0	heat and power co-generation, natural gas, conventional power plant, 100MW electrical	electricity, high voltage	RoW	0.7256	1.0884
Magnetic bar	electricity consumption	kWh	0.05	/	0	heat and power co-generation, natural gas, conventional power plant, 100MW electrical	electricity, high voltage	RoW	0.7256	0.03628
Spray coating	electricity consumption	kWh	1.38889E-05	/	0	heat and power co-generation, natural gas,	electricity, high voltage	RoW	0.7256	1.01E-05

						conventional power plant, 100MW electrical				
Vacuum	liquid PDMS	kg	0.001	/	0	polydimethylsiloxane production	polydimethylsiloxane	GLO	21.6533	0.0217
	electricity consumption	kWh	0.7	/	0	heat and power co-generation, natural gas, conventional power plant, 100MW electrical	electricity, high voltage	RoW	0.7256	0.5079
Total	1.667120749									

Supplementary Table 5. Learned linear equations for nonlinearity issue.

Original Eqs	Correspond Fig.	Transformed Eqs	Correspond Fig.
$y_{\text{batch } 1^{st}} = \frac{47.817}{1 + e^{-0.174(x-23.346)}} + 238.839$	**Supplementary Fig. 10a**	$\begin{cases} V_{\text{batch } 1^{st}} = \frac{\text{Ln}\left(\frac{47.817}{y - 238.839} - 1\right)}{-0.174} \\ U = x \end{cases}$ $V_{\text{batch } 1^{st}} = U - 23.346$	**Supplementary Fig. 10b**
$y_{\text{batch } 2^{nd}} = \frac{41.073}{1 + e^{-0.192(x-22.924)}} + 239.780$		$\begin{cases} V_{\text{batch } 2^{nd}} = \frac{\text{Ln}\left(\frac{41.073}{y - 239.780} - 1\right)}{-0.192} \\ U = x \end{cases}$ $V_{\text{batch } 2^{nd}} = U - 22.924$	
$y_{\text{batch } 3^{rd}} = \frac{35.935}{1 + e^{-0.185(x-23.268)}} + 214.863$		$\begin{cases} V_{\text{batch } 3^{rd}} = \frac{\text{Ln}\left(\frac{35.935}{y - 214.863} - 1\right)}{-0.185} \\ U = x \end{cases}$ $V_{\text{batch } 3^{rd}} = U - 23.268$	

Supplementary Table 6. Comparison of sensor performance.

Ref.	Sensor Type	Sensor Active Materials	Reliable Lifetime Cycles (φ) <5.0%
Araromi, O. A. *et al. Nature*, 2020, **587**, 219–224[4].	Strain sensor	Carbon fiber polymer composites (CFPC)	<100 cycles
Cho, C. *et al. Nat. Electron.,* 2021, **4**, 126–133[5].	Strain sensor	Gold (Au) /two-layer graphene	<1,000 cycles
Jiang, Z. *et al. Nat. Electron.,* 2022, **5**, 784–793[6].	Strain sensor	Gold (Au)	<600 cycles
Wang, X. *et al. Adv. Mater.,* 2020, **32**, 2000351[7].	Strain sensor	Graphite /carbon nanotube (CNT) composite	<1,000 cycles
Wang, M. *et al. Nat. Electron.*, 2020, **3**, 563–570[8].	Strain sensor	Single-walled carbon nanotubes (SWCNTs)	<500 cycles
Yang, H. *et al. Nat. Commun.,* 2024, **15**, 1636[9].	Strain sensor	Single-walled carbon nanotubes (SWCNTs)	<5,000 cycles
Yang, H. *et al. Nat. Commun.*, 2022, **13**, 5311[10].	Strain sensor	MXene	<5 cycles
Shen, Z. *et al. Adv. Mater.,* 2022, **34**, 2203650[11].	Strain Sensors	PEDOT:PSS nanofibers	<1,000 cycles
Wang, C. et al. *ACS Nano,* 2023, **17**, 23194–23206[12].	Strain Sensors	PAMD-NaCl conductive hydrogel	<1,800 cycles
Lee, D. H. et al. *ACS Nano,* 2024, **18**, 32255–32265[13].	Strain Sensors	Laser-induced graphene decorated with ZnO NPs	<2,000 cycles
This Work	Strain sensor	Carbon-waste	>20,000 cycles

Supplementary Movies.

Supplementary Movie 1. Flexible robot arm tracking. This video demonstrates the performance of a sensor-integrated flexible robotic arm during its swing motion while executing sorting/assembly tasks. The D_Former model utilizes real-time feedback from the integrated sensors to estimate the spatial position of the arm's tail gripper. By comparing the predicted trajectory with the actual one, the model's high efficiency in compensating for signal attenuation and signal batch inconsistency is fully validated, effectively reducing estimation errors and demonstrating that the developed ML tool significantly enhances the sustainability of sensor performance.

Supplementary movie 2. Soft quadruped robot trajectory estimating. This video demonstrates the practical application of a sensor-integrated soft quadruped robot performing navigation tasks on artificial terrain under pneumatic actuation. All legs are equipped with CW sensors for real-time monitoring of the robot's movement, while the D_Former model tracks the robot's main body position based on sensor feedback. By comparing the predicted trajectory with the actual trajectory, the video clearly validates that the model's predictions closely match the ground truth throughout the entire navigation process.

Supplementary References

1. Yang, H. *et al.* Automatic strain sensor design via active learning and data augmentation for soft machines. *Nat. Mach. Intell.* **4**, 84–94 (2022).
2. Amjadi, M., Pichitpajongkit, A., Lee, S., Ryu, S. & Park, I. Highly stretchable and sensitive strain sensor based on silver nanowire–elastomer nanocomposite. *ACS nano* **8**, 5154–5163 (2014).
3. Chun, S., Choi, Y. & Park, W. All-graphene strain sensor on soft substrate. *Carbon* **116**, 753–759 (2017).
4. Araromi, O. A. *et al.* Ultra-sensitive and resilient compliant strain gauges for soft machines. *Nature* **587**, 219–224 (2020).

5. Cho, C. *et al.* Strain-resilient electrical functionality in thin-film metal electrodes using two-dimensional interlayers. *Nat. Electron.* **4**, 126–133 (2021).

6. Jiang, Z. *et al.* A 1.3-micrometre-thick elastic conductor for seamless on-skin and implantable sensors. *Nat. Electron.* **5**, 784–793 (2022).

7. Wang, X. Q. *et al.* Somatosensory, light-driven, thin-film robots capable of integrated perception and motility. *Adv. Mater.* **32**, 2000351 (2020).

8. Wang, M. *et al.* Gesture recognition using a bioinspired learning architecture that integrates visual data with somatosensory data from stretchable sensors. *Nat. Electron.* **3**, 563–570 (2020).

9. Yang, H. *et al.* Computational design of ultra-robust strain sensors for soft robot perception and autonomy. *Nat. Commun.* **15**, 1636 (2024).

10. Yang, H. *et al.* Topographic design in wearable MXene sensors with in-sensor machine learning for full-body avatar reconstruction. *Nat. Commun.* **13**, 5311 (2022).

11. Shen, Z. *et al.* High-stretchability, ultralow-hysteresis conducting polymer hydrogel strain sensors for soft machines. *Adv. Mater.* **34**, 2203650 (2022).

12. Wang, C. *et al.* High-saline-enabled hydrophobic homogeneous cross-linking for extremely soft, tough, and stretchable conductive hydrogels as high-sensitive strain sensors. *ACS Nano* **17**, 23194–23206 (2023).

13. Lee, D. H., Miyashita, T., Xuan, Y. & Takei, K. Ultrasensitive and stretchable strain sensors based on laser-induced graphene with ZnO nanoparticles. *ACS Nano* **18**, 32255–32265 (2024).