Physics-Informed Recurrent Neural Networks for Soft Pneumatic Actuators

Wentao Sun, Nozomi Akashi, Yasuo Kuniyoshi, *Member, IEEE*, and Kohei Nakajima

***Abstract*—Replacing sensors with indirect sensing techniques contributes to retaining the flexibility of soft robots. By combining physical models with recurrent neural networks (which we term a physics-informed recurrent neural network [PIRNN] approach), we implemented a hybrid prediction scheme on two typical soft pneumatic actuators: a McKibben pneumatic artificial muscle and a pneumatic-based soft finger made of silicone. The results showed that this hybrid scheme robustly enhanced the prediction accuracy to a great extent, even when combined with an inaccurate physical model. We also present the broad applicability of the PIRNN approach, showing its effectiveness for diverse types of RNNs and soft robotics platforms. Our work fills the gaps in the literature by applying a physics-informed machine-learning approach to practical engineering problems in soft robotics.**

***Index Terms*—Soft robot applications, soft sensors and actuators, modeling, control, and learning for soft robots.**

I. Introduction

ROBOTICS has far-reaching potential that can replicate and even extend human actions, freeing us from tiresome, demanding, and dangerous work. Relatively simple physical models and equations of motion allow rigid robots to operate with high accuracy, high speed, and a heavy payload. As a result, most industrial robots today rely on rigid materials. However, rigid robots are not flexible or safe enough to interact with certain environments or humans [1], [2]. Inspired by how animals exploit the deformability of soft structures to adapt to complex natural environments, engineers incorporated soft technologies into robotic designs [3]. For instance, many artificial soft limbs are now in the experimental stage [4]–[6].

A direct way to identify the shape of soft robots is using sensors that measure changing variables (e.g., curvature, deformation, and stress) when the soft robots move. However, installing sensors could impair the softness of the system and alter the robots' mechanical characteristics and functionality [7]. Thus, there are numerous benefits to replacing sensors and indirectly predicting the state of the soft robots based on the input sequence, freeing the platform from mechanical constraints [8], which is called indirect sensing.

One of the biggest issues in input-driven prediction for soft robots is finding methods and models that are both accurate to experimental values and computationally viable. We are currently unable to find exact physical models for soft robots because of the time-dependent nonlinear dynamics of soft material and the high dimensionality in mechanics. Most work is based on equations of motion with several error-correction models and has adopted empirical or semi-analytical approaches [9], which makes the modeling work convoluted, cumbersome, and not applicable for different robots. Although finite element analysis techniques based on such physical models have been widely used for simulating soft robots' dynamics, there is an exponentially increasing computational burden for increasingly accurate models [1].

Machine learning-based approaches provide an encouraging alternative to traditional models in the soft robotics field, as they can be more effective in solving problems related to nonlinearity [10]. These approaches are still in the early stages of development. It is often reported that simply enlarging the neural network size has limited returns on the accuracy of prediction while causing the computational cost to explode [11]. Furthermore, data-driven neural networks usually require a huge amount of training data, which are often difficult to collect on hardware platforms [12]. This situation is more crucial when it is applied to physical soft robotics platforms. Compared to rigid materials, the soft materials change over time and will therefore perform differently.

To address the deficiencies in the physical and machine learning-based models for soft robots, the physics-informed neural network (PINN) presented by Raissi *et al.* [13] is a promising solution that has been demonstrated through a collection of classical problems in fluids, quantum mechanics, and reaction–diffusion systems. The main idea of the PINN is to exploit the information from physical models with neural networks, i.e., taking the numerical results from physical models as the external input dimensions of the neural networks [14]. This learning scheme significantly reduces the computational load compared to training the network from scratch. The PINN is compatible with any underlying physical laws that govern a given dataset; however, the practical application of PINN in engineering is still unexamined.

Manuscript received February 24, 2022; accepted April 28, 2022. Date of publication May 27, 2022; date of current version June 6, 2022. This letter was recommended for publication by Associate Editor D. Aukes and Editor Y.-L. Park upon evaluation of the reviewers' comments. This work was partially based on the results obtained from a project commissioned by the New Energy and Industrial Technology Development Organization (NEDO). The work of Kohei Nakajima was supported in part by JSPS KAKENHI under Grant JP18H05472 and in part by JST CREST under Grant JPMJCR2014. *(Corresponding author: Kohei Nakajima.)*

Wentao Sun, Yasuo Kuniyoshi, and Kohei Nakajima are with the Graduate School of Information Science and Technology, University of Tokyo, Tokyo 113-8656, Japan (e-mail: sun@isi.imi.i.u-tokyo.ac.jp; kuniyosh@isi.imi.i.u-tokyo.ac.jp; k-nakajima@isi.imi.i.u-tokyo.ac.jp).

Nozomi Akashi is with the Graduate School of Science, University of Kyoto, Kyoto 606-8501, Japan (e-mail: akashi.nozomi.84z@st.kyoto-u.ac.jp).

Digital Object Identifier 10.1109/LRA.2022.3178496

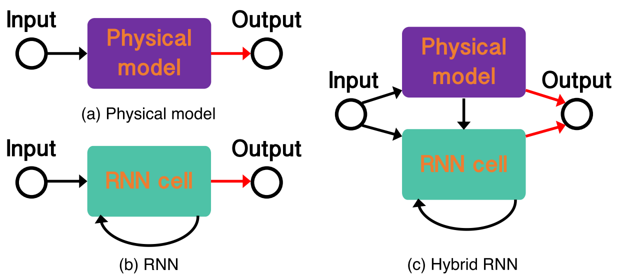


Fig. 1. The information flow for (a) the physical model, (b) the RNN model, and (c) the hybrid RNN model. In (c), RNN uses responses from the physical model as additional inputs.

We went into this unexplored area of the practical soft robots in our recent studies to demonstrate that physics-informed approach in terms of switching readouts significantly enhanced the prediction accuracy [15], [16]. In the current study, we modified a hybrid forecasting scheme proposed by Pathak *et al.* [17], which combines a knowledge-based model and a machine learning technique. Fig. 1 shows the schematic of the hybrid scheme. In their previous work, Pathak *et al.* used an imperfect Lorenz equation as the physical model to predict the time sequence of a correct Lorenz system. In the current paper, we used this hybrid scheme in a realistic mechanical system, using a physical model as the knowledge base. We combined a simple and imperfect knowledge-based model of soft actuators with three state-of-the-art recurrent neural network (RNN) models—the echo state network (ESN), the long short-term memory (LSTM) and the gated recurrent unit (GRU) models—in this hybrid scheme to predict the readouts of sensors for two different soft pneumatic actuators. The actuators were a pneumatic artificial muscle (PAM) and a pneumatic-based soft finger (PSF), which exhibit typical behaviors of soft actuators including stretching and bending. We refer to this network architecture as the *physics-informed recurrent neural networks (PIRNN)*.

Regarding to the accuracy of the predicted sequence in both PAM test and PSF test, our results show that all three hybrid predictors (knowledge-based model combined with ESN, LSTM, and GRU) significantly outperformed both the physical model-based predictor and the RNN predictors (ESN, LSTM, and GRU), even with a much larger network size.

The rest of letter paper is organized as follows. In Section II, we introduce the two typical soft pneumatic actuators and the datasets of the sensors used in the test. In Section III, we provide an overview of the three methods for predicting the readouts of the sensors. In Section IV, we describe how we designed the experiments and processed the data. In Section V, we demonstrate our hybrid approaches as a PIRNN in three RNN models and two pneumatic actuators. We also try to reveal the connections between different models.

II. Related Works and Datasets

We experimented on the open-source datasets from two real robotics platforms: readouts of the sensors from a PAM platform and a PSF platform. This section introduces the two platforms and the sensors in detail. Note that since our motivation in this paper is to introduce a novel machine learning architecture, it is important to demonstrate our approach using open-source datasets so that everyone can fairly compare the performance and understand the range of validity of our approach.

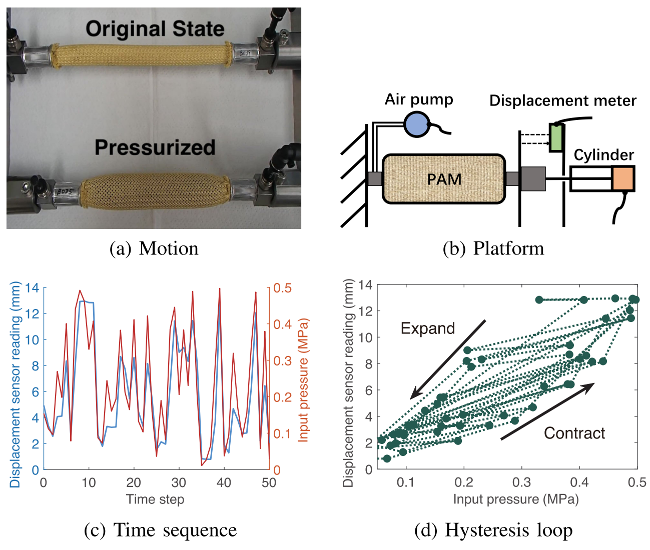


Fig. 2. Pneumatic artificial muscle.

A. Pneumatic Artificial Muscle

Technically speaking, the McKibben PAM is essentially an elastomer (e.g., rubber) inner tube surrounded by a double-helix-braided cord sheath [18]. It contracts in length when pressurized, as shown in Fig. 2(a).

We used the open-source dataset from Akashi *et al.* [19]. As Fig. 2(b) shows, the pressure in the PAM was controlled by an air pump and measured by a pressure sensor. A laser displacement meter was implemented to measure the axial deformation of the PAM.

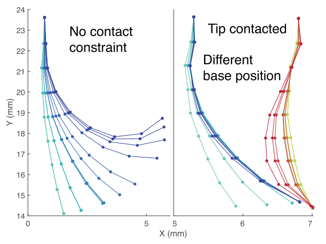
The load of the PAM was fixed at 100 N. The pressure signal was randomly generated in the range of $[0, 0.5]$ MPa. Fig. 2(c) shows the control signal (the input) and the readout of the displacement sensor (the output). Fig. 2(d) shows the hysteresis loop of the PAM.

B. Pneumatic-Based Soft Finger

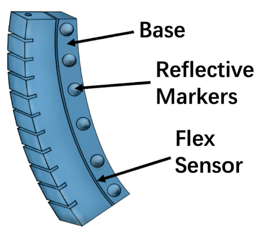
The PSF is a soft actuator consisting of a series of pleated embedded channels and chambers and an inelastic base layer. When pressurized, the channels and chambers inflate, and the whole finger bends, as shown in Fig. 3(a) [9].

We used the open-source dataset from Loo *et al.* [7]. The PSF was controlled by the pressure signal. As Fig. 3(b) shows, the resistance of the embedded flex sensor was influenced by the bending angle of the PSF and converted to voltage readings in the circuit.

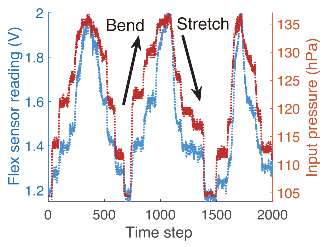
The PSF was tested with and without contact constraint on the tip. The horizontal distance between the base of the PSF and the constraint ranges from $[0, 2]$ mm in the datasets to



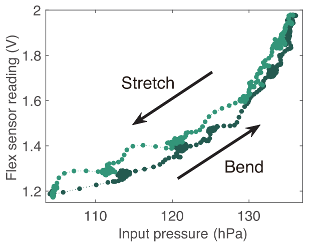
(a) Motion



(b) Platform



(c) Time sequence



(d) Hysteresis loop

Fig. 3. Pneumatic-based soft finger.

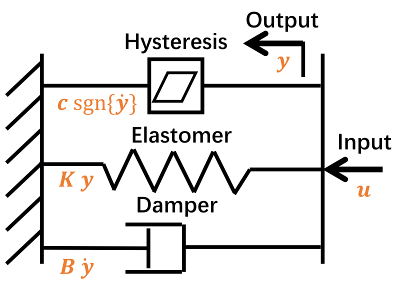


Fig. 4. Physical model of the soft robots.

test the response of PSF under different contact conditions. We introduced the base position of the PSF as an extra input into the neural networks in the PSF test with the contact constraint. The pressure was a stepped quasi-periodic signal. Fig. 3(c) shows the readout sequence of the pressure sensor (the input) and the flex sensor (the output). Fig. 3(d) shows the hysteresis loop in a bending and stretching cycle.

III. Prediction Methods

In this section, we discuss A) a simple and universal physical model of the soft pneumatic actuators, B) three RNN models for predicting the state of the pneumatic actuators, and C) how we combined three such models to make a PIRNN. The information flow of the three kinds for models is shown in Fig. 1.

A. Physical Model

The elastic force of the soft materials represents the static characteristics of the soft robots, and we added an elastomer into the model. Viscous and Coulomb friction are strongly related to the dynamic characteristics of the soft robots [7], [20], [21]. The former is rate dependent, and we use a damper to model it. The latter is rate independent, and we simplified it to a Preisach model of hysteresis.

As Fig. 4 shows, to combine the static and dynamic characteristics of the soft robots, we used a first-order linear system with a rate-independent hysteresis as a physical model of the soft robots:

$$B\dot{y} + K(y - y_0) + c \cdot \mathrm{sgn}\{\dot{y}\} = u \tag{1}$$

where u and y are the input and output of the system, respectively; B is the damping factor; K is the proportionality constant; and c is the Coulomb friction.

We used the explicit Euler method to predict the output in a discrete-time sequence:

$$\hat{y}'_t = \frac{u_t - c \cdot \mathrm{sgn}\{y_t - y_{t-1}\} + Bt^{-1} y_{t-1}}{Bt^{-1} + K} \tag{2}$$

where t is the time interval, and $\hat{y}'_t$ is the predicted output from the physical model of the soft robots.

Our algorithm also introduces a stiction, the maximum value of which was equal to the Coulomb friction.

B. Recurrent Neural Networks

RNNs were developed to capture long-term dependencies in sequential data [11]. RNNs are considered a suitable network for replacing sensors because their temporal dynamic behavior is designed for time-dependent tasks. The RNN model has been implemented widely in the prediction of the state of soft robots [8], [22].

We used an Elman RNN to predict the output of the target sensor. The time evolution of the RNN is given by:

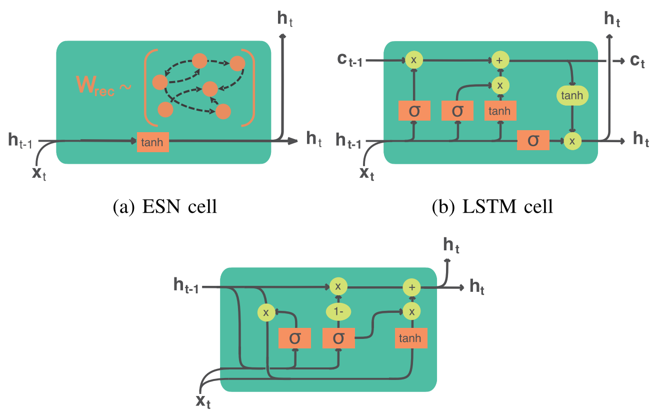
$$\mathbf{x_t} = [u_t],\ \mathbf{h_t} = f(\mathbf{x_0}, \ldots, \mathbf{x_t}),\ \hat{y}_t = W_{out}[1;\ \mathbf{h_t}] \tag{3}$$

where u_t and $\hat{y}_t$ are the input and predicted output at timestep t, respectively; $\mathbf{h_t}$ is the hidden state; and W_{out} is the hidden-to-output weights. The recurrent mapping f can be presented as different RNN cells. Fig. 5 shows the three types of RNN cells used in this letter.

The ESN is a state-of-the-art reservoir computing model. It is conceptually simple and computationally inexpensive, which meets the requirement of physical implementations for edge computing [23]–[25]. The time evolution of the ESN cell is given by:

$$\mathbf{h_t} = \tanh(W_{in}\mathbf{x_t} + W_{rec}\mathbf{h_{t-1}}) \tag{4}$$

where W_{in} and W_{rec} are randomly generated according to certain restrictions. The input scale σ controls the elements in W_{in} that are randomly generated from the uniform distribution $[-\sigma, \sigma]$. The internal weight matrix W_{rec} is a random sparse matrix with degree 3 (average number of connections in the adjacency matrix of the reservoir), which is computationally efficient and also accurate when compared with the full-connected matrix. The spectral radius ρ is defined as the largest magnitude eigenvalue of W_{rec}.



(a) ESN cell (b) LSTM cell

(c) GRU cell

Fig. 5. The information flow for the (a) ESN, (b) LSTM, and (c) GRU cells. Rounded rectangles and circles denote entry-wise operations, while rectangles denote layer operations. *The schematics are modified from Guillaume Chevalier, CC BY-SA 4.0, via Wikimedia Commons.*

Since the W_{in} and W_{rec} were not trained, we could directly calculate the weights of the output layers by ridge regression:

$$W_{out} = (X^T X + \lambda I)^{-1} X^T Y \tag{5}$$

where X is the design matrix, Y is the target matrix, and λ is the coefficient of regularization.

The LSTM was introduced in order to alleviate the vanishing gradient problem of Elman RNN [26]. The GRU, as a variant of the LSTM, has a more simplistic structure [27]. These two types of RNN cells have been widely used for time series predictions in soft robotics [7], [9].

The time evolution of the LSTM cell is:

$$\begin{aligned}
\mathbf{f_t} &= \sigma(W_f \mathbf{x_t} + U_f \mathbf{h_{t-1}} + \mathbf{b_f}) \\
\mathbf{i_t} &= \sigma(W_i \mathbf{x_t} + U_i \mathbf{h_{t-1}} + \mathbf{b_i}) \\
\widetilde{\mathbf{c}}_\mathbf{t} &= \tanh(W_c \mathbf{x_t} + U_c \mathbf{h_{t-1}} + \mathbf{b_c}) \\
\mathbf{c_t} &= \mathbf{f_t} \odot \mathbf{c_{t-1}} + \mathbf{i_t} \odot \widetilde{\mathbf{c}}_\mathbf{t} \\
\mathbf{o_t} &= \sigma(W_o \mathbf{x_t} + U_o \mathbf{h_{t-1}} + \mathbf{b_o}) \\
\mathbf{h_t} &= \mathbf{o_t} \odot \tanh(\mathbf{c_t})
\end{aligned} \tag{6}$$

where $\mathbf{f_t}$, $\mathbf{i_t}$, and $\mathbf{o_t}$ are the forget, input, and output gates, respectively; $\mathbf{x_t}$ is the input; $\mathbf{h_t}$ is the hidden state; and $\mathbf{c_t}$ is the cell state. Weight matrices W_f, W_i, W_c, and W_h and biases $\mathbf{b_f}$, $\mathbf{b_i}$, $\mathbf{b_c}$, and $\mathbf{b_o}$ were trained using the backpropagation through time (BPTT) algorithm.

The time evolution of the GRU cell is:

$$\begin{aligned}
\mathbf{z_t} &= \sigma(W_z[\mathbf{h_{t-1}}, \mathbf{x_t}] + \mathbf{b_z}) \\
\mathbf{r_t} &= \sigma(W_r[\mathbf{h_{t-1}}, \mathbf{x_t}] + \mathbf{b_r}) \\
\widetilde{\mathbf{h}}_\mathbf{t} &= \tanh(W_h[\mathbf{r_t} \odot \mathbf{h_{t-1}}, \mathbf{x_t}] + \mathbf{b_h}) \\
\mathbf{h_t} &= (1 - \mathbf{z_t}) \odot \mathbf{h_{t-1}} + \mathbf{z_t} \odot \widetilde{\mathbf{h}}_\mathbf{t}
\end{aligned} \tag{7}$$

where $\mathbf{x_t}$ is the input; $\mathbf{z_t}$ is the update gate vector; $\mathbf{r_t}$ is the reset gate vector; and $\widetilde{\mathbf{h}}_\mathbf{t}$, $\mathbf{h_t}$ is the hidden state. Weight matrices W_z, W_r, and W_h, and biases $\mathbf{b_z}$, $\mathbf{b_r}$, and $\mathbf{b_h}$ were trained using the BPTT algorithm.

We used the MATLAB deep-learning toolbox to train the LSTM and GRU models using the minibatch BPTT algorithm with Adam optimizer. Ridge regression on ESN and hybrid ESN was also implemented with MATLAB.

C. *Hybrid Scheme*

We modified Pathak *et al.*'s knowledge-based model (which they referred to as a hybrid prediction scheme) to incorporate the physical model into RNNs [17]. Their work involved the ESN model and closed-loop prediction for mathematical models. We extended the hybrid scheme to the ESN, LSTM, and GRU models and open-loop prediction for the real sensor data of soft robots.

The hybrid scheme introduces the prediction from the physical model $\hat{y}'_t$ as an external dimension in both the input layer and the output layer of the RNN:

$$\mathbf{x_t} = [u_t;\ \hat{y}'_t],\ \hat{y}_t = W_{out}[1;\ \mathbf{h_t};\ \hat{y}'_t] \tag{8}$$

There were no changes in the RNN cells except for the dimensions of the input vector $\mathbf{x_t}$.

IV. Implementation

In this section, we discuss A) the evaluation method of the models, B) how we optimized the parameters to make a fair evaluation, and C) the experimental setup for the different tests.

A. *Evaluation*

We evaluated the performance of the models by calculating the normalized mean square error (NMSE) between the predicted outputs and the real readouts of the sensors:

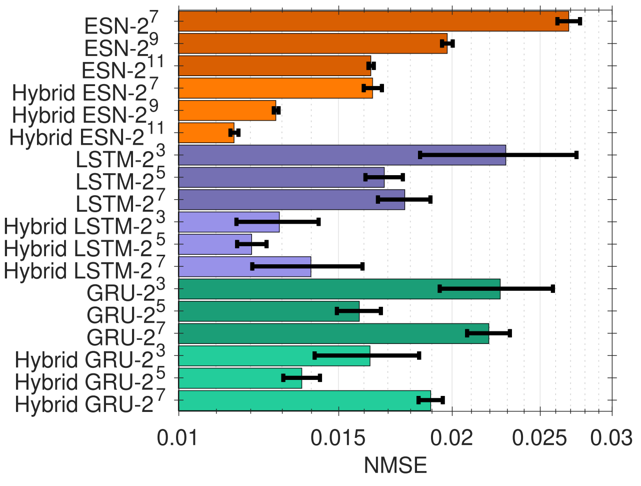
$$\text{NMSE} = \frac{\langle(\hat{y}_t - y_t)^2\rangle_t}{\langle(y_t - \langle y_t\rangle_t)^2\rangle_t} \tag{9}$$

where $\langle\cdot\rangle_t$ is the average of the variable through time t.

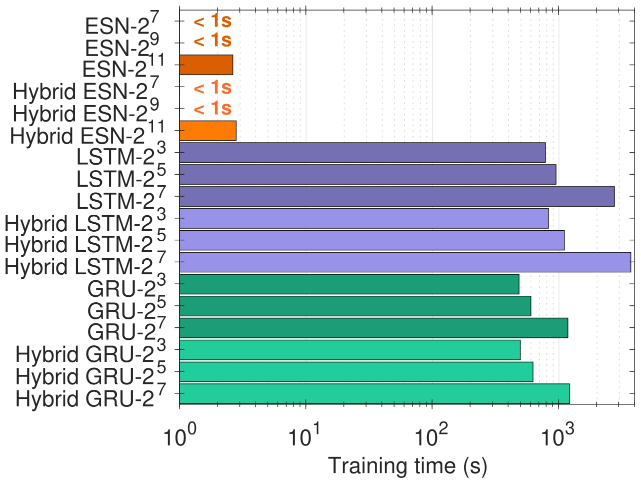
B. *Optimization of the Hyperparameters*

Although the performance of the ESN is affected by numerous hyperparameters [28], we noticed that researchers have rarely paid attention to the global implications of hyperparameters, i.e., when they changed one hyperparameter, they often fixed all of the other hyperparameters at a preset value. We suggest an optimization for the hyperparameters to exploit the potential of the networks and allow for a fair evaluation of the ESN and other RNN models.

In the PAM test, we focused on the dimensions of the hidden state $\mathbf{h_t}$ defined in (3), which we referred to as the hidden state size for the LSTM, the hybrid LSTM, the GRU, and the hybrid GRU cells. As for the ESN and hybrid ESN, we referred to it as the reservoir size per convention. We changed the reservoir size of the ESN cell and the hybrid ESN cell in a geometric sequence: 2^7, 2^9, and 2^{11}. For the LSTM, hybrid LSTM, GRU, and hybrid GRU cells, we changed the hidden state size in the sequence: 2^3, 2^5, and 2^7. In the PSF test, we were focused on the different



(a) NMSE in the test dataset



(b) Training time

Fig. 6. Bayesian optimization was repeated 10 times for each RNN model and hybrid RNN model to calculate (a) the average NMSE in prediction (bar plot) and its standard deviation (error bar) of the optimized models and (b) the average training time of one iteration in Bayesian optimization. Specifically, the numeric suffixes of the model name are the hidden state size and reservoir size of the RNN cell.

TABLE I
RANGE AND METHOD OF THE BAYESIAN OPTIMIZATION, INCLUDING THE SPECTRAL RADIUS ρ, INPUT SCALE σ, INITIAL LEARNING RATE γ, BATCH SIZE m, DAMPING FACTOR B, AND THE COULOMB FRICTION c. THE TEMPORARY VARIABLE a WAS GENERATED IN THE GIVEN RANGE AND THEN ASSIGNED TO THE PARAMETERS FOR THE GIVEN METHOD

Parameter	$\rho^{(*)}$	$\sigma^{(*)}$	$\gamma^{(\dagger)}$	$m^{(\dagger)}$	$B^{(\S)}$	$c^{(\S)}$
Range	$(0, 1.1)$	$(-1, 1)$	$(-4, 0)$	$[0, 10]$	$[0, 1]$	$[0, 0.3]$
Method	a	10^a	10^a	2^a	a	a

$(*)$ ESN and hybrid ESN models.
$(\dagger)$ LSTM, GRU, hybrid LSTM, and hybrid GRU models.
$(\S)$ Physical model.

load conditions of the PSF, so we fixed the reservoir size at 2^{11} for the ESN and hybrid ESN cells and the hidden state size at 2^7 for the other RNN cells. In each hidden state size and reservoir size setting, we conducted Bayesian optimization for the other prominent hyperparameters, which has been demonstrated to be more efficient than traditional grid optimization or random optimization [29]. Bayesian optimization algorithm attempted to minimize the NMSE in test dataset after each training process in bounded hyperparameter domains we set as shown in Table I. Bayesian optimization was implemented in *Experiment Manager* in MATLAB with default settings.

For the physical model, we conducted a linear fitting between the prediction from (2) and the target to get the y_0. Because the proportional hyperparameters K, B, and c led to the same prediction, we fixed K to 1 and optimized the other two hyperparameters B and c.

C. Experimental Setup

To train the different models, we normalized the input and output time series in the datasets to a range of $[0, 1]$. For the prediction results presented in Section V, the time series are rescaled to the original range. In both the PAM and PSF tests, the lengths of the training, evaluating, and wash-out sequences were 12000, 3000, and 300, respectively. The number of epochs in the BPTT was 1000. For each model we tested, Bayesian optimization was implemented for 160 iterations and repeated 10 times to evaluate the performance of the prediction. Hyperparameters not mentioned in the current paper were initialized with the default settings in MATLAB.

V. RESULTS AND DISCUSSION

We acquired the best hyperparameter settings and weights for each RNN cell with different hidden state sizes using Bayesian optimization. In Part A, we present the result of the PAM test and demonstrate the enhancement of the hybrid scheme with different RNN cells. In Part B, we present the result of the PSF test with different contact conditions to demonstrate the broad applicability of the hybrid scheme on the indirect sensing of soft robots. In Part C, we reveal some of the connections between the physical model, the RNN model, and the hybrid RNN model.

A. PAM Test

Fig. 6(a) presents a plot of the NMSE values of different RNN cells in the PAM test. We also tested the physical model exclusively with B and c optimized, and the NMSE of the optimized physical model was 0.0377. We found that the hybrid scheme enhanced the performance of the original RNN models and the physical model to a large extent for all of the neural network settings. LSTM, GRU, and their hybrid forms trained by BPTT each had a prediction accuracy comparable to ESN with a much larger reservoir size. However, training LSTM, GRU, and their hybrid forms with the BPTT algorithm took much longer than training ESN, even with a large reservoir size, as shown in Fig. 6(b).

For larger hidden state sizes in the RNN cells, we need a greater number of epochs in the BPTT training to make the network converge and to get good prediction accuracy. We tested the LSTM, GRU, and their hybrid forms with 100 epochs.

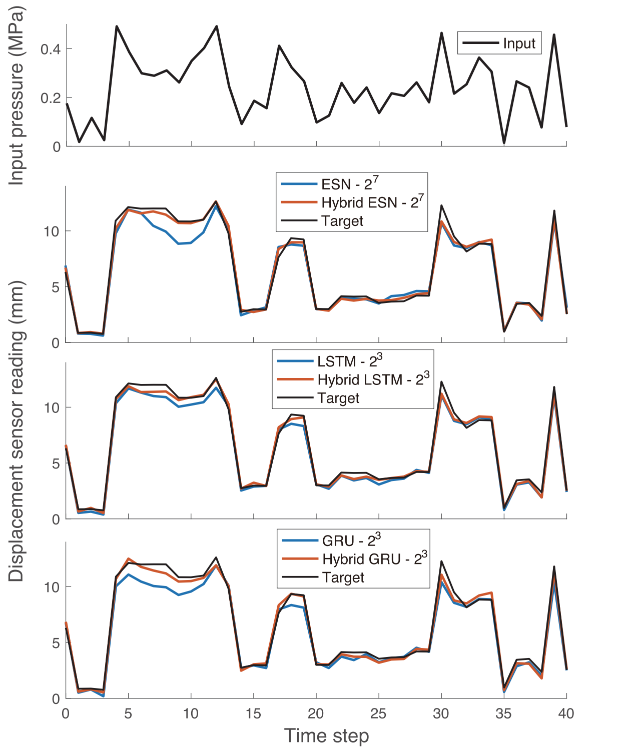


Fig. 7. Predicted displacement of different RNN cells in the PAM test. Numeric suffixes are the hidden state sizes and reservoir sizes.

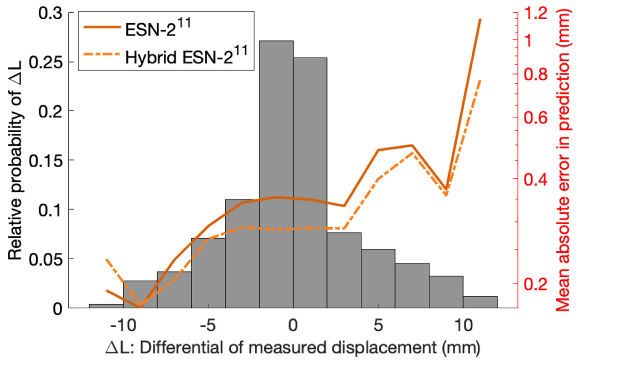
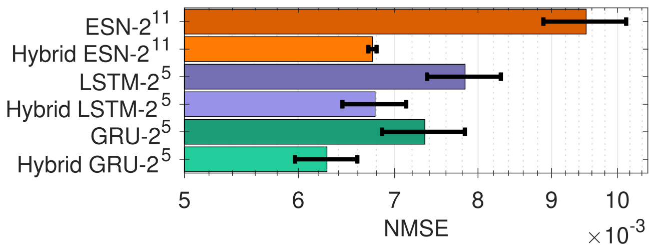
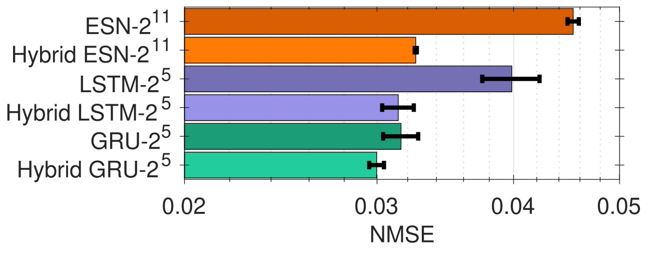


Fig. 8. Distribution of the differential of the measured displacement and the MAE of prediction for each interval.



(a) No contact constraint



(b) Tip contacted constrained

Fig. 9. Bayesian optimization was repeated 10 times for the different models to calculate the average NMSE in the prediction (bar plot) and its standard deviation (error bar) of the optimized models. (a) No contact constraint in PSF. (b) The tip of the PSF was contact constrained.

The prediction accuracy of the LSTM, GRU, and their hybrid forms with a hidden state size of 2^3 outperformed those with hidden state sizes of 2^5 and 2^7. Thus, it is not sufficient to train LSTM, GRU, and their hybrid forms with 1000 epochs in out experimental setting, which explains the inferior performance of LSTM-2^7, hybrid LSTM-2^7, GRU-2^7, and hybrid GRU-2^7 shown in Fig. 6(a).

The results of our test align with those of Vlachas *et al.* [11], shwoing the limited prediction enhancement in LSTM and GRU with large hidden state sizes. In both our tests and theirs, enlarging the reservoir size of ESN reliably increased the prediction accuracy while keeping the training time low, however, the memory cost would explode, which is not acceptable in edge computing.

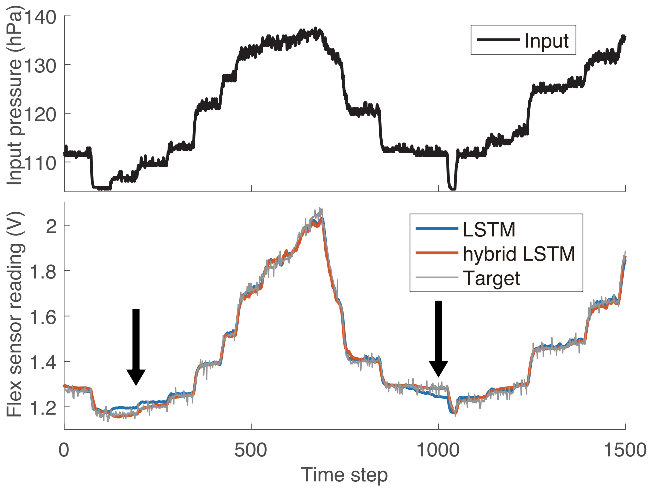
Fig. 7 is a plot of the output sequences of the target and prediction of different RNN models and their hybrid forms. We found that our hybrid scheme played an important role in predicting the hysteresis, i.e., when the input changed direction, the hybrid RNN prediction was more accurate than that of the original RNN.

Fig. 8 presents a plot of the distribution of the differential of measured displacement ΔL together with the mean absolute error (MAE) in ESN and hybrid ESN prediction for each ΔL interval. As a consequence of the Coulomb friction of soft materials, it is difficult to change the state of motion of PAM with small input variations, and the probability of ΔL being around 0 is relatively high. We found that the hybrid ESN worked much better than the original ESN when ΔL was close to 0, which means the physical information, i.e., damping effect and Coulomb friction from the physical model, helped in predicting the hysteresis and deadband of the soft robots. However, it is puzzling that the MAE was smaller when $\Delta L < 0$ (extension of PAM) than when $\Delta L > 0$ (contraction of PAM). We also observed this phenomenon for the predictions of the other RNN and hybrid RNN models, and we will explore its mechanism in future work.

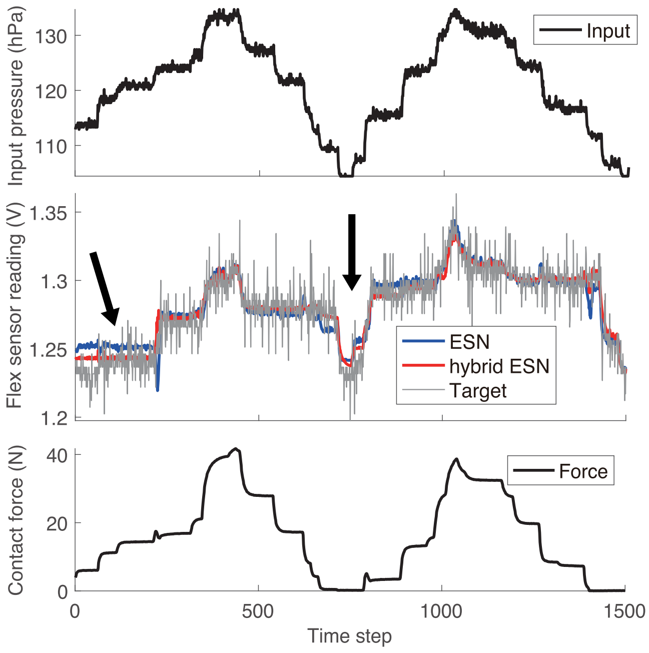
B. PSF Test

We first tested the physical model exclusively with B and c optimized, and the NMSE of the optimized physical model was 0.0969. This was inaccurate, as the NMSE of the linear fitting between the input and output was 0.1031.

We implemented the hybrid scheme in the PSF test using different contact conditions. We compared their performance with the original RNN model in Fig. 9(a), in which the PSF has no contact constraint, and in 9(b), in which the tip of the PSF



(a) No contact constraint



(b) Tip contacted constrained

Fig. 10. Predicted output sequence for the PSF test with (a) no contact constraint and (b) tip contact constrained.

was contact constrained. We found that the physical information was helpful for enhancing the prediction accuracy of the hybrid RNNs, even though the physical model was inaccurate in the PSF test. LSTM and GRU worked well for predicting the readout of the flex sensor in the PSF platform, and the enhancement of the hybrid scheme in these two models was not as significant as the enhancement in ESN.

Fig. 10 presents a plot of the predicted sequence and the target, including the contact force from the sensor in the contact constraint condition to indicate the contact condition of the PSF. Force information was not introduced in any model. The figure shows that the prediction in the hybrid scheme outperformed that of the original RNN for the different contact conditions of the PSF (see black arrows). The readout of the flex sensor jittered more intensely when the tip of the PSF was contacted constrained, contributing to a larger NMSE in the prediction.

TABLE II
OPTIMIZED HYPERPARAMETERS FOR THE DIFFERENT HYBRID RNN MODELS IN THE PAM TEST

Model	ρ	σ	B	c	NMSE	NMSE*
ODE	-	-	0.031	0.117	-	0.0381
Hybrid ESN-2^7	0.112	1.376	0.002	0.201	0.0164	0.0559
Hybrid ESN-2^9	0.139	2.088	0.000	0.208	0.0128	0.0614
Hybrid ESN-2^{11}	0.675	0.773	0.685	0.063	0.0115	0.6308
Model	γ	m	B	c	NMSE	NMSE*
Hybrid LSTM-2^3	0.032	7	0.023	0.100	0.0129	0.0446
Hybrid LSTM-2^5	0.013	15	0.000	0.130	0.0120	0.0652
Hybrid GRU-2^3	0.038	31	0.010	0.138	0.0163	0.0399
Hybrid GRU-2^5	0.013	11	0.019	0.038	0.0137	0.1085

NMSE is for the prediction of the whole system.
NMSE* is for the prediction of the physical model.

When the tip of the PSF was contact constrained, the accuracy of the prediction was much lower than for the no contact constraint condition because the input pressure had no feedback on the contact information for the PSF platform. Obtaining a high prediction accuracy for PSF with contact constraint could be achieved by introducing contact information into the neural networks in the form of a physical quantity, e.g., the resistance of the PSF.

C. Connections Between the Models

To test if a better physical model would contribute to a better hybrid RNN model, we compared the optimized hyperparameters and the performance of different hybrid RNN models, as shown in Table II. We found that among these optimized hybrid RNN models, the prediction accuracy of the physical model in the hybrid scheme was high when the hidden state size and reservoir size were small. As the hidden state size and reservoir size increased, the physical model became less reliable. This result suggests that a good physical model is not always required to generate good performance in our approach.

VI. CONCLUSION

In this letter, we introduced the PINN approach to real engineering problems, i.e., indirect sensing in soft robots. We proposed an input-driven hybrid forecasting scheme, termed PIRNN, to predict the readouts of the sensors in two pneumatic actuators to meet the engineering requirements of soft robots.

The most valuable contribution of this letter is demonstrating the potential and broad applicability of the PINN to soft robots. We started from a simple physical model that matched multiple soft robots made of nonlinear material. We demonstrated that whether the physical model works efficiently or poorly (e.g., it worked well in the PAM test but inaccurately in the PSF test), the type of RNN (e.g., ESN, LSTM, or GRU) affects the performance of our scheme. Further, this enhancement is compelling. As shown in Fig. 6, the hybrid RNN outperformed the original RNN with the same hidden state size and reservoir size, while the training time was almost the same. This suggests that the hybrid scheme could reduce computational and memory costs to a great extent, as hybrid RNN guarantees the same performance while reducing the hidden state size and reservoir size. Overall, our PIRNN model is based on a simple and universal physical model of soft robots, which is suitable for multiple RNN models and provides high efficiency.

Our paper made some other important contributions. 1) Our results suggest the comparative strength of the ESN when applied in soft robotics and edge computing. The dynamics of soft robots may change after a long period of use, so we need to retrain the models or design adaptive networks. In our test, the training time of the ESN was much lower than that of the LSTM and the GRU, while the prediction accuracy was higher, as Fig. 6(b) shows. Besides, the relatively simple structure of ESN enables substantial modifications. So we recommended ESN and hybrid ESN as an effective predictor of soft robots. 2) We demonstrated how the hybrid RNN reorganizes the prediction of the physical model and showed that an effective physical model may not always lead to an effective PIRNN.

The application of applying the PINN in soft robotics is still new. We hope to realize indirect sensing and replace the rigid sensory devices on soft robots with input-driven models. Our future work aims to further improve and explain PINN models. Although our result showed that our hybrid scheme works for LSTM and GRU, there is still room for improvement. For example, 1000 epochs is not sufficient to train LSTM and GRU with a hidden state size of 128 to converge. Further, we could make use of a gradient descent technique to optimize the hyperparameters in the hybrid LSTM and hybrid GRU [30]. As for the contact condition in PSF, we trained it separately in this study. Since the input pressure had no feedback from the contact condition, we want to use other physical quantity, e.g., resistance, to enable an automatic discrimination of the contact condition in the network, which will be included in our future work.

References

[1] J. Walker *et al.*, “Soft robotics: A review of recent developments of pneumatic soft actuators,” *Actuators*, vol. 9, no. 1, 2020, Art. no. 3.

[2] G. Alici, “Softer is harder: What differentiates soft robotics from hard robotics,” *MRS Adv.*, vol. 3, no. 28, pp. 1557–1568, 2018.

[3] S. Coyle, C. Majidi, P. LeDuc, and K. J. Hsia, “Bio-inspired soft robotics: Material selection, actuation, and design,” *Extreme Mech. Lett.*, vol. 22, pp. 51–59, 2018.

[4] P. Polygerinos, Z. Wang, K. C. Galloway, R. J. Wood, and C. J. Walsh, “Soft robotic glove for combined assistance and at-home rehabilitation,” *Robot. Auton. Syst.*, vol. 73, pp. 135–143, 2015.

[5] Y.-L. Park, J. Santos, K. G. Galloway, E. C. Goldfield, and R. J. Wood, “A soft wearable robotic device for active knee motions using flat pneumatic artificial muscles,” in *Proc. IEEE Int. Conf. Robot. Automat.*, 2014, pp. 4805–4810.

[6] T. Nagaoka, Z. Mao, K. Takemura, S. Yokota, and J.-W. Kim, “ECF (electro-conjugate fluid) finger with bidirectional motion and its application to a flexible hand,” *Smart Mater. Struct.*, vol. 28, no. 2, 2019, Art. no. 025032.

[7] J. Y. Loo, Z. Y. Ding, V. M. Baskaran, S. G. Nurzaman, and C. P. Tan, “Robust multimodal indirect sensing for soft robots via neural network-aided filter-based estimation,” *Soft Robot.*, 2021, doi: 10.1089/soro.2020.0024.

[8] R. Sakurai *et al.*, “Emulating a sensor using soft material dynamics: A reservoir computing approach to pneumatic artificial muscle,” in *Proc. 3rd IEEE Int. Conf. Soft Robot.*, 2020, pp. 710–717.

[9] T. G. Thuruthel, B. Shih, C. Laschi, and M. T. Tolley, “Soft robot perception using embedded soft sensors and recurrent neural networks,” *Sci. Robot.*, vol. 4, no. 26, 2019, Art. no. eaav1488. [Online]. Available: https://www.science.org/doi/10.1126/scirobotics.aav1488

[10] D. Kim *et al.*, “Review of machine learning methods in soft robotics,” *Plos One*, vol. 16, no. 2, 2021, Art. no. e0246102.

[11] P. R. Vlachas *et al.*, “Backpropagation algorithms and reservoir computing in recurrent neural networks for the forecasting of complex spatiotemporal dynamics,” *Neural Netw.*, vol. 126, pp. 191–217, 2020.

[12] C. C. Johnson *et al.*, “Using first principles for deep learning and model-based control of soft robots,” *Front. Robot. AI*, vol. 8, 2021, Art. no. 654398.

[13] M. Raissi, P. Perdikaris, and G. E. Karniadakis, “Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations,” *J. Comput. Phys.*, vol. 378, pp. 686–707, 2019.

[14] G. E. Karniadakis, I. G. Kevrekidis, L. Lu, P. Perdikaris, S. Wang, and L. Yang, “Physics-informed machine learning,” *Nature Rev. Phys.*, vol. 3, no. 6, pp. 422–440, 2021.

[15] W. Sun, N. Akashi, Y. Kuniyoshi, and K. Nakajima, “Physics-informed reservoir computing with autonomously switching readouts: A case study in pneumatic artificial muscles,” in *Proc. Int. Symp. Micro-NanoMehatronics Hum. Sci.*, 2021, pp. 1–6.

[16] W. Sun, N. Akashi, Y. Kuniyoshi, and K. Nakajima, “Self-organization of physics-informed mechanisms in recurrent neural networks: A case study in pneumatic artificial muscles,” in *Proc. IEEE 5th Int. Conf. Soft Robot.*, 2022, pp. 409–415.

[17] J. Pathak *et al.*, “Hybrid forecasting of chaotic processes: Using machine learning in conjunction with a knowledge-based model,” *Chaos: An Interdiscipl. J. Nonlinear Sci.*, vol. 28, no. 4, 2018, Art. no. 041101.

[18] B. Tondu, “Modelling of the mckibben artificial muscle: A review,” *J. Intell. Mater. Syst. Struct.*, vol. 23, no. 3, pp. 225–253, 2012.

[19] N. Akashi *et al.*, “Input-driven bifurcations and information processing capacity in spintronics reservoirs,” *Phys. Rev. Res.*, vol. 2, no. 4, 2020, Art. no. 043303.

[20] W. Huang, X. Huang, C. Majidi, and M. K. Jawed, “Dynamic simulation of articulated soft robots,” *Nature Commun.*, vol. 11, no. 1, pp. 1–9, 2020.

[21] H. Chaoui, P. Sicard, and W. Gueaieb, “Ann-based adaptive control of robotic manipulators with friction and joint elasticity,” *IEEE Trans. Ind. Electron.*, vol. 56, no. 8, pp. 3174–3187, Aug. 2009.

[22] G. Soter, A. Conn, H. Hauser, and J. Rossiter, “Bodily aware soft robots: Integration of proprioceptive and exteroceptive sensors,” in *Proc. IEEE Int. Conf. Robot. Automat.*, 2018, pp. 2448–2453.

[23] M. Lukoševičius, “A practical guide to applying echo state networks,” in *Neural Networks: Tricks of the Trade*. Berlin, Germany: Springer, 2012, pp. 659–686.

[24] K. Nakajima, “Physical reservoir computing—an introductory perspective,” *Japanese J. Appl. Phys.*, vol. 59, no. 6, 2020, Art. no. 060501.

[25] K. Nakajima and I. Fischer, *Reservoir Computing—Theory, Physical Implementations, and Applications.* Berlin, Germany: Springer, 2021.

[26] S. Hochreiter, “The vanishing gradient problem during learning recurrent neural nets and problem solutions,” *Int. J. Uncertainty, Fuzziness Knowl.-Based Syst.*, vol. 6, no. 2, pp. 107–116, 1998.

[27] K. Cho, B. Van Merriënboer, D. Bahdanau, and Y. Bengio, “On the properties of neural machine translation: Encoder-decoder approaches,” in *Proc. SSST-8, 8th Workshop Syntax, Semantics Struct. Stat. Transl.*, Doha, Qatar: Association for Computational Linguistics, 2014, pp. 103–111.

[28] H. Jaeger, “The ‘echo state’ approach to analysing and training recurrent neural networks-with an erratum note,” German National Research Center for Information Technology GMD, Bonn, Germany, Tech. Rep., vol. 148, no. 34, p. 13, 2001.

[29] J. Snoek, H. Larochelle, and R. P. Adams, “Practical Bayesian optimization of machine learning algorithms,” in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 25, 2012, pp. 2951–2959.

[30] D. Maclaurin, D. Duvenaud, and R. Adams, “Gradient-based hyperparameter optimization through reversible learning,” in *Proc. Int. Conf. Mach. Learn.*, 2015, pp. 2113–2122.