IEEE Robotics and Automation Letters (RA-L) paper, presented at ICRA 2023, London, UK. Cite as RA-L paper.

DNN-based Predictive model for a Batoid-inspired Soft Robot

Guangtong Li, Thileepan Stalin, Van Tien Truong, and Pablo Valdivia y Alvarado

***Abstract*—Soft robots have a unique potential to harness advanced functionalities through materials engineering, chemistry, and advanced fabrication. However, modeling and control of soft robot bodies is challenging due to non-linearities and time-dependencies of materials physico-chemical properties. With the rapid development of artificial intelligence technologies, deep neural networks (DNN) have become an essential tool for exploring the relationships between inputs and outputs of challenging systems under complex environmental conditions. In this work, rather than physically modeling a soft robotic system, we treat the entire system, including its environment, as a complex but deterministic input-output system, and we use DNNs to estimate these relationships. As an application example, our training results show that DNNs can accurately simulate the physical properties of an underwater bio-inspired soft robot. Validation experiments show that measured propulsive forces are in good agreement with target values predicted by DNNs. Our experiments show the potential of using DNNs to accomplish rapid modeling of bio-inspired propulsion and facilitate control.**

***Index Terms*—Deep neural networks, Soft robot control, Bio-inspired locomotion**

I. Introduction

TRADITIONAL approaches for control of mechatronic systems usually combine a model of the system dynamics with algorithms that compensate for environmental disturbances and errors in the system predictions to regulate desired outputs. This approach has been successful with traditional robots, as their rigid body dynamics can be easily modeled, their non-linearities can usually be compensated, and their structures usually present a manageable number of degrees of freedom for online computation. In contrast, soft robots have highly compliant multi-material structures with non-linear time-dependent physico-chemical properties, and infinite numbers of degrees of freedom due to their continuum nature. Therefore, their body dynamics are more challenging to model [1]–[4]. Although a variety of approaches have been explored to control soft robots [5], [6], their typical large deformations and infinite number of degrees of freedom create difficulties for precise control [7]. Model-based control of soft robots with hysteresis and uncertain state interactions is a time consuming and tedious process [8]. The lack of technical and methodological guidance in soft robotics dynamic modeling makes building a valid model still a problematic task [9]. Models contain complex material and structural characteristics that can only be applied to specific tasks and are challenging to generalize [10] and small changes in both environment and structure can significantly affect the validity of previously derived models [11]. An additional challenge for soft robots is their inherent under-actuation, as actuation often only controls discrete sections and the overall system dynamics is heavily influenced by environment interactions. This property further makes soft robots hard to control in unstable environments. All these challenges make the control of complex movements in soft robots an open ended problem.

DNNs can adopt different network structures such as multi-layer perceptron (MLP), convolutional neural network (CNN), and recurrent neural networks (RNN) [12] among others. Long Short-Term Memory networks (LSTM) is a type of RNN and is widely used for time sequence data [13]. In this study, MLP and LSTM structures are used to build predictive models due to their simplicity in training and suitability for time series data. MLP are feed-forward neural networks trained with the standard back propagation algorithms [14] and have been the preferred neural network topology in various studies since no special assumptions on the data are needed, making them suitable for various applications [15], [16]. As one of the classical deep learning models [17], RNNs are a derivative of feed-forward neural networks that can use their internal state (memory) to process variable sequences of inputs [18], [19]. As a variant of RNNs, LSTM networks overcome RNNs lack of long-term memory, inability to handle long sequences [20], [21], and are extensively used in natural language processing [22].

These difficult-to-model dynamics are an ideal candidate for universal function approximations using deep neural networks (DNNs) [23], [24]. Various research groups have used neural networks and central pattern generators (CPG) to design and control robot locomotion and behavior. Hyatt et al [25] used learned non-linear discrete-time models to achieve accurate control in a six degree-of-freedom soft robot. Johnson et al

Manuscript received: August, 10, 2021; Accepted November, 22, 2021.

This paper was recommended for publication by Editor Cecilia Laschi upon evaluation of the Associate Editor and Reviewers' comments. This research project was supported by A*STAR under its Science and Engineering Research Council (SERC) Awards 1822500053 and W2025d0243, and by SUTD's Digital Manufacturing and Design (DManD) Centre, and SUTD's International Design Centre (IDC) under Grant Nos. RGDM1620401, RGMD1620501, and IDG31600101. G.Li was supported by SUTD's PhD Fellowship and T.Stalin was supported by SUTD's President's Graduate Fellowship. (Corresponding author: P. Valdivia y Alvarado)

G. Li and T. Stalin are with the Engineering and Product Development Pillar, Singapore University of Technology and Design, Singapore 487372 (e-mail: guangtong_li@mymail.sutd.edu.sg; thileepan_stalin@mymail.sutd.edu.sg)

V.T. Truong is with SUTD's Digital Manufacturing and Design (DManD) Centre, Singapore 487372 (e-mail: vantien_truong@sutd.edu.sg)

P. Valdivia Y Alvarado is with the Engineering and Product Development Pillar, Singapore University of Technology and Design, Singapore 487372, and also with SUTD's Digital Manufacturing and Design (DManD) Centre, Singapore 487372 (e-mail: pablov@sutd.edu.sg)

Digital Object Identifier (DOI): see top of this page.

IEEE Robotics and Automation Letters (RA-L) paper, presented at ICRA 2023, London, UK. Cite as RA-L paper.

[26] combined deep learning and physics-based analytical models to model a soft continuum manipulator. Kiguchi et al [27] used neural networks for modeling a rigid robotic manipulator and highlighted the efficiency of neural networks over traditional genetic programming (GP). Gay et al [28] used a neural network to build a model-free feedback controller that improved the locomotion and balance control of a quadruped robot. Thomas et al [29] developed dynamic models for a soft robotic arm to optimize its trajectory based on machine learning. Liu et al [30] used a Deep neural network to control the goal-reaching and tracking behaviors of soft snake robots. Nakajima et al [31]–[34] explored how the diverse dynamics generated by soft octopus arms can be effectively used for machine learning purposes with reservoir computing.

The main focus in this study is to investigate if a sequence of control inputs needed to precisely achieve a sequence of target propulsive forces of a soft robot in a rapidly changing environment can be predicted. To address this question, we consider a soft robot and its environment as a single system which we approximate using non-linear neural networks. This approach simplifies the modeling process and enables real time predictions that can be used for robot control.

As an application example, this study focuses on the use of deep learning tools to model the system dynamics of an under-actuated soft batoid robot [35]–[38]. The robot mimics the morphology and swimming kinematics of rajiform batoids. We demonstrate that DNNs can be accurately trained to predict required robot actuation input sequences to achieve specific desired propulsive force profiles.

The paper is organized as follows: Section II introduces the soft batoid-inspired robot platform as well as a description of the input control signal's selection for its actuation mechanism. The experiment setup and force data acquisition are described in Section III. Section IV presents the MLP and LSTM training methods and the validation of the predicted control signals. Finally, the results of this study and suggested future work are summarized in Section V.

II. Robot Platform

A. Soft batoid robot

Several research groups have developed soft bio-inspired robots to mimic natural locomotion [39]–[42]. In this paper, we use a soft batoid-like robot developed previously by our group (see Fig. 1) [38], [43]. The robotic platform consists of two flapping mechanisms with curvature control enabled by servomotors [44]. Major electronics including a microcontroller, a rechargeable battery, and a wireless charger are enclosed within an 3D-printed shell (see Fig. 1a). A non-permeable soft silicone skin (EcoFlex 00-30) covers the printed shell, protecting the robot mechanisms in aquatic environments, and forms two large under-actuated soft fins (see Fig. 1a). Each flapping mechanism is driven by a servomotor whose rotations force the flapper to bend up (during upstrokes) and down (during downstrokes). A flapper's frequency and angular position can be controlled through the servomotor's angular frequency and angular position respectively. Flapper motions excite fin oscillations which in turn produce the necessary forces and torques for underwater propulsion.

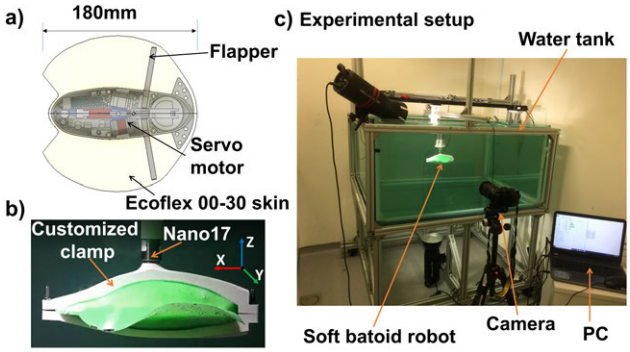

Fig. 1. Soft robot platform and experimental setup: (a) CAD model (top view) displaying robot internal components; (b) Side view showing the robot mounted to the 6-axis load cell; (c) Instrumented tank used for force measurements.

B. Control signal generation

In this study, the three component forces ($F_{x,out}$, $F_{y,out}$, $F_{z,out}$) and three torques ($T_{x,out}$, $T_{y,out}$, $T_{z,out}$) generated by an individual flapping fin are recorded to be used as data for training and predictions (see reference frame in Fig. 1b). Due to the fin's symmetrical arrangement, the generated lateral force ($F_{y,out}$) of each fin acts in the same plane but in opposite directions. Therefore, their net components mutually cancel if the fins are actuated in phase and with the same control inputs. The servomotors used in the robot have an active angular range of $\pi\ rad$ (i.e. they can rotate from 0 to $180deg$). When the servomotors are positioned at $90deg$, the flapper attached to them is in a horizontal resting position. During experiments, the maximum input angle range is set to $60deg$ (i.e $\pm 30deg$ from the horizontal resting position) to prevent the flapper from excessive bending. Each robot input sequence (servomotor commanded angular positions) has a series of 41 angular data points generated with alternating lower bound values of $[60-85deg]$ and upper bound values $[95-120deg]$ to produce fin flapping. Angular position values from $85-95deg$ are omitted as they will only produce small flapping amplitudes resulting in negligible forces. Each servomotor angular position command controls the fins to make a flapping motion (i.e. an up-stroke or a down-stroke). The last servomotor angular position (number 41) is always set to $90deg$ to bring the flapper back to its horizontal resting position. The input sequence is uploaded to the robot microcontroller (Bluno Nano microcontroller, DFrobot, CN) wirelessly via Bluetooth Dongle. In addition to the servomotor angular positions (θ_{input}), the robot input signal also contains angular velocity (ω_{input}) commands. The series of angle commands (θ_{input}) is in the format,

$$\theta_{input} = [119,\ 63,\ 114,\ 66,\ 105,\ 77,\ 111,\ .....] \quad (1)$$

The commanded servomotor angular velocities (ω_{input}) are defined as,

$$\omega_{input} = \frac{(\theta_{t+1} - \theta_t)}{\mathbf{T}} \quad (2)$$

where θ_{t+1} is the target servo motor angular position and θ_t is the current servo motor angular position. The time constant $\mathbf{T}$ is chosen to match the servo motor max speed and it ensures

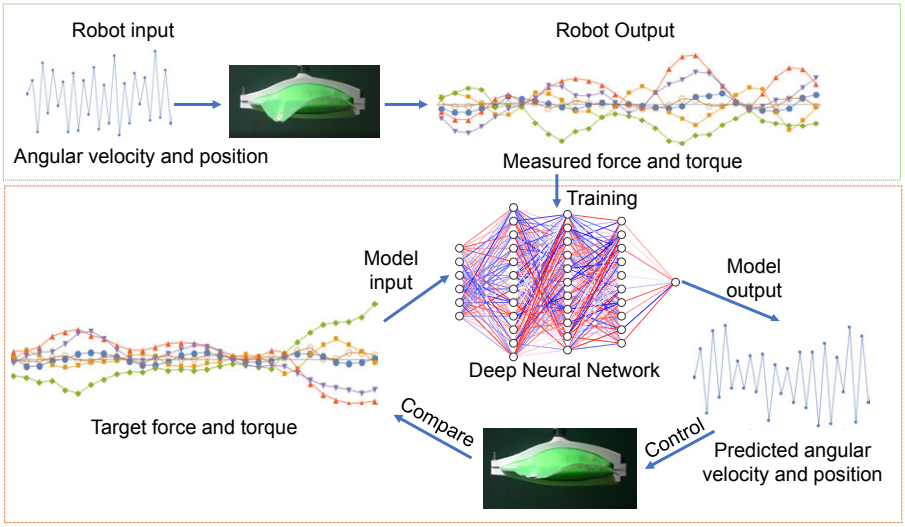

Fig. 2. Steps involved in the generation of predictive models for a batoid-inspired soft robot.

that each flapping motion uses the same time and stays within a safe speed range.

III. Experiment Setup

Experiments were carried out inside a water tank ($1.2m \times 1.2m \times 0.6m$) by fixing the robot to a 3D printed clamp connected to a 6-axis load cell (see Fig. 1b and 1c). The clamp was custom designed to hold the robot along its body length without interfering with its fin motions. The 6-axis force/torque sensor (Nano17, ATI Industrial Automation Inc, USA) attached to the clamp is used to measure the forces and torques generated during flapping. Data is recorded, using a single fin, at a sampling rate of $120Hz$ (ATI DAQ F/T software) which enables logging of roughly 29 force/torque (F/T) data points for each flapping motion.

Each experiment was carried out with 10 different robot input sequences. Each robot input sequence lasted for 9.9 seconds with a pause of 5 seconds between each sequence to ensure any wakes created inside the tank subsided and did not influence the force data acquisition. Different input signals were tested on the robot, and the collected F/T data was used for training and validation. In total, 10 experiments were conducted to collect 100 F/T data sets from 100 different robot input sequences.

IV. Model Training

A. Dataset generation

The creation of a dataset is the first step in performing neural network training, and the dimensions of the dataset determine the structure of the neural network. The entire concept used for the study is shown in Fig. 2. A total of 10 experiments were carried out and the collected F/T dataset can be represented as $\{X_{(0)}, ..., X_{(4100)}\}$, $X \in \mathbb{R}^{29\times6}$. The size of the overall dataset is $(4100, 29, 6)$, where 4100 represents the total angular data points for fin flapping for 10 experiments, 29×6 represents the 6-Dimensional F/T data $(F_x, F_yF_z, T_x, T_y, T_z)$ in an array of 29 elements between two successive angular data points. The dataset was sub-divided into a training set (6 experiments), a validation set (2 experiments), and a test set (2 experiments).

Fig. 3 shows the recorded F/T data. The presence of noise creates a harmful interference on the data-sensitive neural network (see Fig. 3a and 3c) as it would be treated as data features by the neural network model during training, which would cause random errors in the verification tests. A Fourier transform was performed on the acquired data to identify and remove noise above $30Hz$ (see Fig. 3b and 3d). The filtered data retains information of the system dynamics while eliminating noise. Neural network training results also show that noise cancellation improves model accuracy by about 15% and enhances the model's generalization (i.e. reduces random errors) so that the trained model displays similar performance with a training dataset and a validation dataset.

Force and torque signals have different units and their absolute values differ by a wide margin. This can be seen in Fig.3. Neural networks are more inclined to focus on the effects of variables with larger absolute values. Regularization is hence necessary for force and torque data to play an equally important role in the prediction task. Therefore, the entire dataset was reshaped from $(4100, 29, 6)$ to $(4100 \times 29, 6)$ and the force data was regularized using the relation,

$$X_{mnReg} = \frac{X_{mn} - X_{Mean}}{X_{Max} - X_{Min}} \tag{3}$$

where X_{mnReg} is the value of the m^{th} reading in the n^{th} dimension after regularization, $m \in [0, 1, 2, ..., 118900]$, $n \in [0, 1, 2, ..., 6]$. X_{Mean} is the mean value of the whole dataset in the n^{th} dimension, X_{Max} and X_{Min} are the maximum and minimum values of the whole dataset in the n^{th} dimension respectively. The regularization maps the force and torque values to a range between -1 and 1 and assigns them the same weight in model training.

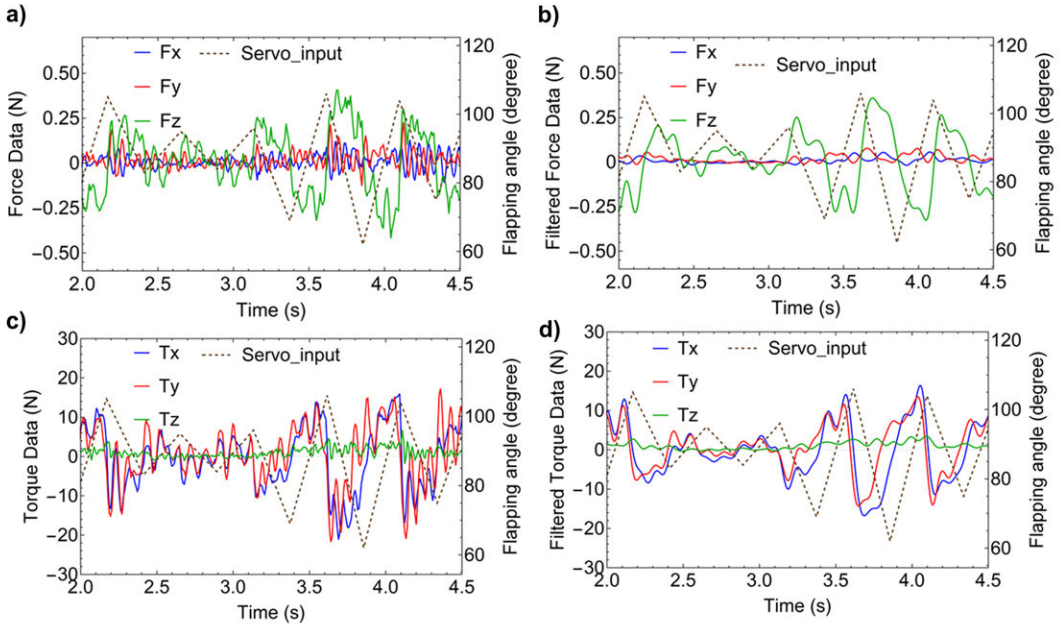

Fig. 3. Robot F/T Data: (a) Unfiltered force data; (b) Filtered force data (Low-pass filter with a $30Hz$ cut off frequency); (c) Unfiltered torque data; (d) Filtered torque data (Low-pass filter with $30Hz$ cut off frequency).

B. Deep Learning models

MLPs usually combine many different functions together. For example,

$$f(x) = f^{(m)}(f^{(m-1)}(f^{(\ldots)}(f^{(1)}(x)))) \quad (4)$$

indicates m functions ($f^{(1)}$ to $f^{(m)}$) connected in a chain to form $f(x)$ where m is the depth of the model. Each example x (input F/T data) is matched with a label y (output angular velocity). The training progress drives $f(x)$ to match target label y by analyzing the given x. In this study, the 29×6 6-Dimensional F/T data was chronologically flattened into a one-dimensional vector of length 174 as the model input, enabling the use of MLPs for this time-series data. LSTM is a sturdy structure used extensively in natural language processing due to its long-term memory capability. The collected force data is a time-series data, and it can be treated similarly to natural language. Therefore, LSTM was selected as one of the neural network models in this study. In this study, general MLP and LSTM models are built to predict the robot input signal, and their performance was compared based on training time and prediction accuracy.

C. Other settings

In terms of model size, there are 100 nodes in each hidden layer for the MLP and LSTM models. The size and ability of the model can be modified by adjusting the number of hidden layers. A deeper network structure helps in reducing the problem by using less parameters [45]. Compared with linear outputs, non-linear transformations can fit more complex models. Applying activation functions in the hidden layers can also help the network learn complex data and make sense of non-linear and complicated mappings between the inputs and corresponding outputs [46].

Currently, the Rectified Linear Unit (ReLU) is the most successful and widely-used activation function [47]–[49] as it proves to be helpful in deep neural networks to achieve minimal rates of convergence [50]. ReLU avoids and rectifies the vanishing gradient problem and is less computationally expensive than other popular activation functions such as TanH and Sigmoid [51]. In this study, ReLU was selected as the activation function in all the hidden layers,

$$ReLU(\chi) = max(0, \chi) \quad (5)$$

where χ is the input variable of the ReLU function. The loss $\mathcal{L}$ is used to describe the gap between the predicted value $f_\theta(\chi)$, and the target label ν. The loss is defined as,

$$\mathcal{L} = g(f_\theta(\chi), \nu) \quad (6)$$

In this task, the target is to minimize the average error in all the flapping motions, therefore a Mean Absolute Error (MAE) function was used as g to calculate $\mathcal{L}$,

$$MAE = \frac{1}{m}\sum_{i=1}^{m}|f_\theta(x_i) - y_i| \quad (7)$$

By learning from the training dataset $\mathbb{D}^{training}$, the neural network can get a group of parameters α^* to achieve minimum $\mathcal{L}$.

$$\alpha^* = \underbrace{\arg\min}_{\alpha} g(f_\alpha(x), y) \quad (8)$$

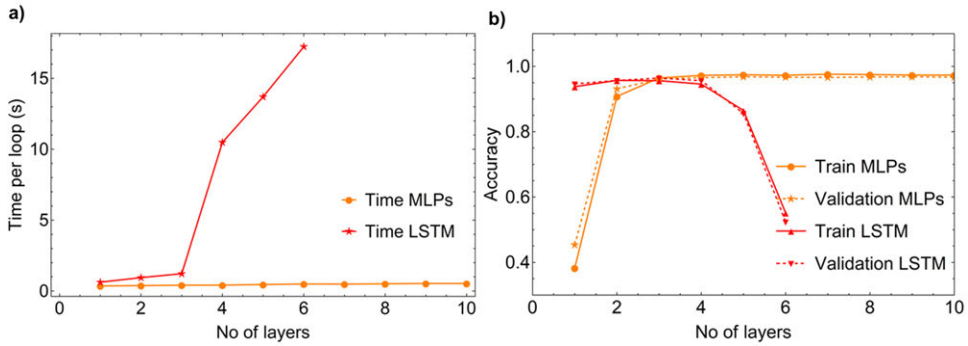

Fig. 4. Performance of MLP and LSTM networks: (a) Time taken for training the network versus number of hidden layers; (b) Training accuracy and validation accuracy of the neural network model versus number of layers.

Finally, to maintain a balance between training efficiency and training stability, Adam is chosen as the network's optimizer. Adam performs well in non-stationary objectives with noisy gradients and is computationally efficient [52].

V. Results

A series of target force and torque values are given as inputs to the MLP model to predict required angular positions and angular velocities (i.e servomotor commands). Subsequently, the predicted values were used to control the robot and verify whether the resulting measured forces and torques matched the target values.

A. Performance of MLP and LSTM networks

Prediction accuracies are found using the relation,

$$1 - \frac{1}{M}\sum_{1}^{M}\frac{|\omega_{pre} - \omega_{true}|}{|\omega_{true}|} \tag{9}$$

where ω_{pre} and ω_{true} are the predicted angular velocity and true angular velocity, and M is the size of the test dataset. The prediction accuracies for MLP and LSTM based networks are 96.8% and 95.7% respectively. MLPs can achieve high prediction accuracies with a wide range of layer numbers, enabling a simplified neural network design structure. In terms of training efficiency, MLP have faster convergence and shorter training time when compared to LSTM.

Fig. 4 shows the differences in prediction accuracy and training time between LSTM and MLP models. Both model types present good generalization abilities since there are no large differences between their training and validation losses. MLP networks with only one or two hidden layers have poor fitting capabilities due to the small number of parameters. When the number of hidden layers increases to more than three, the fitting ability becomes stable, and the prediction accuracy is high. When the parameter scale reaches a certain limit, increasing the number of hidden layers results in excess performance which cannot improve the model's accuracy. The training time for MLP networks remains small as the number of hidden layers increases.

Contrary to MLP network characteristics, LSTM networks have higher prediction accuracies when the number of hidden layers is small. With an increase in the number of hidden layers, the training time of the LSTM model increases, and the prediction accuracy decreases rapidly.

B. Angular velocity prediction

The magnitude and direction of flapping angular velocity directly influence the robot's propulsive forces. The predicted angular velocity also determines the angular position evaluation. In this study, models with up to 50 hidden layers were tested, but accuracy saturates after 5 layers, and improvement is not significant subsequently (See Fig. 4). The model's complexity as well as the training time increases with increase in number of layers. Considering performance, complexity and training time, the MLP model with five layers and 47,901 parameters was chosen for the remaining experiments. During the training progress, the training and validation loss decrease dramatically in the first 10 epochs and remain stable after 60-80 epochs. If the validation loss does not improve for 30 consecutive training epochs, early stopping is used to restore the best model weights and prevent overfitting.

Fig. 5 shows that the MLP model performs well in both the training dataset and the testing dataset. The model shows excellent generalization capabilities without over-fitting problems.

C. Original Position Prediction

Equation 2 can be used to derive angular position from angular velocity information. The predicted angular position (θ_{t+1}) depends on the current angular position (θ_t and the angular velocity (ω_{input}), so the accumulation of small errors in each step can result in significant variations over time.

Fig. 6 shows the predicted angular position with a relatively significant error. The mean deviation in this cycle is of $6.54deg$, and the deviation of the last node is $12.63deg$. The small error in the angular velocity prediction can accumulate into a significant error at later stages of the predicted data series.

D. Adjusted angular position

Despite error accumulation (see Fig.6), the angular position prediction results are accurate in most groups. As discussed

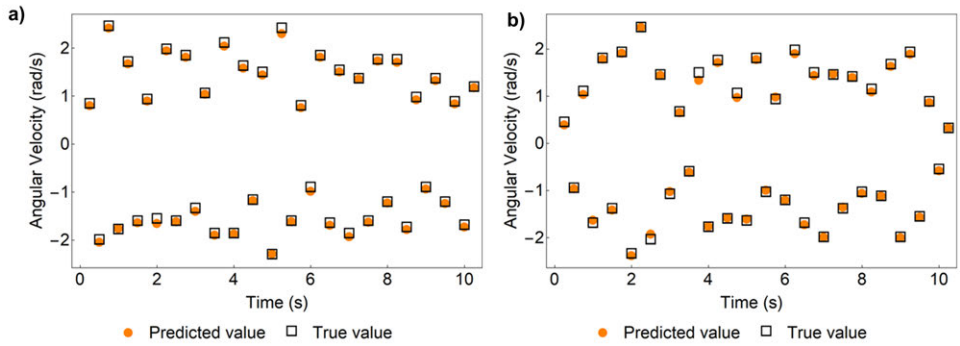

Fig. 5. Angular velocity prediction: (a) Predicted and true angular velocity values in training dataset; (b) Predicted and true angular velocity values in testing dataset. The dataset used for testing was not used during the neural network training and validation.

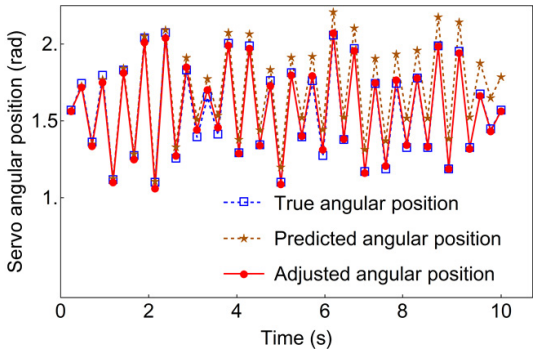

Fig. 6. Plot shows the comparison between true angular position, predicted angular position and adjusted angular position over a period of one robot input sequence

in section II.B, the angular position ends with $90deg$ in every cycle. Predictions will slightly deviate from this value due to the accumulation of deviations over time. By setting the predicted value of the end node to $90deg$ and adjusting all previous predicted node values proportionally, the overall error can be minimized. The correction is implemented using the relation,

$$\hat{\theta}'_{n,out} = \hat{\theta}_{n,out} + \frac{n}{N} E_{final} \tag{10}$$

where $\hat{\theta}_{n,out}$ and $\hat{\theta}'_{n,out}$ are the predicted angular position of the n^{th} node before and after the adjustment, N is the total number of flapping motions in one robot input sequence (e.g. $N = 41$), and E_{final} is the deviation value in the last node. If the angular position values are not within the upper and lower limit after adjustment, they are replaced with the angular boundary positions (60 or 120 respectively).

After adjustment, the prediction results were significantly improved. In the test case shown in Fig. 6, the mean deviation is reduced from $6.54deg$ to $1.04deg$. In other scenarios, the motion characteristics of the robot can also be used to correct the prediction results.

E. Force Data Validation

Validation experiments show that the measured forces and torques generated by the robot are in excellent agreement with the target forces and torques when the robot is controlled using the predicted robot input sequences generated by the deep neural networks (see Fig. 7). This is referred to as *Validation Group 1*. An entirely new robot input sequence was tested twice to estimate the magnitude of random errors. This is referred to as *Validation Group 2*. In *Validation Group 2*, the first and second tests are referred to as $test1$ and $test2$ respectively. The results in both validation groups show that the values for T_x, T_y, T_z, and F_z match very well their target values, while more noticeable errors can be seen in the values for F_x and F_y with respect to their target values (see Fig. 7, Fig. 8). This difference in F_x and F_y could be attributed to their small absolute values, making the percentage errors relatively larger. Dynamic time warping (DTW) was used to quantify the similarity between the target and achieved F/T sequences, it can eliminate the influence of the phase difference between the two sets of measured data Q and C which have same length K by finding the best alignment between them.

$$DTW(Q,C) = \underbrace{\arg\min}_{W=w1,..,w_k,..,w_K} \sqrt{\sum_{k=1,w_k=(i,j)}^{K} (q_i - c_j)^2} \tag{11}$$

In this study, DTW is used to calculate the degree of fitness between the two sets of Force/Torque data, a larger DTW value indicates a larger difference between the two sets of data, and vice versa. The percentage error of validation group 1 and validation group 2 is given by,

$$\frac{DTW(f_{target}(t), f_{validation}(t))}{DTW(f_{target}(t), f(t) = 0)} \times 100\% \tag{12}$$

where f_{target} and $f_{validation}$ are functions used to fit the target force data and corresponding robot force output data, $f(t) = 0$ represents a line that coincides with the horizontal axis.

Figure 8 shows that, compared with the inevitable random errors, the error caused by the deviation of the control signal accounts for a relatively small proportion of the total error. The experimental results show that our DNN prediction model can provide accurate control signal prediction for the soft robot to achieve the target forces and torques.

VI. Conclusions

In this study, DNN models were used to generate actuation inputs to control a soft bio-inspired robot. The control inputs

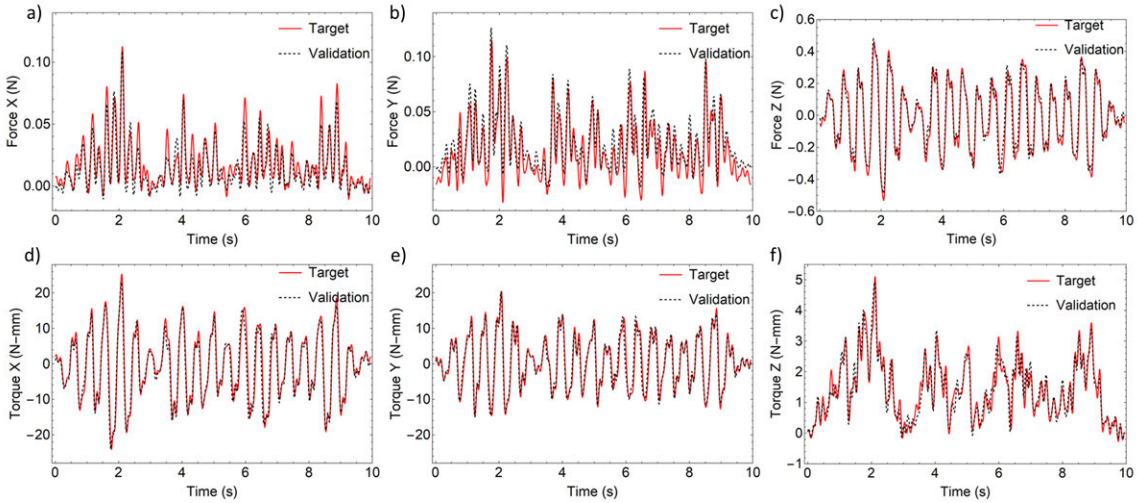

Fig. 7. Validation Group 1: Comparison of target and validation of F/T data with predicted input signal: (a) F_x; (b) F_y; (c) F_z; (d) T_x; (e) T_y; (f) T_z.

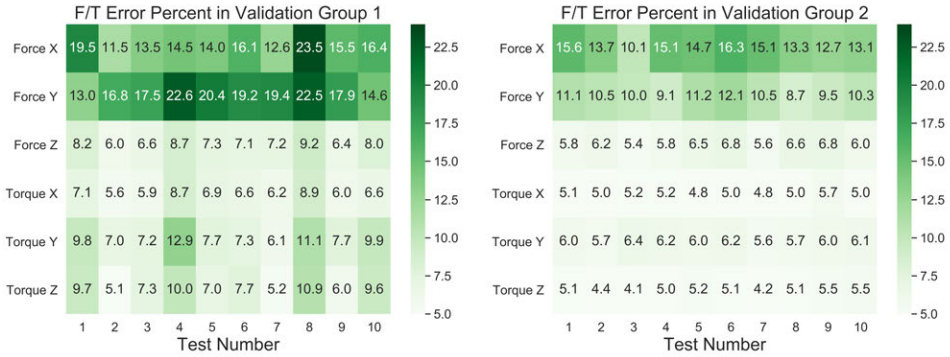

Fig. 8. Ten independent tests were performed on both validation group 1 and validation group 2 to verify the repeatability. The error percentage of each test is calculated using Eqn. 12. The values in the left figure reflect the overall error level (error caused by the DNN prediction deviation and uncontrollable errors), and the values in the right figure reflect the uncontrollable error level. The error level of validation group 1 is generally higher than that of validation group 2, the gap between 2 groups' accuracy reflect the error caused by the DNN prediction deviation. The results show this experiment has a good repeatability.

predicted by the DNN models led to robot generated propulsive forces and torques in good agreement with target values. This approach provides a new perspective for control of bio-inspired soft robots. Unlike traditional physics-based models, DNN models can provide simple input-output relationships for the complex dynamics found in soft bodies with infinite degrees of freedom. The DNN model structure's determination, adequate filtering of the training dataset, and the size of dataset will affect the final model quality. Various optimizations can be made in these areas. In this study, a simple approach using MLP and LSTM was successful in predicting adequate control inputs for fairly complex propulsive forces and torques profiles. This approach can be useful for pre-planned maneuvers or for feed-forward control, where traditional control approaches could be used for disturbance rejection. In more realistic environments with changing currents and flow conditions, the model prediction accuracy would suffer. To guarantee performance, a more robust model structure and broader training data with reinforced learning, as well as closed loop disturbance rejection should be explored. In the future, our group will generalize this method to include more challenging inputs such as pulse trains or non periodic static configurations to mimic various robot maneuvers. In addition, there is a vast scope for implementing real-time training by using sensor signals [53], [54] to train models and explore real-time adaptation to changes in the environment and improve prediction accuracy.

REFERENCES

[1] G. M. Whitesides, "Soft robotics," *Angewandte Chemie International Edition*, vol. 57, no. 16, pp. 4258–4273, 2018.
[2] B. Mazzolai, L. Margheri, M. Cianchetti, P. Dario, and C. Laschi, "Soft-robotic arm inspired by the octopus: Ii. from artificial requirements to innovative technological solutions," *Bioinspiration & biomimetics*, vol. 7, no. 2, p. 025005, 2012.
[3] R. Pfeifer, M. Lungarella, and F. Iida, "The challenges ahead for bio-inspired'soft'robotics," *Communications of the ACM*, vol. 55, no. 11, pp. 76–87, 2012.
[4] C. Majidi, "Soft robotics: a perspective—current trends and prospects for the future," *Soft Robotics*, vol. 1, no. 1, pp. 5–11, 2014.
[5] C. Laschi and M. Cianchetti, "Soft robotics: new perspectives for robot bodyware and control," *Frontiers in bioengineering and biotechnology*, vol. 2, p. 3, 2014.
[6] C. M. Best, M. T. Gillespie, P. Hyatt, L. Rupert, V. Sherrod, and M. D. Killpack, "A new soft robot control method: Using model predictive control for a pneumatically actuated humanoid," *IEEE Robotics & Automation Magazine*, vol. 23, no. 3, pp. 75–84, 2016.

[7] S. Bhagat, H. Banerjee, Z. H. Tse, and H. Ren, "Deep reinforcement learning for soft, flexible robots: Brief review with impending challenges," *Robotics*, vol. 8, no. 1, p. 4, 2019.

[8] M. T. Gillespie, C. M. Best, E. C. Townsend, D. Wingate, and M. D. Killpack, "Learning nonlinear dynamic models of soft robots for model predictive control with neural networks," *2018 IEEE International Conference on Soft Robotics (RoboSoft)*, 2018.

[9] F. Renda, M. Giorelli, M. Calisti, M. Cianchetti, and C. Laschi, "Dynamic model of a multibending soft robot arm driven by cables," *IEEE Transactions on Robotics*, vol. 30, no. 5, pp. 1109–1122, 2014.

[10] R. J. Webster III and B. A. Jones, "Design and kinematic modeling of constant curvature continuum robots: A review," *The International Journal of Robotics Research*, vol. 29, no. 13, pp. 1661–1683, 2010.

[11] H. Zhang, R. Cao, S. Zilberstein, F. Wu, and X. Chen, "Toward effective soft robot control via reinforcement learning," in *International Conference on Intelligent Robotics and Applications*, pp. 173–184, Springer, 2017.

[12] J. Schmidhuber, "Deep learning in neural networks: An overview," *Neural networks*, vol. 61, pp. 85–117, 2015.

[13] Y. Bengio, *Learning deep architectures for AI*. Now Publishers Inc, 2009.

[14] G. Panchal, A. Ganatra, Y. Kosta, and D. Panchal, "Behaviour analysis of multilayer perceptrons with multiple hidden neurons and hidden layers," *International Journal of Computer Theory and Engineering*, vol. 3, no. 2, pp. 332–337, 2011.

[15] B. M. Wilamowski, "Neural network architectures and learning algorithms," *IEEE Industrial Electronics Magazine*, vol. 3, no. 4, pp. 56–63, 2009.

[16] B. K. Bose, "Neural network applications in power electronics and motor drives—an introduction and perspective," *IEEE Transactions on Industrial Electronics*, vol. 54, no. 1, pp. 14–33, 2007.

[17] Y. Bengio, I. Goodfellow, and A. Courville, *Deep learning*, vol. 1. MIT press, 2017.

[18] S. Dupond, "A thorough review on the current advance of neural network structures," *Annual Reviews in Control*, vol. 14, pp. 200–230, 2019.

[19] S. Hochreiter, Y. Bengio, P. Frasconi, J. Schmidhuber, *et al.*, "Gradient flow in recurrent nets: the difficulty of learning long-term dependencies," 2001.

[20] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[21] F. A. Gers, J. Schmidhuber, and F. Cummins, "Learning to forget: Continual prediction with lstm," 1999.

[22] M. Sundermeyer, R. Schlüter, and H. Ney, "Lstm neural networks for language modeling," in *Thirteenth annual conference of the international speech communication association*, 2012.

[23] K. Chin, T. Hellebrekers, and C. Majidi, "Machine learning for soft robotic sensing and control," *Advanced Intelligent Systems*, vol. 2, no. 6, p. 1900171, 2020.

[24] D. Kim, S.-H. Kim, T. Kim, B. B. Kang, M. Lee, W. Park, S. Ku, D. Kim, J. Kwon, H. Lee, *et al.*, "Review of machine learning methods in soft robotics," *Plos one*, vol. 16, no. 2, p. e0246102, 2021.

[25] P. Hyatt, D. Wingate, and M. D. Killpack, "Model-based control of soft actuators using learned non-linear discrete-time models," *Frontiers in Robotics and AI*, vol. 6, 2019.

[26] C. C. Johnson, T. Quackenbush, T. Sorensen, D. Wingate, and M. D. Killpack, "Using first principles for deep learning and model-based control of soft robots," *Frontiers in Robotics and AI*, vol. 8, 2021.

[27] K. Kiguchi, H.-H. Jang, and T. Fukuda, "Identification of robot manipulators using neural networks and genetic programming," in *IEEE SMC'99 Conference Proceedings. 1999 IEEE International Conference on Systems, Man, and Cybernetics (Cat. No. 99CH37028)*, vol. 4, pp. 802–806, IEEE, 1999.

[28] S. Gay, J. Santos-Victor, and A. Ijspeert, "Learning robot gait stability using neural networks as sensory feedback function for central pattern generators," in *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 194–201, Ieee, 2013.

[29] T. G. Thuruthel, E. Falotico, F. Renda, and C. Laschi, "Learning dynamic models for open loop predictive control of soft robotic manipulators," *Bioinspiration & biomimetics*, vol. 12, no. 6, p. 066003, 2017.

[30] X. Liu, R. Gasoto, C. Onal, and J. Fu, "Learning to locomote with deep neural-network and cpg-based control in a soft snake robot," *arXiv preprint arXiv:2001.04059*, 2020.

[31] K. Nakajima, H. Hauser, R. Kang, E. Guglielmino, D. G. Caldwell, and R. Pfeifer, "A soft body as a reservoir: case studies in a dynamic model of octopus-inspired soft robotic arm," *Frontiers in computational neuroscience*, vol. 7, p. 91, 2013.

[32] K. Nakajima, T. Li, H. Hauser, and R. Pfeifer, "Exploiting short-term memory in soft body dynamics as a computational resource," *Journal of The Royal Society Interface*, vol. 11, no. 100, p. 20140437, 2014.

[33] K. Nakajima, H. Hauser, T. Li, and R. Pfeifer, "Information processing via physical soft body," *Scientific reports*, vol. 5, p. 10487, 2015.

[34] K. Nakajima, H. Hauser, T. Li, and R. Pfeifer, "Exploiting the dynamics of soft materials for machine learning," *Soft Robotics*, vol. 5, no. 3, p. 339–347, 2018.

[35] P. V. y Alvarado, S. Chin, W. Larson, A. Mazumdar, and K. Youcef-Toumi, "A soft body under-actuated approach to multi degree of freedom biomimetic robots: A stingray example," in *2010 3rd IEEE RAS & EMBS International Conference on Biomedical Robotics and Biomechatronics*, pp. 473–478, IEEE, 2010.

[36] P. V. y Alvarado, "Hydrodynamic performance of a soft body under-actuated batoid robot," in *2011 IEEE International Conference on Robotics and Biomimetics*, pp. 1712–1717, IEEE, 2011.

[37] A. Cloitre, V. Subramaniam, N. Patrikalakis, and P. V. y Alvarado, "Design and control of a field deployable batoid robot," in *2012 4th IEEE RAS & EMBS International Conference on Biomedical Robotics and Biomechatronics (BioRob)*, pp. 707–712, IEEE, 2012.

[38] T. Truong, V. Viswanathan, V. Joseph, and P. V. y Alvarado, "Design and characterization of a fully autonomous under-actuated soft batoid-like robot," in *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 5826–5831, IEEE, 2019.

[39] F. Boyer, M. Porez, F. Morsli, and Y. Morel, "Locomotion dynamics for bio-inspired robots with soft appendages: Application to flapping flight and passive swimming," *Journal of Nonlinear Science*, vol. 27, no. 4, pp. 1121–1154, 2017.

[40] N. N. Goldberg, X. Huang, C. Majidi, A. Novelia, O. M. O'Reilly, D. A. Paley, and W. L. Scott, "On planar discrete elastic rod models for the locomotion of soft robots," *Soft robotics*, vol. 6, no. 5, pp. 595–610, 2019.

[41] Y.-Y. Xiao, Z.-C. Jiang, X. Tong, and Y. Zhao, "Biomimetic locomotion of electrically powered "janus" soft robots using a liquid crystal polymer," *Advanced Materials*, vol. 31, no. 36, p. 1903452, 2019.

[42] C. Wang, K. Sim, J. Chen, H. Kim, Z. Rao, Y. Li, W. Chen, J. Song, R. Verduzco, and C. Yu, "Soft ultrathin electronics innervated adaptive fully soft robots," *Advanced Materials*, vol. 30, no. 13, p. 1706695, 2018.

[43] N. Boddeti, T. Van Truong, V. S. Joseph, T. Stalin, T. Calais, S. Y. Lee, M. L. Dunn, and P. Valdivia y Alvarado, "Optimal soft composites for under-actuated soft robots," *Advanced Materials Technologies*, vol. 6, no. 8, p. 2100361, 2021.

[44] K. S. Sekar, M. Triantafyllou, and P. V. y Alvarado, "Flapping actuator inspired by lepidotrichia of ray-finned fishes," in *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 1120–1126, IEEE, 2014.

[45] G. F. Montufar, R. Pascanu, K. Cho, and Y. Bengio, "On the number of linear regions of deep neural networks," in *Advances in neural information processing systems*, pp. 2924–2932, 2014.

[46] S. Sharma, "Activation functions in neural networks," *towards data science*, vol. 6, 2017.

[47] R. H. Hahnloser, R. Sarpeshkar, M. A. Mahowald, R. J. Douglas, and H. S. Seung, "Digital selection and analogue amplification coexist in a cortex-inspired silicon circuit," *Nature*, vol. 405, no. 6789, pp. 947–951, 2000.

[48] K. Jarrett, K. Kavukcuoglu, M. Ranzato, and Y. LeCun, "What is the best multi-stage architecture for object recognition?," in *2009 IEEE 12th international conference on computer vision*, pp. 2146–2153, IEEE, 2009.

[49] V. Nair and G. E. Hinton, "Rectified linear units improve restricted boltzmann machines," in *Icml*, 2010.

[50] J. Schmidt-Hieber, "Nonparametric regression using deep neural networks with relu activation function," *arXiv preprint arXiv:1708.06633*, 2017.

[51] P. Ramachandran, B. Zoph, and Q. V. Le, "Searching for activation functions," *arXiv preprint arXiv:1710.05941*, 2017.

[52] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[53] P. V. Alvarado, V. Subramaniam, and M. Triantafyllou, "Performance analysis and characterization of bio-inspired whisker sensors for underwater applications," in *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 5956–5961, 2013.

[54] A. Vaish, S. Y. Lee, and P. V. y. Alvarado, "Mechanical fourier transform using an array of additively manufactured soft whisker-like sensors," in *2019 International Conference on Robotics and Automation (ICRA)*, pp. 9410–9415, 2019.