Proceedings of the 8th International Conference of Control Systems, and Robotics (CDSR'21)
~~Niagara Falls, Canada~~ Virtual Conference– May 23-25, 2021
Paper No113
DOI: 10.11159/cdsr21.113

Deep Learning-based Robot Control using Recurrent Neural Networks (LSTM; GRU) and Adaptive Sliding Mode Control

Raj Patel[1], Meysar Zeinali[2], Kalpdrum Passi[3],

[1]Department of Mathematics & Computer Science, Laurentian University, Sudbury, Ontario, Canada
rpatel7@laurentian.ca
[2]School of Engineering, Laurentian University, Sudbury, Ontario, Canada
mzeinali@laurentian.ca
[3]Department of Mathematics & Computer Science, Laurentian University, Sudbury, Ontario, Canada
kpassi@cs.laurentian.ca

Abstract - A phenomenal increase in computational power made deep learning possible for real-time applications in recent years. Non-linearity, external disturbances, and robustness are significant challenges in robotics. To overcome these challenges, robust adaptive control is needed, which requires manipulator inverse dynamics. Deep Learning can be used to construct the inverse dynamic of a manipulator. In this paper, robust adaptive motion control is developed by effectively combining existing adaptive sliding mode controller (ASMC) with Recurrent Neural Network such as Long Short-Term Memory (LSTM) and Gated Recurrent Unit (GRU). A supervised learning approach is applied to train the LSTM and GRU model, which replaced the inverse dynamic model of a manipulator in model-based control design. The LSTM-based inverse dynamic model constructed using input-output data obtained from a simulation of a dynamic model of the two-links robot. The deep-learning-based controller applied for trajectory tracking control, and the results of the proposed Deep Learning-based controller are compared in three different scenarios: ASMC only, LSTM or GRU only, and LSTM or GRU with ASMC (with and without disturbance) scenario. The primary strategy of designing a controller with LSTM or GRU is to get better generalization, accuracy enhancement, compensate for fast time-varying parameters and disturbances. The experimental results depict that without tuning parameters proposed controller performs satisfactorily on unknown trajectories and disturbances.

Keywords: Robot Learning Control, Deep Learning, Recurrent Neural Network, Long Short-Term Memory, Gated Recurrent Unit, Adaptive Sliding Mode Control

1. Introduction

Artificial Intelligence and Deep Learning are becoming the key to modern robotics. Deep Learning has demonstrated incredible success in image and video processing, object detection, recommender systems, and natural language processing. Deep Learning is a part of a more comprehensive class of machine learning techniques in which the model learns from the data. Each layer extracts new features where a type of learning includes supervised, unsupervised, and semi-supervised. In a deep neural network, top layers learn higher-level features where the last layers learn lower-level features. Yann LeCunn et al. [1] used the back-propagation technique to recognize hand-written digits taken from the US Mail. Krizhevsky, Sutskever and Hinton, proposed a deep convolutional neural network in [2] that made a significant impact on image recognition. H. Su et al. applied the Deep Convolutional Neural Network approach to managing redundancy control [3].

On the other hand, from simulation in the lab to real-world work, robots face extreme challenges. Increasing Human-Robot Interaction requires higher perception and precision for the robot to ensure the safety of human life. Human behaviour interpretation is a highly complicated task, for which human-made solutions are tough to formulate [4]. As the industry grows, robots need to handle various work, which is difficult for many learning algorithms. In such a scenario, a system having the ability to learn from data plays a significant role. In the robotics and control field, numerous researchers have conducted extensive research on intelligent control, adaptive sliding mode control, adaptive control, and chattering-free SMC using fuzzy-logic-based dynamic model of a robot [5]–[9].

The adaptive controller has a problem in handling the unmodelled dynamic and parameter convergence. The well-known variable structure control method, SMC, has been proposed to satisfy robust tracking but the chattering problem still arises,

leading to actuator break down or inefficient control performance. M. Zeinali proposed a chattering free SMC using a fuzzy model combined with Adaptive Sliding Mode Control [10].

In the last decade, the Neural Network-based approach has increased due to its ability to approximate any function. The combination of classical control and advancement in deep learning-based methods began the development of new techniques for research. A Neural Network-based approach to learn the flexible manipulator's inverse dynamics presented in Zeinali and Wang [11]. The Neural Network-based approach needs higher computational power compared to a classical controller. B. Ozyer introduced an Adaptive Fast Sliding Controller combined with Neural Network and Global Fast sliding Mode Controller [12]. S. Bansal, et al. used Neural Network in learning quadrotor dynamics for flight control [13]. S. Edhah et al. used a new greedy Layer-wise approach in which they separated hidden layers to train separately [14]. After training, they merged all hidden layers as a single network. However, application of deep learning techniques to estimate the inverse dynamic of the manipulator and build a robust controller is fairly new and needs more investigation, in particular, combination of ASMC with deep learning-based model needs more research and have not been fully investigated in the literature.

In this paper a robust adaptive motion control is proposed by effectively combining of existing adaptive sliding mode controller (ASMC) [10-11] with Recurrent Neural Network such as Long Short-Term Memory (LSTM) and Gated Recurrent Unit (GRU). A supervised learning approach is applied to train the LSTM and GRU model, which replaces the approximately known inverse dynamic model of a manipulator in model-based control design. As an increasing Neural Network layer gets an accurate and generalized model, the loss function gradient diminishes to zero, making it hard to train, resulting in the vanishing gradient problem. The main reason for the vanishing gradient is transforming a large variation in input data into a small variation input (e.g., 0 to 1, -1 to 1), which means a large change in input data results in a minimal change and the derivative of that becomes close to zero [15].

Humans have biological intelligence which processes information incrementally while keeping an internal model of what it is processing, built from prior knowledge and continuously updated as new knowledge comes in [16]. Recurrent Neural Network uses the same idea, keep the information in a cell. RNN-based models have a problem remembering Long Term dependencies [17], which is solved by the advanced versions of RNN called Long short-term memory [18]. N. Liu et al. used an LSTM-based deep learning algorithm to model robot inverse dynamics for Smart Cities and Factories [19]. This paper organized as follows: Section 2 illustrates problem formulation and the Adaptive Sliding Mode Controller structure. Section 3 includes a brief description of LSTM and the proposed deep learning structure. Simulation results and comparison of different models included in Section 4. Section 5 concludes the paper and future work.

2. Problem Definition

The n-link manipulator's general dynamic model can be defined by following a second-order system equation in joint space angles [10].

$$\tau = M(q)\ddot{q} + C(q,\dot{q})\dot{q} + F(q,\dot{q}) + G(q) + d(t) \tag{1}$$

where q, $\dot{q}$, and $\ddot{q}$ are the joint position, velocity, and acceleration vectors, respectively. $M(q) \in \mathrm{R}^{n\times n}$, $C(q,\dot{q}) \in \mathrm{R}^{n\times n}$, $F(\dot{q}) \in \mathrm{R}^{n\times 1}$, $G \in \mathrm{R}^{n\times 1}$, $d(t) \in \mathrm{R}^{n\times 1}$, $and\ \tau$ refers to inertia matrix, which is positive definite, Coriolis terms, Viscus and Coulomb friction coefficient, gravitational term, bounded disturbances, and torque, respectively. Viscus (F_v) and Coulomb (F_c) friction term defined as the following equation:

$$F(q,\dot{q}) = F_v\dot{q} + F_c \tag{2}$$

Assuming $h(q,\dot{q}) = C(q,\dot{q})\dot{q} + F(q,\dot{q}) + G(q)$, the equation (1) can be defined in the following way:

$$\tau = M(q)\ddot{q} + h(q,\dot{q}) + d(t) \tag{3}$$

Due to the presence of external disturbance, payload variation, and friction terms, it is extremely difficult to model the exact dynamic of the robot mathematically, therefore, dynamic model is an approximation of actual robot and the components defined in (1) can be rewritten as follows:

$$\tau = \left(\widehat{M}(q) + \triangle M(q)\right)\ddot{q} + \left(\hat{C}(q,\dot{q}) + \triangle C(q,\dot{q})\right)\dot{q} + \left(\hat{F}(q,\dot{q}) + \triangle F(q,\dot{q})\right) + \left(\hat{G}(q) + \triangle G(q)\right) + d(t) \tag{4}$$

And

$$M(q) = \widehat{M}(q) + \triangle M(q);\ C(q) = \hat{C}(q,\dot{q}) + \triangle C(q,\dot{q});\ F(q) = \hat{F}(q,\dot{q}) + \triangle F(q,\dot{q});\ \ G(q) = \hat{G}(q) + \triangle G(q) \tag{5}$$

Here, $\widehat{M},\ \hat{C},\ \hat{F},\ and\ \hat{G}$ are estimated values of inertia matrix, Coriolis terms, friction terms, and gravity terms whereas $\triangle M$, $\triangle C,\ \triangle F,\ and\ \triangle G$ are an unknown portion of these terms. In this work, it is assumed that the friction term $F(q)$ is completely unknown without the loss of gravity. Therefore, equation (1) can be written as follows:

$$\tau = \hat{\tau}(q,\dot{q},\ddot{q},t) + \delta(q,\dot{q},\ddot{q},t) \tag{6}$$

$$\hat{\tau}(q,\dot{q},\ddot{q},t) = \widehat{M}(q)\ddot{q} + \hat{C}(q,\dot{q},t)\dot{q} + \hat{G}(q) \tag{7}$$

Here, $\delta(q,\dot{q},\ddot{q};t)$ is an unknown function that combines various uncertain terms into one lumped uncertainty term, and defined as follows

$$\delta(q,\dot{q},\ddot{q},t) = \triangle M(q)\ddot{q} + \triangle C(q,\dot{q},t)\,\dot{q} + \triangle G(q) + \Delta F(q,\dot{q},t) + d(t) \tag{8}$$

Equation (7) is the nominal dynamics of the robot. In this paper, first, the goal is to construct the nominal dynamics of the manipulator, using deep-learning techniques, and drive a formulation for the controller using this deep learning–based model instead of the components given in (7), and then use the learning capability of this AI-based dynamic model and capture the unknown dynamic of the robot over time true off-line and online training To design the controller and analyze the closed-loop system's stability, it is assumed that the following assumption holds.

Assumption: The uncertainty vector $\delta(q,\dot{q},\ddot{q},t)$ and its partial derivatives are bounded in Euclidian norm as:

$$\|\delta(q,\dot{q},\ddot{q},t)\| \le \rho(q,\dot{q},\ddot{q},t) \tag{9}$$

where $\rho(q,\dot{q},\ddot{q},t)$ is the unknown bounding function of the uncertainties, $\|\ .\ \|$ is Euclidian norm. For the detailed information about the stability, the reader is recommended to refer [10-11].

2.1 Controller Design

The controller proposed in this paper is built based on adaptive sliding mode control presented in reference [5-6], as:

$$\tau_{ASMC} = \widehat{M}\ddot{q}_r + \hat{C}(q,\dot{q},t)\dot{q}_r + \hat{G}(q) - KS - \Gamma\int S\,dt \tag{10}$$

In above equation it is assumed that the friction component of robot dynamic is included in the lumped uncertainty term and it is estimated using adaptive term of the controller. In equation (10) K and Γ are design parameters and they are symmetric positive definite diagonal matrix and $S(q,t)$ is the sliding variable which is defined as follows:

$$S(q,t) = (\frac{d}{dt} + \Lambda)^2(\int edt) = \dot{e} + 2\Lambda e + \Lambda^2\int edt \tag{11}$$

And $$e = q - q_d, \dot{e} = \dot{q} - \dot{q}_d \quad (12)$$

where Λ is an $n \times n$ symmetric positive definite diagonal constant matrix and $e = q - q_d, \dot{e} = \dot{q} - \dot{q}_d$ are the tracking error and the rate of error, respectively. q_d and q are the desired and measured joint variables, respectively. The integral of error is included to ensure zero offset error. Control law given in equation (10) has two component: ***i***) a model-based component, i.e., $\tau_m = \hat{M}\ddot{q}_r + \hat{C}\dot{q}_r + \hat{G}$, which can be constructed based on the available knowledge of the dynamic model of the robot manipulator; and ***ii***) a sliding function-based proportional-integral component $\tau_{PI} = -KS - \Gamma \int S dt$, which replaces the discontinuous term in traditional sliding mode, and is constructed based on dynamic behavior of sliding function, for detail please refer to [5-6]. In this work the goal is to replace the model-based component:

$$\tau_m = \hat{M}\ddot{q}_r + \hat{C}\dot{q}_r + \hat{G} \quad (13)$$

with the component calculated using deep learning techniques and is called τ_{NN} in this paper. In fact, τ_{NN} is the data-driven parameterized model of the robot constructed using deep leaning techniques and through off-line training based on data collected from the robot, in which parameters of the model are the weights of the network. Deep learning-based component is a nonlinear function of joint variables and is as:

$$\tau_{NN} = f(q, \dot{q}, \ddot{q}) \quad (14)$$

Where, the input variables are position, velocity, and acceleration of each joint $(q, \dot{q}, \ddot{q})$, and the output of the neural network is the torque τ_{NN} of the corresponding joint. Replacing the model-based component with neural network-based component in equation (10), control law cab be written as:

$$\tau_{ASMC} = \tau_{NN} - KS - \Gamma \int S\, dt \quad (15)$$

The block diagram of the proposed controller is shown in Figure 1.

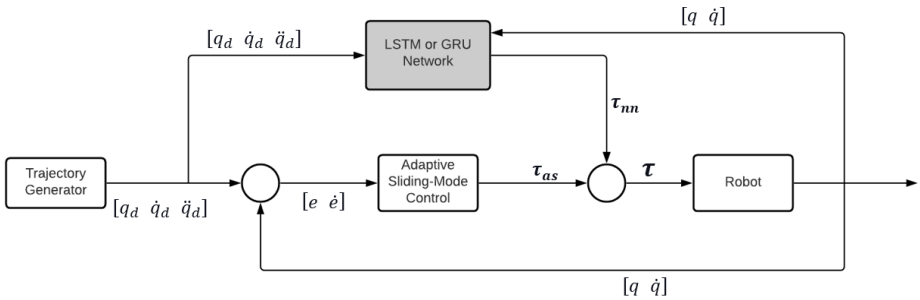

Figure 1: Block diagram of the controller

3. Proposed Deep-Learning Methodology

3.1 Recurrent Neural Network

Depends on the problem, there are numerous versions of RNN presented by researchers such as one-to-one, one-to-many, many-to-many, and many-to-one, whereas some advanced versions include Long Short-Term Memory and Gated Recurrent Unit. In some exceptional cases like a singular-plural problem in speech recognition, the current cell calculation also depends on the next cell's upcoming input. A particular version of Recurrent Neural Network known as Bidirectional Recurrent Neural N plays a significant role in accurate prediction in such cases.

A. Long Short-Term Memory (LSTM)

Nowadays, there has been incredible success using Recurrent Neural Network in various problems such as speech recognition, image captioning, language modelling. The traditional neural network-based models have a shortcoming in remembering older situations. In contrast, Recurrent Neural Network-based models such as LSTM can persist information to use with sequence datasets, making them unique. Long Short Term Memory networks are a unique kind of recurrent neural networks introduced by Hochreiter & Schmidhuber [18] in 1997. In theory, classic RNNs can keep track of arbitrary long-term dependencies in the input sequences. Because of the computations involved in the development of backpropagation method, which uses finite-precision numbers while training a classic (or "vanilla") RNN, the gradients which are back-propagated can "vanish" or "explode".

As shown in Figure 2, a sequence input layer and a LSTM layer are the core component of the LSTM Network. LSTM looks at the input x and outputs a value h, and a loop allows information to pass from one stage to another. As shown in Figure 3, the horizontal line running through the top of the diagram is responsible for persisting information over time. With only a few linear interactions, it runs straight down the entire chain.

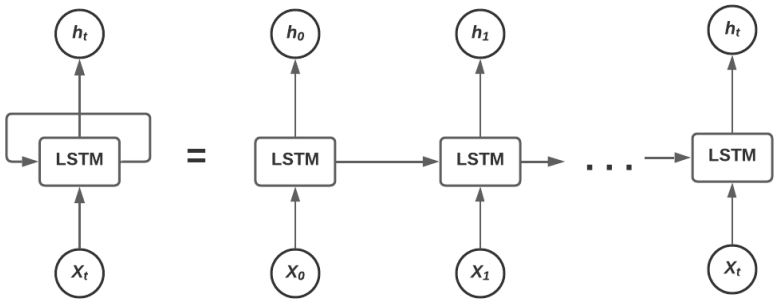

Figure 2: Sequential nature of LSTM

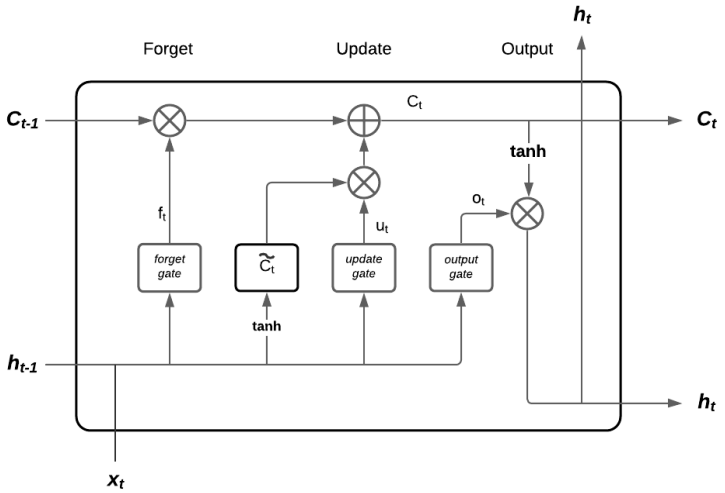

Figure 3: Structure of LSTM cell. This figure is revised from [18]

Mathematically LSTM can be represented by the following equations:

$$f_t = \sigma\big(w_f[h_{t-1}, x_t] + b_f\big) \quad (16)$$

$$u_t = \sigma(w_u[h_{t-1}, x_t] + b_u) \quad (17)$$

$$\tilde{c}_t = \tanh(w_c[h_{t-1}, x_t] + b_c) \quad (18)$$

$$o_t = \sigma(\omega_o[h_{t-1}, x_t] + b_o) \quad (19)$$

$$c_t = u_{t^*}\tilde{c}_t + f_{t^*}c_{t-1} \quad (20)$$

$$h_t = o_{t^*}\tanh(c_t) \quad (21)$$

Where $f_t, u_t,$ and o_t are forget gate, update gate, and output gate, respectively. x_t is the input to the current cell. Here, h_{t-1} and c_{t-1} are the output from the last LSTM cell. σ represents a sigmoid activation function, and tanh stands for tangent activation function. $w_f,\ w_u,\ w_c,\ and\ w_o$ are weight matrix, and $b_f, b_u, b_c, and\ b_o$ refer to bias vectors. f_t is multiplied with c_{t-1} which means based on that how much previous information is useful in current prediction. Also, h_t is the output of each LSTM cell (that is the output torque calculated using the network) and h_{t-1} is output from previous cell and used in the inner computation of LSTM cell.

B. Gated Recurrent Unit (GRU)

GRU is another recurrent neural network that learns meaningful representation from sequence data [20]. The GRU is similar to LSTM without an output gate. GRU cell contains reset gate (r_t), update gate (z_t), and candidate gate ($\hat{h}_t$).

$$r_t = \sigma_s(W_{xr}X_t + b_r + W_{hr}h_{t-1}) \quad (22)$$

$$z_t = \sigma_s(W_{xz}X_t + b_z + W_{hz}h_{t-1}) \quad (23)$$

$$\hat{h}_t = tanh(W_{xh}X_t + b_h + W_{hh}(r_t \odot h_{t-1})) \quad (24)$$

$$h_t = (1 - z_t) \odot \hat{h}_t + z_t \odot h_{t-1} \quad (25)$$

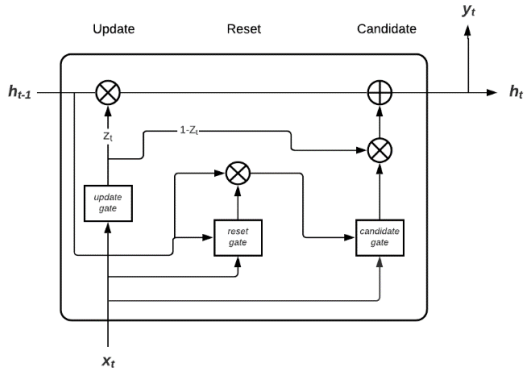

Figure 4: Structure of GRU cell. This figure is revised from [20]

Where $r_t, z_t, and\ \hat{h}_t$ are stands for reset gate, update gate, and candidate gate, respectively. Moreover, h_t is the output of each GRU cell (that is the output torque calculated using the network) and h_{t-1} is output from previous cell and used in the

inner computation of GRU cell. The weight matrix for input gate to reset gate, update gate, and candidate gate are defined as $W_{xr}, W_{xz}, and\ W_{xh}$. Here σ_s is sigmoid activation function.

3.2 Proposed Deep Learning Architecture

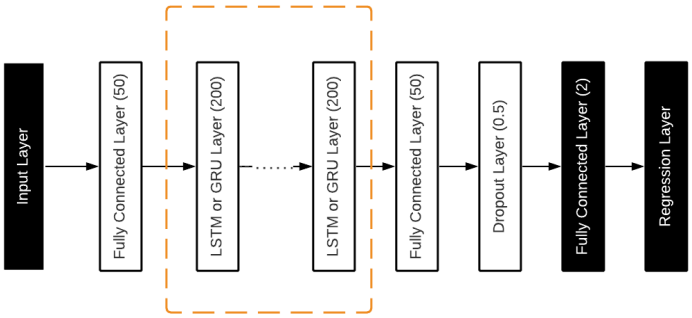

Figure 5: Deep Learning Structure.

As shown in Figure 5, the proposed architecture contains a sequence input layer, LSTM layer, fully connected layer, and dropout layer. Here, the sequence input layer has 6 neurons (position, velocity, and acceleration for Joint 1 and 2), LSTM with 200 neurons, fully connected layer with 50 neurons. The dropout layer utilized to prevent the overfitting situation with a dropout ratio of 0.5. The state activation function and gate activation function are tanh and sigmoid, respectively. Glorot is used as an input weight initializer for recurrent and fully connected layers.

Sigmoid and Hyperbolic tangent activation function can be defined as following:

$$\sigma(x) = \frac{1}{1+ e^{x}} \tag{26}$$

$$tanh(x) = \frac{e^{x} - e^{-x}}{e^{x} + e^{-x}} \tag{27}$$

All inputs are normalized using z-score normalization. In z-score normalization, the input value is normalized based on a mean and standard deviation of input.

$$x' = \frac{x - \bar{x}}{\sigma_x} \tag{28}$$

4. Simulation

To evaluate performance, we compared our approach with existing approaches on various uncertainties and disturbances to measure adaptive capability, error convergence, and robustness of the controller. The exact dynamic model of the manipulator is not known because of uncertainty and external disturbance. In this context, the dynamic model of a 2-DOF robot is as follows

$$\begin{bmatrix}\tau_1 \\ \tau_2\end{bmatrix} = \begin{bmatrix}\widehat{M}_{11} & \widehat{M}_{12} \\ \widehat{M}_{21} & \widehat{M}_{22}\end{bmatrix}\begin{bmatrix}\ddot{q}_1 \\ \ddot{q}_2\end{bmatrix} + \begin{bmatrix}\hat{C}_{11} & \hat{C}_{12} \\ \hat{C}_{21} & \hat{C}_{22}\end{bmatrix}\begin{bmatrix}\dot{q}_1 \\ \dot{q}_2\end{bmatrix} + \begin{bmatrix}\hat{g}_1 \\ \hat{g}_2\end{bmatrix} \tag{29}$$

where, $\widehat{M}_{11} = a_1 + 2a_3\cos(q_2) + 2a_4\sin(q_2), \quad \widehat{M}_{12} = a_2 + a_3\cos(q_2) + a_4\sin(q_2),$

$$\widehat{M}_{21} = \widehat{M}_{12}, \quad \widehat{M}_{22} = a2, \qquad \hat{C}_{11} = -h\dot{q}_2 + v1, \qquad \hat{C}_{12} = -h(\dot{q}_1 + \dot{q}_2), \qquad \hat{C}_{21} = h\dot{q}_1, \qquad \hat{C}_{22} = v2,$$

$$h = a_3 \sin(q_2) - a_4 \cos(q_2), \qquad \hat{g}_1 = a_5 \cos(q_1) + a_6 \cos(q_1 + q_2),$$

$$\hat{g}_2 = a_6 * \cos(q_1 + q_2), \qquad a_1 = I_1 + m_1 l_{c1}^2 + I_e + m_e l_{ce}^2 + m_e l_1^2, \qquad a_2 = I_e + m_e l_{ce}^2$$

$$a_3 = m_e l_1 l_{ce} \cos(30), a_4 = m_e l_1 l_{ce} \sin(30), a_5 = (m_1 l_{c1} + m_e l_1) g, a_6 = \frac{m_e l_{ce} g}{\cos\left(\frac{\pi}{6}\right)}$$

$$Here, \qquad K = \begin{bmatrix} 400 & 0 \\ 0 & 200 \end{bmatrix}, \qquad \zeta = \begin{bmatrix} 25 & 0 \\ 0 & 15 \end{bmatrix}, \qquad \Gamma = \begin{bmatrix} 20000 & 0 \\ 0 & 10000 \end{bmatrix}, \qquad \Lambda = 50, \qquad d(t) = \begin{bmatrix} 250 & 0 \\ 0 & 100 \end{bmatrix}$$

Combining LSTM or GRU with ASMC as follows

$$\tau_{LSTM} = \tau_{GRU} = h_t \tag{32}$$

$$\tau_{LSTM+\,ASMC} = \tau_{LSTM} - KS - \Gamma \int S\, dt \tag{33}$$

$$\tau_{GRU+ASMC} = \tau_{GRU} - KS - \Gamma \int S\, dt \tag{34}$$

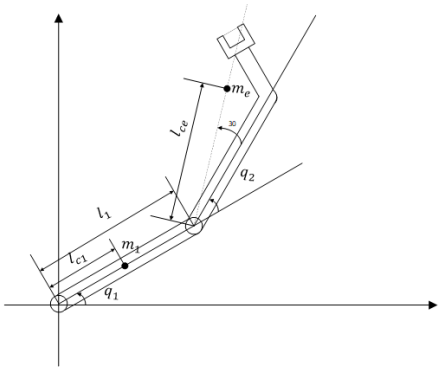

Figure 6: This figure is revised from [5].

LSTM and GRU are trained on 27 trajectories with a sampling time of 0.003 sec for 500 epochs and tested with 0.001 sec for 10 sec on 3 new trajectories. The initial learning rate and $L2$ regularization are 0.005 and 0.0005, respectively.

Table 2. Model performance with and without external disturbance (in degree)

Model	RMSE for Joint 1	RMSE for Joint 2	Average RMSE	Disturbance
ASMC	0.0413	0.0616	0.0514	No
LSTM	4.7291	22.6367	13.6829	No
GRU	5.0640	17.4393	11.2516	No
LSTM + ASMC	0.1608	0.1810	0.1709	No
GRU + ASMC	0.1598	0.1790	0.1694	No
ASMC	0.0429	0.0685	0.0557	t = 7 sec
LSTM	5.3851	49.0522	27.2186	t = 7 sec
GRU	8.5020	35.4534	21.9777	t = 7 sec
LSTM + ASMC	0.1613	0.1824	0.1718	t = 7 sec
GRU + ASMC	*0.1603*	*0.1804*	*0.1703*	*t = 7 sec*

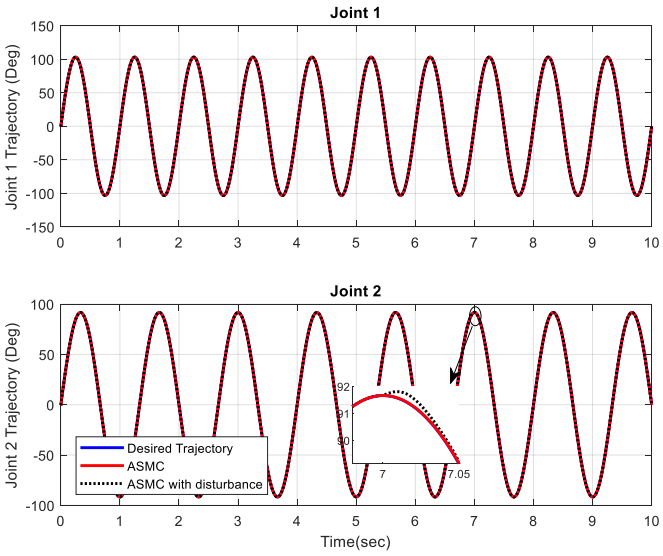

Figure 7(a): ASMC controller performance with and without disturbance.

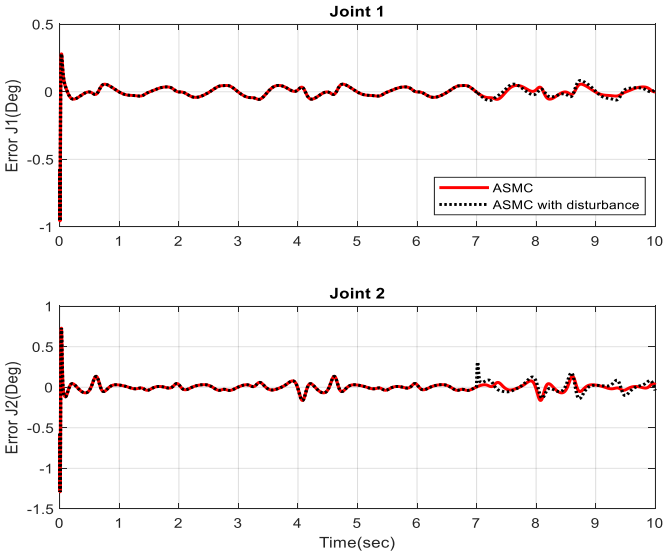

Figure 7(b): ASMC controller Tracking error with and without disturbance.

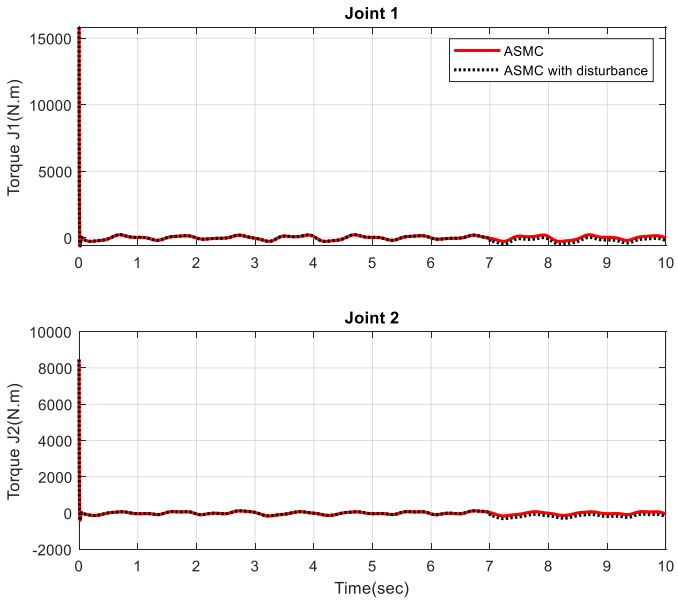

Figure 7(c): ASMC controller Torque with and without disturbance.

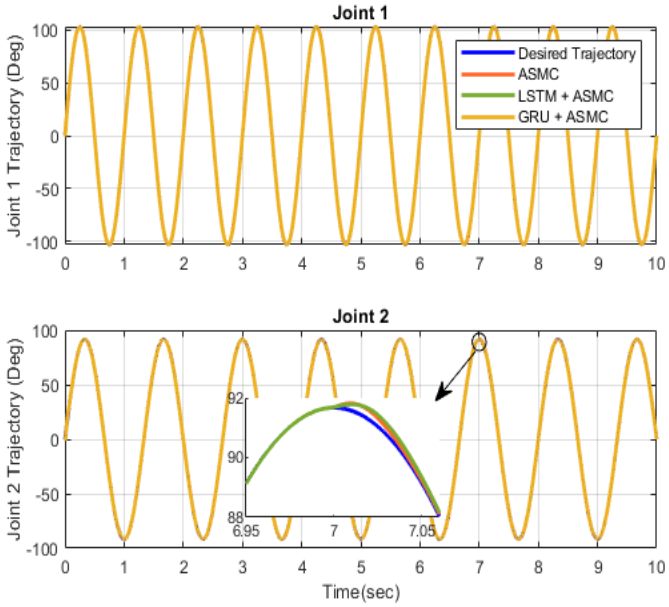

Figure 8(a): Comparison of ASMC, LSTM with ASMC only, and GRU with ASMC (with disturbance).

Figure 7 describes the Adaptive Sliding Mode Controller with and without disturbance. The external disturbance T_d applied at t = 7 sec and in 0.1 sec it follows the desired trajectory again, which illustrates that the ASMC controller can handle external disturbance situation. Figure 8 depicts the comparison of ASMC, LSTM with ASMC, and GRU with ASMC. At t = 7 sec, error increases due to the disturbance for LSTM with a peak value of 0.4 degree and ASMC with 2.5 degree. Using only the LSTM or GRU model results in higher tracking error. After combining with ASMC, both of the models precisely following the desired trajectory. Following the disturbance, recurrent neural network-based models have less error than ASMC, as shown in Figure 8(b).

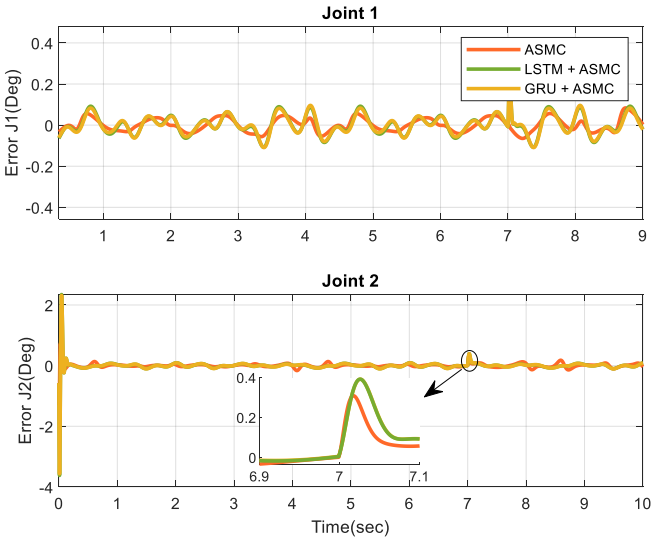

Figure 8(b): Comparison of Error (with disturbance)

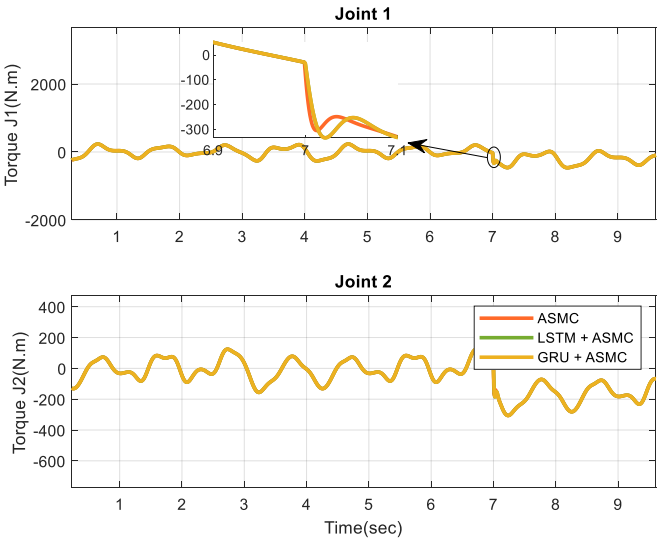

Figure 8(c): Comparison of Torque (with disturbance)

5. Conclusion

To conclude, to verify and test the proposed controller, the trajectory tracking results of ASMC is compared with a combination of LSTM or GRU and ASMC. Our experimental results show that without tuning the parameters for the Deep Learning models, control law combining LSTM or GRU with ASMC generates acceptable results for scenarios with and without disturbance. The results show that ASMC controller has lower tracking error, but with more fluctuation and a higher torque value whereas the proposed controller generates torque based on knowledge of dynamic not based on tracking error, and this ,in turn, reduces the fluctuation and chattering in control torque and tracking error. In this Deep Learning-based model, it is found that 0.003, and 0.005 learning rate are the optimum values and we also noted that increasing the epoch rises the training time. Deep learning model construction and data cleaning positively influence the output value of the network. Using position, velocity, and acceleration data, the deep learning model can precisely estimate the torque value required to follow the desired trajectory. This study shows that up to 90% of the required torque can be generated with LSTM or GRU model, instead of using Lgrangian model of the robot.

References

[1] Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel, “Backpropagation Applied to Handwritten Zip Code Recognition.”, Neural Computation 1, 1989, pp. 541-551

[2] A. Krizhevsky, I. Sutskever, and G. Hinton, “Imagenet classification with deep convolutional neural networks”, Proc. Advances in neural information processing systems, 2012, pp. 1097-1105

[3] H. Su, W. Qi, C. Yang, A. Aliverti, G. Ferrigno, and E. De Momi, “Deep Neural Network Approach in Human-Like Redundancy Optimization for Anthropomorphic Manipulators”, IEEE Access, vol. 7, pp. 124207-124216, 2019, DOI: 10.1109/ACCESS.2019.2937380.

[4] A. I. Károly, P. Galambos, J. Kuti, and I. J. Rudas, “Deep Learning in Robotics: Survey on Model Structures and Training Strategies”, IEEE Transactions on Systems, Man, and Cybernetics: Systems, vol. 51, no. 1, pp. 266-279, Jan. 2021, DOI: 10.1109/TSMC.2020.3018325.

[5] M. Zeinali, and L. Notash, “Adaptive sliding mode control with uncertainty estimator for robot manipulators”, Mechanism and Machine Theory, Volume 45, Issue 1, 2010, pp. 80-90

[6] M. Zeinali, “Adaptive Chattering-Free Sliding Mode Control Design for Robot Manipulators Based on Online Estimation of Uncertainties and Its Experimental Verification”, JOURNAL OF MECHATRONICS 3 (2), 85-97, 2015.

[7] O. Carman and P. Husek, “Adaptive fuzzy sliding mode control for electro-hydraulic servo mechanism”, Expert Systems with Applications., Volume 39, 2012, pp. 10269-10277

[8] D. Zhao, S. Li, and F. Gao, “A new terminal sliding mode control for robotic manipulators.”, IFAC Proceedings Volumes, Volume 41, Issue 2, 2008, pp. 9888-9893

[9] F.T. Mrad and S. Ahmad, “Adaptive control of flexible joint robots using position and velocity feedback”, International Journal of Control, Volume 55, 1992, pp. 1255-1277

[10] M. Zeinali, “Adaptive chattering-free sliding mode control design using fuzzy model of the system and estimated uncertainties and its application to robot manipulators”, 2015 International Workshop on Recent Advances in Sliding Modes (RASM), Istanbul, 2015, pp. 1-6, DOI: 10.1109/RASM.2015.7154652.

[11] M. Zeinali and H. Wang, “New Methodology to Design Learning Control for Robots Using Adaptive Sliding Mode Control and Multi-Model Neural Networks”, Proceedings of the 5th International Conference of Control, Dynamic Systems, and Robotics (CDSR'18) Niagara Falls, Canada – June 7 – 9, Paper No. 140 DOI: 10.11159/cdsr18.140, 2018.

[12] B.Ozyer, “Adaptive fast sliding neural control for robot manipulator.", Turk J Elec Eng & Comp Sci, 28, 2020, pp. 3154-3167

[13] S. Bansal, A. K. Akametalu, F. J. Jiang, F. Laine, and C. J. Tomlin, “Learning quadrotor dynamics using neural network for flight control”, 2016 IEEE 55th Conference on Decision and Control (CDC), Las Vegas, NV, 2016, pp. 4653-4660, DOI: 10.1109/CDC.2016.7798978.
[14] S. Edhah, S. Mohamed, A. Rehan, M. AlDhaheri, A. AlKhaja, and Y. Zweiri, “Deep Learning Based Neural Network Controller for Quad Copter: Application to Hovering Mode”, 2019 International Conference on Electrical and Computing Technologies and Applications (ICECTA), Ras Al Khaimah, United Arab Emirates, 2019, pp. 1-5, DOI:10.1109/ICECTA48151.2019.8959776.
[15] Sepp. Hochreiter, “The Vanishing Gradient Problem During Learning Recurrent Neural Nets and Problem Solutions”, International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems., 1998, 6. 107-116. DOI:10.1142/S0218488598000094.
[16] F. Chollet, “Deep Learning with Python”, 2018, p. 196
[17] Y. Bengio, P. Simard, and P. Frasconi, “Learning long-term dependencies with gradient descent is difficult”, IEEE Transactions on Neural Networks, vol. 5, no. 2, pp. 157-166, March 1994, DOI: 10.1109/72.279181.
[18] S. Hochreiter and J. Schmidhuber, “Long Short-Term Memory”, Neural Computation, vol. 9, no. 8, pp. 1735-1780, Nov. 1997, DOI: 10.1162/neco.1997.9.8.1735.
[19] N. Liu, L. Li, B. Hao, L. Yang, T. Hu, T. Xue, and S. Wang, “Modeling and Simulation of Robot Inverse Dynamics Using LSTM-Based Deep Learning Algorithm for Smart Cities and Factories”, IEEE Access, vol. 7, pp. 173989-173998, 2019, DOI: 10.1109/ACCESS.2019.2957019.
[20] K. Cho, B. V. Merrienboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio, “Learning phrase representations using RNN encoder-decoder for statistical machine translation.”, arXiv preprint arXiv:1406.1078 (2014).