Available online at www.sciencedirect.com

ScienceDirect

IFAC PapersOnline CONFERENCE PAPER ARCHIVE

IFAC PapersOnLine 55-15 (2022) 148–153

Adaptive Control of Robotic Manipulators using Deep Neural Networks

Irfan Ganie and S. Jagannathan

Electrical and Computer Engineering Department,
Missouri University of Science and Technology,
Rolla, MO 65409, USA,
(e-mail: iag76b@mst.edu).

Abstract: In this paper, we present a lifelong deep learning-based control of robotic manipulators with nonstandard adaptive laws using singular value decomposition (SVD) based direct tracking error driven (DTED) approach. Moreover, we incorporate concurrent learning (CL) to relax persistency of excitation condition and elastic weight consolidation (EWC) for lifelong learning on different tasks in the adaptive laws. Simulation results confirm theoretical conclusions.

Keywords: SVD, deep neural networks, elastic weight consolidation, concurrent learning, lifelong learning

1. INTRODUCTION

Advanced control techniques are necessary in order for the robot manipulators to be applied successfully in industry applications. The well-known computed torque scheme from Lewis et al. (1998) for robot manipulator control suffers from performance degradation in the presence of model uncertainties. The addition of a neural network (NN) in the feedback-loop compensates from smooth uncertainties.

In Bengio et al. (2010), it is shown that NN can approximate continuous functions uniformly to an arbitrary degree of accuracy. For the approximation, an adaptive weight tuning law should be derived, and closed-loop stability must be demonstrated. Many NN weight updates, such as gradient descent in Benesty et al.(1992), have been introduced. Although the tracking performance is acceptable for a single task they do not focus on learning performance. Available online NN control techniques from Lewis et al. (1998) are unsuitable for lifelong learning, which requires a continuous stream of available information due to changes in tasks and dynamics. In addition, established online NN robot control schemes in Werbos et al. (1991), use either a single-layer NN with basis functions which is difficult to select for an unknown system or two-layer NN in Lewis et al. (1998) with nonstandard weight tuning.Convergence due to the addition of hidden layers is not straightforward.

Nonlinear functions, which can be approximated by a NN having a polynomial number of nodes with p layers, may need an exponential number of nodes with $p-1$ layers as shown in Bengio et al.(2010). Moreover, in Bengio et al. (2010) it was shown that deep NN with more than two hidden layers can approximate any nonlinear function smoothly. A two-layer NN based control technique in continuous time in Lewis et al. (1998), is presented using Taylor series expansion. Extension of this method by Lewis et al. (1998) to more hidden layers are challenging and is not reported.

In the literature, backpropagation as shown in Baldi et al. (1995) using stochastic gradient descent method (SGD) and supervisory learning is widely employed. Though backpropagation method can be used with deep NN, however, it is proven to be unsuitable for feedback control applications due to the difficulty in finding target values and demonstrating stability analysis. Due to the number of layers, a deep NN trained by backpropagation using SGD can face vanishing and exploding gradient Zhang et al. (2018), Wang et al (2021).

In addition, in order to show the convergence of the weight estimates and robustness in the presence of parameter drift, a persistency of excitation (PE) condition is normally required as shown in Bitmead et al. (1984) and Narendra et al. (1989). Since it is difficult to either verify or guarantee a PE condition, modified updates have been introduced in Narendra et al. (1989). While PE condition can be relaxed in Lewis et al. (1998) with these methods from Narendra et al. (1989), a uniformly ultimately bounded (UUB) performance can only be determined. In addition, it is not well understood how to monitor the PE effects for a deep NN with hidden layers.

Recently, an alternate condition in Chaudhary et al. (2010) known as the interval excitation condition, which is similar to PE, but only over a fixed time interval can be utilized by using online data for traditional adaptive control. This concurrent learning method is shown to provide exponentially parameter convergence in the ideal case of no disturbances and unmodeled dynamics. However, it is not used in conjunction with deep NN weight tuning for adaptive NN control. In addition, when the number of tasks is increased, the NN with online learning suffers from catastrophic forgetting Kirkpatrick et al. (2016); hence the performance gets degraded.

The main contribution of this paper has three folds. First a deep learning based NN control scheme is introduced wherein multiple hidden layers are tuned simultaneously online and without any offline learning phase. The multiple layered NN weight adjustment is accomplished by using tracking error directly instead of propagating the error backward in the case of traditional SGD-based backpropagation. With the proposed SVD based direct tracking error driven (DTED) approach, the

10.1016/j.ifacol.2022.07.623

vanishing and explosive gradients are not observed since the error is directly used at each layer.

Secondly, an experience replay buffer with sufficiently rich online data is used as part of deep NN based weight adaptation laws for relaxing the PE condition. This is different from the Chowdhary et al. (2010) in a sense that SVD based updates are used instead of the gradient.

Finally, we add an additional term in the weight tuning from elastic weight consolidation (EWC) to mitigate forgetting. It will be shown that with the additional term, the overall system maintains the performance while maintaining the closed-loop stability. Simulation results by using a *n*-link robot manipulator confirm theoretical conclusions.

Notation: Throughout the paper, the notations adopted are $\mathbb{R}$ denotes the set of real numbers, $\|.\|$ denotes the Euclidean norm and induced two norms for a vector and matrix respectively, $\lambda_{min}, \lambda_{max}$ denote the minimum and maximum eigen values respectively, tr (‘) means trace of a matrix, and I represent identity matrix. Next, the manipulator dynamics are revisited before introducing the proposed method.

2. ROBOT DYNAMICS

The dynamics of manipulator having *n* degrees-of-freedom is given in Lewis et al. (1998) as

$$M(p)\ddot{p} + V_m(p,\ \dot{p})\,\dot{p} + G\,(p) + F_d(\dot{p}) + \tau_d = \tau, \quad (1)$$

where $p,\ \dot{p}(t),\ \ddot{p}(t) \in R^n$ represent joint angle, angular velocity, acceleration, respectively, $M(p) \in \mathbb{R}^{n\times n}$ is the symmetric positive definite inertia matrix, $V_m(p,\dot{p}) \in \mathbb{R}^{n\times 1}$ is a Coriolis and centripetal torque matrix, $G(p) \in \mathbb{R}^{n\times 1}$ is the gravitational vector, $\tau \in \mathbb{R}^{n\times 1}$ is the joint actuator torque, $\tau_d \in \mathbb{R}^{n\times 1}$ denotes the unknown bounded disturbance and $F_d(\dot{p}) \in \mathbb{R}^{n\times 1}$ is the viscous friction. The robot dynamics has the following properties:

Property 1: $\dot{M} - 2V_m$ is skew symmetric.

Property 2: The unknown disturbance satisfies $\|\tau_d\| < b_d, b_d > 0$.

The objective is to develop a deep or multilayer NN controller so that the joint angles, p, of the robotic manipulator can track the reference input $p_d \in \mathbb{R}^n$ with bounded and small error while mitigating the unstable gradient problem and maintaining lifelong learning.

3. PROPOSED METHOD

A deep learning-based control method for robotic manipulators in the continuous-time domain using a direct tracking error-driven approach (DTED) via singular value decomposition (SVD) is being developed to enhance performance and for lifelong learning. We can use *n*-layers using this method, however for the sake of simplicity four-layer NN is considered. The weight matrix for each layer is denoted as V, W, B, D for the first, second, third, and fourth layers, respectively. The robotic filtered tracking error dynamics and the deep NN learning approach using SVD are introduced next.

The vanishing and exploding gradient as discussed in Zhang et al. (2018) that normally occurs in deep NN, as the name suggests, must be mitigated. Usually, a special type of activation function, RELU, is employed whereas in this paper we will use SVD, sigmoid and nonstandard weight tuning law to mitigate the instability of the gradients by using the singular values. Next, four-layer NN based control design is explained.

3.1 Four-layer Neural Network-based Control Design

Define tracking error and filtered tracking error as

$$e = p_d - p, \quad (2)$$

$$r = \dot{e} + \wedge e, \quad (3)$$

where $\wedge \in \mathbb{R}^{n\times n}$ is a positive diagonal matrix, $e(t) \in \mathbb{R}^n$ is the position tracking error, $r(t) \in \mathbb{R}^n$ is the filtered tracking error. Taking the derivative of (3) with respect to time and multiplying by $M(p)$ and substituting in (1), we get

$$M\dot{r} = -V_m r - \tau + \tau_d + f, \quad (4)$$

where

$$f(X) = M(p)(\ddot{p}_d + \ \wedge \dot{e}) + V_m(p,\dot{p})(\dot{p}_d + \wedge e) + G(p) + F(\dot{p}).$$

The vector $X = [1, \bar{X}] \in \mathbb{R}^{n_1+1}, \bar{X} = \left[x_1, \ldots x_{n_1}\right]^T \in \mathbb{R}^{n1}$ in our case $\bar{X} \equiv [e^T\ \dot{e}^T\ p_d^T\ \dot{p_d}^T\ \ddot{p}_d^T]$ which can be measured.

A general sort of approximate based controller is designed as shown in Lewis et al. (1998) as

$$\tau = \hat{f} + K_v r - v, \quad (5)$$

where $\hat{f}(x)$ is an estimate of $f(x)$, $K_v = K_v^T > 0$ is the $n \times n$ gain matrix and v provides robustness in the face of higher order terms. Substituting (5) in (4) yields

$$M\dot{r} = -V_m r - K_v r + \tilde{f} + \tau_d + v, \quad (6)$$

where $\tilde{f}(X) = f(X) - \hat{f}(X)$, the unknown nonlinear function approximation error given by a four-layer NN.

Considering $\hat{y} = [\hat{y}_1, \ldots \hat{y}_{n2}]^T \in \mathbb{R}^{n2}$. Let a NN estimate of $f(x)$ by using a four -layer NN is given by

$$\hat{f}(X) = \hat{D}^T\sigma_1\left(\hat{W}^T\sigma_2\left(\hat{Z}^T\sigma_3\left(\hat{V}^T X\right)\right)\right) + \epsilon_1, \quad (7)$$

where ϵ_1 is the NN reconstruction error that is considered to be bounded such that $\|\epsilon_1\| < \epsilon_N$, where ϵ_N is known constant and $\hat{V}, \hat{Z}, \hat{W}, \hat{D}$ represent the actual NN weights given by tuning scheme. The first hidden layer has m_1 neurons, the second hidden layer has m_2 neurons, the third hidden layer has m_3 neurons, the NN weight matrices $\hat{V} \in \mathbb{R}^{(n_1+1)\times m_1}, \hat{Z} \in \mathbb{R}^{(m_1+1)\times m_2}, \quad \hat{W} \in \mathbb{R}^{(m_2+1)\times m_3}, \hat{D} \in \mathbb{R}^{(m_3+1)\times n_2}$. We use $\sigma_1(.) \in \mathbb{R}^{m_3+1}, \sigma_2(.) \in \mathbb{R}^{m_2+1}, \sigma_3(.) \in \mathbb{R}^{m_3+1}$. Let’s define $\tilde{\sigma}_k = \sigma_k - \hat{\sigma}_k, \tilde{V} = V - \hat{V}, \tilde{W} = W - \hat{W}, \tilde{Z} = Z - \hat{Z}, \tilde{D} = D - \hat{D}$. Let’s define $\sigma_1 = \sigma(W^T\sigma_2), \sigma_2 = \sigma(Z^T\sigma_3), \sigma_3 = \sigma(V^T X)$.

Moreover, we express $\hat{\sigma}_1' = \frac{d\sigma_1(s)}{ds}|\hat{W}^T\hat{\sigma}_2,\ \widehat{\sigma_2}' = \frac{d\sigma_2}{ds}\Big|s = \hat{Z}^T\hat{\sigma}_2,\ \widehat{\sigma}_3 = \frac{d\sigma_3(s)}{ds}\Big| s = \hat{V}^T X$.

Using value of $\tilde{f} = f - \hat{f}$ in (6)

$$M\dot{r} = -(K_v + V_m)r + f - \hat{f} + v + \tau_d \quad (8)$$

$$M\dot{r} = -(K_v + V_m)r - \hat{D}^T\sigma\left(\hat{W}^T\sigma\left(\hat{Z}^T\sigma\left(\hat{V}^T X\right)\right)\right) + D^T\sigma\left(W^T\sigma\left(Z^T\sigma(V^T X)\right)\right) + \epsilon + \tau_d + v. \quad (9)$$

Following the procedure Lewis et al (1996) one can write

$$f - \hat{f} = D^T\sigma_1\left(W^T\sigma_2\left(Z^T\sigma_3(V^T X)\right)\right) - \hat{D}^T\sigma_1\left(\hat{W}^T\sigma_2\left(\hat{Z}^T\sigma_3\left(\hat{V}^T X\right)\right)\right) + \epsilon \quad (10)$$

$$f - \hat{f} = D^T\sigma_1 - \hat{D}^T\hat{\sigma}_1 + \epsilon\ . \quad (11)$$

Add and subtract $D^T\hat{\sigma}_1$ in (9) we get

$$f - \hat{f} = D^T\tilde{\sigma}_1 - \hat{D}^T\hat{\sigma}_1 + D^T\hat{\sigma}_1 + \epsilon\,, \tag{12}$$

add and subtract $\hat{D}^T\tilde{\sigma}_1$ in (11)

$$f - \hat{f} = \tilde{D}^T\,\hat{\sigma}_1 + \tilde{D}^T\tilde{\sigma}_1 + \hat{D}\tilde{\sigma}_1 + \epsilon\,. \tag{13}$$

Taylor series expansion of $\sigma_1\left(W^T\sigma_2\left(Z^T\sigma_3(V^TX)\right)\right)$ at $\hat{W}^T\sigma_2\left(\hat{Z}^T\sigma_3\left(\hat{V}^TX\right)\right)$, $\sigma_2\left(Z^T\sigma_3(V^TX)\right)$ at $\hat{Z}^T\sigma_3\left(\hat{V}^TX\right)$ and $\sigma_3\left(\hat{V}^TX\right)$ at $\hat{V}^TX$ can be written as

$$\sigma_1\left(W^T\sigma_2\left(Z^T\sigma_3(V^TX)\right)\right) = \sigma_1\left(\hat{W}^T\sigma_2\left(\hat{Z}^T\sigma_3\left(\hat{V}^TX\right)\right)\right) + \hat{\sigma}_1'\left[W^T\sigma_2\left(Z^T\sigma_3(V^TX)\right) - \hat{W}^T\sigma_2\left(\hat{Z}^T\sigma_3\left(\hat{V}^TX\right)\right)\right] + O_1(.),$$

$$\sigma_2\left(Z^T\sigma_3(V^TX)\right) = \sigma_2\left(\hat{Z}^T\sigma_3\left(\hat{V}^TX\right)\right) + \widehat{\sigma_2}'\left[Z^T\sigma_3(V^TX) - \hat{Z}^T\sigma_3\left(\hat{V}^TX\right)\right] + O_2(.)$$

$$\sigma_3(V^TX) = \hat{\sigma}_3 + \hat{\sigma}_3'\tilde{V}^TX + O_3(.)\,. \tag{14}$$

Note that $O_1(.)$, $O_2(.)$, $O_3(.)$ are short notations for $O(W^T\sigma_2\left(Z^T\sigma_3(V^TX)\right) - \hat{W}^T\sigma_2\left(\hat{Z}^T\sigma_3\left(\hat{V}^TX\right)\right))$, $O\left(Z^T\sigma_3(V^TX) - \hat{Z}^T\sigma_3\left(\hat{V}^TX\right)\right)$, $O(\tilde{V}^TX)$ respectively.

From (14) it is evident that

$$\tilde{\sigma}_1 = \widehat{\sigma_1}'\left[W^T\sigma_2 - \hat{W}^T\hat{\sigma}_2\right] + O_1(.)\,, \tag{15}$$

$$\tilde{\sigma}_2 = \hat{\sigma}_2'\left[Z^T\sigma_3 - \hat{Z}^T\hat{\sigma}_3\right] + O_2(.)\,, \tag{16}$$

$$\tilde{\sigma}_3 = \hat{\sigma}_3'[\tilde{V}^TX] + O_3(.)\,, \tag{17}$$

Substituting (15) into (12)

$$f - \hat{f} = \tilde{D}^T\left(\hat{\sigma}_1 + \hat{\sigma}_1'\left[W^T\sigma_2 - \hat{W}^T\hat{\sigma}_2\right] + O_1(.)\right) + \hat{D}^T\tilde{\sigma}_1 + \epsilon \tag{18}$$

where

$$W^T\sigma_2 = \left(\tilde{W} + \hat{W}\right)(\tilde{\sigma}_2 + \hat{\sigma}_2) = \tilde{W}\tilde{\sigma}_2 + \tilde{W}\hat{\sigma}_2 + \hat{W}\tilde{\sigma}_2 + \hat{W}\widehat{\sigma_2}\,. \tag{19}$$

Using (19) in (18) and rearranging (18) yields

$$f - \hat{f} = \tilde{D}^T\left(\hat{\sigma}_1 - \hat{\sigma}_1'\hat{W}^T\hat{\sigma}_2 + \hat{\sigma}_1'\hat{W}^T\tilde{\sigma}_2\right) + \hat{D}^T\tilde{\sigma}_1 + \left[\tilde{D}^T\left(\hat{\sigma}_1'\left[\tilde{W}^T\sigma_2 + \hat{W}^T\hat{\sigma}_2\right] + O_1(.)\right) + \epsilon\right]. \tag{20}$$

From (19), (20) can be written as

$$f - \hat{f} = \tilde{D}^T\left(\hat{\sigma}_1 - \hat{\sigma}_1'\hat{W}^T\hat{\sigma}_2 - \hat{\sigma}_1'\hat{W}^T\hat{\sigma}_2'\hat{Z}^T\hat{\sigma}_3 - \hat{\sigma}_1'\hat{W}^T\hat{\sigma}_2'\hat{Z}^T\left(\hat{\sigma}_3'\left(\hat{V}^TX\right)\right)\right) + \hat{D}^T\hat{\sigma}_1'\tilde{W}^T\left(\hat{\sigma}_2 - \hat{\sigma}_2'\hat{Z}^T\hat{\sigma}_3 - \hat{\sigma}_2'\hat{Z}^T\hat{\sigma}_3'\hat{V}^TX\right) + \hat{D}^T\hat{\sigma}_1'\tilde{W}^T\hat{\sigma}_2'\tilde{Z}^T\left(\hat{\sigma}_3 - \hat{\sigma}_3'\hat{V}^TX\right) + \hat{D}^T\left(\widehat{\sigma_1}'\hat{W}^T\hat{\sigma}_2'\hat{Z}^T\widehat{\sigma_3}'\tilde{V}X\right) + \hat{D}^T\widehat{\sigma_1}'W^TO_2(.) + \hat{D}^T\hat{\sigma}_1'\tilde{W}^T\hat{\sigma}_2'Z^T\sigma_3 + \hat{D}^T\widehat{\sigma_1}'\,\hat{W}^T\hat{\sigma}_2'Z^TO_3(.) + \hat{D}^T\hat{\sigma}_1'\hat{W}^T\hat{\sigma}_2'\tilde{Z}^T\hat{\sigma}_3'V^TX + \left[\tilde{D}^T(\hat{\sigma}_1'W^T\sigma_2) + \tilde{D}\hat{\sigma}_1'\hat{W}^T\hat{\sigma}_2'\hat{Z}^T\hat{\sigma}_3 + \tilde{D}^T\left(\hat{\sigma}_1'\hat{W}^T\widehat{\sigma_2}'\hat{Z}^T\hat{\sigma}_3'\hat{V}^TX\right) + D^TO_1(.) + \epsilon\right]. \tag{21}$$

On further simplifying (21) we get

$$f - \hat{f} = \tilde{D}^T\left(\hat{\sigma}_1 - \hat{\sigma}_1'\hat{W}^T\hat{\sigma}_2 - \hat{\sigma}_1'\hat{W}_1^T\hat{\sigma}_2'\hat{Z}^T\hat{\sigma}_3 - \hat{\sigma}_1'\hat{W}_1^T\hat{\sigma}_2'\hat{Z}^T\hat{\sigma}_3'\hat{V}^TX\right) + \hat{D}^T\hat{\sigma}_1'\tilde{W}^T\left(\hat{\sigma}_2 - \hat{\sigma}_2'\hat{Z}^T\hat{\sigma}_3 - \hat{\sigma}_2'\hat{Z}^T\hat{\sigma}_3'\hat{V}^TX\right) + \hat{D}^T\hat{\sigma}_1'\tilde{W}^T\hat{\sigma}_2'\tilde{Z}^T\left(\hat{\sigma}_3 - \hat{\sigma}_3'\hat{V}^TX\right) + \hat{D}^T\left(\widehat{\sigma_1}'\hat{W}^T\hat{\sigma}_2'\hat{Z}^T\widehat{\sigma_3}'\tilde{V}X\right) + \bar{\epsilon}, \tag{22}$$

with

$$\bar{\epsilon} = \hat{D}^T\widehat{\sigma_1}'W^TO_2(.) + \hat{D}^T\hat{\sigma}_1'\tilde{W}^T\hat{\sigma}_2'Z^T\sigma_3 + \hat{D}^T\widehat{\sigma_1}'\,\hat{W}^T\hat{\sigma}_2'Z^TO_3(.) + \hat{D}^T\hat{\sigma}_1'\hat{W}^T\hat{\sigma}_2'\tilde{Z}^T\hat{\sigma}_3'V^TX + \left[\tilde{D}^T(\hat{\sigma}_1'W^T\sigma_2) + \tilde{D}\hat{\sigma}_1'\hat{W}^T\hat{\sigma}_2'\hat{Z}^T\hat{\sigma}_3 + \tilde{D}^T\left(\hat{\sigma}_1'\hat{W}^T\widehat{\sigma_2}'\hat{Z}^T\hat{\sigma}_3'\hat{V}^TX\right) + D^TO_1(.) + \epsilon\right]. \tag{23}$$

Following the procedure from Lewis et al. (1996) (Lemma 4.3.1), and using the fact that $\sigma_k(.)$ and its derivatives are bounded, and from (13-14) we can show that

$$\|O_3(.)\| \le c_o + c_1\|X\| + c_2\left\|\hat{V}\right\|_F\|X\|, \tag{24}$$

$$\|O_2(.)\| \le c_3 + c_4\left\|\hat{Z}\right\|_F\,, \tag{25}$$

$$\|O_1(.)\| \le c_5 + c_6\left\|\hat{W}\right\|_F, \tag{26}$$

where $c_i, i \in \{0,..6\}$ are positive constants

Using the value of $D^T\sigma_1\left(W^T\sigma_2\left(Z^T\sigma_3(V^TX)\right)\right) - \hat{D}^T\sigma_1\left(\hat{W}^T\sigma_2\left(\hat{Z}^T\sigma_3\left(\hat{V}^TX\right)\right)\right)$ from (28) in (9) we get

$$M\dot{r} = -(K_v + V_m)r + \tilde{D}^T\left(\hat{\sigma}_1 - \hat{\sigma}_1'\hat{W}^T\hat{\sigma}_2 - \hat{\sigma}_1'\hat{W}_1^T\hat{\sigma}_2'\hat{Z}^T\hat{\sigma}_3 - \hat{\sigma}_1'\hat{W}_1^T\hat{\sigma}_2'\hat{Z}^T\hat{\sigma}_3'\hat{V}^TX\right) + \hat{D}^T\hat{\sigma}_1'\tilde{W}^T\left(\hat{\sigma}_2 - \hat{\sigma}_2'\hat{Z}^T\hat{\sigma}_3 - \hat{\sigma}_2'\hat{Z}^T\hat{\sigma}_3'\hat{V}^TX\right) + \hat{D}^T\hat{\sigma}_1'\hat{W}^T\hat{\sigma}_2'\tilde{Z}^T\left(\hat{\sigma}_3 - \hat{\sigma}_3'\hat{V}^TX\right) + \hat{D}^T\left(\widehat{\sigma_1}'\hat{W}^T\hat{\sigma}_2'\hat{Z}^T\widehat{\sigma_3}'\tilde{V}X\right) + \bar{\epsilon} + \tau_d + v. \tag{27}$$

This $M\dot{r}$ is used in the stability analysis of the system using the proposed update laws and control input. Next the traditional backpropagation for four-layer NN is explained.

The backpropagation method shown in Baldi et al. (1995) provides a way to compute the gradient of the cost function. Let x being the NN input, and a^l is the corresponding activation function for the input layer. For layers 2,3, ... l define $z^l = w^la^{l-1} + b^l$ and $a^l = \sigma(z^l)$.

Compute the output error as

$$\delta^l = \nabla_aC \odot \sigma'(z^l)\,. \tag{28}$$

The activation a_j^l of the jth neuron in the lth layer is related to activations in the$(l-1)th$ layer through this equation

$$a_j^l = \sigma(\textstyle\sum_k w_{jk}^l\, a_k^{l-1} + b_j^l)\,. \tag{29}$$

For each layer l= $l-1, l-2\ldots$,2 define δ^l as

$$\delta^l = ((w^{l+1})^T\delta^{l+1}) \odot \sigma'(z^L)\,. \tag{30}$$

Now consider a four-layer NN with V, Z, W, D the weights for the 1st ,2nd, 3rd, and 4th layer, respectively and the output of 1st ,2nd, 3rd, and 4th layer as $z_l = \sigma(V^Tx)$, $h_l = \sigma(Zz_l)$, $g_l = \sigma(WF_i)$, $y = \sigma(DG_k)$ where x is the input. Now errors can be calculated as

$$\delta^4 = (a^L - y) \odot \sigma'(Dg_l), \tag{31}$$

$$\delta^3 = D^T\delta^4 \odot \sigma'\left(Wh_l\right), \tag{32}$$

$$\delta^2 = W^T\delta^3 \odot \sigma'(Zz_l), \tag{33}$$

$$\delta^1 = Z^T\delta^2 \odot \sigma'(Vx). \tag{34}$$

Update laws can be defined as

$$\dot{D} = \sigma(WF_i)(\delta^4)^T, \tag{35}$$

$$\dot{W} = \sigma(Zz_l)(\delta^3)^T, \tag{36}$$

$$\dot{Z} = \sigma(V^Tx)(\delta^2)^T, \tag{37}$$

$$\dot{V} = x(\delta^1)^T, \tag{38}$$

where $a^L - y$ is the output error which is used to tune the NN weights in the case of traditional SGD-based backpropagation. Replace $a^L - y = r$ as defined in (3) for filtered tracking error-based backpropagation since there are no targets that can be generated for the nonlinear dynamics. Normally for traditional backpropagation method with SGD, the NN output will be compared with a target output and this error is utilized to tune the NN layered weights. In contrast in the case of

tracking error-based backpropagation, the filtered tracking error is backpropagated back for tuning the layered weights.

But as the number of layers increases, the vanishing gradient problem still arises in the traditional and tracking error-based backpropagation method, which must be mitigated to improve the learning effectiveness. Therefore, a SVD based DTED approach is introduced next.

3.2 Weight Update Laws Using Singular Value Decomposition

The weight update laws for four layers are given below

$$\dot{\hat{D}} = C\big(\hat{\sigma}_1 - \dot{A}_d\widehat{W}^T A_b - \dot{A}_d\widehat{W}^T\dot{A}_b\hat{Z}^T A_v - \dot{A}_d\widehat{W}^T\dot{A}_b\hat{Z}^T\dot{A}_v\hat{V}^T X\big)r^T, \quad (39)$$

$$\dot{\widehat{W}} = F\widehat{D}^T\dot{A}_d\big(A_b - \dot{A}_b\hat{Z}^T A_v - \dot{A}_b\hat{Z}^T\dot{A}_v\hat{V}^T X\big)r^T, \quad (40)$$

$$\dot{\hat{Z}} = Q\widehat{D}^T\dot{A}_d\widehat{W}^T\dot{A}_b\big(A_v - \dot{A}_v\hat{V}^T X\big)r^T\ , \quad (41)$$

$$\dot{\hat{V}} = G\widehat{D}^T\big(\dot{A}_d\widehat{W}^T\dot{A}_b\hat{Z}^T\dot{A}_v X\big)r^T\ , \quad (42)$$

$$A_v = svd\left(\sigma\big(\hat{V}^T X\big)\right), \quad (43)$$

$$A_b = svd(\sigma\left(\hat{Z}^T\sigma\big(\hat{V}^T X\big)\right), \quad (44)$$

$$A_d = svd\left(\sigma\left(\widehat{W}^T(\sigma(\hat{Z}^T\sigma\big(\hat{V}^T X\big)\right)\right), \quad (45)$$

where $\dot{A}_d, \dot{A}_b, \dot{A}_v$ are the matrix updates of the matrices A_d, A_b, A_v after taking SVDs respectively, X is the input defined in (4) and r is the tracking error defined in (3). C, F, G and Q are positive definite tuning rate matrix. The SVD updates for these matrices can be obtained as follows.

Lemma 1: A matrix 'A 'which can be either square or rectangular can be decomposed as USV^T as shown in Wang et al. (2021) by applying SVD. If A is a $m \times n$ matrix of rank *k*, then U is $m \times k$, S is $k \times k$ diagonal matrix of singular values, V is $n \times k$. When the matrix 'A' is varying with respect to time, and the matrices V and U satisfies

$$U^T U = V^T V = I_k, \quad (46)$$

and the derivative of A can be written as

$$\dot{A} = \dot{U}SV^T + U\dot{S}V^T + US\dot{V}^T. \quad (47)$$

The constraints in (46) imply that $\dot{U}$, $\dot{V}$ are also constrained. Let's focus on U first, consider the right part of the equation (46) to get

$$\dot{U}^T U + U^T\dot{U} = 0, \quad (48)$$

$$\dot{U}^T U = -U^T\dot{U}, \quad (49)$$

Hence matrix $d\omega_U = U^T\dot{U}$ is skew-symmetric. Let $m \times (m-k)$ is U_ϵ such that [U U_ϵ] is an orthogonal matrix then $\dot{U}$ can be computed as

$$\dot{U} = Ud\omega_U + U_\epsilon dK_U\ , \quad (50)$$

where dK_U is $(m-k) \times k$ matrix, which is not constrained, now $\dot{V}$ can be expressed in a similar way as

$$\dot{V} = Vd\omega_V + V_\epsilon dK_V, \quad (51)$$

$dK_v\ is\ (n-k)\ \times k$ matrix, and $d\omega_V = V^T\dot{V}$ is $k \times k$ skew-symmetric matrix. Left multiplying equation (51) by U^T and right multiplying by V we get,

$$U^T\dot{A}V = d\omega_U S + \dot{S} + Sd\omega_V^T\ , \quad (52)$$

$d\omega_U$ and $d\omega_V$ are skew-symmetric, so they have zero diagonal elements; hence $d\omega_U$S and S$d\omega_V^T$ must also have zero diagonal elements. This means that we can split (52) into two components, let's say, $\dot{P} = U^T\dot{A}V$, and we will use $\odot$ it as a Hadamard product. So, the diagonal product of (52) is

$$\dot{S} = I_k \odot \dot{P}\ , \quad (53)$$

and off-diagonal as

$$\overline{I_k} \odot \dot{P} = d\omega_U S - Sd\omega_V, \quad (54)$$

where $\overline{I_k}$ is $k \wedge k$ matrix with zero diagonals and ones every other place. Now taking the transpose of (54)

$$\overline{I_k} \odot \dot{P}^T = -Sd\omega_U + d\omega_V S\ . \quad (55)$$

Now right multiply equation (51) by S and left multiply equation (52) by S and then add we get

$$\overline{I_k} \odot \left[\dot{P}S + S\dot{P}^T\right] = d\omega_U S^2 - S^2 d\omega_U, \quad (56)$$

This is solved by

$$d\omega_U = F \odot \left[\dot{P}S + S\dot{P}^T\right], \quad (57)$$

where

$$F_{ij} = \begin{cases} \frac{1}{\left(s_j^2 - s_i^2\right)} & i \neq j \\ 0 & i = j \end{cases}\ , \quad (58)$$

where s_j, s_i are the singular values of S.

Similarly,

$$d\omega_V = F \odot \left[S\dot{P} + \dot{P}^T S\right], \quad (59)$$

Similarly, if we have to find dK_U , we left multiply equation (47) by U_ϵ^T

$$U_\epsilon^T\dot{A} = dK_U SV^T, \quad (60)$$

$$dK_U = U_\epsilon^T\dot{A}VS^{-1}, \quad (61)$$

similarly,

$$dK_V = V_\epsilon^T\dot{A}US^{-1}\ , \quad (62)$$

where

$$U_\epsilon U_\epsilon^T = I - UU^T\ , \quad (63)$$

$$V_\epsilon V_\epsilon^T = I - VV^{T,} \quad (64)$$

Now using (63), (64) in (60), (61) and replacing the values in (50),(51) and (53) we get

$$\dot{U} = U(F \odot [U^T\dot{A}VS + SV^T\dot{A}U] + (I_m - UU^T)\dot{A}VS^{-1} \quad (65)$$

$$\dot{S} = I_k \odot \left[U^T\dot{A}V\right], \quad (66)$$

$$\dot{V} = V(F \odot [SU^T\dot{A}V + V^T\dot{A}^T US]) + (I_n - VV^T)\dot{A}^T US^{-1} \quad (67)$$

Using (65-67) in (47) we get $\dot{A}$,where $\odot$ is the Hadamard product, *I* is the identity matrix and S contains singular values of A, *F* consists of zero diagonal and non-zero off diagonal elements. We shall control the off-diagonal elements of F by regulating the singular values of A for gradient stabilization. As demonstrated in (58), if the difference in singular values of A is small, F will be very big, resulting in an explosion gradient, whereas if the difference is very large, F will be near to zero, resulting in a vanishing gradient. The SVD-based technique keeps the difference of the singular values of the updated matrix close to one for gradient stabilization and by updating the gradient continuously with respect to time.

Remark 1: This Lemma will be employed to generate NN weight tuning laws by using SVD of the gradients of the NN activation function. As a consequence, we will show that vanishing and explosive gradient problems can be mitigated. Next the definition of PE condition as shown in Narendra et al. (1989) is introduced.

Definition 1. A bounded vector signal $\sigma(t)$ is persistently exciting over an interval $[t, t+T], T > 0$ and $t > t_0$ if $\gamma > 0$ exists such that

$$\int_t^{t+T} \sigma(t)\,\sigma^T(t)d\tau \geq \gamma I. \quad (68)$$

For τ to be bounded, it is important to show that *r* is bounded and the weights are bounded; we can easily show that r is bounded but in order for the weights to be bounded, PE condition as shown in Narendra et al. (1989) is required.

The PE condition is difficult to verify or guarantee. Moreover, from the above equation (68), it is not suitable to monitor it online. So, to overcome this, a new set of adaptive laws based on concurrent learning are used.

3.3 Relaxation of PE using Concurrent Learning

The tracking error *r* will converge to a compact set with origin is the equilibrium point. However, the NN weight error convergence cannot be guaranteed in the presence of disturbances and unmodeled dynamics without the PE condition as given in Narendra et al. (1989). When there are disturbances and unmodeled dynamics, NN weights obtained online may be unbounded, so this unboundedness of weights may lead to drifting of weight estimates.

In this paper, a concurrent learning (CL) scheme, as discussed in Chaudhary et al. (2010) is used. The basic idea of CL is to use the recorded system dynamics data. This yields a negative definite parameter estimation error term by which the convergence of parameters is obtained provided the condition of finite excitation (only a finite time excitation is required) is satisfied. Also, in online learning, this condition can be checked by checking the positivity of the smaller singular value of the regressor matrix. Let i = {1,2, 3…. p}, denotes the index of data points stored say i and Γ is positive definite learning matrix. The concurrent learning-based weight update laws can be written as

$$\dot{\hat{D}} = C\big(A_d - \dot{A}_d\widehat{W}^T A_b - \dot{A}_d\widehat{W}_1^T\dot{A}_b\hat{Z}^T A_v - \dot{A}_d\widehat{W}_1^T\dot{A}_b\hat{Z}^T\dot{A}_v\hat{V}^T X\big)r^T - \zeta_1\hat{D}\,, \quad (69)$$

$$\dot{\widehat{W}} = F\hat{D}^T\dot{A}_d\big(A_b - \dot{A}_b\hat{Z}^T A_v - \dot{A}_b\hat{Z}^T\dot{A}_v\hat{V}^T X\big)r^T - \zeta_2\widehat{W}, \quad (70)$$

$$\dot{\hat{Z}} = Q\hat{D}^T\dot{A}_d\widehat{W}^T\dot{A}_b\big(A_v - \dot{A}_v\hat{V}^T X\big)r^T - \zeta_3\hat{Z}\,, \quad (71)$$

$$\dot{\hat{V}} = G\hat{D}^T\big(\dot{A}_d\widehat{W}^T\dot{A}_b\hat{Z}^T\dot{A}_v X\big)r^T - \zeta_4\hat{V}\,, \quad (72)$$

where

$$\zeta_1 = \sum_{l=1}^{p} C\big(A_d\,(i) - \dot{A}_d(i)\widehat{W}^T A_b(i) - \dot{A}_d(i)\widehat{W}_1^T\dot{A}_b(i)\hat{Z}^T A_v(i) - \dot{A}_d(i)\widehat{W}_1^T\dot{A}_b(i)\hat{Z}^T\dot{A}_v(i)\hat{V}^T X(i)\big)\,,$$

$$\zeta_2 = \sum_{i=1}^{p} F\hat{D}^T\dot{A}_d(i)\big(A_b(i) - \dot{A}_b(i)\hat{Z}^T A_v(i) - \dot{A}_b(i)\hat{Z}^T\dot{A}_v(i)\hat{V}^T X(i)\big)\,,$$

$$\zeta_3 = \sum_{i=1}^{p} Q\hat{D}^T\dot{A}_d(i)\widehat{W}^T\dot{A}_b(i)\big(A_v(i) - \dot{A}_v(i)\hat{V}^T X(i)\big)\,,$$

$$\zeta_4 = \sum_{i=1}^{p} G\hat{D}^T\big(\dot{A}_d(i)\widehat{W}^T\dot{A}_b(i)\hat{Z}^T\dot{A}_v(i)X(i)\big)\,,$$

Define matrix of $\zeta = diag\{\zeta_1\ \zeta_2\ \zeta_3\ \zeta_4\}$.

Condition 1

$$\exists\ \lambda' > 0, \exists T > \Delta t: \forall t \geq T,\ \lambda_{min}\{\zeta\} \geq \lambda.'$$

So, from the above condition, the system trajectories are required to be finite time exciting say the finite time is *T* at which the ζ has non-zero determinants or is full rank, after this the data that is recorded during $t \in [0, T]$ is utilized for all *t* greater than T. Using the concurrent learning in weight update laws the tracking error *r* will converge to a compact set around zero and the NN weight error convergence is guaranteed when the condition 1 satisfies without requiring PE condition. Proof of stability has been omitted because of space constraints.

Though the online learning schemes presented so far ensure stability of the closed-loop system, they do not help with lifelong learning wherein forgetting aspect is studied. Elastic weight consolidation (EWC) is used to overcome the issue of catastrophic forgetting as shown in Kirkpatrick et al. (2016). EWC slows the learning on certain weights on the basis of their importance for previous tasks.

3.4 Lifelong Learning using Elastic Weight Consolidation

In SGD, the NN will learn one task only forgetting all other ones learned before this is known as catastrophic forgetting. So, to overcome this issue, EWC using the Bayesian approach in Kirkpatrick et al. (2016) is proposed. The EWC consists of adding a penalty term to the update laws which constrains the NN weights to stay in the area of low error around the optimal weights of previous task, hence prevents catastrophic forgetting while learning the new task. In Kirkpatrick et al. (2016), the probability of parameter θ given two different task training data $\phi_A\ and\ \phi_B$ using Bayes rule can be obtained

$$logp\big(\theta|\phi_{A,}\,\phi_B\big) = logp(\phi_B|\theta) + logp(\theta|\phi_A)logp(\phi_B) \quad (73)$$

The left side of the above equation contains the posterior distribution term which cannot be computed. So, EWC approximates it by a gaussian distribution having mean as given by parameters θ_A^*. Hence, loss function using EWC is shown below

$$L(\theta) = L_b(\theta) + \frac{\lambda}{2}\sum_i \Omega_i(\theta_i - \theta_{Ai}^*)^2, \quad (74)$$

where θ is the parameter (weights and biases), θ_A^* is the parameter that gives low error for task A, $L_b(\theta)$ represents the loss for task B,λ is design parameter and Ω is the Fisher matrix.

$$\Omega_i = E_x\left[\left(\frac{\partial}{\partial\theta_i}logp(\phi|\theta)\right)^2 \mid \theta\right], \quad (75)$$

the loss function in (74) is used for two or more tasks; the regularizer added in the above loss function prevents the essential weights from deviating away from the consolidation values when they are learning different tasks. The EWC converge the weights of two tasks using the above loss function to a point where the error for the two tasks is very low. So, the update law for the four layers will be given by

$$\dot{\hat{D}} = C\big(A_d - \dot{A}_d\widehat{W}^T A_b - \dot{A}_d\widehat{W}^T\dot{A}_b\hat{Z}^T A_v - \dot{A}_d\widehat{W}^T\dot{A}_b\hat{Z}^T\dot{A}_v\hat{V}^T X\big)r^T - \lambda\Omega_i(D_i - D_\gamma^*)\,, \quad (76)$$

$$\dot{\widehat{W}} = F\hat{D}^T\dot{A}_d\big(A_b - \dot{A}_b\hat{Z}^T A_v - \dot{A}_b\hat{Z}^T\dot{A}_v\hat{V}^T X\big)r^T - \lambda\Omega_i(W_i - W_\gamma^*)\,, \quad (77)$$

$$\dot{\hat{Z}} = Q\hat{D}^T\dot{A}_d\widehat{W}^T\dot{A}_b\big(A_v - \dot{A}_v\hat{V}^T X\big)r^T - \lambda\Omega_i\big(Z_i - Z_\gamma^*\big), \quad (78)$$

$$\dot{\hat{V}} = G\hat{D}^T\big(\dot{A}_d\widehat{W}^T\dot{A}_b\hat{Z}^T\dot{A}_v X\big)r^T - \lambda\Omega_i\big(V_i - V_\gamma^*\big), \quad (79)$$

where λ is the design parameter and W_i, Z_i, D_i are the value of ith weight in present task and $V_\gamma^*, W_\gamma^*, Z_\gamma^*$, are the values of ith weight after the previous task training is completed. Proof of stability has been omitted because of space constraints.

4. SIMULATION RESULTS AND DISCUSSION

In this section, the effectiveness of the proposed method is illustrated using simulation results. The two link robotic arm dynamics is considered from (1) for two different tasks of different mass of the links. The task 1 is run for 0 to 12 seconds, the masses are mass m_1=0.8; mass m_2=2.3; a_1=1; a_2=1; g=9.8;

$K_v = 20eye(2);\ \Gamma = 1;\ \lambda = 5eye(2); \mathrm{F} = \mathrm{diag}\{30,30\}; \wedge = \mathrm{diag}\{4,4\}; \kappa = 0.2;$ amp=1; The task 2 is run for 13 to 25 seconds, the link masses considered for task 2 include m_1=0.1; mass m_2=1; a_1=1; a_2=1; g=9.8; all other values are same, after 25 seconds task 1 is run again using the same mass values for the links in order to evaluate the forgetting property.

The activation function used is sigmoid, bias taken is one; six neurons are used in input layer, four neurons are used in each hidden layer and two neurons are used in output layer; $p_{d_1}(t) = \sin(t)$; $p_{d2}(t) = \cos(t)$;initial conditions are chosen randomly from (0-0.1). The masses are changed in order to show that for each different task the dynamics change. The change in dynamics due to the manipulator performing a task is employed to illustrate the forgetting by a NN controller.

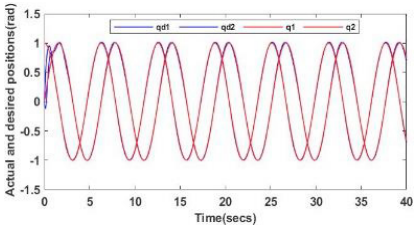

Fig. 1. Actual and desired joint angles with EWC and CL-based DTED approach.

Fig. 1 shows the tracking performance is effective when EWC and CL are used with the proposed method. It is seen the performance improvement with DTED method after EWC and CL are added even after the tasks are changed thus indicating the forgetting effects are reduced.

Fig. 2 shows the comparison of errors between tracking error-based backpropagation and DTED methods. It is shown that the DTED-based approach has less error as compared to backpropagation and even with a change in mass values of the links. Fig. 3 shows the error plots using EWC and CL; it is seen that errors are significantly less when EWC and CL are added to the proposed method. So the proposed method with EWC and CL performs better.

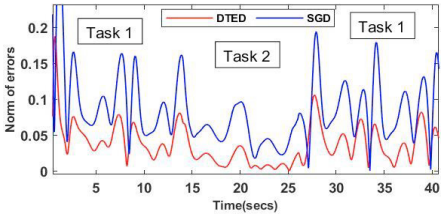

Fig. 2. Comparison of errors using DTED and filtered tracking error-based SGD.

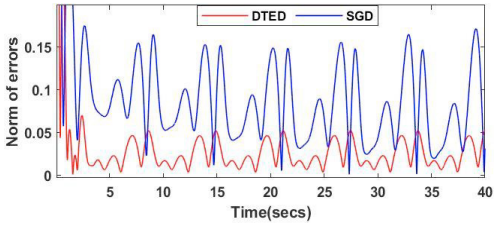

Fig. 3. Errors when EWC and CL are included to DTED approach.

From Fig. 4, it is shown that the control inputs vary with link masses change; it can be seen that the DTED-based approach requires less control effort while ensuring low error as compared to tracking error-based backpropagation.

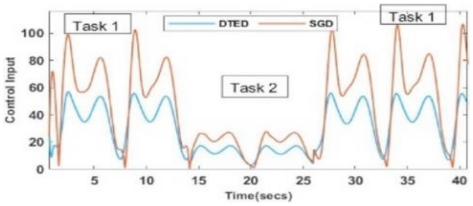

Fig. 4. Control input using unsupervised SGD and DTED.

5. CONCLUSION AND FUTURE WORK

The DTED approach uses filtered tracking error to tune weights of all the layers and therefore can be extended to any number of NN layers and relaxes the need for basis function selection. The proposed SVD based DTED approach results in better performance as compared to the tracking error version of backpropagation method. The addition of layers improves tracking accuracy at the expense of computation. Preliminary results show the lifelong learning functionality with the DTED approach due to EWC. The use of an experience replay buffer was able to generate the convergence of the tracking error and guarantee the convergence of the weights when the buffer could be filled with sufficiently rich data.

Acknowledgements: The effort undertaken was or is sponsored by the Office of Naval Research Grant N00014-21-1-2232 and Army Research Office Grant W911NF-21-2-0260.

REFERENCES

Baldi, P. (Jan. 1995). "Gradient descent learning algorithm overview: a general dynamical systems perspective," in *IEEE Transactions on Neural Networks*, vol. 6, no. 1,pp.182-195.

Benesty, J. and Duhamel, P. (Dec.1992). "A fast exact least mean square adaptive algorithm," in *IEEE Transactions on Signal Processing*, vol. 40, no. 12, pp. 2904-2920.

Bengio, Y. and Roux (2010). "Deep belief networks are compact universal approximators", *Neural computation, vol. 22, no.8,* pp.2192-2207.

Bitmead, R. (March 1984). "Persistence of excitation conditions and the convergence of adaptive schemes," in *IEEE Transactions on Information Theory*, vol. 30, no. 2, pp. 183-191.

Chowdhary, G. and Johnson, E. (2010). "Concurrent learning for convergence in adaptive control without persistency of excitation," *49th IEEE Conference on Decision and Control (CDC)*.

Kirkpatrick, J., Pascanu, Razvan., Rabinowitz, Neil.,Veness, Joel. (2016). "Overcoming catastrophic forgetting in neural nets", *Proceedings of the National Academy of Sciences*.

Lewis, F.L., Jagannathan, S., and Yesilderek. (1998). *Neural network control of robot Manipulators And Non-Linear Systems*.

Mhaskar, N. and Poggio, Tomaso. (2016). "Deep vs. shallow networks: An approximation theory perspective." *Analysis and Applications,* vol. 14, no. 6, pp. 829-848.

Narendra, K. S., and Annaswamy, Anuradha M. (1989). *Stable Adaptive Systems*. Prentice-Hall, Englewood Cliffs.

Wang, W., Dang, Z., Hu, Y., Fua, P. (2021). "Robust differentiable SVD", *IEEE Transactions on pattern analysis and machine learning*.

Werbos, P. J. (Jan. 1991). "An overview of neural networks for control," in *IEEE Control Systems Magazine*, vol. 11, no. 1, pp. 40-41.

Zhang, J., Dhillon, I. S. (2018). "Stablizing gradients for deep neural networks via efficient SVD parmeterization" *Proceedings of 35thIinternational Conference on Machine Learning*.