An Analytic End-to-End Deep Learning Algorithm based on Collaborative Learning

Sitan LI and Chien Chern CHEAH

Abstract— **In most control applications, theoretical analysis of the systems is crucial in ensuring stability or convergence, so as to ensure safe and reliable operations and also to gain a better understanding of the systems for further developments. However, most current deep learning methods are black-box approaches that are more focused on empirical studies. Recently, some results have been obtained for convergence analysis of end-to end deep learning based on non-smooth ReLU activation functions, which may result in chattering for control tasks. This paper presents a convergence analysis for end-to-end deep learning of fully connected neural networks (FNN) with smooth activation functions. The proposed method therefore avoids any potential chattering problem, and it also does not easily lead to gradient vanishing problems. The proposed End-to-End algorithm trains multiple two-layer fully connected networks concurrently and collaborative learning can be used to further combine their strengths to improve accuracy. A classification case study based on fully connected networks and MNIST dataset was done to demonstrate the performance of the proposed approach. Then an online kinematics control task of a UR5e robot arm was performed to illustrate the regression approximation and online updating ability of our algorithm.**

Index Terms— **End-to-End, Deep learning, Sigmoid, Robot kinematics**

I. Introduction

Deep learning networks are widely applied in various applications due to their exceptional performance, but their use in control tasks is limited by the difficulty of analyzing the convergence. The commonly used methods for training Deep Neural Networks (DNNs) are backpropagation and gradient descent [1] [2], but they are considered as black-box approaches that offer no assurance of convergence. In robot control, for example, convergence of error is crucial for ensuring that the robot moves accurately and smoothly. Otherwise, the robot may move erratically or fail to reach its intended target, which can be dangerous in industrial applications. Disregarding the theoretical convergence of deep learning could hinder its progress as a dependable and trustworthy technology. Therefore, it is crucial to develop an analytic learning algorithm for DNNs that can analyze convergence of the algorithm and ensure the safe and robust deployment of deep learning.

In recent years, researchers from machine learning field have started to analyze convergence of deep neural networks in optimization aspect. In [3], it was demonstrated that an over-parameterized deep neural network with rectified linear unit (ReLU) activation functions achieves global minima for training loss in a binary classification task. In [4], it was proved that, with Gram Matrix structure and over-parameterized neural networks (NNs), training loss can converge to zero through GD. In [5], it analyzed over-parameterized DNNs and showed that stochastic gradient descent(GD) method is capable of finding a global minimum for non-smooth ReLU and also other smooth activation functions. However, results in [3] [4] [5] are based on assumptions of over-parameterized networks , which are hard to implement in real control systems.

Layerwise learning is a promising approach for analyzing deep neural networks, whereby the network is divided into multiple layers or blocks and trained sequentially. In [6], the convergence analysis was provided for deep linear networks based on a layer-wise learning using block coordinate gradient descent. However, compared to deep nonlinear networks deep linear networks are not preferred in practical scenarios as linear ones have poor approximation ability. In [7], an analytical layer-wise approach was proposed for fully connected networks and was applied to classification and robot kinematic control tasks. In [8], a layer-wise deep learning algorithm with convergence analysis was proposed for convolutional neural networks.

However, the implementation on layerwise learning requires repeating procedure of adding one layer at a time, which is therefore limited to repetitive tasks. For more general tasks, End-to-End deep learning methods are therefore required. In [9], real-time weight adaptation laws were developed based on the Lyapunov-based stability analysis for a deep feed-forward neural network. However, it mainly focuses on the control of dynamic systems and classification tasks are not considered. In [10], an End-to-End learning algorithm was developed for DNNs for both image classification tasks and real-time control of robotic systems. The convergence analysis was based on non-smooth ReLU activation function and its variants.

The non-smooth activation function ReLU are widely applied in existing end-to-end deep learning methods, but it is not differentiable when the input is zero. This limits the implementation on control systems as it results in chattering of input. A smooth activation function like sigmoid or Tahn is more feasible as they are differentiable at any point, which therefore does not result in any chattering problem in actual implementation. However, one main issue that limits the use of the some smooth activation functions like Sigmoid activation function in end-to-end deep learning is that the gradients can easily vanish when the magnitude of input becomes too large. This means that the derivative of the function becomes extremely close to 0, which can result in exponentially decreasing gradients as they propagate through the layers of the deep FNN.

In this paper, we proposed an end-to-end deep learning method with convergence analysis based on smooth nonlinear activation functions. The difficulty in this convergence analysis is due to the nonlinearities of the activation functions in the hidden layers. Unlike existing end-to-end deep learning methods, the proposed learning method does not result in vanishing gradients easily, even when sigmoid activation functions are used. The proposed learning method is developed based on the collaborative learning of several sub-systems. The sub-systems are updated by a two-layer update law concurrently. We show that the sub-systems can also be combined to form a more accurate and robust predictive model by leveraging the diversity and complementary strengths of the sub models using collaborative learning. It is a powerful machine learning technique that involves multiple systems working together to achieve better accuracy than an individual system can achieve alone. Collaborative learning [11] [12] is a powerful machine learning technique that involves multiple systems working together to achieve better accuracy than an individual system can achieve alone. Some results have been obtained in the literature (see [11] and [12] and the references therein) but the previous works do not provide any convergence analysis. However, the previous works did not provide any convergence analysis. This paper presents a theoretical convergence analysis for the proposed end-to-end collaborative system. To demonstrate the efficacy of the proposed learning algorithm, a case study was first done on a classification task based on fully connected networks and MNIST dataset. Then a regression problem was also performed on the kinematics of a UR5e robot arm.

II. End-to-End Deep Learning with Collaborative Learning

A collaborative deep fully connected network is designed by combining $n-1$ sub-systems as shown in Fig 1. In the first sub-system, the input goes through one hidden layer, as shown in Fig 1(a). The input weight matrix $\hat{\mathbf{W}}_1$ and the pseudo weight matrix of output layer $\hat{\mathbf{W}}_1^{\triangleright}$ are updated together by the proposed updating law. Fig 1(b) shows the learning in the second sub-system with weight matrix $\hat{\mathbf{W}}_2$ and pseudo weight matrix $\hat{\mathbf{W}}_2^{\triangleright}$. The input $\boldsymbol{x}_2$ of the second sub-system is formed by passing $\boldsymbol{x}_1$ through the first weight matrix $\hat{\mathbf{W}}_1$. Fig 1(c) shows the updating in the jth sub-system with weight matrix $\hat{\mathbf{W}}_j$ and pseudo weight matrix $\hat{\mathbf{W}}_j^{\triangleright}$.The learning of last weights takes place at $n-1$th sub-system as shown in Fig 1(d), where two weights $\hat{\mathbf{W}}_{n-1}$ and $\hat{\mathbf{W}}_{n-1}^{\triangleright}$ are learned instead of pseudo weights. In this way, all weights $\hat{\mathbf{W}}_1, \hat{\mathbf{W}}_2, \dots, \hat{\mathbf{W}}_{n-1}$ are updated simultaneously based on $n-1$ sub-systems by the input data $\boldsymbol{x}(k)$. All $n-1$ sub-systems are connected to a fully connected layer to do the final classification. The weight matrix of the last fully connected layer $\hat{\mathbf{W}}_n$ is updated concurrently with the sub-systems.

After all the sub-systems have been updated using data $\boldsymbol{x}_1(k)$, the weight matrix $\hat{\mathbf{W}}_1$ is shared with remaining subsystems as illustrated in Fig 1, $\hat{\mathbf{W}}_2$ is shared with remaining subsystems from 3 to $n-1$, and similarly, the weight matrix $\hat{\mathbf{W}}_j$ is shared with remaining subsystems from $j+1$ to $n-1$. The same updating procedure is then repeated for all sub-systems based on next data $\boldsymbol{x}_1(k+1)$. When the entire set of data has been passed through the neural network, one training epoch is finished. The sub-systems are trained for multiple epochs until it reaches convergence. The remaining part of this section presents the update laws for simultaneously updating all weight matrices. $\hat{\mathbf{W}}_1, ..., \hat{\mathbf{W}}_n$.

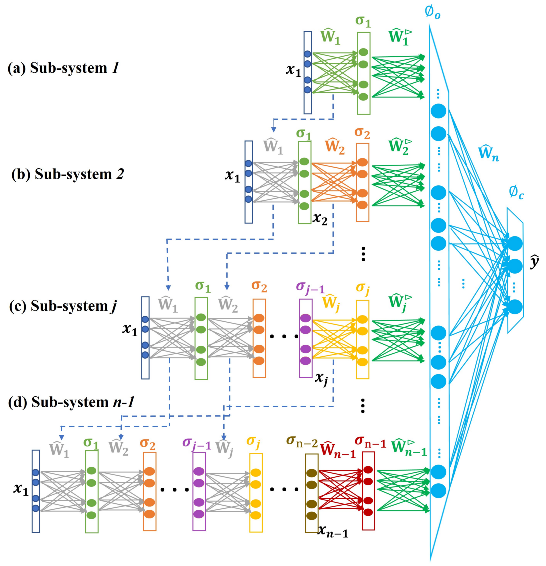


Fig. 1. Collaborative End to End learning method, where the collaborative FNN is combined of $n-1$ sub-systems

For all sub-systems, the input data is $\boldsymbol{x}_1(k)$. For ease of representation, the input to the last 2 layers of jth sub system $\boldsymbol{x}_j(k)$, $j=2,..,n-1$, can be calculated by passing each input data $\boldsymbol{x}_1(k)$ through estimated weights $\hat{\mathbf{W}}_1(k), \hat{\mathbf{W}}_2(k), \dots, \hat{\mathbf{W}}_{j-1}(k)$ shared from previous sub-systems:

$$\boldsymbol{x}_j(k) = \hat{\boldsymbol{\sigma}}_{j-1}(k) \tag{1}$$

$$\boldsymbol{\sigma}_{j-1}(k) = \boldsymbol{\sigma}_{j-1}(\hat{\mathbf{W}}_{j-1}(k)\boldsymbol{\sigma}_{j-2}(\dots\boldsymbol{\sigma}_2(\hat{\mathbf{W}}_2(k)\boldsymbol{\sigma}_1(\hat{\mathbf{W}}_1(k)\boldsymbol{x}_1(k)))\dots)) \tag{2}$$

where estimated weights are denoted as $\hat{\mathbf{W}}_1(k), \hat{\mathbf{W}}_2(k), \dots, \hat{\mathbf{W}}_n(k)$ from first layer to last layer of the jth sub-system and $\boldsymbol{\sigma}_j$ denotes the activation functions of the jth hidden layer, which can be any smooth and differentiable activation functions like Sigmoid activation function, Tahn activation function .

For each subsystem, the last two weight matrices (see Fig 1) are updated by a two-layer update law. The estimated weight matrices $\hat{\mathbf{W}}_j$ and $\hat{\mathbf{W}}_j^{\triangleright}$ of all sub-systems and the weight matrix $\hat{\mathbf{W}}_n$ of the last layer are updated concurrently.

In the jth sub-system, the estimated output of this sub-system $\hat{\boldsymbol{y}}_j(k)$ is constructed by the estimated input weights $\hat{\mathbf{W}}_j(k)$ and output weights $\hat{\mathbf{W}}_j^{\triangleright}(k)$ as follows:

$$\hat{\boldsymbol{y}}_j(k) = \phi_o\big(\hat{\mathbf{W}}_j^{\triangleright}(k)\boldsymbol{\sigma}_j(\hat{\mathbf{W}}_j(k)\boldsymbol{x}_j(k))\big) \tag{3}$$

where the activation functions for output layer are denoted as ϕ_o. There exists ideal unknown weight matrices $\mathbf{W}_j$ and $\mathbf{W}_j^{\triangleright}$

for each epoch such that the actual output $\boldsymbol{y}(k)$ is represented as:

$$\boldsymbol{y}(k) = \boldsymbol{\phi}_o\big(\mathbf{W}_j^{\triangleright}\boldsymbol{\sigma}_j(\mathbf{W}_j\boldsymbol{x}_j(k))\big) \tag{4}$$

The estimated output of the collaborative multi-layer fully connected network (MLFN), which combines $n-1$ sub-systems is presented as follows:

$$\hat{\boldsymbol{y}}_n(k) = \boldsymbol{\phi}_c(\hat{\mathbf{W}}_n(k)\hat{\boldsymbol{\phi}}_{sub}(k)) \tag{5}$$

where $\hat{\boldsymbol{\phi}}_{sub}(k) = [\hat{\boldsymbol{y}}_1(k), \hat{\boldsymbol{y}}_2(k), .., \hat{\boldsymbol{y}}_j(k), .., \hat{\boldsymbol{y}}_n(k)]^T$, and the activation functions of the last layer of collaborative network are defined as $\boldsymbol{\phi}_c$. There exists an ideal weight matrix $\mathbf{W}_n$ for each epoch such that the ideal output of the collaborative network is represented as follows:

$$\boldsymbol{y}(k) = \boldsymbol{\phi}_c\big(\mathbf{W}_n\hat{\boldsymbol{\phi}}_{sub}(k)\big) \tag{6}$$

The weight matrices $\hat{\mathbf{W}}_j$ and $\hat{\mathbf{W}}_j^{\triangleright}$ are then updated by two learning laws based on data $\boldsymbol{x}_j(k)$ and error. The estimation error $\boldsymbol{e}_j(k)$ for the jth sub-system is defined as the difference between its estimated output in (3) and the ideal output (4) in as:

$$\begin{aligned}\boldsymbol{e}_j(k) =& \boldsymbol{y}(k) - \hat{\boldsymbol{y}}_j(k) \qquad (7)\\ =& \boldsymbol{\phi}_o\big(\mathbf{W}_j^{\triangleright}\boldsymbol{\sigma}_j(\mathbf{W}_j\boldsymbol{x}_j(k))\big)\boldsymbol{\phi}_o\big(\hat{\mathbf{W}}_j^{\triangleright}(k)\boldsymbol{\sigma}_j(\hat{\mathbf{W}}_j(k)\boldsymbol{x}_j(k))\big)\end{aligned}$$

For updating the last weight matrix $\mathbf{W}_n$, from (5) and (6) the error $\boldsymbol{e}_n(k)$ is formulated as:

$$\boldsymbol{e}_n(k) = \boldsymbol{\phi}_c\big(\mathbf{W}_n\hat{\boldsymbol{\phi}}_{sub}(k)\big) - \boldsymbol{\phi}_c\big(\hat{\mathbf{W}}_n(k)\hat{\boldsymbol{\phi}}_{sub}(k)\big) \tag{8}$$

Let

$$\boldsymbol{\delta}_j(k) = \mathbf{W}_j^{\triangleright}\boldsymbol{\sigma}_j(k) - \hat{\mathbf{W}}_j^{\triangleright}(k)\hat{\boldsymbol{\sigma}}_j(k) \tag{9}$$

where $\boldsymbol{\sigma}_j(k) = \boldsymbol{\sigma}_j(\mathbf{W}_j\boldsymbol{x}_j(k))$ and $\hat{\boldsymbol{\sigma}}_j(k) = \boldsymbol{\sigma}_j(\hat{\mathbf{W}}_j(k)\boldsymbol{x}_j(k))$, then equation (9) can be expressed as:

$$\boldsymbol{\delta}_j(k) = \hat{\mathbf{W}}_j^{\triangleright}(k)\Delta\boldsymbol{\sigma}_j(k) + \Delta\mathbf{W}_j^{\triangleright}(k)\hat{\boldsymbol{\sigma}}_j(k) + \Delta\mathbf{W}_j^{\triangleright}(k)\Delta\boldsymbol{\sigma}_j(k) \tag{10}$$

where $\Delta\boldsymbol{\sigma}_j(k) = \boldsymbol{\sigma}_j(k) - \hat{\boldsymbol{\sigma}}_j(k)$ and $\Delta\mathbf{W}_j^{\triangleright}(k) = \mathbf{W}_j^{\triangleright} - \hat{\mathbf{W}}_j^{\triangleright}(k)$. In addition, let

$$\boldsymbol{\delta}_n(k) = \mathbf{W}_n\hat{\boldsymbol{\phi}}_{sub}(k) - \hat{\mathbf{W}}_n(k)\hat{\boldsymbol{\phi}}_{sub}(k) = \Delta\mathbf{W}_n(k)\hat{\boldsymbol{\phi}}_{sub}(k) \tag{11}$$

where $\Delta\mathbf{W}_n(k) = \mathbf{W}_n - \hat{\mathbf{W}}_n(k)$. The activation functions $\boldsymbol{\phi}$ (including $\boldsymbol{\phi}_o$ and $\boldsymbol{\phi}_c$) can be selected as monotonically increasing smooth activation functions with upper bounds like Sigmoid activation function, Tahn activation function or Identity activation function. Since they are monotonically increasing, they have the upper bounds $f_{\phi M}$. The errors $\boldsymbol{e}_j(k)$,$\boldsymbol{e}_n(k)$ in (7),(8) and $\boldsymbol{\delta}_j(k)$,$\boldsymbol{\delta}_n(k)$ in (10),(11) have the following properties:

i, the sign of ith elements of $\boldsymbol{e}_j(k)$ and $\boldsymbol{\delta}_j(k)$ are the same, $j = 1, 2..., n$, i.e.

$$e_{j_i}(k)\delta_{j_i}(k) \geq 0, \ \forall i = 1..p \tag{12}$$

ii, the absolute value of the ith elements of $\boldsymbol{e}_j(k)$ are no more than $f_{\phi M}$ times the ith elements of $\boldsymbol{\delta}_j(k)$, $j = 1, 2..., n$, i.e.

$$|e_{j_i}(k)| \leq f_{\phi M}|\delta_{j_i}(k)|, \ \forall i = 1..p \tag{13}$$

The estimated weights of sub-systems are updated using $\boldsymbol{e}_j(k)$ and the output weight matrix $\hat{\mathbf{W}}_n(k+1)$ is updated using $\boldsymbol{e}_n(k)$ as:

$$\hat{\mathbf{W}}_j(k+1) = \hat{\mathbf{W}}_j(k) + \alpha_j\mathbf{S}_j(k)\hat{\mathbf{W}}_j^{\triangleright T}(k)\mathbf{L}_j(k)\boldsymbol{e}_j(k)\boldsymbol{x}_j^T(k) \tag{14}$$

$$\hat{\mathbf{W}}_j^{\triangleright}(k+1) = \hat{\mathbf{W}}_j^{\triangleright}(k) + \alpha_j^{\triangleright}\mathbf{L}_j(k)\boldsymbol{e}_j(k)\hat{\boldsymbol{\sigma}}_j^T(k) \tag{15}$$

$$\hat{\mathbf{W}}_n(k+1) = \hat{\mathbf{W}}_n(k) + \alpha_n\mathbf{L}_n(k)\boldsymbol{e}_n(k)\hat{\boldsymbol{\phi}}_{sub}^T(k) \tag{16}$$

where $\alpha_j^{\triangleright}$, $\alpha_j, j = 1, .., n-1$ and α_n are constant non-negative scalars, $\mathbf{S}_j(k) = diag[\boldsymbol{\sigma}'_{j,1}(k), ..., \boldsymbol{\sigma}'_{j,i}(k), ..., \boldsymbol{\sigma}'_{j,h_j}(k)]$ are diagonal matrices where $\boldsymbol{\sigma}'_{j,i}(k)$ are the gradients of activation functions $\boldsymbol{\sigma}_{j,i}(k)$, $\mathbf{L}_j(k)$ and $\mathbf{L}_n(k)$ are positive diagonal matrices. After each update, the weight matrix $\hat{\mathbf{W}}_j$ is shared with the sub-systems from $j+1$ to $n-1$.

To analyze the convergence of the deep FNN, an objective function $V(k)$ of the whole network is proposed as the summation of objective functions of all sub-systems as follows:

$$\begin{aligned}V(k) =& \textstyle\sum_{j=1}^{n-1}\mathbf{Tr}\Big(\Delta\mathbf{W}_j^{\triangleright T}(k)\Delta\mathbf{W}_j^{\triangleright}(k)\Big) + \sum_{j=1}^{n-1}\mathbf{Tr}\Big(\Delta\mathbf{W}_j^T(k)\Delta\mathbf{W}_j(k)\Big)\\ &+ \mathbf{Tr}\Big(\Delta\mathbf{W}_n^T(k)\Delta\mathbf{W}_n(k)\Big) \qquad (17)\end{aligned}$$

where $\mathbf{Tr}$ represents the trace of the matrix. From (17), for $(k+1)$th data, the objective function $V(k+1)$ can be similarly formulated. Substituting (14),(15) and (16) into $V(k+1)$ and taking the difference with $V(k)$ in (17), it can be shown that :

$$\begin{aligned}&\Delta V(k) = V(k+1) - V(k)\\ &= -2\alpha_j^{\triangleright}\textstyle\sum_{j=1}^{n-1}\hat{\boldsymbol{\sigma}}_j^T(k)\Delta\mathbf{W}_j^{\triangleright T}(k)\mathbf{L}_j(k)\boldsymbol{e}_j(k)\\ &+ \boldsymbol{e}_j^T(k)\Big(\textstyle\sum_{j=1}^{n-1}\alpha_j^{\triangleright^2}\|\hat{\boldsymbol{\sigma}}_j(k)\|^2\mathbf{L}_j^T(k)\mathbf{L}_j(k)\\ &+ \textstyle\sum_{j=1}^{n-1}\alpha_j^2\|\boldsymbol{x}_j(k)\|^2(\mathbf{L}_j^T(k)\hat{\mathbf{W}}_j^{\triangleright}(k)\mathbf{S}_j^2(k)\hat{\mathbf{W}}_j^{\triangleright T}(k)\mathbf{L}_j(k))\Big)\boldsymbol{e}_j(k)\\ &- 2\alpha_n\hat{\boldsymbol{\phi}}_{sub}^T(k)\Delta\mathbf{W}_n^{\triangleright T}(k)\mathbf{L}_n(k)\boldsymbol{e}_n(k)\\ &+ \boldsymbol{e}_n^T(k)\alpha_n^2\|\hat{\boldsymbol{\phi}}_{sub}(k)\|^2\mathbf{L}_n^T(k)\mathbf{L}_n(k)\boldsymbol{e}_n(k)\\ &- 2\alpha_j\textstyle\sum_{j=1}^{n-1}\boldsymbol{e}_j^T(k)(\mathbf{L}_j^T(k)\hat{\mathbf{W}}_j^{\triangleright}(k)\mathbf{S}_j(k))\Delta\mathbf{W}_j(k)\boldsymbol{x}_j(k) \qquad (18)\end{aligned}$$

The training process is divided into 2 phases: a pretraining phase and a fine-tuning phase. The pretrain [7] process is conducted by randomly initializing $\hat{\mathbf{W}}_j(k) = \bar{\mathbf{W}}_j(k)$ in (3) and (4) and then fix them by setting $\alpha_j = 0$ in (14). Then only train output weight matrix $\hat{\mathbf{W}}_j^{\triangleright}(k)$ and $\hat{\mathbf{W}}_n(k)$ in initial epochs using update laws (15) and (16) to reduce errors. Since $\hat{\mathbf{W}}_j(k)$ is fixed as $\bar{\mathbf{W}}_j(k)$, then $\hat{\boldsymbol{\sigma}}_j(k)$ are also fixed as $\bar{\boldsymbol{\sigma}}_j(k)$.Then there exist ideal weight matrices $\mathbf{W}_j^{\triangleright}$ for each epoch such that equation (7) and (9) become:

$$\begin{aligned}\boldsymbol{e}_j(k) =& \boldsymbol{y}(k) - \hat{\boldsymbol{y}}_j(k) \qquad (19)\\ =& \boldsymbol{\phi}_o\big(\mathbf{W}_j^{\triangleright}\bar{\boldsymbol{\sigma}}_j(\bar{\mathbf{W}}_j\boldsymbol{x}_j(k))\big) - \boldsymbol{\phi}_o\big(\hat{\mathbf{W}}_j^{\triangleright}(k)\bar{\boldsymbol{\sigma}}_j(\bar{\mathbf{W}}_j(k)\boldsymbol{x}_j(k))\big)\end{aligned}$$

$$\boldsymbol{\delta}_j(k) = \mathbf{W}_j^{\triangleright}\bar{\boldsymbol{\sigma}}_j(k) - \hat{\mathbf{W}}_j^{\triangleright}(k)\bar{\boldsymbol{\sigma}}_j(k) = \Delta\mathbf{W}_j^{\triangleright}(k)\bar{\boldsymbol{\sigma}}_j(k) \tag{20}$$

where $\hat{\boldsymbol{\sigma}}_j(k) = \bar{\boldsymbol{\sigma}}_j(k)$ in (18) in pretraining. Substitute (20) and (11) into (18), with $\alpha_j = 0$, the change in objective function for pretraining reduces to :

$$\begin{aligned}&\Delta V_{pre}(k) = -2\alpha_j^{\triangleright} \textstyle\sum_{j=1}^{n-1} \boldsymbol{\delta}_j^T(k)\mathbf{L}_j(k)\boldsymbol{e}_j(k) - 2\alpha_n \boldsymbol{\delta}_n^T(k)\mathbf{L}_n(k)\boldsymbol{e}_n(k)\\ &+ \boldsymbol{e}_j^T(k) \textstyle\sum_{j=1}^{n-1} \alpha_j^{\triangleright^2} \|\bar{\boldsymbol{\sigma}}_j(k)\|^2 \mathbf{L}_j^T(k)\mathbf{L}_j(k)\boldsymbol{e}_j(k)\\ &+ \boldsymbol{e}_n^T(k)\alpha_n^2 \|\hat{\boldsymbol{\phi}}_{sub}(k)\|^2 \mathbf{L}_n^T(k)\mathbf{L}_n(k)\boldsymbol{e}_n(k) \qquad (21)\end{aligned}$$

Let L_{jM}, L_{nM} be the maximum eigenvalues of $\boldsymbol{L}_j(k), \boldsymbol{L}_n(k)$. Since $\bar{\boldsymbol{\sigma}}_j(k)$ and $\hat{\boldsymbol{\phi}}_{sub}(k)$ are bounded, there exist positive constants d_j^{pre} and d_n^{pre} such that:

$$\begin{aligned}&\boldsymbol{e}_j^T(k) \textstyle\sum_{j=1}^{n-1} \alpha_j^{\triangleright^2} \|\bar{\boldsymbol{\sigma}}_j(k)\|^2 \mathbf{L}_j^T(k)\mathbf{L}_j(k)\boldsymbol{e}_j(k)\\ &+\boldsymbol{e}_n^T(k)\alpha_n^2 \|\hat{\boldsymbol{\phi}}_{sub}(k)\|^2 \mathbf{L}_n^T(k)\mathbf{L}_n(k)\boldsymbol{e}_n(k) \qquad (22)\\ &\leq \textstyle\sum_{j=1}^{n-1} \alpha_j^{\triangleright^2} d_j^{pre} L_{jM}^2 \|\boldsymbol{e}_j(k)\|^2 + \alpha_n^2 d_n^{pre} L_{nM}^2 \|\boldsymbol{e}_n(k)\|^2\end{aligned}$$

where d_j^{pre} is the norm bound for any $\| \bar{\boldsymbol{\sigma}}_j(k)\|^2$ and d_n^{pre} is the norm bound for any $\|\hat{\boldsymbol{\phi}}_{sub}(k)\|^2$. Substituting inequality (22) into the second and third row of (21) and using inequalities in (12), (13) in the first row of (21) , we have:

$$\begin{aligned}&\Delta V_{pre}(k) \leq -2\alpha_j^{\triangleright} \textstyle\sum_{j=1}^{n-1} \boldsymbol{\delta}_j^T(k)\mathbf{L}_j(k)\boldsymbol{e}_j(k) + \textstyle\sum_{j=1}^{n-1} \alpha_j^{\triangleright^2} d_j^{pre} L_{jM}^2 \|\boldsymbol{e}_j(k)\|^2\\ &-2\alpha_n \boldsymbol{\delta}_n^T(k)\mathbf{L}_n(k)\boldsymbol{e}_n(k) + \alpha_n^2 d_n^{pre} L_{nM}^2 \|\boldsymbol{e}_n(k)\|^2\\ &\leq -\textstyle\sum_{j=1}^{n-1} \alpha_j^{\triangleright} (\frac{2L_{jm}}{f_{\phi M}} - \alpha_j^{\triangleright} d_j^{pre} L_{jM}^2) \|\boldsymbol{e}_j(k)\|^2\\ &- \alpha_n (\frac{2L_{nm}}{f_{\phi M}} - \alpha_n d_n^{pre} L_{nM}^2) \|\boldsymbol{e}_n(k)\|^2 \qquad (23)\end{aligned}$$

where L_{jm}, L_{nm} are the minimum eigenvalues of matrix $\boldsymbol{L}_j(k), \boldsymbol{L}_n(k)$. It can be shown that if $\frac{2L_{jm}}{f_{\phi M}} - \alpha_j^{\triangleright} d_j^{pre} L_{jM}^2 > 0$ and $\frac{2L_{nm}}{f_{\phi M}} - \alpha_n d_n^{pre} L_{nM}^2 > 0$ then $\Delta V_{pre}(k) \leq 0$, which guarantees the convergence in pretraining.

Theorem. Consider the deep collaborative fully connected neural network given by (6) with the update laws (14), (15) and (16) in the fine-tuning phase. Let the non-negative scalar α_j in (14) be chosen as $\alpha_j = \alpha_j^{\triangleright}$ and $\mathbf{L}_j(k)$ and $\mathbf{L}_n(k)$ in (14), (15) and (16) be chosen to satisfy the following conditions:

$$\frac{2L_{jm}}{f_{\phi M}} - \alpha_j^{\triangleright} d_j L_{jM}^2 > 0, \frac{2L_{nm}}{f_{\phi M}} - \alpha_n d_n L_{nM}^2 > 0 \qquad (24)$$

where L_{jM} and L_{nM} are the maximum eigenvalues of $\mathbf{L}_j(k)$ and $\mathbf{L}_n(k)$, d_j and d_n are positive scalars. The output training errors $\boldsymbol{e}_j(k)$ and $\boldsymbol{e}_n(k)$ converge towards 0 as $k \to \infty$.

Proof. After pretraining, update law (14) is activated by setting α_j to equal to non-zero constants $\alpha_j^{\triangleright}$. In this fine-tuning process, all layers' weights are updated together to further reduce output error and improve performance. From (10) and (11), we have:

$$\Delta\mathbf{W}_j^{\triangleright}(k)\hat{\boldsymbol{\sigma}}_j(k) = \boldsymbol{\delta}_j(k) - \hat{\mathbf{W}}_j^{\triangleright}(k)\Delta\boldsymbol{\sigma}_j(k) - \Delta\mathbf{W}_j^{\triangleright}(k)\Delta\boldsymbol{\sigma}_j(k) \qquad (25)$$

$$\Delta\mathbf{W}_n^{\triangleright}(k)\hat{\boldsymbol{\phi}}_{sub}(k) = \boldsymbol{\delta}_n(k) \qquad (26)$$

Substituting (25) and (26) into (18), we have:

$$\begin{aligned}&\Delta V(k) = -\, 2\alpha_j^{\triangleright} \textstyle\sum_{j=1}^{n-1} \boldsymbol{\delta}_j^T(k)\mathbf{L}_j(k)\boldsymbol{e}_j(k)\\ &+ \boldsymbol{e}_j^T(k)\Big(\textstyle\sum_{j=1}^{n-1} \alpha_j^{\triangleright^2} \|\hat{\boldsymbol{\sigma}}_j(k)\|^2 \mathbf{L}_j^T(k)\mathbf{L}_j(k)\\ &+\textstyle\sum_{j=1}^{n-1} \alpha_j^{\triangleright 2} \|\boldsymbol{x}_j(k)\|^2 (\mathbf{L}_j^T(k)\hat{\mathbf{W}}_j^{\triangleright}(k)\mathbf{S}_j^2(k)\hat{\mathbf{W}}_j^{\triangleright T}(k)\mathbf{L}_j(k))\Big)\boldsymbol{e}_j(k)\\ &-2\alpha_n \boldsymbol{\delta}_n^T(k)\mathbf{L}_n(k)\boldsymbol{e}_n(k)\\ &+\boldsymbol{e}_n^T(k)\alpha_n^2 \|\hat{\boldsymbol{\phi}}_{sub}(k)\|^2 \mathbf{L}_n^T(k)\mathbf{L}_n(k)\boldsymbol{e}_n(k)\\ &+ 2\alpha_j^{\triangleright} \textstyle\sum_{j=1}^{n-1} \boldsymbol{e}_j^T(k)\mathbf{L}_j^T(k)\hat{\mathbf{W}}_j^{\triangleright}(k)\Big(\Delta\boldsymbol{\sigma}_j(k) - \mathbf{S}_j(k)\Delta\mathbf{W}_j(k)\boldsymbol{x}_j(k)\Big)\\ &+ \textstyle\sum_{j=1}^{n-1} 2\alpha_j^{\triangleright} \Delta\boldsymbol{\sigma}_j^T(k)\Delta\mathbf{W}_j^{\triangleright T}(k)\mathbf{L}_j(k)\boldsymbol{e}_j(k) \qquad (27)\end{aligned}$$

Consider the terms in the second last row in (27), using Taylor expansion we have:

$$\begin{aligned}\Delta\boldsymbol{\sigma}_j(k) &= \boldsymbol{\sigma}_j(\mathbf{W}_j\boldsymbol{x}_j(k)) - \boldsymbol{\sigma}_j(\hat{\mathbf{W}}_j(k)\boldsymbol{x}_j(k))\\ &= \mathbf{S}_j(k)\Delta\mathbf{W}_j(k)\boldsymbol{x}_j(k) + \mathbf{O}^t\big(\Delta\mathbf{W}_j(k)\boldsymbol{x}_j(k)\big)\end{aligned} \qquad (28)$$

where $\mathbf{S}_j(k)$ are diagonal matrices with elements of gradients of activation functions $\boldsymbol{\sigma}_{j,i}(k)$, $\mathbf{O}^t\big(\Delta\mathbf{W}_j(k)\boldsymbol{x}_j(k)\big)$ are summation of high order terms. In the fine-tuning phase after pre-training phase, the errors are sufficiently small and hence the higher order terms in last 2 terms of (27), which are O^3 and more are negligible as compared to the other terms which are of O^2.

Then equation (27) becomes:

$$\begin{aligned}&\Delta V(k) = -\, 2\alpha_j^{\triangleright} \textstyle\sum_{j=1}^{n-1} \boldsymbol{\delta}_j^T(k)\mathbf{L}_j(k)\boldsymbol{e}_j(k) - 2\alpha_n \boldsymbol{\delta}_n^T(k)\mathbf{L}_n(k)\boldsymbol{e}_n(k)\\ &+ \boldsymbol{e}_j^T(k)\Big(\textstyle\sum_{j=1}^{n-1} \alpha_j^{\triangleright^2} \|\hat{\boldsymbol{\sigma}}_j(k)\|^2 \mathbf{L}_j^T(k)\mathbf{L}_j(k)\\ &+\textstyle\sum_{j=1}^{n-1} \alpha_j^{\triangleright 2} \|\boldsymbol{x}_j(k)\|^2 (\mathbf{L}_j^T(k)\hat{\mathbf{W}}_j^{\triangleright}(k)\mathbf{S}_j^2(k)\hat{\mathbf{W}}_j^{\triangleright T}(k)\mathbf{L}_j(k))\Big)\boldsymbol{e}_j(k)\\ &+ \boldsymbol{e}_n^T(k)\alpha_n^2 \|\hat{\boldsymbol{\phi}}_{sub}(k)\|^2 \mathbf{L}_n^T(k)\mathbf{L}_n(k)\boldsymbol{e}_n(k)\end{aligned} \qquad (29)$$

Since the smooth activation functions $\boldsymbol{\sigma}$ and $\boldsymbol{\phi}_{sub}$ are saturated, therefore $\hat{\boldsymbol{\sigma}}_j(k)$, $\hat{\boldsymbol{\phi}}_{sub}(k)$ and $\boldsymbol{x}_j(k)$ are bounded. Since the initial weights matrices $\hat{\mathbf{W}}_j^{\triangleright}(0)$, $\hat{\mathbf{W}}_j(0)$ and $\hat{\mathbf{W}}_n(0)$ are bounded, so according to update laws (14), (15) and (16)(refer to Appendix), there exist positive constants d_j, d_n such that :

$$\begin{aligned}&\boldsymbol{e}_j^T(k)\Big(\sum_{j=1}^{n-1} \alpha_j^{\triangleright^2} \|\hat{\boldsymbol{\sigma}}_j(k)\|^2 \mathbf{L}_j^T(k)\mathbf{L}_j(k) \qquad (30)\\ &+\textstyle\sum_{j=1}^{n-1} \alpha_j^{\triangleright 2} \|\boldsymbol{x}_j(k)\|^2 (\mathbf{L}_j^T(k)\hat{\mathbf{W}}_j^{\triangleright}(k)\mathbf{S}_j^2(k)\hat{\mathbf{W}}_j^{\triangleright T}(k)\mathbf{L}_j(k))\Big)\boldsymbol{e}_j(k)\\ &+ \boldsymbol{e}_n^T(k)\alpha_n^2 \|\hat{\boldsymbol{\phi}}_{sub}(k)\|^2 \mathbf{L}_n^T(k)\mathbf{L}_n(k)\boldsymbol{e}_n(k)\\ &\leq \textstyle\sum_{j=1}^{n-1} \alpha_j^{\triangleright^2} d_j L_{jM}^2 \|\boldsymbol{e}_j(k)\|^2 + \alpha_n^2 d_n L_{nM}^2 \|\boldsymbol{e}_n(k)\|^2\end{aligned}$$

where d_j is the norm bound for any $\|\boldsymbol{x}_j(k)\|^2 \|\hat{\mathbf{W}}_j^{\triangleright}(k)\mathbf{S}_j(k)\|^2$ and d_n is the norm bound for any $\|\hat{\boldsymbol{\phi}}_{sub}(k)\|^2$.

Therefore, (29) becomes:

$$\begin{aligned}&\Delta V(k) \leq -2\alpha_j^{\triangleright} \textstyle\sum_{j=1}^{n-1} \boldsymbol{\delta}_j^T(k)\mathbf{L}_j(k)\boldsymbol{e}_j(k) + \textstyle\sum_{j=1}^{n-1} \alpha_j^{\triangleright^2} d_j L_{jM}^2 \|\boldsymbol{e}_j(k)\|^2\\ &-2\alpha_n \boldsymbol{\delta}_n^T(k)\mathbf{L}_n(k)\boldsymbol{e}_n(k) + \alpha_n^2 d_n L_{nM}^2 \|\boldsymbol{e}_n(k)\|^2 \qquad (31)\end{aligned}$$

Using inequalities in (12), (13) in the first term of (31), we have:

$$-2\boldsymbol{\delta}_j^T(k)\mathbf{L}_j(k)\boldsymbol{e}_j(k) \leq \frac{-2\alpha_j^{\triangleright} L_{jm}}{f_{\phi M}} \|\boldsymbol{e}_j(k)\|^2 \qquad (32)$$

$$-2\boldsymbol{\delta}_n^T(k)\mathbf{L}_n(k)\boldsymbol{e}_n(k) \leq \frac{-2\alpha_n L_{nm}}{f_{\phi M}}\|\boldsymbol{e}_n(k)\|^2 \tag{33}$$

Rearranging the equation (31), we can get:

$$\Delta V(k) \leq -\sum_{j=1}^{n-1}\alpha_j^{\triangleright}(\frac{2L_{jm}}{f_{\phi M}} - \alpha_j^{\triangleright} d_j L_{jM}^2)\|\boldsymbol{e}_j(k)\|^2$$
$$-\alpha_n(\frac{2L_{nm}}{f_{\phi M}} - \alpha_n d_n L_{nM}^2)\|\boldsymbol{e}_n(k)\|^2 \tag{34}$$

as seen from equation (34), when the conditions in (24) are satisfied, then the two terms on the right-hand side of (34) are negative definite in $\boldsymbol{e}_j(k)$ and $\boldsymbol{e}_n(k)$, and hence $\Delta V(k) \leq 0$. Since $V(k)$ is non-negative and bounded from below, then it is converging as k increases, which also indicates that all errors are converging for each epoch. This condition in (24) adjusts the gain matrix $\mathbf{L}_j(k)$ after it is randomly initialized in each update. The proof is complete.

III. Case Studies

In this section, a case study using MNIST dataset was first done to show the performance of the approximation ability in classification tasks. Then an online robot kinematics control task was done using an industrial robot UR5e to illustrate the regression approximation ability and online adaptation ability of the proposed algorithm.

A. MNIST Dataset Case Studies

Case Studies were performed on a fully connected network based on handwritten number dataset MNIST. MNIST dataset is a 10 classes image dataset with the input image size of $28 * 28$. The case study is done on a fully connected neural network with 3 subsystems (subsystem I:784-Sig-150-Sig-10-Sig, subsystem II:784-Sig-150-Sig-100-Sig-10-Sig, subsystem III:784-Sig-150-Sig-100-Sig-50-Sig-10-Sig) connected by a collaborative layer (see Fig 1 also):

Unlike SGD, where learning rate can be selected from several empirical trials, the choice of $\mathbf{L}_j(k)$ in proposed algorithm in each step can be automatically reduces by the condition given in (24) and convergence can always be assured. The pretraining was first conducted by setting $\alpha_j = 0$ in (14) and $\mathbf{L}_j = \mathbf{L}_n = diag(0.01, ..., 0.01)$ for 10 epochs and then by setting $\alpha_j^{\triangleright} = \alpha_j = \alpha_n = 1$ the fine-tuning process was conducted. To obtain the optimal test performance for both methods, we have tuned the hyperparameters of both SGD and the proposed method and compared their results.

TABLE I

Test results on Collaborative learning of Fully connected networks with MNIST

Network	SGD		sub-system		Collaborative	
	Train acc	Test acc	Train acc	Test acc	Train acc	Test acc
sub-system I	99.5	98.2	99.3	98.2	99.8	98.7
sub-system III	99.9	98.3	99.6	98.4		
sub-system II	99.8	89.3	99.5	98.3		
sub-system IV	Diverge	Diverge	99.2	98.1	99.85	98.6
sub-system V	Diverge	Diverge	99.0	98.0		

The training and testing accuracies are shown in Table I compared between SGD and the proposed method. The sub-system column refers to each sub-system performance as shown in Fig 1(a)-(d). It can be observed from the last row of Table I that the four hidden layer fully connected neural network trained with SGD has already dropped to 89.3 percent due to the gradient vanishing problem of using Sigmoid activation. However, our proposed learning algorithm allows the use of Sigmoid activation for deeper neural networks without gradient vanishing problem. The results also show that by using collaborative learning, the testing accuracy has a noticeable improvement. To further illustrate the problem of gradient vanishing using Sigmoid activation functions, we tested the proposed method and SGD with deeper networks (subsystem IV:784-Sig-150-Sig-100-Sig-50-Sig-50-Sig-50-Sig-10-Sig, subsystem V:784-Sig-150-Sig-100-Sig-50-Sig-50-Sig-50-Sig-50-Sig-10-Sig) as stated in the fourth and fifth row of Table I. It was observed that SGD could not converge with deeper layers, but the collaborative network with deeper subsystems included could converge and achieve a final accuracy of 98.6%.

It is shown that compared with SGD, our result on the MNIST can achieve similar performance. From Table I, with collaborative learning, the accuracy has a noticeable improvement. This demonstrates the potential of using smooth activation functions in deep control systems.

B. Online Jacobian matrix approximation task on UR5E

Jacobian matrix, mapping from joint space to Cartesian space, directly gives feedback from Cartesian space and therefore is crucial to real-time control [13]. In this section, the Jacobian matrix of UR5E robot arm with unknown kinematics is approximated by a FNN using the proposed algorithm.

Let k represent the sampling time, the Cartesian space end effector velocities $\dot{\boldsymbol{x}}$ and joint velocities $\dot{\boldsymbol{q}}$ has the following relationship:

$$\dot{\boldsymbol{x}}(k) = \mathbf{J}(\boldsymbol{q}(k))\dot{\boldsymbol{q}}(k) \tag{35}$$

where $\mathbf{J}(\boldsymbol{q}(k))$ is the Jacobian matrix.

Jacobian matrix is approximated by deep FNNs using the proposed algorithm. The estimated Jacobian matrices $\hat{\mathbf{J}}_j$ are retrieved from the jth, $j = 1, ..., n-1$ sub-system as (3) and the velocity in Cartesian space is approximated by the collaborated network as (5) :

$$\begin{aligned}\hat{\dot{\boldsymbol{x}}}_j(k) &= \hat{\mathbf{J}}_j(\boldsymbol{q}(k), \hat{\mathbf{W}}_j^{\triangleright}, \hat{\mathbf{W}}_j)\dot{\boldsymbol{q}}(k) \\ &= \textstyle\sum_{h=1}^{r} \boldsymbol{\phi}_{o_h}\big(\hat{\mathbf{W}}_j^{\triangleright}(k)\hat{\boldsymbol{\sigma}}_j(k)\big)\dot{q}_m(k)\end{aligned} \tag{36}$$

$$\hat{\dot{\boldsymbol{x}}_n}(k) = \textstyle\sum_{h=1}^{r} \boldsymbol{\phi}_{c_h}\big(\hat{\mathbf{W}}_n(k)\hat{\boldsymbol{\phi}}_{sub}(k)\big)\dot{q}_m(k)) \tag{37}$$

where $\hat{\boldsymbol{\sigma}}_j(k) = \boldsymbol{\sigma}_j(\hat{\mathbf{W}}_j(k)\boldsymbol{q}_j(k)))$, $\hat{\boldsymbol{\phi}}_{sub}(k) = [\hat{\dot{\boldsymbol{x}}}_1(k), \hat{\dot{\boldsymbol{x}}}_2(k), .., \hat{\dot{\boldsymbol{x}}}_j(k), .., \hat{\dot{\boldsymbol{x}}}_n(k)]^T$, r denotes the number of joints, $\boldsymbol{q}_j(k)$ is the input joint angle of the jth sub-system, $\dot{q}_m(k)$ is the mth element of joint velocity $\dot{\boldsymbol{q}}(k)$, the activation functions ϕ_c and ϕ_o are chosen as linear activation function in the following online kinematics task.

The output error $\boldsymbol{e}_j(k)$, which is formulated using (36), is utilized to update the weights of the jth sub-system. The collaborative layer is updated based on $\boldsymbol{e}_n(k)$ formulated using (37):

$$\boldsymbol{e}_j(k) = \dot{\boldsymbol{x}}(k) - \hat{\dot{\boldsymbol{x}}}_j(k) \tag{38}$$
$$= \sum_{h=1}^{r} \phi_{o_h}\big(\mathbf{W}_j^{\triangleright}(k)\boldsymbol{\sigma}_{j-1}(k)\big)\dot{q}_m(k)) - \sum_{h=1}^{r} \phi_{o_h}\big(\hat{\mathbf{W}}_j^{\triangleright}(k)\hat{\boldsymbol{\sigma}}_{j-1}(k)\big)\dot{q}_m(k)$$

$$\boldsymbol{e}_n(k) = \dot{\boldsymbol{x}}(k) - \hat{\dot{\boldsymbol{x}}}_n(k) \tag{39}$$
$$= \sum_{h=1}^{r} \phi_{c_h}\big(\mathbf{W}_n(k)\hat{\boldsymbol{\phi}}_{sub}(k)\big)\dot{q}_m(k)) - \sum_{h=1}^{r} \phi_{c_h}\big(\hat{\mathbf{W}}_n(k)\hat{\boldsymbol{\phi}}_{sub}(k)\big)\dot{q}_m(k))$$

where $\dot{\boldsymbol{x}}(k) = \sum_{h=1}^{r} \phi_{o_h}\big(\mathbf{W}_j^{\triangleright}(k)\boldsymbol{\sigma}_j(k)\big)\dot{q}_m(k))$ and $\dot{\boldsymbol{x}}(k) = \sum_{h=1}^{r} \phi_{c_h}\big(\mathbf{W}_n(k)\hat{\boldsymbol{\phi}}_{sub}(k)\big)\dot{q}_m(k)$, $\boldsymbol{\sigma}_j(k) = \boldsymbol{\sigma}_j(\mathbf{W}_j(k)\boldsymbol{q}_j(k)))$. Comparing with (7) and (8), the error $\boldsymbol{e}_j(k)$ and $\boldsymbol{e}_n(k)$ are identical to the output error of the sub-systems. Therefore, the errors $\boldsymbol{e}_j(k)$ and $\boldsymbol{e}_n(k)$ are updated following the update laws in (14), (15) and (16) with guaranteed convergence.

Then, after all the subsystems have been trained, The Jacobian matrix $\hat{\mathbf{J}}_{n-1}(\boldsymbol{q}(k), \hat{\mathbf{W}}_{n-1}^{\triangleright}, \hat{\mathbf{W}}_{n-1})$ of the $n-1$th sub-system alone is applied in the online kinematic control to accommodate changes. The reference joint velocity $\dot{\boldsymbol{q}}$ for online control is defined as :

$$\dot{\boldsymbol{q}}(k) = \hat{\mathbf{J}}_{n-1}^{\dagger}(\boldsymbol{q}(k), \hat{\mathbf{W}}_{n-1}^{\triangleright}, \hat{\mathbf{W}}_{n-1})(\dot{\boldsymbol{x}}_d(k) - \alpha\Delta\boldsymbol{x}(k)) \tag{40}$$

where $\alpha\Delta\boldsymbol{x}(k) = \boldsymbol{x}(k) - \boldsymbol{x}_d(k), \alpha$ is a positive scalar, $\boldsymbol{x}_d(k), \dot{\boldsymbol{x}}_d(k)$ represents the desired position and velocity of the end effector in the task space $\hat{\mathbf{J}}_{n-1}^{\dagger}$ is the pseudoinverse matrix of $\hat{\mathbf{J}}_{n-1}$. By multiplying (40) with $\hat{\mathbf{J}}_{n-1}(\boldsymbol{q}(k), \hat{\mathbf{W}}_{n-1}^{\triangleright}, \hat{\mathbf{W}}_{n-1})$, we have:

$$\hat{\mathbf{J}}_{n-1}((\boldsymbol{q}(k), \hat{\mathbf{W}}_{n-1}^{\triangleright}, \hat{\mathbf{W}}_{n-1})\dot{\boldsymbol{q}}(k) = \dot{\boldsymbol{x}}_d(k) - \alpha\Delta\boldsymbol{x}(k) \tag{41}$$

Subtracting (35) and (41) we have:

$$\begin{aligned}&\mathbf{J}(\boldsymbol{q}(k))\dot{\boldsymbol{q}}(k) - \hat{\mathbf{J}}_{n-1}(\boldsymbol{q}(k), \hat{\mathbf{W}}_{n-1}^{\triangleright}, \hat{\mathbf{W}}_{n-1})\dot{\boldsymbol{q}}(k)\\ =&\dot{\boldsymbol{x}}(k) - \dot{\boldsymbol{x}}_d(k) + \alpha\Delta\boldsymbol{x}(k) = \Delta\dot{\boldsymbol{x}}(k) + \alpha\Delta\boldsymbol{x}(k)\end{aligned} \tag{42}$$

During online learning, online feedback error is constructed as $\boldsymbol{\varepsilon}(k) = \Delta\dot{\boldsymbol{x}}(k) + \alpha\Delta\boldsymbol{x}(k)$, from (35),(36), we have:

$$\boldsymbol{\varepsilon}(k) = \mathbf{J}(\boldsymbol{q}(k))\dot{\boldsymbol{q}}(k) - \hat{\mathbf{J}}_{n-1}(\boldsymbol{q}(k), \hat{\mathbf{W}}_{n-1}^{\triangleright}, \hat{\mathbf{W}}_{n-1})\dot{\boldsymbol{q}}(k) \tag{43}$$

Hence, in the online kinematics control, update laws (14),(15) based on $\boldsymbol{\varepsilon}(k)$ are used, which ensures the convergence of the online feedback error.

A trajectory tracking control experiment of the robot arm in task space was conducted to show the proposed algorithm's approximation ability in online kinematics control tasks. For approximating the Jacobian matrix in the online task, a deep collaborated FNN was employed and evaluated with following the structure with three subsystems (subsystem I:3-Sig-12-Sig-3-Sig, subsystem II:3-Sig-12-Sig-24-Sig-3-Sig, subsystem III:3-Sig-12-Sig-24-Sig-24-Sig-3-Sig) connected by a collabrative layer(see Fig 1 also):

The input data of the network is $[\boldsymbol{q}, \dot{\boldsymbol{q}}]$ and the output is end-effector velocity $\dot{\boldsymbol{x}}$. All the data are acquired from the robot arm real-time data exchange every 0.01s. In the FNN, Sigmoid activation functions were employed for the hidden layers in each subsystem and identity activation functions were chosen for each output layer.

The online task was to track a circle trajectory $C1$ with the parameters shown in Fig 2(a). The training data were manually collected around circle $C1$. The pretraining was conducted by setting $\alpha_j = 0$ in (14) and $\mathbf{L}_j = \mathbf{L}_n = diag(0.05, ..., 0.05)$ for 25 epochs and then by setting $\alpha_j^{\triangleright} = \alpha_j = \alpha_n = 1$ the fine-tuning process was conducted. The offline weights were later used as starting weights for the online trajectory tracking task. The tracking errors are shown in Fig 2(a). Then from the obtained offline weights, the online updating was performed according to (40). During the online training, all layers' weights were updated concurrently and only the last sub-system was used for online control. As shown in Fig 2(b), after online updating, the tracking errors have been reduced as online learning can further adapt the control system using tracking errors.

The online task was also performed on Leaky ReLU activation function with the same network structure and learning method for comparison purpose. It can be seen from the results in Fig 3(a) that, Leaky ReLU activation function causes chattering problem when the error goes through zero due to it is unsmooth when input is zero. However, as shown in Fig 3(b), smooth activation function Sigmoid does not have this problem and hence it is differentiable .

IV. Conclusion

In this paper, an End-to-End deep learning algorithm based on collaborative learning has been proposed. This learning algorithm allows a deep FNN updating all layer's weight concurrently on $n-1$ subsystems and that the subsystems are combined by a collaborative learning layer. The convergence analysis is provided with smooth activation functions like Sigmoid to avoid chattering input problem in control tasks. We have shown the approximation ability of the proposed method for classification tasks compared with SGD, and the approximation ability for regression task and online kinematics task.

References

[1] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, pp. 436–444, 2015.

[2] A. Shrestha and A. Mahmood, "Review of deep learning algorithms and architectures," *IEEE Access*, vol. 7, pp. 53 040–53 065, 2019.

[3] D. Zou, Y. Cao, D. Zhou, and Q. Gu, "Gradient descent optimizes over-parameterized deep relu networks," *Machine Learning*, vol. 109, no. 3, pp. 467–492, 2020.

[4] S. Du, J. Lee, H. Li, L. Wang, and X. Zhai, "Gradient descent finds global minima of deep neural networks," in *International Conference on Machine Learning*. PMLR, 2019, pp. 1675–1685.

[5] Z. Allen-Zhu, Y. Li, and Z. Song, "A convergence theory for deep learning via over-parameterization," in *International Conference on Machine Learning*. PMLR, 2019, pp. 242–252.

[6] Y. Shin, "Effects of depth, width, and initialization: A convergence analysis of layer-wise training for deep linear neural networks," *Analysis and Applications*, vol. 20, no. 01, pp. 73–119, 2022.

[7] H.-T. Nguyen, C. C. Cheah, and K.-A. Toh, "An analytic layer-wise deep learning framework with applications to robotics," *Automatica*, vol. 135, p. 110007, 2022.

[8] H.-T. Nguyen, S. Li, and C. C. Cheah, "A layer-wise theoretical framework for deep learning of convolutional neural networks," *IEEE Access*, vol. 10, pp. 14 270–14 287, 2022.

[9] O. S. Patil, D. M. Le, M. L. Greene, and W. E. Dixon, "Lyapunov-derived control and adaptive update laws for inner and outer layer weights of a deep neural network," *IEEE Control Systems Letters*, vol. 6, pp. 1855–1860, 2021.

[10] S. Li, H.-T. Nguyen, and C. C. Cheah, "A theoretical framework for end-to-end learning of deep neural networks with applications to robotics," *IEEE Access*, vol. 11, pp. 21 992–22 006, 2023.

[11] H. Wang, N. Wang, and D.-Y. Yeung, "Collaborative deep learning for recommender systems," in *Proceedings of the 21th ACM SIGKDD international conference on knowledge discovery and data mining*, 2015, pp. 1235–1244.

[12] G. Song and W. Chai, "Collaborative learning for deep neural networks," *Advances in neural information processing systems*, vol. 31, 2018.

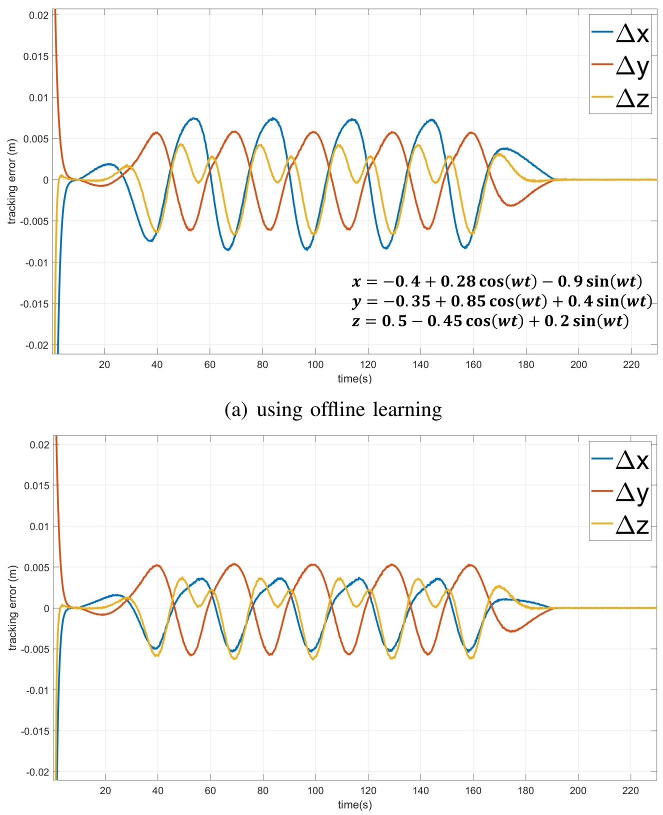
[13] C. C. Cheah and X. Li, *Task-space sensory feedback control of robot manipulators*. Springer, 2015.

Appendix

Consider (29) with $k = 0$, we have

$$\begin{aligned}&\Delta V(0) = V(1) - V(0) = \\ &- 2\alpha_j^{\triangleright} \textstyle\sum_{j=1}^{n-1} \boldsymbol{\delta}_j^T(0)\mathbf{L}_j(0)\boldsymbol{e}_j(0) - 2\alpha_n\boldsymbol{\delta}_n^T(0)\mathbf{L}_n(0)\boldsymbol{e}_n(0) \\ &+ \boldsymbol{e}_j^T(0)\Big(\textstyle\sum_{j=1}^{n-1} {\alpha_j^{\triangleright}}^2 \|\hat{\boldsymbol{\sigma}}_j(0)\|^2\mathbf{L}_j^T(0)\mathbf{L}_j(0) \\ &+ \textstyle\sum_{j=1}^{n-1} {\alpha_j^{\triangleright}}^2 \|\boldsymbol{x}_j(0)\|^2(\mathbf{L}_j^T(0)\hat{\mathbf{W}}_j^{\triangleright}(0)\mathbf{S}_j^2(0)\hat{\mathbf{W}}_j^{\triangleright T}(0)\mathbf{L}_j(0))\Big)\boldsymbol{e}_j(0) \\ &+ \boldsymbol{e}_n^T(0)\alpha_n^2\|\hat{\boldsymbol{\phi}}_{sub}(0)\|^2\mathbf{L}_n^T(0)\mathbf{L}_n(0)\boldsymbol{e}_n(0)\end{aligned} \tag{44}$$

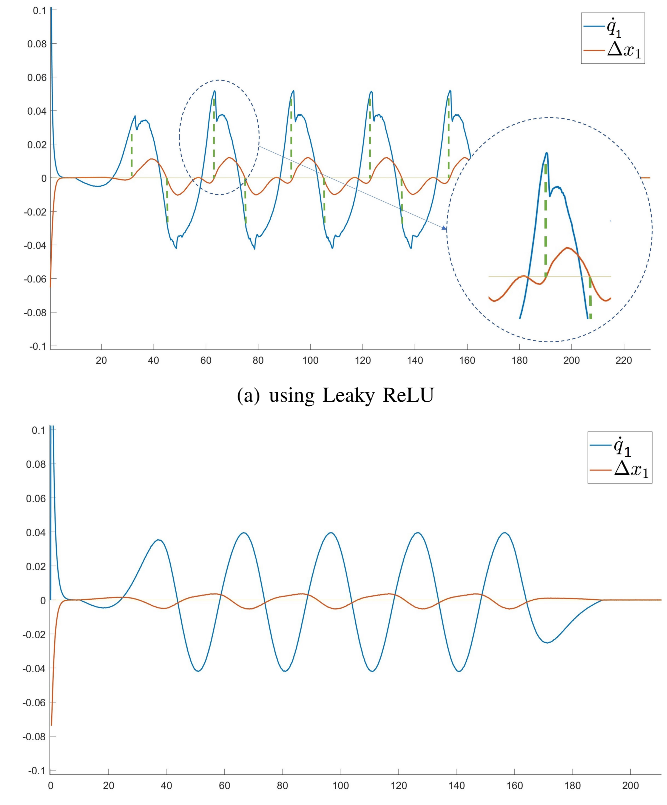
When $k = 0$, since all elements in $\Delta V\,(k)$ are bounded, therefore there exists d_j,d_n such that (30) can be satisfied when $k = 0$. This means that $\Delta V\,(0) < 0$ and also means $\hat{\mathbf{W}}_j\,(1) < \hat{\mathbf{W}}_j\,(0)$, $\hat{\mathbf{W}}_j^{\triangleright}\,(1) < \hat{\mathbf{W}}_j^{\triangleright}\,(0)$. Therefore, by induction, all $\hat{\mathbf{W}}_j\,(k)$ and $\hat{\mathbf{W}}_j^{\triangleright}\,(k)$ bounded. Therefore, for any k, there exists d_j,d_n such that (30) can be satisfied.



(a) using offline learning

(b) using online learning

Fig. 2. Tracking errors of kinematic control



(a) using Leaky ReLU

(b) using Sigmoid

Fig. 3. Reference input and tracking error for kinematic control