2023 62nd IEEE Conference on Decision and Control (CDC)
December 13-15, 2023. Marina Bay Sands, Singapore

Unmatched uncertainty mitigation through neural network supported model predictive control

Mateus V. Gasparino, Prabhat K. Mishra, and Girish Chowdhary

Abstract— **This paper presents a deep learning based model predictive control (MPC) algorithm for systems with unmatched and bounded state-action dependent uncertainties of unknown structure. We utilize a deep neural network (DNN) as an oracle in the underlying optimization problem of learning based MPC (LBMPC) to estimate unmatched uncertainties. Generally, DNNs as oracle are considered difficult to employ with LBMPC due to the technical difficulties associated with the estimation of their coefficients in real time. We employ a dual-timescale adaptation mechanism, where the weights of the last layer of the neural network are updated in real time while the inner layers are trained on a slower timescale using the training data collected online and selectively stored in a buffer. Our results are validated through a numerical experiment on the compression system model of a jet engine. These results indicate that the proposed approach is implementable in real time and carries the theoretical guarantees of LBMPC.**

I. Introduction

Machine learning and Model Predictive Control (MPC) complement each other by compensating drawbacks of each other and making their combination useful for safety critical applications. We refer readers to [1], [2] for excellent surveys on safe learning and robotics.

Learning based Model Predictive Control (LBMPC) [3] became popular due to its improvement over linear MPC in terms of transient response and overshoot with slight expense in processing time [4]. Several interesting applications such as autonomous driving [5], [6], heat, ventilation and air-conditioning systems [7], quad-copter [4], formation control [8], atmospheric pressure plasma jets [9], air-borne wind energy systems [10] etc., contributed in theoretical and practical advancements.

Due to the significant success of and progress in deep learning techniques, this is tempting to use a neural network as an oracle in LBMPC. However, boundedness and differentiability of the oracle is required to hold the results of [3]. Even though a bounded and differentiable neural network can be constructed, the estimation of their weights in real time is generally difficult [11]. Since training of DNN is a time demanding process, real time implementation of DNN supported LBMPC is challenging.

In this article, we demonstrate that the real time implementation of neural network based oracle is not only possible but also computationally efficient by two time scale training. In [12], [13], the training of the output layer and the hidden layers are separated in which the output layer is trained online through adaptation by considering the recently trained hidden layers as a feature basis function and the hidden layers are trained on a parallel machine by keeping the recently updated weights of the output layer fixed. We refer readers to excellent surveys on transfer learning [14], [15]. This method of two time scale training allows us to try different methods of training the output layer and hidden layers independently. In addition, different architectures such as Recurrent Neural Network and Convolutional Neural Network are also supported for the hidden layer. The theoretical guarantees mostly depend on last activation layer and the training mechanism of the linear output layer. The above methods [12], [13] are limited only to those uncertainties that enter into the dynamics through control channel. The present article extends the results of [12], [13] by generalizing the class of uncertainties. Our approach improves the performance of [3] by replacing the L2NW estimator by a DNN. Since PyTorch is popular for DNN implementation and CasAdi for MPC, we address some non-trivial issues related to their interface also.

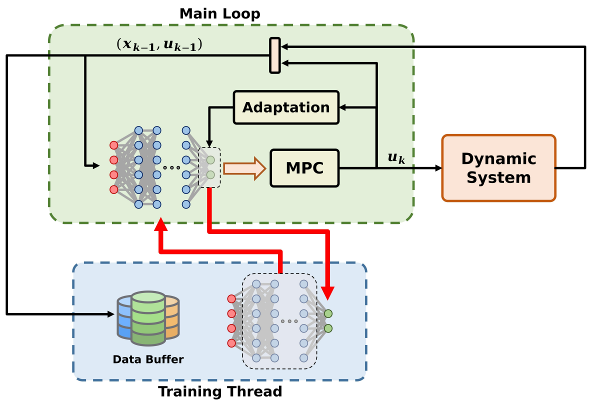

Fig. 1: The neural network on the main loop with fixed hidden layers is connected with MPC and transmits the weights of output layer at time sequence $(T_k)_{k\in\mathbb{Z}_+}$ to the neural network on the training thread. This neural network (on the training thread) transmits the weights of hidden layer to the neural network (on the main loop) at time instants $(t_j)_{j\in\mathbb{Z}_+}$ after its j^{th} training.

This article is organized as follows. The problem statement is given in §II. The architecture of neural network and its training mechanism are explained in §III. LBMPC and its properties are presented in §IV and §V, respectively. We validate our results in §VI and conclude in §VII.

We let $\mathbb{R}$ denote the set of real numbers, $\mathbb{N}$ the set of non-

We gratefully acknowledge financial support from NSF CNS grant 1954556 and USDA-NIFA grant 2021-67021-34418.

M. V. Gasparino and G. Chowdhary are with Computer Science Department, University of Illinois at Urbana Champaign (UIUC), USA. {mvalve2,girishc}@illinois.edu

P. K. Mishra is with Chemical Engineering Department, Massachusetts Institute of Technology (MIT), Cambridge, USA. pkmishra@mit.edu

negative integers and $\mathbb{Z}_+$ the set of positive integers. For a given vector v and positive (semi)-definite matrix $M \geq \mathbf{0}$, $\|v\|_M^2$ is used to denote $v^\top M v$. For a given matrix A, the trace, the largest eigenvalue and pseudo-inverse are denoted by $\mathrm{tr}(A)$, $\lambda_{\max}(A)$ and $A^\dagger$, respectively. By notation $\|A\|$ and $\|A\|_\infty$, we mean the 2−norm and ∞−norm when A is a vector; induced 2−norm and ∞−norm when A is a matrix, respectively. A vector or a matrix with all entries 0 is represented by $\mathbf{0}$ and I is the identity matrix of appropriate dimensions. We let $M^{(i)}$ denote the i^{th} column of a given matrix M.

II. Problem setup

Let us consider a discrete time dynamical system

$$x_{t+1} = Ax_t + Bu_t + h(x_t, u_t); \quad t \in \mathbb{N}, \tag{1}$$

(1-a) $x_t \in \mathcal{X} \subset \mathbb{R}^d$, $u_t \in \mathbb{U} \subset \mathbb{R}^m$; $\quad d, m \in \mathbb{Z}_+$,
(1-b) $h : \mathcal{X} \times \mathbb{U} \to \mathbb{R}^d$ is a continuous and (possibly) nonlinear function, which represents state-action dependent unmatched uncertainty (or modeling error),
(1-c) $h(x,u) \in \mathbb{W}$ for every $x \in \mathcal{X}$ and $u \in \mathbb{U}$,
(1-d) $\mathcal{X}, \mathbb{U}$ and $\mathbb{W}$ are polytopes,
(1-e) the matrix pair (A, B) is stabilizable.

The matrices A and B represent our domain knowledge or prior knowledge about the system dynamics, and the continuous function h represents the unknown component of the system dynamics[1]. LBMPC [3] has been developed to improve the closed-loop performance of tube-based robust MPC by modifying the cost function in the underlying optimization problem. The cost function is modified by learning (or estimating) the unknown function h with the help of data. In this article, we address issues associated with the use of DNN as an estimator of h. Our main focus is to use DNN in such a way that it can be implemented in real time on a hardware with limited computational power. The general problem description of this article is as follows:

Problem Statement 1: Present a stabilizing, robust and real-time implementable control framework for (1), which respects physical constraints (1-a), optimizes a given performance index, and reduces the effect of unmatched uncertainties by using a trainable DNN.

III. Deep neural network

Any continuous function h on a compact set $\mathcal{X} \times \mathbb{U}$ can be approximated with a desired accuracy by a multi-layer network with number of layers $L \geqslant 2$ [16, §7.1]. We can represent $h(x_t, u_t)$ with the help of a neural network such that

$$h(x,u) = W_L^\top \psi_L \left[W_{L-1}^\top \psi_{L-1} \left[\cdots \left[\psi_1(x,u) \right] \right] \right] + \varepsilon^*(x,u), \tag{2}$$

where $x \in \mathcal{X}$ and $u \in \mathbb{U}$. ψ_i and W_i, for $i = 1, \ldots, L$, are the activation functions in the i^{th} layer and the corresponding ideal weights, respectively.

[1]For example, when a non-linear dynamics is linearized by the matrix pair (A, B), h represents the linearization error.

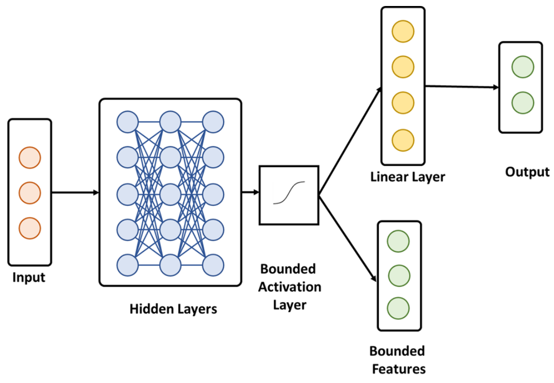

Fig. 2: The neural network architecture can use an arbitrary number of layers and neurons. The input layer has the same number of neurons as the experience data. The output layer has as many neurons as system states. The hidden layers can have any architecture provided suitable adjustments are made to facilitate their training.

Let us define $\phi^*(x,u) := \psi_L \left[W_{L-1}^\top \psi_{L-1} \left[\cdots \left[\psi_1(x,u) \right] \right] \right]$ and $W^* := W_L$, then

$$h(x_t, u_t) = W^{*\top} \phi^*(x_t, u_t) + \varepsilon^*(x_t, u_t), \tag{3}$$

where $W^* \in \mathbb{R}^{(n_L+1) \times d}$ denotes the weights of the output layer. There are n_L number of neurons in the last hidden layer. The first row of W^* represents the bias term in the output layer and the first element of $\phi^* \in \mathbb{R}^{n_L+1}$ is 1.

The major challenge is associated with real-time implementation of DNN because its training takes much more time than the sampling interval of fast hardware like quadcopter. Therefore, we address this issue with the help of two DNN's as shown in Fig. 1. Both DNNs have same architecture as in Fig. 2. The DNN in the main loop is located on the main machine and the other DNN is located on some secondary (or remote) machine. We update the weights of the output layer on the main machine in real time at each time instant with the help of a weight update law while keeping the weights of hidden layers fixed. The hidden layers are trained on a secondary machine by using the approach [17] in which the weights of the output layer are copied from the main machine at the start of the training and remain fixed during the training. Once the training of DNN on a secondary machine is complete, new weights of hidden layers are updated on the main machine and remain fixed until the new set of weights are again obtained from the secondary machine. Training details about the output layer and hidden layers are provided in §III-A and §III-B, respectively.

Let $(t_j)_{j \in \mathbb{Z}_+}$ represent an increasing time sequence of instants when weights of hidden layers are updated on main machine. Therefore, we get the following expression:

$$h(x_t, u_t) = W^{*\top} \phi_j(x_t, u_t) + \varepsilon_j(x_t, u_t), \tag{4}$$

where $\varepsilon_j(x_t, u_t) = W^{*\top} \left(\phi^*(x_t, u_t) - \phi_j(x_t, u_t) \right) + \varepsilon^*(x_t, u_t)$. We can assume that $\left\| \phi_j(x,u) \right\|$ will be bounded for each j, $u \in \mathbb{R}^m$ and $x \in \mathbb{R}^d$ due to the presence of bounded activation layer in Fig. 2 consisting bounded neurons, i. e. sigmoidal, tanh, etc. Since h is bounded due to (1-c), we can assume that ideal weights W^* in the output layer are also bounded. We make the following assumption:

Assumption 1: There exist $\bar{W}_i > 0$ for $i = 1, \dots, d$, and $\sigma, \bar{\varepsilon} > 0$ such that $\|W^{*(i)}\| \leqslant \bar{W}_i$, for $i = 1, \dots, d$, and $\|\phi_j(x,u)\| \leqslant \sigma$ for every $x \in \mathcal{X}, u \in \mathbb{U}$ and $j \in \mathbb{N}$.
The above assumption is standard in literature [12], [18], [19]. If the neural network is not minimal then the ideal weights may not be unique. However, for the neural-adaptive controller design only the existence of ideal weights is assumed, which is always guaranteed when h is a continuous function on a compact set [16, §7.1]. A priori knowledge about the bounds on the ideal weights W^* of the output layer is useful to avoid parameter drift phenomenon.

At $t_0 = 0$, weights of DNN on both machines are randomly initialized with desired bound on the output layer weights as per the Assumption 1. Therefore, for $j \in \mathbb{N}$, we have

$$\hat{h}_t(x_t,u_t) := K_t^\top \phi_j(x_t,u_t) \text{ for } t \in \{t_j, t_j+1, \dots, t_{j+1}-1\}. \quad (5)$$

We update K_t in an unsupervised manner on main machine while collecting the training data for the DNN on secondary machine. In the next subsections we provide the relevant details of the training of DNN.

A. Adaptive learning of W^* on the main machine

For $t \in \{t_j, t_j+1, \dots, t_{j+1}-1\}$, the output of DNN is given by (5) and the bounded features are given by $\phi_j(x_t,u_t)$ at time $t+1$ [2]. We get the estimated state

$$\hat{x}_{t+1} = Ax_t + Bu_t + \hat{h}_t(x_t,u_t). \quad (6)$$

We can compute the error $\tilde{x}_{t+1} = \hat{x}_{t+1} - x_{t+1}$. For the online training of the output layer, we employ the projection based robust weight update law and refer readers to [20, Chapter 10] for other methods. For a given learning rate $0 < \gamma < 1$, we first modify K_t by taking $\tilde{x}_{t+1}$ into account to get $\bar{K}_{t+1}$ and then project $\bar{K}_{t+1}$ to ensure its boundedness as follows:

$$\begin{aligned} \bar{K}_{t+1} &= K_t - \gamma \frac{\phi_j(x_t,u_t)}{\|\phi_j(x_t,u_t)\|^2} \tilde{x}_{t+1}^\top, \\ K_{t+1}^{(i)} &= \operatorname{Proj} \bar{K}_{t+1}^{(i)} = \begin{cases} \bar{K}_{t+1}^{(i)} & \text{if } \|\bar{K}_{t+1}^{(i)}\| \leqslant \bar{W}_i \\ \frac{\bar{W}_i}{\|\bar{K}_{t+1}^{(i)}\|} \bar{K}_{t+1}^{(i)} & \text{otherwise.} \end{cases} \end{aligned} \quad (7)$$

The implications of (7) are discussed in §V.

B. Supervised learning of ϕ^* on a secondary machine

In this section, we explain the training data and the loss function required for the supervised training of the DNN on the secondary machine. For a given state-action pair (x_t,u_t) as input at time t, the label $h(x_t,u_t)$ is computed at time $t+1$ by the relation:

$$h(x_t,u_t) = x_{t+1} - Ax_t - Bu_t. \quad (8)$$

These pairs (x,u) and corresponding labels $h(x,u)$ are stored in a training buffer until the buffer is full. In particular, a full buffer consists of $\big((x^i,u^i), h(x^i,u^i)\big)$ for $i = 1, \dots, p_{\max}$.

[2]This is important for implementation to note that u_t is computed at time t but we need to use $\phi_j(x_t,u_t)$ and $\hat{h}(x_t,u_t)$ at time $t+1$.

Once the buffer is full, new data is stored by replacing some old data. Several methods are proposed in the literature to increase the richness of the training buffer; see [21]–[23] and references therein. We refer readers to [24] for different methods of designing replay buffer.

Let $(T_k)_{k\in\mathbb{Z}_+}$ be an increasing time sequence. At time T_k, the weight of output layer of the primary neural network are copied in the secondary neural network. During the training, the output layer weights in the secondary network remain fixed. Therefore, we are interested in finding the weights $W_{1:L-1} := W_1, \dots, W_{L-1}$ which minimize the following cost for a given input (x,u) and label $h(x,u)$:

$$\ell((x,u), W_{1:L-1}) := \left\| h(x,u) - K_{T_k}^\top \psi_L \left[W_{L-1}^\top \psi_{L-1} \left[\cdots [\psi_1(x,u)] \right] \right] \right\|^2.$$

Let M represent the number of training samples and $\mathcal{D}_k := \big((x^i,u^i), h(x^i,u^i)\big)_{i=1}^M$ is training data consisting M data points randomly sampled from the buffer for the k^{th} training. The following loss function is considered for the training of DNN:

$$\mathcal{L}(\mathcal{D}_k, W_{1:L-1}) = \frac{1}{M} \sum_{i=0}^{M} \ell\left((x^i,u^i), W_{1:L-1}\right).$$

IV. Model predictive controller

We first fix an optimization horizon $N \in \mathbb{Z}_+$. Let x^r, u^r be a reference (or equilibrium) state-action pair. For given positive definite matrices $Q, R \succ 0$, $P \succ 0$ is the solution of the following Lyapunov equation:

$$(A + B\mathcal{K})^\top P (A + B\mathcal{K}) - P = -(Q + \mathcal{K}^\top R \mathcal{K}), \quad (9)$$

where $\mathcal{K}$ is such that $A + B\mathcal{K}$ is Schur stable. We define the following cost

$$\psi(z_{0:N+1}, v_{0:N}) := \|z_N - x^r\|_P^2 + \sum_{i=0}^{N-1} \|z_i - x^r\|_Q^2 + \|v_i - u^r\|_R^2.$$

Let us define $R_{i+1} = (A + B\mathcal{K})R_i \oplus \mathbb{W}$ with $R_0 = \{0\}$. For $i = 0, \dots, N-1$, we impose the following constraints:

$$\begin{aligned} &\bar{z}_{i+1} = A\bar{z}_i + Bv_i \\ &\bar{z}_i \in \mathcal{X} \ominus R_i, \quad v_i \in \mathbb{U} \ominus \mathcal{K}R_i \\ &\bar{z}_N \in \Omega \ominus R_N, \end{aligned} \quad (10)$$

where Ω is a disturbance invariant set. At each time t, we measure the state x_t of the system (1) and solve the following optimization problem:

$$\begin{aligned} \min_{(c_i)_{i=0}^{N-1}} \quad & \psi(z_{0:N+1}, v_{0:N}) \\ \text{s.t.} \quad & z_0 = \bar{z}_0 = x_t \\ & v_i = \mathcal{K}\bar{z}_i + c_i \text{ for } i \in \mathbb{Z}_{[0,N-1]} \\ & z_{i+1} = Az_i + Bv_i + \hat{h}_t(z_i,v_i) \text{ for } i \in \mathbb{Z}_{[0,N-1]} \\ & \text{Eq. (10).} \end{aligned} \quad (11)$$

By solving the above problem, we get $v_0 = \mathcal{K}x_t + c_0$. We set $u_t = v_0$ and apply u_t to the system (1).

V. Properties of LBMPC

For the purpose of analysis, we define $\tilde{K}_t := K_t - W^*$, $\tilde{x}_{t+1} := \hat{x}_{t+1} - x_{t+1} = \hat{h}_t(x_t,u_t) - h(x_t,u_t) = \tilde{K}_t^\top \phi_j(x_t,u_t) - \varepsilon_j(x_t,u_t)$ and $\bar{W} := \sum_{i=1}^m \bar{W}_i^2$. We recall the following definition:

Definition 1 ([25], page 117): The vector sequence $(s_t)_{t\in\mathbb{N}}$ is called μ small in mean square sense if it satisfies $\sum_{t=k}^{k+N-1} \|s_t\|^2 \leqslant Nc_0\mu + c_0'$ for all $k \in \mathbb{Z}_+$, a given constant $\mu \geqslant 0$ and some $N \in \mathbb{Z}_+$, where $c_0, c_0' \geqslant 0$.
We make the following assumption:

Assumption 2: There exists $\bar{\varepsilon} > 0$ such that

$$\left\|\varepsilon_j(x,u)\right\| \leqslant \bar{\varepsilon} \text{ for each } (x,u) \in \mathcal{X}\times\mathbb{U} \text{ and } j \in \mathbb{N}.$$

Since h is bounded due to (1-c), W^*, ϕ_j due to Assumption 1 and K_t due to (7), the above assumption is trivially satisfied. We have the following result that says that the estimation error $\tilde{x}_{t+1} = \hat{h}_t(x_t,u_t) - h(x_t,u_t)$ is small in mean square sense.

Lemma 1: Consider the dynamical system (1), weight update law (7). Let the assumptions 1 and 2 hold. Let us define $V_a(K_t) := \frac{1}{\gamma}\operatorname{tr}(\tilde{K}_t^\top \tilde{K}_t)$. Then for all t,

(i) $V_a(K_t) \leqslant \frac{4}{\gamma}\bar{W}$,

(ii) $V_a(K_{t+1}) - V_a(K_t) \leqslant -\frac{1-\gamma}{\sigma^2}\|\tilde{x}_{t+1}\|^2 + \left\|\varepsilon_j(x_t,u_t)\right\|^2$,

(iii) $\tilde{x}_t$ is $\bar{\varepsilon}^2$ small in mean square sense with $c_0 = \frac{\sigma^2}{1-\gamma}$ and $c_0' = \frac{4c_0}{\gamma}\bar{W}$ as per the Definition 1.

Proof: (Proof of Lemma 1)

(i) Since $V_a(K_t) = \frac{1}{\gamma}\operatorname{tr}(\tilde{K}_t^\top \tilde{K}_t) = \frac{1}{\gamma}\sum_{i=1}^m \left\|K_t^{(i)} - W^{*(i)}\right\|^2 \leqslant \frac{2}{\gamma}\sum_{i=1}^m \left\|K_t^{(i)}\right\|^2 + \left\|W^{*(i)}\right\|^2 \leqslant \frac{4}{\gamma}\sum_{i=1}^m \bar{W}_i^2 = \frac{4}{\gamma}\bar{W}$.

(ii) By substituting $\tilde{K}_{t+1} = \bar{K}_{t+1} - W^* + K_{t+1} - \bar{K}_{t+1}$ in $V_a(K_{t+1})$ and defining $\alpha_t := (K_{t+1} - \bar{K}_{t+1})^\top(K_{t+1} - \bar{K}_{t+1}) + 2(K_{t+1} - \bar{K}_{t+1})^\top(\bar{K}_{t+1} - W^*) = -(K_{t+1} - \bar{K}_{t+1})^\top(K_{t+1} - \bar{K}_{t+1}) + 2(K_{t+1} - \bar{K}_{t+1})^\top(K_{t+1} - W^*)$, we get

$$\begin{aligned}V_a(K_{t+1}) &= \frac{1}{\gamma}\operatorname{tr}(\tilde{K}_{t+1}^\top \tilde{K}_{t+1})\\ &= \frac{1}{\gamma}\operatorname{tr}\left((\bar{K}_{t+1} - W^*)^\top(\bar{K}_{t+1} - W^*)\right) + \frac{1}{\gamma}\operatorname{tr}(\alpha_t).\end{aligned}$$

One important property of the projection is the following [25, (4.61)]:

$$(W^{*(i)} - K_t^{(i)})^\top(\bar{K}_t^{(i)} - K_t^{(i)}) \leqslant 0 \text{ for each } i = 1,\ldots,m. \tag{12}$$

Since $(K_{t+1}^{(i)} - \bar{K}_{t+1}^{(i)})^\top(K_{t+1}^{(i)} - W^*) \leqslant 0$ due to (12), we can ensure $\operatorname{tr}(\alpha_t) \leqslant 0$. Therefore,

$$\begin{aligned}V_a(K_{t+1}) &\leqslant \frac{1}{\gamma}\operatorname{tr}\left((\bar{K}_{t+1} - W^*)^\top(\bar{K}_{t+1} - W^*)\right)\\ &= V_a(K_t) + \frac{\gamma}{\left\|\phi_j(x_t,u_t)\right\|^2}\operatorname{tr}\left(\tilde{x}_{t+1}\tilde{x}_{t+1}^\top\right) - \frac{1}{\left\|\phi_j(x_t,u_t)\right\|^2}\\ &\quad\times\operatorname{tr}\left(\tilde{K}_t^\top\phi_j(x_t,u_t)\tilde{x}_{t+1}^\top + \tilde{x}_{t+1}\phi_j(x_t,u_t)^\top\tilde{K}_t\right).\end{aligned}$$

By substituting $\tilde{K}_t^\top\phi_j(x_t,u_t) = \tilde{x}_{t+1} + \varepsilon_j(x_t,u_t)$ in the above inequality, we get

$$\begin{aligned}V_a(K_{t+1}) &\leqslant V_a(K_t) + \frac{1}{\left\|\phi_j(x_t,u_t)\right\|^2}\Big(\gamma\|\tilde{x}_{t+1}\|^2 - 2\operatorname{tr}(\tilde{x}_{t+1}\\ &\qquad + \varepsilon_j(x_t))\tilde{x}_{t+1}^\top\Big)\\ &= V_a(K_t) + \frac{1}{\left\|\phi_j(x_t,u_t)\right\|^2}\Big(\gamma\|\tilde{x}_{t+1}\|^2 - 2\tilde{x}_{t+1}^\top(\tilde{x}_{t+1}\\ &\qquad + \varepsilon_j(x_t,u_t))\Big)\\ &= V_a(K_t) + \frac{1}{\left\|\phi_j(x_t,u_t)\right\|^2}\Big((\gamma-2)\|\tilde{x}_{t+1}\|^2\\ &\qquad - 2\tilde{x}_{t+1}^\top\varepsilon_j(x_t,u_t)\Big)\\ &\leqslant V_a(K_t) + \frac{1}{\left\|\phi_j(x_t,u_t)\right\|^2}\left((\gamma-1)\|\tilde{x}_{t+1}\|^2 + \left\|\varepsilon_j(x_t,u_t)\right\|^2\right)\\ &= V_a(K_t) - \frac{1-\gamma}{\left\|\phi_j(x_t,u_t)\right\|^2}\|\tilde{x}_{t+1}\|^2 + \frac{\left\|\varepsilon_j(x_t,u_t)\right\|^2}{\left\|\phi_j(x_t,u_t)\right\|^2}\\ &\leqslant V_a(K_t) - \frac{1-\gamma}{\sigma^2}\|\tilde{x}_{t+1}\|^2 + \left\|\varepsilon_j(x_t,u_t)\right\|^2,\end{aligned}$$

where the last inequality is due to $1 \leqslant \left\|\phi_j(x_t,u_t)\right\|^2 \leqslant \sigma^2$. Therefore,

$$V_a(K_{t+1}) - V_a(K_t) \leqslant -\frac{1-\gamma}{\sigma^2}\|\tilde{x}_{t+1}\|^2 + \left\|\varepsilon_j(x_t,u_t)\right\|^2.$$

(iii) Consider Lemma 1-(ii) to get

$$\begin{aligned}\frac{1-\gamma}{\sigma^2}\|\tilde{x}_{t+1}\|^2 &\leqslant -V_a(K_{t+1}) + V_a(K_t) + \left\|\varepsilon_j(x_t,u_t)\right\|^2\\ &\leqslant -V_a(K_{t+1}) + V_a(K_t) + \bar{\varepsilon}^2.\end{aligned}$$

By summing from $t = k$ to $k+N-1$ in both sides, we get

$$\begin{aligned}\frac{1-\gamma}{\sigma^2}\sum_{t=k}^{k+N-1}\|\tilde{x}_{t+1}\|^2 &\leqslant V_a(K_k) + N\bar{\varepsilon}^2,\\ &\leqslant \frac{4}{\gamma}\bar{W} + N\bar{\varepsilon}^2.\end{aligned}$$

Therefore, $\tilde{x}_t$ is $\bar{\varepsilon}^2$ small in mean square sense with $c_0 = \frac{\sigma^2}{1-\gamma}$ and $c_0' = \frac{4c_0}{\gamma}\bar{W}$ as per the Definition 1.

■

We recall the following definition:

Definition 2 ([26]): A system is robust asymptotically stable around x^r if there exists a class-$\mathcal{KL}$ function β and for every $\varepsilon > 0$ there exists $\delta > 0$ such that $\|h(x_t,u_t)\| \leqslant \delta \implies \|x_t - x^r\| \leqslant \beta(\|x_t - x^r\|, t) + \varepsilon$ for all t.

Theorem 1 (Recursive feasibility and stability): The control law of LBMPC (11) is robust asymptotically stable with respect to the disturbances. The closed-loop system (1) under LBMPC (11) is input-to-state stable. The optimization problem (11) is recursively feasible.

Proof: Since $\hat{h}_t$ is a continuous and uniformly bounded function due to the construction, the result follows from [3, Theorem 2]. The second result is a direct implication of [27, Lemma 3.5]. The recursive feasibility of (11) is due to [3, Theorem 1]. In particular, if $(c_0, c_1, \ldots, c_{N-1})$ is an optimizer of (11) at some time t then $(c_1, \ldots, c_{N-1}, 0)$ will be a feasible solution at time $t+1$. ■

VI. Numerical experiment

We consider the compression system of a jet engine, which exhibits instabilities due to rotating stall and surge. The Moore-Greitzer compressor model is given by the following nonlinear dynamics:

$$\begin{aligned}\dot{z} &= -y + z_c + 1 + \frac{3}{2}z - \frac{1}{2}z^3 \\ \dot{y} &= \frac{1}{\beta^2}\left(z + 1 - r\sqrt{z}\right),\end{aligned} \tag{13}$$

where z is mass flow, y is pressure rise, $\beta > 0$ is a constant and r is the throttle opening. Similar to [3], we assume that r is controlled by a second order actuation with transfer function $r(s) = \frac{\omega_n^2}{s^2+2\zeta\,\omega_n s+\omega_n^2}u(s)$, where u is the input. Our simulation parameters are $\beta = 1, z_c = 0, \zeta = \frac{1}{\sqrt{2}}, \omega_n = 10\sqrt{10}$. We have the following constraints:

$$\begin{aligned} &z \in [0,1] \quad y \in [1.1875, 2.1875] \\ &r \in [0.1547, 2.1547] \quad \dot{r} \in [-20, 20] \\ &u \in [0.1547, 2.1547] \end{aligned} \tag{14}$$

The state x of the system is represented by $x = \begin{bmatrix} z & y & r & \dot{r}\end{bmatrix}^\top \in \mathbb{R}^4$. The nonlinear model (13) is linearized around the equilibrium point $x_e = \begin{bmatrix}0.5 & 1.6875 & 1.1547 & 0\end{bmatrix}^\top$ and discretized with sampling time $T = 0.05$ seconds. We chose $T = 0.05$ to make it slightly larger than the solver time in one optimization problem corresponding to linear MPC for its online implementation. We use a learning rate of 0.001 to train the deep neural network in parallel.

We employed CasADI for the symbolic representation of the underlying optimization problem of MPC (11). This symbolic representation is done offline and therefore, helpful in online implementation of MPC by updating only the measured state. Since PyTorch has inbuilt tools for the training of neural networks, we want to use it along with CasADI. We design a neural network on main machine with the help of CasADI and that on the secondary machine by PyTorch.

Another difficulty is associated with the computation of terminal set Ω and reachable sets R_i's. The computation of R_i is very hard for large i. MATLAB based tools like MPT are not available in python for polytopic manipulation. The python library pytope has very limited capability and is under development. Therefore, we followed the approach of [28]. Let the set $\mathcal{X} \ominus \mathbb{W}$, $\mathcal{X}$ and $\mathbb{U}$ be given by

$$\begin{aligned}\mathcal{X} \ominus \mathbb{W} &:= \{x \in \mathbb{R}^d \mid F_p x \leqslant h_p\} \\ \mathcal{X} &:= \{x \in \mathbb{R}^d \mid F_x x \leqslant h_x\} \\ \mathbb{U} &:= \{x \in \mathbb{R}^m \mid F_u x \leqslant h_u\},\end{aligned}$$

where F_p, F_x, F_u are suitable matrices and h_p, h_x, h_u are suitable vectors required to provide the half-space representations of above sets. The set Ω and R_i are approximated in [28].

We are interested in comparing our proposed approach with the state-of-the-art [3], which employs L2NW as an estimator. We implemented L2NW with the help of CasADI in terms of symbolic variables. One of the inputs for L2NW is a data buffer of fixed size, used as function parameters to learn the uncertainties. In an online implementation, this data buffer is empty at the beginning and its size increases with time. Therefore, we initialize a fixed size buffer in such a way that default values do not affect the outcome of the L2NW estimator. Our simulation parameters are same as in [3] except the sampling time T. Figures 3 and 4 demonstrate that the proposed approach has faster transient response than that of linear MPC and smaller overshoot than that of [3]. It is demonstrated in Fig. 5 that the proposed approach and linear MPC both have comparable solver time but that in [3] is larger than the sampling time at the beginning.

Only drawback in [3] is the use of L2NW estimator, which makes the underlying optimization problem of LBMPC computationally demanding and therefore hard to implement on a small machines. Since the approach [3] gives choice to use any estimator, authors used linear estimators in [4], [28] instead of L2NW estimator. Our approach of using DNN as an estimator is not only computationally less demanding due to the parallel processing but surprisingly the underlying optimization problems have similar solver time as linear MPC (Fig. 5). Therefore, the proposed approach can be implemented on small machines.

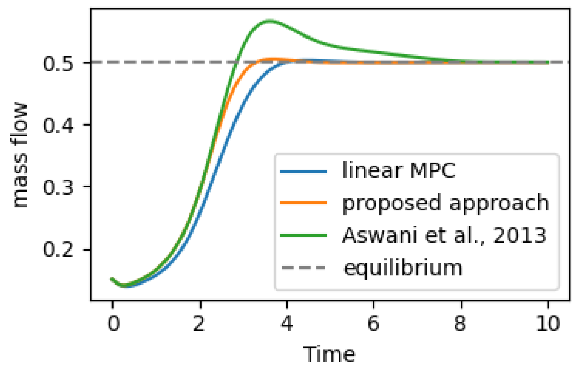

Fig. 3: Proposed approach and [3] both have faster transient response than that of linear MPC but the proposed approach has smaller overshoot than that of [3].

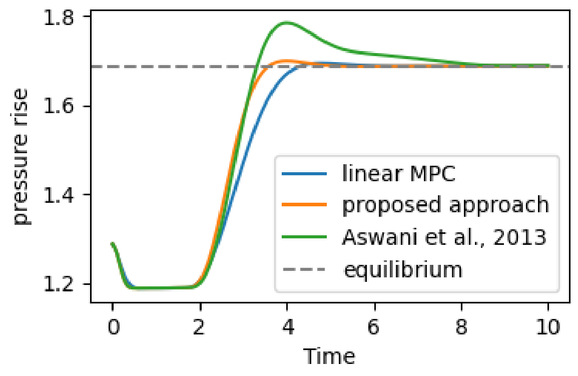

Fig. 4: Proposed approach and [3] both have faster transient response than that of linear MPC but the proposed approach has smaller overshoot than that of [3].

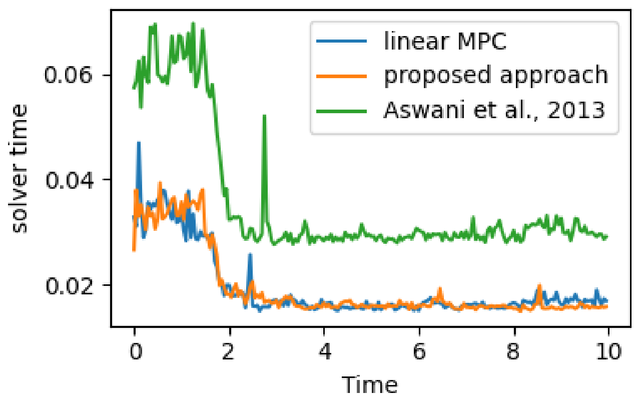

Fig. 5: Solver time in both linear MPC and proposed approach is smaller than that of [3].

VII. Conclusion

This article demonstrates that DNN can be used in the framework of LBMPC by dual time scale training and using bounded neurons in the last activation layer. The proposed approach is able to provide a faster solution because a DNN can be trained on a separate machine. Since LBMPC allows any estimator as an oracle, different DNN architectures can be investigated for different class of problems by following the proposed approach. Some interesting extensions may be possible along the lines of vision based navigation [29], stochastic MPC [30], [31] and Bayesian Neural Network [32].

References

[1] L. Hewing, K. P. Wabersich, M. Menner, and M. N. Zeilinger, "Learning-based model predictive control: Toward safe learning in control," *Annual Review of Control, Robotics, and Autonomous Systems*, vol. 3, pp. 269–296, 2020.

[2] L. Brunke, M. Greeff, A. W. Hall, Z. Yuan, S. Zhou, J. Panerati, and A. P. Schoellig, "Safe learning in robotics: From learning-based control to safe reinforcement learning," *Annual Review of Control, Robotics, and Autonomous Systems*, vol. 5, pp. 411–444, 2022.

[3] A. Aswani, H. Gonzalez, S. S. Sastry, and C. Tomlin, "Provably safe and robust learning-based model predictive control," *Automatica*, vol. 49, no. 5, pp. 1216–1226, 2013.

[4] P. Bouffard, A. Aswani, and C. Tomlin, "Learning-based model predictive control on a quadrotor: Onboard implementation and experimental results," in *International Conference on Robotics and Automation*. IEEE, 2012, pp. 279–284.

[5] T. Zieger, H. H. Nguyen, E. Schulz, T. Oehlschlägel, and R. Findeisen, "Non-diverging neural networks supported tube model predictive control," in *61st Conf. on Decision and Control*. IEEE, 2022, pp. 3066–3073.

[6] J. Ding, K. Li, and K. Hedrick, "A robust lane-keeping 'co-pilot' system using LBMPC method," *SAE International Journal of Passenger Cars-Electronic and Electrical Systems*, vol. 8, no. 2015-01-0322, pp. 219–228, 2015.

[7] A. Aswani, N. Master, J. Taneja, A. Krioukov, D. Culler, and C. Tomlin, "Energy-efficient building HVAC control using hybrid system LBMPC," *IFAC Proceedings Volumes*, vol. 45, no. 17, pp. 496–501, 2012.

[8] A. T. Hafez, S. N. Givigi, S. Yousefi, and A. Noureldin, "Cooperative unmanned aerial vehicles formation via decentralized LBMPC," in *23rd Mediterranean Conference on Control and Automation (MED)*. IEEE, 2015, pp. 377–383.

[9] A. D. Bonzanini, D. B. Graves, and A. Mesbah, "Learning-based SMPC for reference tracking under state-dependent uncertainty: An application to atmospheric pressure plasma jets for plasma medicine," *IEEE Trans. on Control Systems Technology*, vol. 30, no. 2, pp. 611–624, 2021.

[10] C. Eckel, M. Maiworm, and R. Findeisen, "Optimal operation and control of towing kites using online and offline gaussian process learning supported model predictive control," in *American Control Conference*. IEEE, 2022, pp. 2637–2643.

[11] V. N. Vapnik, "An overview of statistical learning theory," *IEEE trans. on neural networks*, vol. 10, no. 5, pp. 988–999, 1999.

[12] G. Joshi and G. Chowdhary, "Deep model reference adaptive control," in *58th Conference on Decision and Control*. IEEE, 2019, pp. 4601–4608.

[13] P. K. Mishra, M. V. Gasparino, A. E. B. Velsasquez, and G. Chowdhary, "Deep model predictive control with stability guarantees," *https://arxiv.org/abs/2104.07171*, 2021.

[14] S. J. Pan and Q. Yang, "A survey on transfer learning," *IEEE Trans. on knowledge and data engineering*, vol. 22, no. 10, pp. 1345–1359, 2010.

[15] F. Zhuang, Z. Qi, K. Duan, D. Xi, Y. Zhu, H. Zhu, H. Xiong, and Q. He, "A comprehensive survey on transfer learning," *Proceedings of the IEEE*, vol. 109, no. 1, pp. 43–76, 2020.

[16] F. W. Lewis, S. Jagannathan, and A. Yesildirak, *Neural network control of robot manipulators and non-linear systems*. Taylor & Francis, London, U.K., 1999.

[17] G. Joshi and G. Chowdhary, "Deep model reference adaptive control," in *58th Conference on Decision and Control*. IEEE, 2019, pp. 4601–4608.

[18] T. Hayakawa, W. M. Haddad, and N. Hovakimyan, "Neural network adaptive control for a class of nonlinear uncertain dynamical systems with asymptotic stability guarantees," *IEEE Trans. on Neural Networks*, vol. 19, no. 1, pp. 80–89, 2008.

[19] X. Yang, D. Liu, Q. Wei, and D. Wang, "Direct adaptive control for a class of discrete-time unknown nonaffine nonlinear systems using neural networks," *International Journal of Robust and Nonlinear Control*, vol. 25, no. 12, pp. 1844–1861, 2015.

[20] I. Landau, R. Lozano, M. M'Saad, and A. Karimi, *Adaptive control: algorithms, analysis and applications*. Springer Science & Business Media, 2011.

[21] L. Hewing, J. Kabzan, and M. N. Zeilinger, "Cautious model predictive control using gaussian process regression," *IEEE Trans. on Control Systems Tech.*, vol. 28, no. 6, pp. 2736–2743, 2019.

[22] B. Scholkopf, S. Mika, C. J. C. Burges, P. Knirsch, K.-R. Muller, G. Ratsch, and A. J. Smola, "Input space versus feature space in kernel-based methods," *IEEE trans. on neural networks*, vol. 10(5), pp. 1000–1017, 1999.

[23] A. Jain, T. Nghiem, M. Morari, and R. Mangharam, "Learning and control using Gaussian processes," in *9th International Conference on Cyber-Physical Systems*. IEEE, 2018, pp. 140–149.

[24] T. De Bruin, J. Kober, K. Tuyls, and R. Babuska, "Experience selection in deep reinforcement learning for control," *Journal of Machine Learning Research*, vol. 19, 2018.

[25] P. Ioannou and B. Fidan, *Adaptive control tutorial*. SIAM, 2006.

[26] G. Grimm, M. J. Messina, S. E. Tuna, and A. R. Teel, "Examples when nonlinear model predictive control is nonrobust," *Automatica*, vol. 40, no. 10, pp. 1729–1738, 2004.

[27] Z. Jiang and Y. Wang, "Input-to-state stability for discrete-time nonlinear systems," *Automatica*, vol. 37, no. 6, pp. 857–869, 2001.

[28] A. Aswani, P. Bouffard, and C. Tomlin, "Extensions of learning-based model predictive control for real-time application to a quadrotor helicopter," in *American Control Conf.* IEEE, 2012, pp. 4661–4666.

[29] M. V. Gasparino, A. N. Sivakumar, Y. Liu, A. E. B. Velasquez, V. A. H. Higuti, J. Rogers, H. Tran, and G. Chowdhary, "Wayfast: Navigation with predictive traversability in the field," *IEEE Robotics and Automation Letters*, vol. 7, no. 4, pp. 10 651–10 658, 2022.

[30] P. K. Mishra, D. Chatterjee, and D. E. Quevedo, "Stochastic predictive control under intermittent observations and unreliable actions," *Automatica*, vol. 118, p. 109012, 2020.

[31] ——, "Stabilizing stochastic predictive control under bernoulli dropouts," *IEEE Transactions on Automatic Control*, vol. 63, no. 6, pp. 1579–1590, 2017.

[32] Y. Bao, K. J. Chan, A. Mesbah, and J. M. Velni, "Learning-based adaptive-scenario-tree model predictive control with improved probabilistic safety using robust bayesian neural networks," *International Journal of Robust and Nonlinear Control*, 2022.