Constrained Deep Learning Based Nonlinear Model Predictive Control

Farshid Asadi¹

Abstract—Learning-based model predictive control (MPC) is an approach designed to reduce the computational cost of MPC. In this paper, a constrained deep neural network (DNN) design is proposed to learn MPC policy for nonlinear systems. Using constrained training of neural networks, MPC constraints are enforced effectively. Furthermore, recursive feasibility and robust stability conditions are derived for the learning-based MPC approach. Additionally, probabilistic feasibility and optimality empirical guarantees are provided for the learned control policy. The proposed algorithm is implemented on the Furuta pendulum and control performance is demonstrated and compared with the exact MPC and the normally trained learning-based MPC. The results show superior control performance and constraint satisfaction of the proposed approach.

I. Introduction

Model predictive control is an advanced control technique based on optimization while considering system constraints [1]–[3]. High computational demand of the nonlinear MPC [4] is a challenge for its wide scalable application. Over the past two decades, various approaches are proposed to solve the scalability challenge. Fast solvers tailored to the structure of the MPC optimization are used in [4]–[6] and implemented on embedded systems e.g., [7], [8]. In addition, soft constraint formulation is proposed to relax the problem and alleviate computational cost of MPC [9], [10], however, they are not widely used because of nonstrict nature of constraint satisfaction. Explicit MPC (EMPC) technique for linear systems is another approach for reducing computational burden problem. The main idea of EMPC is to pre-compute the control laws offline and use them in online operation [11]. However, computational demand and memory requirement in EMPC rise dramatically as the size of the problem increases [12]. A plausible solution to alleviate this, is to use various approximated EMPC methods [13]–[15].

Recent advances in machine learning techniques, such as DNNs [16], bring new opportunities to integrate with EMPC. For example, piece-wise affine property of solution to linear MPC is used to show the learnability of linear MPC policies [17]–[21]. Deep EMPC for nonlinear systems is also proposed in [10], [22]–[24]. Optimality and constraint satisfaction of the learned policy are important concerns of learning-based EMPC. In [19], [22], [24], probabilistic confidence level for feasibility is proposed based on empirical post validation [25]. The works in [10], [20], [21] use neural network structure properties to guarantee control design feasibility, but are limited to linear time-invariant

systems. Furthermore, no explicit method is introduced for imposing constraints on DNN.

To improve constraint satisfaction for deep learning-based EMPC, we consider enforcing constraints during the training process. Up until recently, DNNs output constraints are modeled as soft constraints [26], namely, adding constraints as a penalty to the loss function. However, it is not straightforward to choose weigh factors and constraint satisfaction is not guaranteed [26]. Recently, it became possible to include hard constraints in training DNNs. For example, equality output constraints of DNNs are considered in [26] and the method of Lagrange multiplier is used to train the network. Further, inequality constraints are studied in [27], and a constrained training problem is solved by applying alternating stochastic gradient descent/ascent iteratively.

In some deep EMPC algorithms (e.g., [19], [22], [24]), no constraint is enforced during training stage. On the other hand, some algorithms use relatively computationally heavy methods to enforce constraints in training [17] or only enforce simple bound constraints on the neural network [18]. In this work, we use constrained DNN to learn MPC policy for nonlinear systems while enforcing constraints during the training. Compared with the existing methods of deep learning-based EMPC (e.g., [18], [19], [22], [24]), the proposed approach explicitly enforces constraints during the training process. Unlike the method in [17], the improved constraint satisfaction is achieved without adding extra computational burden in the final learned policy. We also provide recursive feasibility and robust stability analvsis for the learned policy. The proposed constrained deep EMPC algorithm is implemented on an underactuated rotary pendulum and its performance and constraint satisfaction are compared with the exact MPC method and the normally trained deep EMPC. The main contribution of this work lies in the new deep learning-based EMPC for nonlinear systems with improved constraint satisfaction and guaranteed robust stability.

II. PROBLEM STATEMENT

Consider the following discrete-time nonlinear system,

$$x^+ = f(x, u) \tag{1}$$

where $x \in \mathbb{R}^n$ is the state variable, $u \in \mathbb{R}^m$ is the control input, $f : \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}^n$ is a function representing system dynamics with f(0,0) = 0, and x^+ is the state variable at the next time step, respectively. The MPC is formulated,

This work was partially supported by the US NSF under award CNS-1932370.

Author is with Rutgers University, New Brunswick, NJ 08854, USA ¹ fa416@scarletmail.rutgers.edu

for equation (1), via the following optimization problem [1],

$$V_{N}^{*}(\boldsymbol{x}) = \min_{\boldsymbol{U}} V_{N}(\boldsymbol{x}, \boldsymbol{U})$$
s.t.
$$V_{N}(\boldsymbol{x}, \boldsymbol{U}) = \sum_{k=0}^{N-1} l(\boldsymbol{x}_{k}, \boldsymbol{u}_{k}) + V_{f}(\boldsymbol{x}_{N})$$

$$\boldsymbol{x}_{k} = \boldsymbol{\phi}(k; \boldsymbol{x}, \boldsymbol{U}),$$

$$\boldsymbol{x}_{k} \in \mathbb{X}, \boldsymbol{x}_{N} \in \mathbb{X}_{f} \subseteq \mathbb{X}, \boldsymbol{u}_{k} \in \mathbb{U}$$

$$\boldsymbol{x}_{0} = \boldsymbol{x}, k = 0, \dots, N-1$$

$$(2)$$

where k is the time step over control horizon, \boldsymbol{x}_k and \boldsymbol{u}_k are state vector and control input at the k_{th} time step of the control horizon with respect to current time step, $\boldsymbol{U}_k = [\boldsymbol{u}_0^T, \dots, \boldsymbol{u}_{k-1}^T]^T$ is control sequence of length k, and $\boldsymbol{U} = \boldsymbol{U}_N$. Functions $l: \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}_{\geq 0}$ and $V_f: \mathbb{R}^n \to \mathbb{R}_{\geq 0}$ represent stage and terminal cost, respectively. The function $\boldsymbol{\phi}$ denotes the solution of the system in equation (1) at k_{th} time step of horizon when $\boldsymbol{x}_0 = \boldsymbol{x}$ and control sequence \boldsymbol{U} is applied. Sets $\mathbb{X} \subseteq \mathbb{R}^n$, \mathbb{X}_f , and $\mathbb{U} \subseteq \mathbb{R}^m$ denote state, terminal, and control constraints, respectively.

The MPC problem must be solved at each time step and only first element of optimal control sequence $U^*(x) = [u_0^*(x)^T, \dots, u_{N-1}^*(x)^T]^T$ is applied to the system, i.e.,

$$\boldsymbol{u}^*(\boldsymbol{x}) = \boldsymbol{u}_0^*(\boldsymbol{x}) \tag{3}$$

Our main goal is to learn the control policy in equation (3) with a DNN while enforcing constraints of equation (2) during the training process. This way, we obtain a computationally efficient approximate MPC policy with improved constraint satisfaction.

Following standard MPC assumptions [1, pp. 97–98, 114, 116] are considered in the current work.

Assumption 1: System dynamics is locally Lipschitz continuous in its arguments, i.e., for all $x, \bar{x} \in \mathbb{X}$ and $u, \bar{u} \in \mathbb{U}$ there exist some constant $l_f > 0$ such that

$$\|f(x, u) - f(\bar{x}, \bar{u})\| \le l_f \|x - \bar{x}\| + l_f \|u - \bar{u}\|$$
 (4)

Assumption 2: Stage and terminal cost are defined by quadratic forms, that is

$$egin{aligned} l(oldsymbol{x},oldsymbol{u})\coloneqq oldsymbol{x}^\intercal oldsymbol{Q} oldsymbol{x} + oldsymbol{u}^\intercal oldsymbol{R} oldsymbol{u} \ V_f(oldsymbol{x})\coloneqq oldsymbol{x}^\intercal oldsymbol{Q}_f oldsymbol{x} \end{aligned}$$

where $Q \succ 0$, $Q_f \succ 0 \in \mathbb{R}^{n \times n}$ and $R \succ 0 \in \mathbb{R}^{m \times m}$ are positive definite matrices.

Assumption 3: The set $\mathbb{Z} := \mathbb{X} \times \mathbb{U}$ is closed and contains origin. Further, sets \mathbb{X} and \mathbb{U} are defined via the following element-wise inequalities, respectively,

$$X := \{x \in \mathbb{R}^{n} | C_{x}(x) \leq 0\}$$

$$\mathbb{U} := \{u \in \mathbb{R}^{m} | C_{u}(x, u) \leq 0 \text{ for } x \in \mathbb{X}\}$$
(5)

where $C_x: \mathbb{R}^n \to \mathbb{R}^{n_c}$ and $C_u: \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}^{n_u}$ are continuous functions.

Assumption 4: For all $x \in X_f$, there exist a locally Lipschitz continuous $u_f \in \mathbb{U}$ such that

$$egin{aligned} oldsymbol{f}(oldsymbol{x}, oldsymbol{u}_f) &\in \mathbb{X}_f \ V_f\left(oldsymbol{f}(oldsymbol{x}, oldsymbol{u}_f)) - V_f(oldsymbol{x}) \leq -l(oldsymbol{x}, oldsymbol{u}_f) \ V_f\left(oldsymbol{f}(oldsymbol{x}, oldsymbol{u}_f)) \leq \gamma_f V_f(oldsymbol{x}) \end{aligned}$$

where $\gamma_f \in (0,1)$ and the terminal set X_f is defined as

$$X_f := \{ \boldsymbol{x} \in X \mid V_f(\boldsymbol{x}) \le c_f \}$$

for some $c_f > 0$.

Assumption 5: There exist \mathcal{K}_{∞} function $\alpha(.)$ such that

$$V_N^*(\boldsymbol{x}) \le \alpha(\|\boldsymbol{x}\|) \quad \forall \boldsymbol{x} \in \mathcal{X}_N$$

where \mathcal{X}_N is defined in the next section.

III. PRELIMINARY BACKGROUND

DNNs have shown promising potentials in different learning schemes [16]. In this work, deep multilayer perceptron (MLP) is used to approximate the MPC policy in a computationally efficient closed form. A deep MLP can be described in the following form [16].

$$\mathcal{N}(\boldsymbol{x};\boldsymbol{\theta},M,l) = f_{l+1} \circ g_l \circ f_l \circ \cdots \circ g_1 \circ f_1(\boldsymbol{x})$$
 (6)

where $x \in \mathbb{R}^n$ and $\mathcal{N} \in \mathbb{R}^m$ are the network input and output, respectively, M is the number of hidden nodes per layer, l is the number of hidden layers, and θ is the network parameters. The function $f_i(\cdot)$ is a linear affine map between each layer and $g_i(\cdot)$ is nonlinear activation function, $i=1,\ldots,l$. The goal in constrained deep learning is to solve the following constrained minimization problem for θ ,

$$\min_{\boldsymbol{\theta}} \ \mathcal{L}(\boldsymbol{\theta}) = \mathbb{E}\left[L\left(\boldsymbol{y} - \hat{\boldsymbol{y}}(\boldsymbol{x})\right)\right]$$
 s.t. $C(\boldsymbol{\theta}, \boldsymbol{x}, \hat{\boldsymbol{y}}) < 0$ (7)

where L(.) is a loss function, C(.) is constraint on network prediction, $y \in \mathbb{R}^m$ is the vector assigned to x by the underlying function, $\hat{y} = \mathcal{N}(x; \theta, M, l)$ is the neural network prediction for y, and expected value is calculated based on distribution of x. The optimization equation (7) is intractable to solve in this form and scenario optimization [28], that is solving a sampled optimization based on the original problem, is one approach to approximately solve equation (7). Therefore, we consider the sampled version of equation (7).

$$\min_{\boldsymbol{\theta}} \ \mathcal{L}(\boldsymbol{\theta}) = \frac{1}{N_s} \sum_{i=0}^{N_s} L\left(\boldsymbol{y}_i - \hat{\boldsymbol{y}}(\boldsymbol{x}_i)\right)$$
s.t. $C(\boldsymbol{\theta}, \boldsymbol{x}_i, \hat{\boldsymbol{y}}_i) \leq \mathbf{0}, \ i = 1, \dots, N_s$ (8)

where N_s is sample numbers.

To compare empirical realization of optimality and constraint satisfaction of the constrained optimization of equation (7), when using solutions of equation (8), following indicators are used.

$$I_{o}(\boldsymbol{x}) = \begin{cases} 1, & \|\boldsymbol{y} - \hat{\boldsymbol{y}}(\boldsymbol{x})\|_{\infty} \leq \eta \\ 0, & else \end{cases}$$

$$I_{c}(\boldsymbol{x}) = \begin{cases} 1, & C(\boldsymbol{\theta}, \boldsymbol{x}, \hat{\boldsymbol{y}}) \leq 0 \\ 0, & C(\boldsymbol{\theta}, \boldsymbol{x}, \hat{\boldsymbol{y}}) > 0 \end{cases}$$

$$(9)$$

The following proposition will be used later for empirical statistical analysis of optimality and feasibility of the trained networks.

Proposition 1: Let I(x) be a binary indicator of independently and identically distributed (i.i.d.) random variable x, e.g. indicators given in equation (9). Also,let μ be true mean of the given indicator and define $\bar{I} := \frac{1}{N} \sum_{i=1}^{N} I(x_i)$ as its empirical mean over distribution of x. Then using Hoeffding's inequality [25, Theorem 2], the following inequality holds

$$\Pr\left[\Pr\left(I(\boldsymbol{x})=1\right) > \bar{I} - \epsilon_h\right] > \delta_h = 1 - e^{-2N\epsilon_h^2} \quad (10)$$

where Pr(.) represents probability of an event.

To provide recursive feasibility and robust stability analysis of the learned policy, the following standard definitions [1, pp. 96–97] are helpful.

Definition 1 (Implicit control constraint set): The set

$$\mathcal{U}_N(x) := \{ \boldsymbol{U} | (\boldsymbol{x}, \boldsymbol{U}) \in \mathbb{Z}_N \} \tag{11}$$

is called implicit control constraint set for MPC problem defined by equation (2), where $\mathbb{Z}_N \subset \mathbb{X} \times \mathbb{U}^N$ is defined as

$$\mathbb{Z}_{N} := \{ (\boldsymbol{x}, \boldsymbol{U}) | (\boldsymbol{\phi}(k; \boldsymbol{x}, \boldsymbol{U}), \boldsymbol{u}_{k}) \} \in \mathbb{Z}, \forall k \in \mathbb{I}_{0:N-1},$$
$$\boldsymbol{\phi}(N; \boldsymbol{x}, \boldsymbol{U}) \in \mathbb{X}_{f} \}$$
(12)

Definition 2 (N-reachable set): The set

$$\mathcal{X}_N \coloneqq \{ \boldsymbol{x} \in \mathbb{X} | \, \mathcal{U}_N \neq \varnothing \} \tag{13}$$

is called N-reachable set for MPC problem in equation (2). In other words, \mathcal{X}_N is the set of states that the optimization problem in equation (2) has a solution.

Assuming that the assumptions (1) to (5) holds the following theorem can be stated.

Theorem 1 (Asymptotic stability of origin): [1, Theorem 2.19] Suppose that assumptions (1) to (5) hold. Then

a. There exist constants $c_2 > c_1 > 0$ such that for all $\boldsymbol{x} \in \mathcal{X}_N$, we have

$$c_1 \|\mathbf{x}\|^2 \le V_N^*(\mathbf{x}) \le c_2 \|\mathbf{x}\|^2$$

 $V_N^*(\mathbf{f}(\mathbf{x}, \mathbf{u}^*(\mathbf{x}))) - V_N^*(\mathbf{x}) \le -c_1 \|\mathbf{x}\|$

- b. The origin is asymptotically stable in \mathcal{X}_N for the closed loop system, that is defined by $x^+ = f(x, u^*(x))$.
- c. The following inequality holds,

$$V_N^*(\boldsymbol{x}^+) \le \gamma V_N^*(\boldsymbol{x})$$

where $\gamma = (1 - c_1/c_2) \in (0, 1)$.

IV. PROPOSED CONSTRAINED DEEP EMPC

In the proposed learning-based MPC, only learning u^* of equation (3) is considered. So one has

$$\hat{\boldsymbol{u}}^* = \mathcal{N}(\boldsymbol{x}; \boldsymbol{\theta}, M, l) \tag{14}$$

This has two benefits in comparison to learning the whole optimal control sequence, as in some of the referenced literature. First, it makes learning easier due to the smaller learning space. Second, constraint learning reduces to simple single-step constraint checks that facilitates nonlinear constraint

learning. That being said, as it will be shown in the next section, it does not have any effect on the feasibility and robustness of the system. Assuming that $x_0 = x$ is a feasible initial state for equation (2), then the parametric single step constraint can be calculated by considering the collection of constraints on first state and control transition in MPC horizon,

$$C(\boldsymbol{\theta}, \boldsymbol{x}, \hat{\boldsymbol{u}}^*) \le 0 \tag{15}$$

where $C(\boldsymbol{\theta}, \boldsymbol{x}, \hat{\boldsymbol{u}}^*) \in \mathbb{R}^{n_c}$ is defined as

$$C(\theta, x, \hat{u}^*) \coloneqq \left[C_x(x^+)^\intercal, C_u(x, \hat{u}^*)^\intercal \right]^\intercal$$
 (16)

In this way, learning-based MPC problem using the deep neural network can be written as

$$\min_{\boldsymbol{\theta}} \quad \mathcal{L}(\boldsymbol{\theta}) = \frac{1}{N_s} \sum_{i=0}^{N_s} L\left(\boldsymbol{u}^*(\boldsymbol{x}_i) - \hat{\boldsymbol{u}}^*(\boldsymbol{x}_i)\right)
s.t. \quad \boldsymbol{C}_r(\boldsymbol{\theta}, \boldsymbol{x}_i, \hat{\boldsymbol{u}}_i^*) \le 0, \ r \in [1, \dots, n_c]$$
(17)

where C_r refers to r_{th} row of inequality (15). In the above problem one needs to deal with $N_s \times n_c$ inequality constraints. This is not yet tractable, moreover inequality constraints cannot be handled with method of Lagrange multiplier in an straightforward manner. Here, we adopt a method introduced in [27] for constraint conversion and constrained network training. In this regard, note that $C_r(\theta, x, \hat{u}) \leq 0$ and $ReLU\left(C_r(\theta, x, \hat{u})\right) = 0$ are equivalent, where ReLU(x) = max(x, 0) is applied elementwise. Therefore, we can write (17) with equality constraints. Furthermore, because of non-negativeness of ReLU(.) function, $\frac{1}{N_s} \sum_{i=1}^{N_s} ReLU\left(C_r(\theta, x_i, \hat{u}_i)\right) = 0$, is equivalent to $ReLU\left(C_r(\theta, x_i, \hat{u}_i)\right) = 0, \forall i \in N_s$. Thus, number of constraints will be reduced effectively through this process. This can be summarized as

$$\min_{\boldsymbol{\theta}} \quad \mathcal{L}(\boldsymbol{\theta}) = \frac{1}{N_s} \sum_{i=0}^{N_s} L\left(\boldsymbol{u}^*(\boldsymbol{x}_i) - \hat{\boldsymbol{u}}^*(\boldsymbol{x}_i)\right)$$
s.t. $C_r(\boldsymbol{\theta}) = 0, r \in n_c$ (18)

where $C_r(\theta)$ is defined as

$$C_r(\boldsymbol{\theta}) = \frac{1}{N_s} \sum_{i=1}^{N_s} ReLU\left(\boldsymbol{C}_r(\boldsymbol{\theta}, \boldsymbol{x}_i, \hat{\boldsymbol{u}}_i)\right), \forall r \in n_c \quad (19)$$

Now, let us introduce a Lagrange multiplier $\mathbf{\Lambda} = [\lambda_1, \dots, \lambda_{n_c}]^\mathsf{T}$ and convert the constrained minimization of equation (18) to the following maxmin problem,

$$\max_{\mathbf{\Lambda}} \min_{\boldsymbol{\theta}} \mathfrak{L}(\boldsymbol{\theta}, \mathbf{\Lambda}) := \mathcal{L}(\boldsymbol{\theta}) + \sum_{r=1}^{n_c} \lambda_r \mathcal{C}_r(\boldsymbol{\theta})$$
 (20)

The optimization problem of equation (20) can be solved with a two-step alternating approach, that is minimization on θ and maximization on Λ . Let us define the following update rules

$$\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t - \alpha_{\boldsymbol{\theta}} \Delta \boldsymbol{\theta}_t(\boldsymbol{\theta}, \boldsymbol{\Lambda}) \tag{21}$$

$$\mathbf{\Lambda}_{t_0+1} = \mathbf{\Lambda}_{t_0} + \alpha_{\mathbf{\Lambda}} \Delta \mathbf{\Lambda}_{t_0}(\boldsymbol{\theta}, \mathbf{\Lambda}) \tag{22}$$

Algorithm 1:

```
Input: Training data and hyper-parameters i_{\theta}, d, \eta,
            \alpha_{\theta}, \alpha_{\Lambda}^{0}, and threshold.
Output: Trained neural network.
Initialize: \theta randomly, \lambda_r = 0, \forall r \in n_c;
Warm up training:
for i_{\theta} do
     Update \theta using equation (21):
end
Constraint training:
while \|\mathbf{\Lambda}_{t_o} - \mathbf{\Lambda}_{t_o-1}\|_{\infty} > threshold do
     Update \Lambda using equation (22);
     Increment t_o = t_o + 1;
     for s steps do
          Update \theta using equation (21);
          Increment t = t + 1;
     end
     Update s = s + d;
     Update learning rate using \alpha_{\Lambda} = \alpha_{\Lambda}^0 \frac{1}{1+nt};
end
```

where t and t_o are training steps and α_{θ} and α_{Λ} are learning rates. The terms $\Delta\theta(\theta,\Lambda)$ and $\Delta\Lambda(\theta,\Lambda)$ are network weight and Lagrange multiplier update increment, respectively. Any gradient-based algorithm can be used in equation (21), however, only first order gradient-based algorithms can be implemented in equation (22). Algorithm 1 can be used to train the neural network using the given update rules iteratively.

V. ROBUSTNESS AND FEASIBILITY

In this section, we provide robustness and recursive feasibility guarantees of the learned policy. To do this, we use arguments from nominal robustness of standard nonlinear MPC as in [1]. In the sequel, \mathcal{R}_c is defined as the largest level set of $V_N^*(\boldsymbol{x})$ that is contained in \mathcal{X}_N , that is

$$\mathcal{R}_c \coloneqq \{ \max_c | \operatorname{ev}_c\{V_N^*\} | | \operatorname{lev}_c\{V_N^*\} \in \mathcal{X}_N \}$$

where $\operatorname{lev}_a\{V_N^*\} = \{x | V_N^*(x) \leq a\}$. Also, \mathcal{R}_b denotes to the smallest level set of $V_N^*(x)$ that contains \mathbb{X}_f , that is

$$\mathcal{R}_b := \{ \min_b \operatorname{lev}_b\{V_N^*\} | \ \mathbb{X}_f \in \operatorname{lev}_b\{V_N^*\} \}$$

Further, \boldsymbol{x}^* and $\hat{\boldsymbol{x}}^*$ are defined as $\boldsymbol{x}^* \coloneqq \boldsymbol{f}(\boldsymbol{x}, \boldsymbol{u}^*(\boldsymbol{x}))$ and $\hat{\boldsymbol{x}}^* \coloneqq \boldsymbol{f}(\boldsymbol{x}, \hat{\boldsymbol{u}}^*(\boldsymbol{x}))$, respectively. Also, $\phi(k; \boldsymbol{x}) \coloneqq \phi(k; \boldsymbol{x}, \boldsymbol{U}^*(\boldsymbol{x}))$ denotes to the optimal state at k_{th} time step of MPC horizon. Additionally, $\tilde{\boldsymbol{U}}$ and $\tilde{V}_f(\boldsymbol{x}, \boldsymbol{U})$ are defined as

$$ilde{m{U}} \coloneqq egin{bmatrix} m{u}_1^*(m{x})^\intercal, \dots, m{u}_{N-1}^*(m{x})^\intercal, m{u}_f(m{x}) \end{bmatrix}^\intercal \ & ilde{V}_f(m{x}, m{U}) \coloneqq V_f(m{\phi}(N; m{x}, m{U})). \end{aligned}$$

Remark 1: Supposing that assumptions (1), (2) and (4) holds, it directly follows (from their construction) that $V_N(x, U)$ and $\tilde{V}_f(x, U)$ are locally Lipschitz continuous.

That is, for all $x, \bar{x} \in \mathbb{X}$ and $U, \bar{U} \in \mathbb{U}^N$ there exist $\alpha, \alpha_f > 0$ such that

$$||V_N(x, U) - V_N(\bar{x}, \bar{U})|| \le \alpha ||x - \bar{x}|| + \alpha ||U - \bar{U}||$$
 (23)

 $\|\tilde{V}_f(\boldsymbol{x}, \boldsymbol{U}) - \tilde{V}_f(\bar{\boldsymbol{x}}, \bar{\boldsymbol{U}})\| \leq \alpha_f \|\boldsymbol{x} - \bar{\boldsymbol{x}}\| + \alpha_f \|\boldsymbol{U} - \bar{\boldsymbol{U}}\|$ (24) Lemma 1 (Feasibility with learned policy): Suppose that $\boldsymbol{x} \in \mathcal{R}_c$, assumptions (1) to (5) hold and $\|\hat{\boldsymbol{u}}^*(\boldsymbol{x}) - \boldsymbol{u}^*(\boldsymbol{x})\| \leq e$ for all $\boldsymbol{x} \in \mathcal{R}_c$ and that $\alpha_f l_f e \leq (1 - \gamma_f) c_f$. Then, the MPC problem remains feasible using the learned policy, that is the MPC optimization is feasible for $\hat{\boldsymbol{x}}^*$ for all $\boldsymbol{x} \in \mathcal{R}_c$.

Proof: In order to prove the lemma we need to show that $\phi(N; \hat{x}^*) \in \mathbb{X}_f$ holds. Using equation (24) one can write

$$\tilde{V}_f(\hat{\boldsymbol{x}}^*, \tilde{\boldsymbol{U}}) \leq \tilde{V}_f(\boldsymbol{x}^*, \tilde{\boldsymbol{U}}) + \alpha_f \|\hat{\boldsymbol{x}}^* - \boldsymbol{x}^*\|$$
 (25)

Substituting equation (4) into second term in right hand side of equation (25) and using $\|\hat{u}^*(x) - u^*(x)\| \le e$ per assumption, one has

$$\tilde{V}_f(\hat{\boldsymbol{x}}^*, \tilde{\boldsymbol{U}}) \le \tilde{V}_f(\boldsymbol{x}^*, \tilde{\boldsymbol{U}}) + \alpha_f l_f e$$
 (26)

Using assumption (3), definition of $\tilde{V}_f(.)$, and the fact that $\phi(N; x) \in \mathbb{X}_f$ one has

$$\tilde{V}_f(\boldsymbol{x}^*, \tilde{\boldsymbol{U}}) = V_f(\boldsymbol{f}(\boldsymbol{\phi}(N; \boldsymbol{x}), \boldsymbol{u}_f(\boldsymbol{\phi}(N; \boldsymbol{x})))) \le \gamma_f c_f$$
 (27)

Considering definition of $\tilde{V}_f(.)$ and equations (26) and (27) one has

$$V_f(\phi(N; \hat{\boldsymbol{x}}^*, \tilde{\boldsymbol{U}})) = \tilde{V}_f(\hat{\boldsymbol{x}}^*, \tilde{\boldsymbol{U}}) \le \gamma_f c_f + \alpha_f l_f e \qquad (28)$$

Using $\alpha_f l_f e \leq (1 - \gamma_f) c_f$ in equation (28), we can write

$$V_f(\boldsymbol{\phi}(N; \hat{\boldsymbol{x}}^*, \tilde{\boldsymbol{U}})) \le c_f \tag{29}$$

which means that $\phi(N; \hat{x}^*, \tilde{U}) \in \mathcal{X}_f$. By this we showed that the control sequence \tilde{U} takes \hat{x}^* to set \mathcal{X}_f is N steps. Therefore the MPC problem is feasible for $\hat{x}^* = f(x, \hat{u}^*(x))$.

Lemma 2 (Robust invariance of \mathcal{R}_c): Suppose that $x \in \mathcal{R}_c$, assumptions (1) to (5) hold, $\|\hat{u}^*(x) - u^*(x)\| \le e$ for all $x \in \mathcal{R}_c$, $\alpha_f l_f e \le (1 - \gamma_f) c_f$, and $\alpha l_f e \le (1 - \gamma)c$. Then, the set \mathcal{R}_c is robust invariant under the learned policy, that is $\hat{x}^* \in \mathcal{R}_c$.

Proof: Using equation (23), one can write

$$V_N(\hat{\boldsymbol{x}}^*, \tilde{\boldsymbol{U}}) \le V_N(\boldsymbol{x}^*, \tilde{\boldsymbol{U}}) + \alpha \|\hat{\boldsymbol{x}}^* - \boldsymbol{x}^*\|$$
 (30)

Substituting equation (4) into second term in right hand side of equation (30) and using $\|\hat{u}^*(x) - u^*(x)\| \le e$ per assumption, one has

$$V_N(\hat{\boldsymbol{x}}^*, \tilde{\boldsymbol{U}}) \le V_N(\boldsymbol{x}^*, \tilde{\boldsymbol{U}}) + \alpha l_f e \tag{31}$$

From proof of theorem (1) [1, pp. 117], one can write $V_N(\boldsymbol{x}^*, \tilde{\boldsymbol{U}}) \leq V_N^*(\boldsymbol{x}) - c_1 \|\boldsymbol{x}\|^2$. Substituting this in equation (31) and using feasibility of MPC problem for $\hat{\boldsymbol{x}}^*$ one can write

$$V_N^*(\hat{x}^*) \le V_N(\hat{x}^*, \tilde{U}) \le V_N^*(x) - c_1 ||x||^2 + \alpha l_f e$$
 (32)

Using $V_N^*(\boldsymbol{x}) \leq c_2 \|\boldsymbol{x}\|^2$ and $\gamma = 1 - c_1/c_2$ in the above expression one has

$$V_N^*(\hat{\boldsymbol{x}}^*) \le \gamma V_N^*(\boldsymbol{x}) + \alpha l_f e \tag{33}$$

Considering that $x \in \mathcal{R}_c$, definition of \mathcal{R}_c , and $\alpha l_f e \leq (1 - \gamma)c$ in equation (33), results in

$$V_N^*(\hat{\boldsymbol{x}}^*) \le \gamma c + \alpha l_f e \le c \tag{34}$$

The equation (34) means that $\hat{x}^* \in \mathcal{R}_c$. In other words, \mathcal{R}_c is forward invariant under the learned policy when assumptions of the lemma (2) hold.

Remark 2: Lemmas (1) and (2) denote that recursive feasibility of MPC is preserved in \mathcal{R}_c under the learned policy.

Lemma 3 (Robust invariance of \mathcal{R}_b): Suppose that $x \in \mathcal{R}_b$, assumptions (1) to (5) hold, $\|\hat{u}^*(x) - u^*(x)\| \le e$ for all $x \in \mathcal{R}_c$, $\alpha_f l_f e \le (1 - \gamma_f) c_f$, and $\alpha l_f e \le (1 - \gamma)b$. Then, the set \mathcal{R}_b is robust invariant under the learned policy, that is $\hat{x}^* \in \mathcal{R}_b$.

Proof: The proof is similar to proof of lemma (2). Lemma 4 (Decent property of $V_N^*(.)$ in $\mathcal{R}_c \setminus \mathcal{R}_b$): Suppose that $\mathbf{x} \in \mathcal{R}_c \setminus \mathcal{R}_b$, assumptions (1) to (5) hold, $\|\hat{\mathbf{u}}^*(\mathbf{x}) - \mathbf{u}^*(\mathbf{x})\| \le e$ for all $\mathbf{x} \in \mathcal{R}_c$, $\alpha_f l_f e \le (1 - \gamma_f) c_f$, and $\alpha l_f e \le (\gamma^* - \gamma) b$ for some $\gamma^* \in (\gamma, 1)$. Then using the learned policy, $V_N^*(.)$ exponentially decreases along the

trajectory, that is, $V_N^*(\hat{x}^*) \leq \gamma^* V_N^*(x)$. *Proof*: Under the assumptions, lemmas (1) and (2) hold. Then by using $\alpha l_f e \leq (\gamma^* - \gamma) b$ in equation (33), one has

$$V_N^*(\hat{\boldsymbol{x}}) \le \gamma V_N^*(\boldsymbol{x}) + (\gamma^* - \gamma)b \tag{35}$$

Noting that $V_N^*(x) > b$ for $x \in \mathcal{R}_c \setminus \mathcal{R}_b$, equation (35) can be written as

$$V_N^*(\hat{x}) \le \gamma V_N^*(x) + (\gamma^* - \gamma) V_N^*(x) = \gamma^* V_N^*(x)$$
 (36)

This mean that the function $V_N^*(.)$ exponentially decreases for closed loop system with learned policy in the set $\mathcal{R}_c \setminus \mathcal{R}_b$.

Theorem 2 (Robust stability [29] of learned policy): Suppose that assumptions (1) to (5) hold, $\|\hat{u}^*(x) - u^*(x)\| \le e$ for all $x \in \mathcal{R}_c$, $\alpha_f l_f e \le (1 - \gamma_f) c_f$, and $\alpha l_f e \le (\gamma^* - \gamma) b$ for some $\gamma^* \in (\gamma, 1)$. Then, the closed loop system is locally robustly stable for all $x \in \mathcal{R}_c$.

Proof: Because of the assumption, lemmas (1) to (4) hold. Due to this, all trajectories starting in $\mathcal{R}_c \setminus \mathcal{R}_b$ ends up in \mathcal{R}_b and stay there. Thus, the closed loop system under the learned policy is locally robustly stable [29] in \mathcal{R}_c .

VI. RESULTS

The proposed algorithm is implemented on the Furuta pendulum benchmark system. QuanserTM experimental setup is used in the experiments. The Furuta pendulum, can be seen in figure (1), is introduced in [30] and is commonly used as a benchmark for control algorithms [31]. Furuta pendulum is chosen as a benchmark, because of its fast dynamic behavior that makes constraint satisfaction challenging. Further, unlike existing literature, MPC is used as the sole control for the swing-up and stabilization of the system.

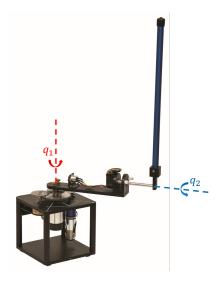


Fig. 1. Furuta pendulum.

The proposed method is compared with the normally trained DNN, e.g. proposed in [23], [24]. Additionally, exact MPC is used as the baseline of comparison in closed-loop control performance. The goal for the learning-based control policy is to perform as closely as possible to the exact MPC, quantitatively and qualitatively.

A. Constrained learning MPC

MPC parameters are chosen as Q = diag (8,3,.2,.1), R = 0.1, and N = 10, based on fine tuning on the experimental setup. Q_f is also chosen as the solution of infinite horizon discrete-time Riccati equation for the given Q, R and linearized dynamics of the system around origin. The MPC constraints are $|v| \le 6$, $|q_1| \le 1$, $|\dot{q}_1| \le 6$, and $|\dot{q}_2| \le 15$.

A deep MLP, with $g(x) = \tanh(x)$ activation function in hidden units, is used. The network has l=8 layers with M=30 nodes in each hidden layer as in equation (6). By this configuration, the network has 5,760 learnable parameters.

To train both networks, 100,000-point and 10,000-point data sets are collected from trajectories with random initial states and lengths for training and testing, respectively. PyTorchTM is used for neural network training. For the normally trained network, the L-BFGS [32] method is used for 2,000 epochs after which error plateaus. To train the constrained network, Algorithm 1 is implemented. A warm-up training with 1,200 epochs was conducted using the L-BFGS method. Then, this is followed by 700 epochs of constraint training with combination of L-BFGS and gradient ascent methods. The training completes with 100 epochs of fine-tuning. In this way, total epochs of training are the same for both neural networks.

B. Statistical analysis of learned policies

Empirical analysis of optimality and constraint satisfaction of the trained policies is done on 100,000 data set. Table table (I) summarizes the comparison results between the

TABLE I
STATISTICS OF LEARNED POLICIES.

Parameter		Normal training	Constraint training
Optimality error	Mean	0.0%	0.0%
	STD	3.3%	3.6%
State Constraint violation		3.14%	0.86%
Control Constraint violation		6.53%	0.07%
Total Constraint violation		9.67%	0.93%

normal and the proposed training approaches. It is clear that the proposed training significantly outperforms the normal training process in various criteria as shown in the table.

Empirical confidence levels are calculated using proposition (1). Choosing $\eta=0.3$, representing five percent of maximum voltage, and $\epsilon_h=0.01$, we have $\delta_h\simeq 0.99$.

Empirical averages $\bar{I}_o(\boldsymbol{x}) = 98.8\%$ and $\bar{I}_c(\boldsymbol{x}) = 90.3\%$ are achieved for the normally trained network. Using these in proposition (1), we obtain following probabilities with 99% confidence level

Normal training :
$$\begin{cases} \Pr\left(I_o(\boldsymbol{x}) = 1\right) > 0.978 \\ \Pr\left(I_c(\boldsymbol{x}) = 1\right) > 0.893 \end{cases}$$

Similarly, $\bar{I}_o(x) = 98.6\%$ and $\bar{I}_c(x) = 99.1\%$ are achieved for proposed constraint trained network. Using these values in Proposition proposition (1), we obtain the following statements with 99% confidence levels

Constraint training :
$$\begin{cases} \Pr\left(I_o(\boldsymbol{x}) = 1\right) > 0.9786 \\ \Pr\left(I_c(\boldsymbol{x}) = 1\right) > 0.981 \end{cases}$$

It is clear that using the constraint learning algorithm increases constraint satisfaction guarantees while maintaining the suboptimality achieved by the normal training.

Deep learning-based MPC has already been proved to be dramatically more computationally efficient than exact MPC even for simple problems [18], [20], [22]. The final structure of the constraint trained network is the same as the normally trained network and thus, they have the same online computational cost. We calculated the average computational time using Matlab™ functions on a personal Core i7™ laptop with 16 GB RAM. The exact MPC takes on average 5 ms, while the deep learning-based MPC takes on average 0.1 ms. This estimation indicates the deep learning-based controller approximately achieves 50 times faster than the normal training MPC. This result is compatible with that presented in [18].

C. Experimental closed loop results

In this section, control performance of the proposed method is compared with normally trained deep MPC and exact MPC on QuanserTM experimental setup. In the test, the pendulum's initial condition is considered as $x_0 = [-1, \pi, 0, 0]$. The results can be seen in figures (2) to (4). As it can be seen in the figures, the proposed constraint training approach has less constraint violations than the normally trained MPC. Furthermore, the amplitude of constraint violation is lesser with the proposed approach, see figure (2).

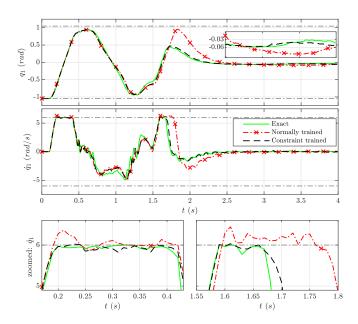


Fig. 2. Arm angle and angular velocity.

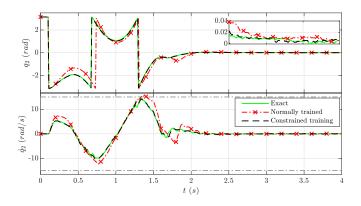


Fig. 3. Pendulum angle and angular velocity.

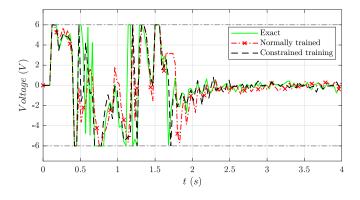


Fig. 4. Motor voltage.

TABLE II
EMPIRICAL STATISTICS OF CONSTRAINT VIOLATION.

Parameter		Normal training	Constraint training
No. of violations	Mean	31	4
	STD	6	2
Max amplitude of violation		8.1%	0.5%

Additionally, there is a difference between the performance of the normally trained network when the arm reaches its angular velocity constraints at 0.2 and 1.6 s. As it is evident in figures (2) and (3) the constraint trained network continues to perform closely to exact MPC, but the normally trained network has a shift in its response after meeting and violating the constraint. To further compare the proposed method, the experiment was repeated 30 times and the constraint violations statistics are summarized in table (II). Based on this, the proposed approach has five times less constraint violations. Further, the maximum amplitude of constraint violations is 40 times lesser than that of normally trained deep MPC.

VII. CONCLUSION

In this work, a constrained deep learning based MPC is proposed. Additionally, conditions for feasibility and robust stability of the learned policy are derived. As it was shown, the proposed constraint training improves constraint satisfaction while does not effectively change the optimality of the learned policy. In addition, superiority of the constraint trained network is shown, through the experiments, in replicating the same behavior as exact MPC. Future work can be done in extending these results to tracking MPC.

ACKNOWLEDGMENTS

The author would like to thank Dr. Jingang Yi for editing the language of the initial draft of the paper.

APPENDIX

The dynamics of the Furuta pendulum can be derived with the Lagrange method [33]. Combining the Lagrangian dynamics with actuator dynamics, the state-space model of the system can be summarized as

$$\dot{\boldsymbol{x}} = \boldsymbol{f}(\boldsymbol{x}) + \boldsymbol{g}(\boldsymbol{x})V \tag{37}$$

where $\boldsymbol{x} = [q_1, q_2, \dot{q}_1, \dot{q}_2]^\mathsf{T}$, V is voltage, and $\boldsymbol{f}(\boldsymbol{x})$ and $\boldsymbol{g}(\boldsymbol{x})$ are defined as

$$\begin{split} \boldsymbol{f}(\boldsymbol{x}) &= \begin{bmatrix} \dot{\boldsymbol{q}} \\ \boldsymbol{M}^{-1}(-\boldsymbol{C}\dot{\boldsymbol{q}}-\boldsymbol{G}) \end{bmatrix}, \quad \boldsymbol{g}(\boldsymbol{x}) = \begin{bmatrix} \mathbf{0}_{2\times 1} \\ \boldsymbol{M}^{-1}\boldsymbol{B} \end{bmatrix} \\ \boldsymbol{M}(\boldsymbol{q}) &= \begin{bmatrix} m_1l_1^2 + I_1 + m_2L_1^2 \\ +m_2l_2^2s^2(q_2) \end{bmatrix} & m_2L_1l_2c(q_2) \\ m_2L_1l_2c(q_2) & m_2l_2^2 + I_2 \end{bmatrix} \\ \boldsymbol{C}(\boldsymbol{q},\dot{\boldsymbol{q}}) &= \begin{bmatrix} 2m_2l_2^2s(q_2)c(q_2)\dot{q}_2 \\ +c_1 + \frac{\eta_m\eta_gK_g^2K_tK_V}{R_m} & -m_2L_1l_2s(q_2)\dot{q}_2 \\ -m_2l_2^2s(q_2)c(q_2)\dot{q}_1 & c_2 \end{bmatrix} \end{split}$$

TABLE III
SYSTEM PARAMETERS.

Parameter	Description
m_1	Mass of arm
m_2	Mass of pendulum
L_1	Length of arm
L_2	Length of pendulum
l_1	CoM of arm
l_2	CoM of pendulum
I_1	Arm moment of inertia
I_2	Pendulum moment of inertia
c_1	Arm viscous coefficient of friction
c_2	Pendulum viscous coefficient of friction
K_q	Gearbox ratio
K_t°	Motor torque constant
K_V	Motor back-emf constant
R_m	Motor winding electrical resistance
η_m	Motor efficiency
η_g	Gearbox efficiency

$$m{G}(m{q}) = egin{bmatrix} 0 \ -m_2 g l_2 s(q_2) \end{bmatrix}, \ m{B} = egin{bmatrix} rac{\eta_m \eta_g K_t K_g}{R_m} \ 0 \end{bmatrix},$$

where $q = [q_1, q_2]^\mathsf{T}$, $s(x) := \sin(x)$, $c(x) := \cos(x)$, and all parameters are given in table (III). Euler discretization is done on equation (37) with sampling time dt = 0.01 s to get discrete time model for MPC optimization.

REFERENCES

- J. B. Rawlings, D. Q. Mayne, and M. Diehl, *Model predictive control:* theory, computation, and design. Nob Hill Publishing Madison, WI, 2017, vol. 2. 1, 2, 3, 4
- [2] X. Yu-Geng, L. De-Wei, and L. Shu, "Model predictive control—status and challenges," *Acta Automat. Sinica*, vol. 39, no. 3, pp. 222–236, 2013. 1
- [3] A. Mirakhorli and B. Dong, "Market and behavior driven predictive energy management for residential buildings," *Sustainable Cities and Society*, vol. 38, pp. 723–735, 2018.
- [4] A. Zanelli, A. Domahidi, J. Jerez, and M. Morari, "Forces nlp: an efficient implementation of interior-point methods for multistage nonlinear nonconvex programs," *Int. J. Control*, vol. 93, no. 1, pp. 13–29, 2020. 1
- [5] J. Mattingley and S. Boyd, "Cvxgen: A code generator for embedded convex optimization," *Optim. Eng.*, vol. 13, no. 1, pp. 1–27, 2012.
- [6] P. Giselsson, M. D. Doan, T. Keviczky, B. De Schutter, and A. Rantzer, "Accelerated gradient methods and dual decomposition in distributed model predictive control," *Automatica*, vol. 49, no. 3, pp. 829–833, 2013.
- [7] S. Lucia, D. Navarro, Ó. Lucía, P. Zometa, and R. Findeisen, "Optimized fpga implementation of model predictive control for embedded systems using high-level synthesis tool," *IEEE Trans. Ind. Informat.*, vol. 14, no. 1, pp. 137–145, 2017.
- [8] J. R. Sabo and A. A. Adegbege, "A primal-dual architecture for embedded implementation of linear model predictive control," in *Proc. IEEE Conf. Decision Control*, 2018, pp. 1827–1832.
- [9] A. Richards, "Fast model predictive control with soft constraints," in Proc. Europ. Control Conf., 2013, pp. 1–6. 1
- [10] J. Drgona, K. Kis, A. Tuor, D. Vrabie, and M. Klauco, "Differentiable predictive control: An mpc alternative for unknown nonlinear systems using constrained deep learning," arXiv preprint arXiv:2011.03699, 2020.
- [11] F. Borrelli, A. Bemporad, and M. Morari, Predictive control for linear and hybrid systems. Cambridge University Press, 2017. 1
- [12] A. Gersnoviez, M. Brox, and I. Baturone, "High-speed and low-cost implementation of explicit model predictive controllers," *IEEE Trans. Contr. Syst. Technol.*, vol. 27, no. 2, pp. 647–662, 2017.
- [13] K. König and M. Mönnigmann, "Accelerating mpc by online detection of state space sets with common optimal feedback laws," arXiv preprint arXiv:2009.08764, 2020.

- [14] L. Hewing, J. Kabzan, and M. N. Zeilinger, "Cautious model predictive control using gaussian process regression," *IEEE Trans. Contr. Syst. Technol.*, vol. 28, no. 6, 2019. 1
- [15] T. Parisini and R. Zoppoli, "A receding-horizon regulator for nonlinear systems and a neural approximation," *Automatica*, vol. 31, no. 10, pp. 1443 – 1451, 1995.
- [16] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. MIT Press, 2016, http://www.deeplearningbook.org. 1, 2
- [17] S. Chen, K. Saulnier, N. Atanasov, D. D. Lee, V. Kumar, G. J. Pappas, and M. Morari, "Approximating explicit model predictive control using constrained neural networks," in *Proc. Amer. Control Conf.*, 2018, pp. 1520–1527.
- [18] X. Zhang, M. Bujarbaruah, and F. Borrelli, "Near-optimal rapid mpc using neural networks: A primal-dual policy learning framework," *IEEE Trans. Contr. Syst. Technol.*, 2020. 1, 6
- [19] B. Karg and S. Lucia, "Efficient representation and approximation of model predictive control laws via deep learning," *IEEE Trans. Cybernetics*, vol. 50, no. 9, pp. 3866–3878, 2020. 1
- [20] E. T. Maddalena, C. d. S. Moraes, G. Waltrich, and C. N. Jones, "A neural network architecture to learn explicit mpc controllers from data," arXiv preprint arXiv:1911.10789, 2019. 1, 6
- [21] J. Ferlez and Y. Shoukry, "Aren: assured relu nn architecture for model predictive control of lti systems," in *Proc. Int. Conf. Hybrid Systems:* Computation and Control, 2020, pp. 1–11.
- [22] M. Hertneck, J. Köhler, S. Trimpe, and F. Allgöwer, "Learning an approximate model predictive controller with guarantees," *IEEE Control Syst. Lett.*, vol. 2, no. 3, pp. 543–548, 2018. 1, 6
- [23] S. Lucia and B. Karg, "A deep learning-based approach to robust non-linear model predictive control," *IFAC-PaperOnLine*, vol. 51, no. 20, pp. 511–516, 2018. 1, 5
- [24] J. Nubert, J. Köhler, V. Berenz, F. Allgöwer, and S. Trimpe, "Safe and fast tracking on a robot manipulator: Robust mpc and neural network

- control," *IEEE Robot. Automat. Lett.*, vol. 5, no. 2, pp. 3050–3057, 2020. 1, 5
- [25] W. Hoeffding, "Probability inequalities for sums of bounded random variables," in *The Collected Works of Wassily Hoeffding*. Springer, 1994, pp. 409–426. 1, 3
- [26] P. Márquez-Neila, M. Salzmann, and P. Fua, "Imposing hard constraints on deep networks: Promises and limitations," arXiv preprint arXiv:1706.02025, 2017.
- [27] Y. Nandwani, A. Pathak, P. Singla et al., "A primal dual formulation for deep learning with constraints," in Advances in Neural Information Processing Systems. NeurIPS, 2019, pp. 12157–12168. 1, 3
- [28] S. Grammatico, X. Zhang, K. Margellos, P. Goulart, and J. Lygeros, "A scenario approach for non-convex control design," *IEEE Trans. Automatic Control*, vol. 61, no. 2, pp. 334–345, 2015.
- [29] C. M. Kellett and A. R. Teel, "Discrete-time asymptotic controllability implies smooth control-lyapunov function," *Systems & control letters*, vol. 52, no. 5, pp. 349–359, 2004. 5
- [30] K. Furuta, M. Yamakita, and S. Kobayashi, "Swing-up control of inverted pendulum using pseudo-state feedback," *Proc. Inst. Mech. Eng.*, *Part I: J. Syst. Control Eng.*, vol. 206, no. 4, pp. 263–269, 1992.
- [31] M. F. Hamza, H. J. Yap, I. A. Choudhury, A. I. Isa, A. Y. Zimit, and T. Kumbasar, "Current development on using rotary inverted pendulum as a benchmark for testing linear and nonlinear control algorithms," *Mech. Syst. Sig. Proc.*, vol. 116, pp. 347–369, 2019. 5
- [32] D. C. Liu and J. Nocedal, "On the limited memory bfgs method for large scale optimization," *Math. Prog.*, vol. 45, no. 1-3, pp. 503–528, 1989. 5
- [33] R. Olfati-Saber, "Nonlinear control of underactuated mechanical systems with application to robotics and aerospace vehicles," Ph.D. dissertation, Massachusetts Institute of Technology, 2001.