Capabilities and limitations of learning of MPC through recurrent neural networks

Guilherme Augusto Silva de Souza[a*], Vitor Caitano Salles[a], Flávio Mendonça Alcântara[a], Song Won Park[a]

[a]*Polytechnic School of the University of São Paulo, São Paulo – SP, Brasil*
**guilherme_souza@usp.br*

ABSTRACT

Model predictive controllers (MPCs) application in embedded systems is limited by solving optimal control problems online, in time, with limited hardware. This work proposes use of recurrent neural networks (RNN), namely gated recurrent unit (GRU) and long short-term memory (LSTM) networks, replacing the predictive controller. The dataset used in training was built upon multiple closed-loop trajectories of an unstable reactor tracking six different setpoints by way of a nonlinear MPC from various starting inputs and states in the setpoints' neighborhood. The resulting networks had their gains compared, as well as their closed-loop performances were compared to and the nonlinear MPC used in generating the dataset.

Palavras-chave: Nonlinear model predictive control, long-short term memory, process control, machine learning.

1. Motivation

Model predictive control is an online advanced control technique which consists of building predictions for a system under its control in order to reach different objectives such as reference tracking, control action smoothing or an economic objective while accounting for restrictions on manipulated or control variables. The detail level of said mathematical model can lead to prohibitive convergence times for the optimization problem, even with optimization strategies such as warm-start and multiple shooting. Orthogonal collocation over finite elements can be used in place of numeric integration methods (where most of the computational effort is exerted), but it may lead to imperfect models that cause closed-loop offset. Adding to these complexities, formulations with granted asymptotic stability and recursive feasibility, which seek to approximate a infinite horizon optimal control problem, may add terminal inequality constraints (Souza and Odloak, 2024) or terminal contractive constraints (Sencio and Souza and Odloak, 2021) which also increase computational demand. Explicit MPC can be used in these cases, carrying with it the computational effort of solving optimization problems over the entire state and input spaces for creating a look-up table for online-use, with its proximity to optimal control law restricted to how fine the search space is partitioned.

Artificial neural networks (ANN) were at first used to learn unknown dynamics from process data, as proposed by Chen and Xiu (2019). Then, ANNs have been applied to replace numeric integration methods in MPC, as detailed by Wu et al. (2019), accelerating optimization convergence by replacing a numeric integrator with an ANN. Approximate explicit MPC solutions have also been proposed by what is refered to in the literature as imitation learning of MPC or approximate MPC. In these works, a neural network is trained to mimetize the control actions a predictive controller would compute over a subset of the search space, skipping entirely the online solution of an optimization problem at every sampling interval. The work of Adhau, Naik and Skogestad (2021) successfully implements constraints during the training step of a feedforward neural network composed of two hidden layers of 160 and 1560 neurons, respectively, using rectified linear units (ReLU) and linear activation functions. The trained network is capable of accounting for constraints of a two-state CSTR reactor. Chen et al. (2022) propose large scale MPC with neural networks with guarantees on recursive feasibility and guaranteed stability, using a network architecture of four layers composed of two up to 32 ReLU units for a two-state double integrator system. The work of Alsmeier et al. (2024) uses a feedforward neural network for a two-state inverted pendulum control. Its architecture consists of two 10-neuron layers with *tanh* activation functions, aiming for approximation error boundedness via incorporating sensitivities of the optimzation problem in the ANN training.

However, at the time of writing, a single use of recurrent neural networks (RNN) for MPC approximations was found. Adhau, Gros and Skogestad (2024) have proposed use of a 128-neuron Leaky ReLU units to learn the control policy of an NMPC operating on a cascaded tanks system, while reinforcement learning was used to reduce closed-loop errors due to plant-model mismatch the RNN may experience. Recurrent networks are more capable of applications that require memory or handling data series as features (Goodfellow and Bengio and Courville, 2016), such as fault detection in chemical processes operation . Taira et al. (2022) propose a Bayesian recurrent neural network for fault detection in a fluid catalytic cracking process, incorporating Bayesian theory in RNN composed of long short-term memory units, first ideated by Hochreiter and Schmidhuber (1997). These units were developed due to difficulties in storing information over long periods of time in a network. Another proposed architecture consists of gated recurrent units (GRU), proposed by Cho et al. (2014) which aims to avoid performance degradation experienced by encoder-decoder networks when the data series becomes too extense.

Deep neural networks (DNN), as the ones mentioned earlier, can handle these by either input tapping or accepting data sequences (*sequenceInputLayer*, in MATLAB), although this strategy does not grant memory to the network, remaining a feedforward network.

The main goal of this work is to implement recurrent neural networks for MPC approximation, or imitation learning of MPC. The networks aim to replace a MPC in closed-loop. This work is organized as follows: Section 2 contains the methodology related to data series generation, network training and testing, as well as benchmarks used to compare networks implemented and their closed-loop performance. Section 3 consists of closed-loop results of the proposed networks. Section 4 contains some concluding remarks as well as suggestions of future works pertaining to MPC imitation.

2. Methodology

2.1. *CSTR plant, NMPC and data series generation*

The jacketed CSTR reactor to be controlled consists of three ODE equations that describe dimensionless reagent concentration, dimensionless reactor temperature and dimensionless jacket temperature, developed by Russo and Bequette (1996).

$$\dot{C}_A = F(\gamma_7 - C_A) - \gamma_6 C_a \exp\left(\frac{T_R}{1+T_R/\gamma_2}\right) \quad (1)$$

$$\dot{T}_R = F(\gamma_8 - T_R) - \gamma_3(T_R - T_C) + \gamma_1\gamma_6 C_a \exp\left(\frac{T_R}{1+T_R/\gamma_2}\right) \quad (2)$$

$$\dot{T}_C = \gamma_4\left(F_C(\gamma_9 - T_C) + \gamma_3\gamma_5(T_R - T_C)\right) \quad (3)$$

With γ = [8.0, 20.0, 0.3, 10.0, 1.0, 0.072, 1.0, 0.0, -1.0]. The NMPC that at first controls this reactor is represented by the following optimization problem:

$$\min_{\Delta u(k|k),\dots,\Delta u(k+N-1|k)} \sum_{j=0}^{N-1} \left\| x(k+j|k) - x_{sp} \right\|_Q^2 + \sum_{j=0}^{N-1} \|\Delta u(k+j|k)\|_R^2 \quad (4)$$

$$\text{s.t. } x(k+j+1|k) = f\left(x(k+j|k), \Delta u(k+j|k)\right),\ j = 0, \dots, N-1, \quad (5)$$

$$u(k+j|k) = u(k+j-1|k) + \Delta u(k+j|k),\ j = 0, \dots, N-1 \quad (6)$$

$$x(k+j|k) \in X, u(k+j|k) \in U, \Delta u(k+j|k) \in U_\Delta,\ j = 0, \dots, N-1 \quad (7)$$

Accounting for state, input and input move constraints, the controller (4)-(7) computes penalized control actions and tracks the reference value x_{sp}. The penalty matrices consist of identity matrices of appropriate dimensions for states and inputs. Horizon length N of 40 was used, in order to approximate an optimal controller. Mathematical model f in this case is a numeric integrator CVODES computing future states with sampling time of 3 seconds. The optimization problem was written in CasADi, and solved via an interior-point algorithm IPOpt in a multiple-shooting manner.

Before generating the state and input trajectories used in training, six unstable equilibria were selected as reference points for tracking simulations. Then, the predicted states and computed inputs were built starting from a fine partitioning (nine points) around the reference points (respectively ±5%, ±5%, and ±7%) and their respective inputs (±5%, and ±7%). The controller was made to control the reactor starting from each of these combinations of states and inputs to all the six references, leading to a data series composed of 40 samples of states, inputs, and reference per trajectory.

The selected network features were then states, inputs and errors (w.r.t. reference), and the chosen network outputs were input moves. Hence, these vectors were submitted to a min-max normalization, with an additional symmetry step for the input moves in order to rule out any biases the network may develop in its

output layer. The largest modulo value was found for each input move, and used as a scaling factor for the minimal value (after multiplying by -1) and maximal value.

2.2. *Network training, validation and testing*

The resulting dataset consists of 354294 state and input trajectories, which in training requires a larger batch size of 512 features. Initial learning rate of 0.001, with learning rate drop factor of 0.2, learning rate drop period of 5 epochs, gradient threshold of 1, norm-2 regularization of 0.0001, with data shuffling at every epoch, validations executed every 1000 iterations and training stopping criteria of 3 validation steps. The training-validation-test division selected was 80%-10%-10%. Training and validation was done via Adam algorithm developed by Kingma and Ba (2015), in MATLAB R2022b developed by The Mathworks Inc. (2022), and testing was done via *predict* and *predictAndUpdateState*, when applicable.

Six different neural networks were proposed. Their architectures are described by the number of network features, the number of neurons in the hidden layers and finalized by the number of network outputs 8-8-8-2 multi-layer perceptron (MLP), 8-16-2 MLP, 8-16-2 long-short term memory (LSTM) and 8-16-2 gated recurrent unit (GRU). The latter was used in a transfer learning experiment where an input restriction was added after training via hyperparameter tuning.

2.3. *Network comparison and closed-loop performance*

MLP network static gains are computed via 2-norm of the product of every fully connected layer weight matrices W_i. This allows identifying which networks are more sensible to noise.

$$G_{MLP} = \left\| \prod_{i=N_{layer}}^{1} W_i \right\| \tag{8}$$

Whereas recurrent networks have their recurrent weight matrices 2-norm taken as well as their weight matrices 2-norm taken. The two norms participate in the overall network dynamic gain as long as the recurrence norm is less than 1:

$$G_{RNN} = \frac{\|W_i\|}{1-\|W_{R,i}\|} \tag{9}$$

Closed-loop performance is compared via three metrics: integral of absolute error (IAE), integral of time-weighted absolute error (ITAE) and control effort (CEff) exerted by the network:

$$IAE = \sum_{j=0}^{N_{sim}} \left| x(k+j|k+j) - x_{sp} \right| \tag{10}$$

$$ITAE = \sum_{j=0}^{N_{sim}} t(k+j) \left| x(k+j|k+j) - x_{sp} \right| \tag{11}$$

$$CEff = \sum_{i=1}^{n_u} \sum_{j=0}^{N_{sim}} |\Delta u_i(k+j|k+j)| \tag{12}$$

Finally, the closed-loop simulation consists of three reference points to be tracked that change every 180 seconds, with the two first references present in the training data set. Evaluation of the networks' generalization capability was tested by way of the third reference to be tracked, which does not belong to the training dataset.

3. Results

Network gains of each proposed architecture in the methodology section was computed, and are organized in Table 1.

Table 1. Network gains for one and two-layer MLP, and recurrent weight and weight norms for GRU and LSTM networks.

Network	Architecture	Gain
1-layer MLP	8-16-2	1.0563
2-layer MLP	8-8-8-2	1.2221
1-layer GRU	8-16-2	1.0046, 1.3181
1-layer LSTM	8-16-2	1.4229, 1.4558

The two-layer MLP shows higher gain, showing higher sensibility to noise and errors in its inputs. Both recurrent networks show recurrent weight norms above unity. This is undesired, because it indicates that recurrence may lead to instability when used in closed-loop. Hence, the global gain of these networks cannot be computed by way of equation (9). This norm can be corrected by either using larger values of L2Regularization in Adam training (which may worsen network performance), incorporating a Lipschitz constraint in training, or limiting the singular values until the dynamic gain reaches a desired value. Limiting the singular values of the recurrent weight matrix by 0.95 leads to a dynamic gain of 39.3908 for the GRU network, while the same

manouver, but limiting singular values by 0.93, lead to a dynamic gain 38.4271 for the LSTM network. These limits for the singular value were chosen so to align the dynamic gain to the number of samples the networks are trained upon, that is, the NMPC horizon length.

Next, we have the closed-loop results of NMPC and trained networks.

Figure 1. Closed-loop states for the unstable jacketed CSTR.

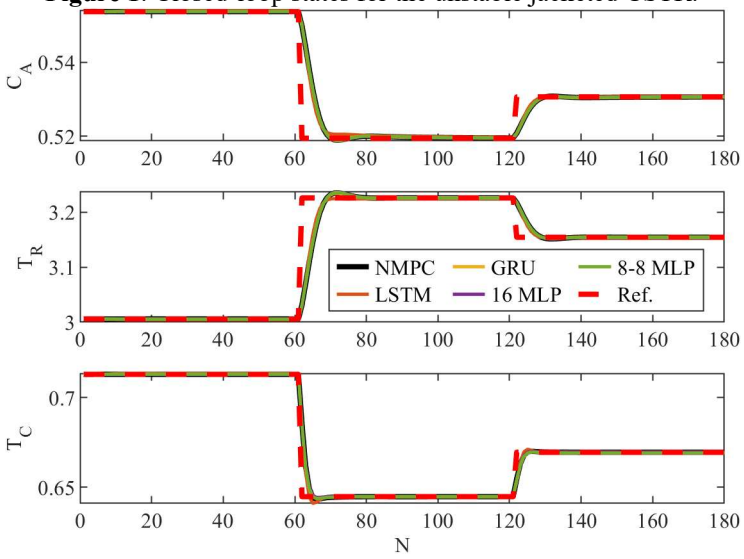
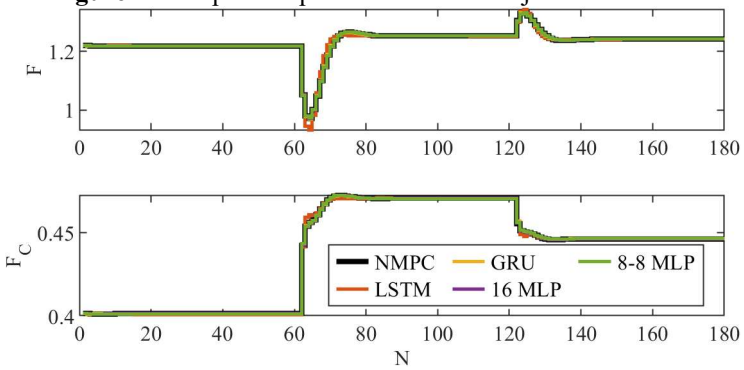


Figure 2. Computed inputs for the unstable jacketed CSTR.



All proposed networks indicate that they have gained generalization capabilities, since all of them have tracked the final reference (which was not in the training, validation or test dataset). Both Figure 1 and 2 show a slight difference in behavior for the LSTM network when compared to other control solutions used. This is due to the singular value manipulation that happened, which affected this network's closed loop performance. In Table 2, the effect of said manipulation becomes clearer as the performance of each control solution is shown.

Table 2. Closed-loop performance indexes for GRU and LSTM networks.

Solution	Architecture	IAE	ITAE	CEff	Mean time [s]
1-layer MLP	8-16-2	1.2034	0.2377	0.8354	0.0013
2-layer MLP	8-8-8-2	1.2230	0.2427	0.8288	0.0013
1-layer GRU	8-16-2	1.2077	0.2422	0.8386	0.0014
1-layer LSTM	8-16-2	1.2897	0.2272	0.9160	0.0014
NMPC	N_p= N_p=40	1.1640	0.2272	0.8313	0.1755

With the metrics in Table 2, the performance difference between the proposed controllers becomes more evident. The computer time gained from replacing an optimization solution by a network was expected. Between the two MLP networks, the two-layer network seems to lead to slower closed-loop tracking: it exerts less control effort while attaining higher absolute error overall. For the recurrent networks, the GRU solution was not as affected by the singular value manipulation nearly as much as the LSTM network, which exerts higher control

effort out of all the solutions and higher integral of absolute error. In the end, none of the networks were able to match the predictive controller in non-time-weighted absolute error.

4. Concluding remarks and future works

Fine tuning and transfer learning seem to be the current topic studied in approximate MPC. Elegant strategies such as Lipschitz-based regularization of the recurrent weight matrices for the recurrent neural networks would be an opportunity for future works in this line of work.

In the construction of this work, it became evident that more than 16 neurons for these approximations did not bring returns in any of these metrics, while hidden layers below 8 neurons (in total) did lead to poor closed-loop performance. The networks tested for this work consisted of 4 up to 128 neurons in its hidden layer. For brevity, the networks shown in this work were limited to 16 neurons (total). Network pruning does not seem to be a concern, as per the work of Adhau, Naik and Skogestad, where the main focus was to satisfy constraints and preserve closed-loop performance.

Normalization strategies, careful selection of features and activation functions were groundbreaking: dropping use of ReLU as well as using state error instead of the reference value as a feature, associated with using input moves instead of input values removed an insistent offset. Deep networks (over three layers) along with dropout layers use unsatisfying performance.

Agradecimentos: This study was financed in part by the Coordenação de Aperfeiçoamento de Pessoal de Nível Superior – Brasil (CAPES) – Finance Code 001. Process 88887.988471/2024-00.

Referências

S. Adhau and V. V. Naik and S. Skogestad: Constrained Neural Networks for Approximate Nonlinear Model Predictive Control, in IEEE 60th Conference on Decision and Control, 2021.
S. Adhau and S. Gros and S. Skogestad, Reinforcement learning based MPC with neural dynamic models, European Journal of Control (80), 2024.
H. Alsmeier and L. Theiner and A. Savchenko and A. Mesbah and R. Findeisen: Imitation Learning of MPC with Neural Networks: Error Guarantees and Sparsification, in IEEE 63rd Conference on Decision and Control, 2024.
S. W. Chen and T. Wang and N. Atanasov and V. Kumar and M. Morari: Large scale model predictive control with neural networks and primal active sets, Automatica (135), 2022.
Z. Chen and D. Xiu: On generalized residual network for deep learning of unknown dynamical systems, Journal of Computational Physics (438), 2021.
K. Cho and B. Merriënboer and D. Bahdanau and Y. Bengio: On the Properties of Neural Machine Translation> Encoder-Decoder Approaches. Proceedings of SSST-8, Eighth Workshop on Syntax, Semantics and Structure in Statistical Translation, p. 103-111, 2014.
I. Goodfellow and Y. Bengio and A. Courville: Deep Learning. MIT Press. 2016.
S. Hochreiter and J. Schimdhuber: Long short-term memory. Neural Computation 9(8), p. 1735-1780, 1997.
D. P. Kingma, J. L. Ba: Adam: A Method for Stochastic Optimization. Proceedings of International Conference on Learning Representations 2015. 2015.
The MathWorks Inc. (2022). MATLAB version: 9.13.0 (R2022b), Natick, Massachusetts: The MathWorks Inc. https://www.mathworks.com
L. P. Russo and B. W. Bequette: Effect of process design on the open-loop behavior of a jacketed exothermic CSTR. Computers & Chemical Engineering 20(4), p.417-426. 1996.
R. R. Sencio and G. A. S. Souza and D. Odloak: A terminal state contractive nonlinear MPC with output zones and input targets. IFAC PapersOnLine 53(2), p. 6025-6030, 2020.
G. A. S. Souza and D. Odloak: Novel terminal region computation method for quasi-infinite horizon NMPC. Computers & Chemical Engineering (189), 2024.
G. R. Taira and S. W. Park and A. C. Zanin and C. R. Porfirio: Fault Detection in a Fluid Catalytic Cracking Process using Bayesian Recurrent Neural Network, IFAC-PapersOnLine 55(7), p-715-720, 2022.
Z. Wu and A. Tran and D. Rincon and P. D. Christofides: Machine learning-based predictive control of nonlinear processes. Part I: Theory, AIChE Journal (65), 2019.