Computers and Chemical Engineering 136 (2020) 106801

Contents lists available at ScienceDirect

Computers and Chemical Engineering

journal homepage: www.elsevier.com/locate/compchemeng

Integrating deep learning models and multiparametric programming

Justin Katz[a,b], Iosif Pappas[a,b], Styliani Avraamidou[b], Efstratios N. Pistikopoulos[a,b,*]

[a] Artie McFerrin Department of Chemical Engineering, Texas A&M University, College Station, TX 77843, USA
[b] Texas A&M Energy Institute, Texas A&M University, College Station, TX 77843, USA

ARTICLE INFO

Article history:
Received 30 October 2019
Revised 29 January 2020
Accepted 27 February 2020
Available online 28 February 2020

Keywords:
Deep learning
Neural networks
Rectified linear units
Mixed-integer linear programming
Multiparametric programming

ABSTRACT

Deep learning models are a class of approximate models that are proven to have strong predictive capabilities for representing complex phenomena. The introduction of deep learning models into an optimization formulation provides a means to reduce the problem complexity and maintain model accuracy. Recently it has been shown that deep learning models in the form of neural networks with rectified linear units can be exactly recast as a mixed-integer linear programming formulation. However, developing the optimal solution of problems involving mixed-integer decisions in online applications remains challenging. Multiparametric programming alleviates the online computational burden of solving an optimization problem involving bounded uncertain parameters. In this work, a strategy is presented to integrate deep learning and multiparametric programming. This integration yields a unified methodology for developing accurate surrogate models based on deep learning and their offline, explicit optimal solution. The proposed strategy is demonstrated on the optimal operation of a chemostat.

1. Introduction

Deep learning is a method to approximate complex systems and tasks by exploiting copious amounts of data to develop rigorous mathematical models. These approximate models are based on neural networks which are popular in the chemical engineering literature (Himmelblau, 2000; Shang and You, 2019). As data-driven modeling techniques are seeing increased value and are being adopted by industry (Tran et al., 2018), the incorporation of deep learning into optimization formulations is paramount.

Surrogate modeling for optimization, both as a tool (Beykal et al., 2018; Kim and Boukouvala, 2019) and as a means for approximate model development (Tso et al., 2019) is an established field. In the chemical engineering literature, the use of neural networks as surrogate models has found success in a variety of contexts (Chiang et al., 2017), such as (i) modeling (Shokry et al., 2018; Hough et al., 2017), (ii) optimization and control (Schweidtmann et al., 2019; Himmelblau, 2008; Wu et al., 2019b; 2019c), (iii) regression (Himmelblau, 2000), and (iv) classification (Himmelblau, 2000). In all of these applications, the developed artificial neural network (ANN) model is used to represent a complex nonlinear process. However obtaining the global solution for the corresponding optimization problem incorporating a neural network poses a significant computational burden due to the inherent nonconvexity introduced (Rister and Rubin, 2017). Hence, incorporating neural networks in real-time applications remains a major challenge.

The ability to use neural network models as surrogates in an optimization formulation is contingent on utilizing a strategy to obtain a quality optimal solution. Current literature demonstrates an interest in obtaining 'good' solutions to optimization formulations incorporating deep learning models. Schweidtmann et al. (2019) proposed a global optimization strategy based on McCormick relaxations to identify the global minimum to optimization problems involving ANNs. Pfrommer et al. (2018) utilized a stochastic genetic algorithm to find the minimum for a process involving textile draping where a neural network was utilized as a surrogate model. Nagata and Chu (2003) developed a surrogate neural network model for a fermentation process, and optimal operating conditions were identified using a genetic algorithm.

Deep learning has a distinct advantage compared to other surrogate modeling techniques, such as response surfaces, Kriging models, Bayesian networks, and radial basis functions (Asher et al., 2015). Because of the highly connected structure of deep learning models, they are naturally adept at expressing complex functional relationships. Their ability to approximate a function to an arbitrary level of accuracy is because there is an exponential number of piecewise connected hyperplanes based on the size of the neural network (Chen et al., 2018; Montufar et al., 2014).

Given an optimization problem with highly complex and nonlinear components, neural networks with rectified linear activation

* Corresponding author at: Artie McFerrin Department of Chemical Engineering, Texas A&M University, College Station, TX 77843, USA.
E-mail address: stratos@tamu.edu (E.N. Pistikopoulos).

https://doi.org/10.1016/j.compchemeng.2020.106801
0098-1354/

units (ReLU), that have been shown to have high performance for regression based problems (Eckle and Schmidt-Hieber, 2019), are of interest to be incorporated into optimization formulations as surrogate models. In recent work, it was shown that neural networks utilizing ReLU activation functions can be represented exactly in a mixed-integer linear programming (MILP) formulation (Fischetti and Jo, 2018; Grimstad and Andersson, 2019a). An exact recasting of neural networks with ReLU activation functions as an MILP formulations allows a new avenue for incorporating deep learning models into optimization based formulations. With this capability, the gap between model accuracy and computational performance is reduced.

Previous work has demonstrated integrating neural networks with ReLU activation functions into optimization formulations. However, in online applications where time is a critical factor, determining the optimal solution is challenging because of the inherent nonconvexity of the resulting discrete optimization formulation. In these time critical applications, multiparametric programming is a proven methodology to alleviate the online computational burden by developing the optimal solution offline. By incorporating more advanced surrogate models in multiparametric optimization formulations, the benefits of the developed parametric solution are improved, namely (i) the ability to obtain the optimal solution without having to solve an optimization problem each time the uncertain parameter is identified, (ii) having the explicit map of solution a priori, and (iii) having the explicit functional relationship between optimization variables and uncertain parameters. In addition, the explicit map of solution provides benefits in numerous applications including, multiparametric model predictive control (Bemporad et al., 2002; Katz et al., 2018; Lee and Chang, 2018), scheduling (Kopanos and Pistikopoulos, 2014), bilevel programming (Avraamidou and Pistikopoulos, 2019), multilevel decision making (Burnak et al., 2018; 2019), portfolio selection (Steuer et al., 2006), and instance-weighted support vector machines (Karasuyama et al., 2012).

Complete algorithms regarding the development of the multiparametric solution is available for multiparametric linear (mpLP) (Jones and Morrari, 2006), quadratic (mpQP) (Gupta et al., 2011; Oberdieck et al., 2017; Ahmadi-Moshkenani et al., 2018), mixed-integer linear (mpMILP) (Wittmann-Hohlbein and Pistikopoulos, 2013; Charitopoulos et al., 2018), and mixed-integer quadratic programming (mpMIQP) problems (Oberdieck and Pistikopoulos, 2015). In these multiparametric model formulations, a key detail is their dependence on linear or piecewise linear constraints. Therefore, to incorporate more complex phenomena in parametric formulations surrogate modeling is required. Developing accurate surrogate models to represent nonlinear functional relationships is non-trivial, and deep learning models based on ReLU activation functions bridge this gap. Given that complete theory regarding mpMILPs and algorithmic strategies are available in online solvers (Oberdieck et al., 2016b; Herceg et al., 2013), the integration of neural networks involving ReLU activation functions and multiparametric programming is a natural step.

The rest of the work is organized as follows. First an overview of neural networks and ReLU activation functions is provided. Then a discussion on multiparametric programming is presented, followed by the framework for integrating neural networks and multiparametric programming. Next, a demonstration of the framework is showcased on a chemostat model. Finally conclusions and future directions are presented.

2. Key contribution

The focus of this work is to approximate nonlinear functions in optimization formulations with neural networks composed of ReLU activation functions. In particular, the key contribution is the integration of these deep learning models and multiparametric programming.

Given is a nonlinear optimization formulation in the form of Eq. (1).

$$\begin{aligned} \min_{x} \quad & c^T x \\ \text{s.t.} \quad & f(\theta, x) \leq 0 \\ & x \in X^q \\ & \theta \in \Theta^m \end{aligned} \tag{1}$$

where x is the vector of optimization variables, c is the linear cost coefficient, f is the vector of linear/nonlinear constraints, and θ is the vector of uncertain parameters that are known at the time of solving the nonlinear optimization problem. The sets X^q and Θ^m are closed polyhedral sets of the q optimization variables and the m uncertain parameters respectively.

Determining the optimal solution at every parameter realization (e.g. model predictive control, reactive scheduling) for this nonlinear optimization problem is computationally expensive. Multiparametric programming is a technique that transforms an optimization formulation involving bounded, uncertain parameters to an explicit functional relationship between the optimal optimization variables and these parameters. The optimal multiparametric solution is described by the different combinations of active constraints which can occur for every feasible uncertain parameter value, θ, realization. Based on the various active sets, the overall parameter space is partitioned indicating where in the parameter space each optimal multiparametric solution holds. To realize the benefits of multiparametric programming, a surrogate model is developed to replace the nonlinear function in Eq. (1). The model accuracy of the surrogate model is ensured by utilizing neural networks with ReLU activation functions. Neural networks are inherently nonconvex, nonlinear functions making them difficult to incorporate into optimization problems, however a recent reformulation technique enables their direct use in a mixed-integer linear programming based formulation with no information lost. Following the implementation of the neural network surrogate model, Eq. (1) is approximated by Eq. (2).

$$\begin{aligned} \min_{x,y} \quad & c^T \omega \\ \text{s.t.} \quad & [A\ E]\omega \leq b + F\theta \\ & \omega = [x^T y^T]^T \\ & x \in X^q,\ y \in \mathbb{Z}_2^n \\ & \theta \in \Theta^m \end{aligned} \tag{2}$$

where $A \in \mathbb{R}^{1\times q}$, $E \in \mathbb{R}^{1\times n}$, $F \in \mathbb{R}^{1\times m}$, $b \in \mathbb{R}^{(q+n)\times 1}$. Determining the optimal solution of a mixed-integer linear programming problem is still computationally demanding. Unlike nonlinear programming formulations with uncertain parameters, mixed-integer linear program with bounded uncertain parameters are solvable using state-of-the-art multiparametric algorithms. Therefore, the online computational burden of determining the optimal solution to an MILP at every parameter realization is eliminated. The developed explicit solution produces a map of solutions, Fig. 1, where each region is an affine function that relates the optimal optimization variables to the uncertain parameters.

A summary of the novel framework followed in the work is presented in Fig. 2. The key features include data collection and processing, neural network development, MILP reformulation, and multiparametric programming. The following sections provide necessary information regarding (i) neural networks, (ii)multiparametric programming, and (iii) their integration.

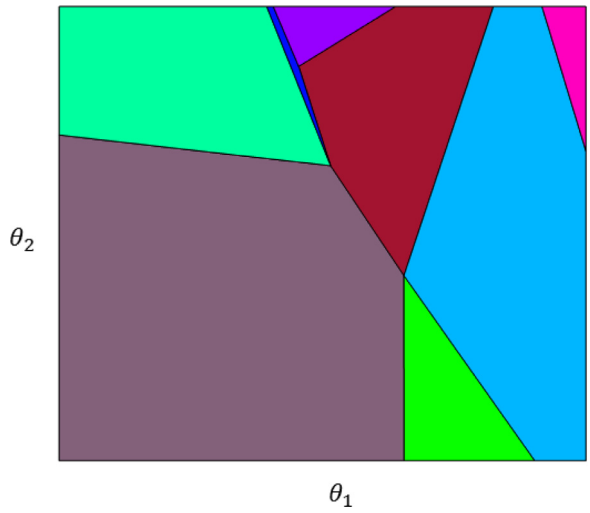

Fig. 1. The framework for integrating deep learning and multiparametric programming, adapted from Katz, 2020.

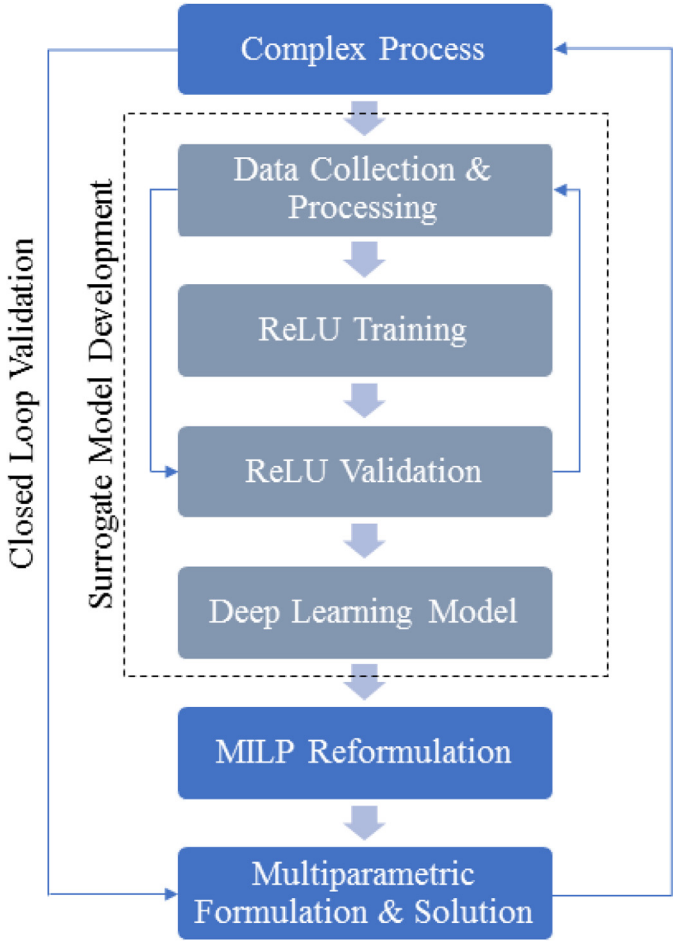

Fig. 2. The framework for integrating deep learning and multiparametric programming, adapted from Katz, 2020.

3. Neural networks

Neural networks are nonparametric mathematical models that are an effective tool to map input data to output data with a high degree of accuracy. The strength of neural networks is their ability to universally approximate any function (Pinkus, 1999). A neural network is defined by its structure, two of which are feedforward and recurrent (Ning and You, 2019). In this work, the structure assumed is feedforward because of its proven performance and straightforward structure, shown in Fig. 3. The feedforward neural network involves inputs, outputs, an input layer, output layer, hidden layers, weighting parameters, and activation functions. The inputs, outputs, input layer, and output layer are defined by the particular process. For example, Fig. 3 has 2 inputs, 1 output, 2 hidden layers, 4 nodes in the first hidden layer, and 2 nodes in the second hidden layer. However, the number of hidden layers and their respective sizes, weighting parameters, and activation functions are all tunable. The number of hidden layers and their respective sizes can be determined through trial and error, and the weighting parameters are typically defined through a local search, such a stochastic gradient descent. Numerous activation functions have shown to be meritous under varying circumstances. In this work, the ReLU activation function is used throughout the manuscript because it can be reformulated as a MILP.

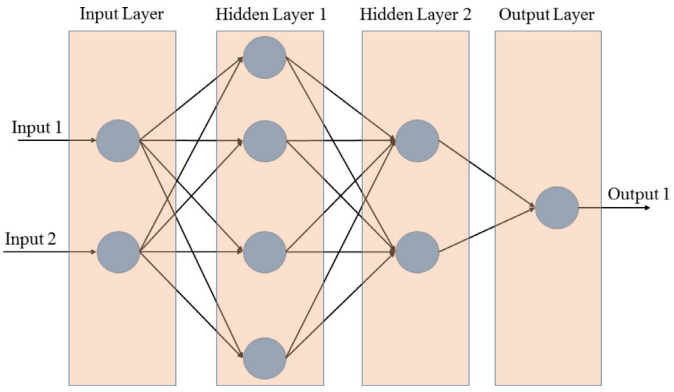

Fig. 3. The structure of a feedforward neural network with inputs, outputs, and hidden layers, adapted from Katz, 2020.

3.1. Training

Training a neural network involves identifying the values for the weights and biases, such that a performance metric is minimized. The training step is critical because it provides the basis for how the neural network will map inputs to outputs. Key factors that must be addressed during training include (i) identifying a performance metric to use, (ii) minimizing the performance metric, and (iii) data selection and processing.

3.1.1. Performance metric

The performance metric is a measure to provide the goodness of fit for the neural network. The metric is based on the deviation between the output of the neural network and the dataset. There are a myriad of performance criteria, such as mean squared error (MSE), normalized mean squared error (NMSE), root mean squared error (RMSE), mean absolute error (MAE), etc. Aftam et al. (2017). These metrics are different formulations for quantifying the deviation between the expected output of the neural network and the output of the dataset. Note that determining which performance criteria should be selected to train the neural network is nontrivial.

3.2. Validation

Model validation ensures the trained model is an accurate representation of the dataset. There are many techniques for model validation for data-driven models, such as (i) data partitioning, (ii) cross-validation (Poole and Mackworth, 2010), and (iii) analyzing

the residual plot. These techniques provide different approaches for ensuring the regressed model is valid over the dataset.

3.3. ReLU activation functions

Each node in a neural network's hidden layer is associated with an activation function. Common activation functions are the hyperbolic tangent, sigmoid, and the rectified linear unit. Selecting a suitable activation function is problem dependant, but the rectified linear unit has consistently displayed an ability to perform well on numerous applications, and is the activation function assumed in this work.

The ReLU activation function is defined in Eq. (3). It is a piecewise linear function alternatively represented by Eq. (4). The alternative representation gives rise to a formalism for introducing a neural network with ReLU activation functions into a mixed-integer optimization formulation.

$$y = max\{0, x\} \tag{3}$$

$$y = \delta x, \;\; \delta = \begin{cases} 1, & \text{if } 0 < x \\ 0, & \text{otherwise} \end{cases} \tag{4}$$

The ReLU activation functions approximates a function by deconstructing the original function into a set of piecewise hyperplanes. The number of piecewise hyperplanes that define the ReLU neural network scales exponentially with the number of hidden layers of the neural network, shown in Eq. (5) (Chen et al., 2018; Montufar et al., 2014).

$$\Omega\left(\left(\frac{n}{n_0}\right)^{L-1} n^{n_0}\right) \tag{5}$$

where L is the number of hidden layers, n_0 is the number of inputs, and n is the number of nodes. The exponential scaling of the number of piecewise hyperplanes is the reason ReLU activation functions in deep learning models are successful. This complexity is demonstrated with Eq. (5). In other words, a deep neural network with ReLU activation functions expresses an arbitrary function with an exponential number of piecewise affine hyperplanes. Fig. 4 shows the connected piecewise affine hyperplane structure for an arbitrary neural network, with two inputs, x and y, and one output, z. Leveraging the deep neural networks ability for accurate function approximation makes it an ideal candidate surrogate model for optimization studies.

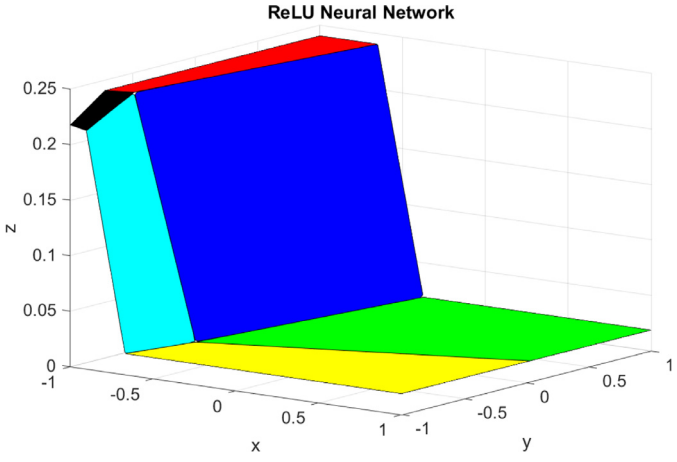

Fig. 4. A neural network with ReLU activation functions and the representative connected hyperplanes, adapted from Katz, 2020.

4. Multiparametric programming

Optimization formulations involving bounded uncertain parameters are defined as a multiparametric programming problem. The multiparametric solution provides a means to derive analytic explicit functional relationships between the optimization variables and these uncertain parameters. Developing the multiparametric solution has several benefits including (i) determining the optimal solution faster in real time as the uncertainty is revealed, (ii) deriving fundamental knowledge of the optimization formulation via the offline map of solution, and (iii) providing the explicit expressions relating optimization variables to uncertain parameters that can be embedded in multi-level optimization frameworks (Burnak et al., 2019). In this work, we focus on mpMILPs, which are used as the exact reformulation of deep neural networks maintaining ReLU activation functions. Because the solution strategy for mpMILPs is based on continuous mpLPs, a comprehensive discussion on the fundamentals and solution strategies for mpLPs is provided in the subsequent sections. Discussions regarding developing the full multiparametric solution for mpMILPs is then presented in Section 4.2.

4.1. Fundamental concepts

A generic mpLP has the following form

$$\begin{aligned} \min_{x} \;\; & c_x^T x \\ \text{s.t.} \;\; & A_i x \le b_i + F_i \theta, \;\; \forall i \in \mathbb{I} \\ & A_j x = b_j + F_j \theta, \;\; \forall j \in \mathbb{J} \\ & \theta \in \Theta := \{\theta \in \mathbb{R}^m \,|\, CR_A \theta \le CR_b\} \\ & x \in \mathbb{R}^n \end{aligned} \tag{6}$$

where the matrices $A_i \in \mathbb{R}^{1\times q}$, $F_i \in \mathbb{R}^{1\times m}$, $A_j \in \mathbb{R}^{1\times q}$, $F_j \in \mathbb{R}^{1\times m}$ and the scalars b_i, b_j correspond to the ith and jth inequality and equality constraints of the sets $\mathbb{I}$ and $\mathbb{J}$ respectively.

The multiparametric solution of Eq. (6) returns a list of critical regions, and each critical region defines affine functions relating the bounded uncertain parameters to the optimal continuous decision variables, Eq. (7).

$$x^* = K_i \theta^* + r_i, \;\; \theta^* \in CR^i = \{CR_A^i \theta \le CR_b^i\} \tag{7}$$

where x^* is the optimal solution at the parameter realization θ^*, CR^i define the ith critical region, and K_i and r_i define the affine expression for the ith critical region.

The development of the multiparametric solution has been addressed using three types of algorithmic approaches (i) geometric-based strategies (Bemporad et al., 2002), (ii) active set-based strategies (Gupta et al., 2011), and (iii) combinations of geometric and active set methodologies (Oberdieck et al., 2017).

Geometric-based algorithms are founded on the following notion: given a critical region, the neighborhoods area is explored to identify adjacent critical regions, Fig 5. These identified critical regions are subsequently utilized to explore their neighboring critical regions, until the full parametric solution is determined.

On the other hand, combinatorial approaches aim to implicitly explore the parameter space through enumeration and efficient pruning criteria of all possible active set which can yield an optimal solution for a feasible parameter realization, Fig 6. Because each optimal active set yields a critical region, the identification of the complete map of solutions is achieved.

Each of the approaches offers its own advantages and drawbacks in regards to developing the solution of the multiparametric optimization problem, and the selection of the algorithmic strategy depends on the structure of each particular problem (Oberdieck et al., 2016a). Such algorithms are provided in

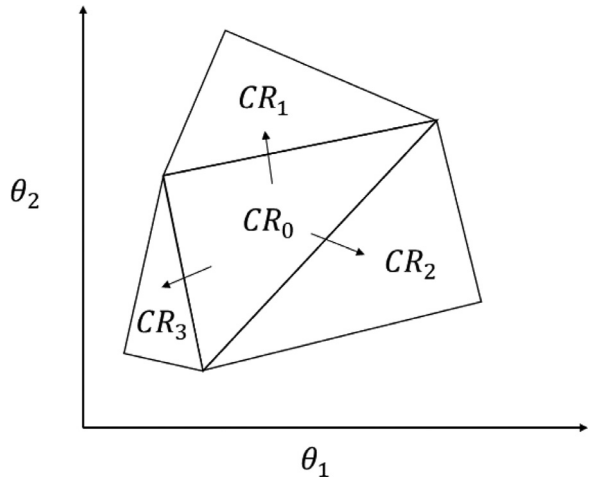

Fig. 5. Illustrative representation of the exploration of the parameter space using a geometrical algorithm for multiparametric programming, adapted from Katz, 2020.

the state-of-the-art software, Parametric OPtimization (POP) Toolbox (Oberdieck et al., 2016b) and Multiparametric Toolbox (MPT) (Herceg et al., 2013).

4.2. Multiparametric mixed-integer linear programming

A vital element for the integration of multiparametric programming and deep learning using ReLU activation functions are mpMILPs, which are represented by Eq. (2). Developing the full multiparametric solution for mpMILPs is significantly more computationally taxing than for mpLPs or mpQPs. However, there exists full theory and algorithms to solve these challenging problems.

An overview of the solution procedure is as follows. Given the mpMILP, a candidate set of critical regions is determined for a particular combination of binary variables. In an iterative manner, each critical region is examined to identify if an improved solution exists for a different binary combination. The comparison procedure is performed by introducing integer cuts into the original mpMILP within the confines of the critical region under examination. The resulting map of solutions provides the optimal critical regions, the optimization variables as functions of the uncertain parameters, and the optimal integer variables, (8).

$$\left.\begin{aligned} x^* = K_i\theta^* + r_i, \\ y^* = y_i, \end{aligned}\right\} \quad \theta^* \in CR^i = \{CR^i_A\theta \le CR^i_b\}. \tag{8}$$

Note that improving mpMILP algorithms is accomplished via strengthening the comparison procedure, with the goal of identifying fewer candidate critical regions.

5. Framework for integration

Current theory regarding multiparametric programming indicates the optimization formulation must have linear or piecewise linear constraints with a linear or convex quadratic objective function. Hence, the description of nonlinear constraints or objective functions that naturally appear in real processes cannot be achieved directly in multiparametric programs. Therefore, surrogate models are a necessity to capture such complex relationships. Because of their strong predictive capabilities, deep learning models using ReLU functions are ideal candidates to be incorporated into mpP formulations.

In the remaining of this section, the framework for integrating deep learning models and multiparametric programming is presented. First the multiparametric nonlinear programming (mpNLP) problem is formulated. The nonlinear equations are then approximated using a deep neural network with ReLU activation functions. The neural network is then recast as a MILP. The mpNLP is reformulated to incorporate the MILP representation of the neural network, yielding a mpMILP. With the developed mpP problem formulated, the problem is solved using the POP toolbox. Further details are provided in the subsequent sections.

5.1. Optimization formulation

Eq. (9) defines a mpNLP with a linear objective and bounded uncertain parameters, θ. In addition, it is assumed the objective function is linear for clarity purposes only, and is not a restriction of the proposed integration strategy.

$$\begin{aligned} \min_x \quad & c^T x \\ \text{s.t.} \quad & h(x,\theta) = 0 \\ & g(x,\theta) \le 0 \\ & x \in X^q \\ & \theta \in \Theta^m \end{aligned} \tag{9}$$

where c is a cost vector, $x \in \mathbb{R}^q$ is the vector of optimization variables, $\theta \in \mathbb{R}^m$ is the vector of uncertain parameters, and h and g are nonlinear relationships between optimization variables and uncertain parameters. Some examples of the nonlinear constraints defined by h and g are thermodynamic relations, disturbance models, or path constraints. The nonlinear constraints defined by h and g are considered to be complex, nonlinear, and nonconvex ensuring the development of the solution for a given parameter realization

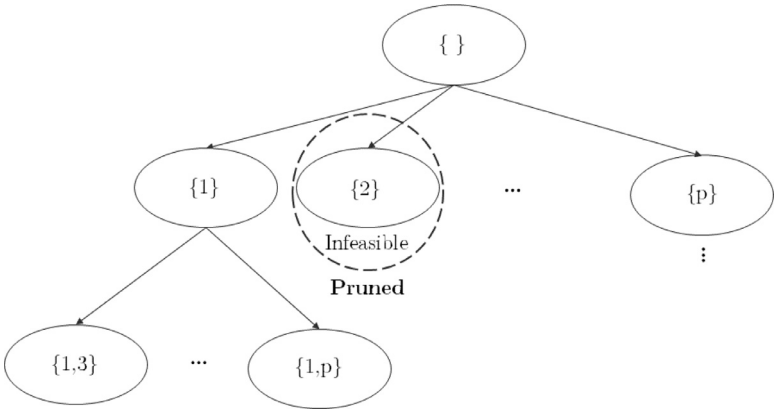

Fig. 6. Illustrative representation of the exploration of the parameter space using a geometrical algorithm for multiparametric programming, adapted from Katz, 2020.

of the resulting NLP is challenging. Reducing the complexity of h and g improves the computational performance of determining the optimal solution for a given parameter realization.

5.2. Neural network model development

The nonlinear functions h and g in Eq. (9) are approximated using deep neural networks with ReLU activation functions. The neural network is developed through data sampling, training, and validation. Note that each nonlinear function can be represented as a deep neural network. However, it may be advantageous to group functions that are related to reduce the total number of neural networks used in the final formulation.

Prior to model development, it is critical to develop a suitable dataset. The input space to the functions h and g are sampled to construct the input/output dataset. The relationship between the inputs of the functions to the outputs is important to understand prior to invoking a sampling procedure. For instance, if the functions being approximated are probability distributions, the sampling procedure should follow Monte Carlo techniques or Markov Chain Monte Carlo methods (van Ravenzwaaij et al., 2018). In the case h and g are defined by a dynamic model, system identification techniques are used to properly sample and excite the system to achieve the desired relationship between the inputs and outputs (Ljung, 1999). Following sampling the space of the nonlinear functions, the dataset is normalized to improve training the neural network model.

Once a suitable input/output dataset is constructed, the neural network is defined by specifying the number of hidden layers and the number of nodes in each hidden layer. Determining the optimal number of hidden layers and nodes is not trivial, and in many cases heuristically searching for a suitable size is sufficient.

5.3. Reformulation

As demonstrated in the literature (Fischetti and Jo, 2018; Grimstad and Andersson, 2019a), a neural network involving ReLU activation functions can be exactly recast as a MILP following a big M formulation. A big M formulation, as shown in Eq. (11), is a strategy to activate or deactivate constraints based on the value of the binary variable y, either 0 or 1. The value of the M parameter is a large number and it is selected depending on the application. Using a MILP formulation allows the neural network to be embedded into an optimization problem directly without the associated difficulty of a composite function with the nonlinear *max* operator. The procedure is presented as follows.

For an arbitrary layer with n nodes, the output takes the form of Eq. (10), where k is the layer, W^k is the matrix of weights for layer k, $\hat{b}^k$ is the vector of biases for layer k, $x^{k-1} \in \mathbb{R}^n$ is the output of the previous layer, and $x^k \in \mathbb{R}^n$ is the output of the current layer. The *max* operator is taken element-wise.

$$x^k = max\{0, W^k x^{k-1} + \hat{b}^k\} \tag{10}$$

The importance of the ReLU activation function is its piecewise linear nature. Therefore, Eq. (10) can be exactly recast in an optimization formulation via the inclusion of binary variables. Eq. (11) is the reformulation of the k^{th} hidden layer in an MILP structure[1]. Eq. (11) represents the neural network structure and not the overall optimization problem. The final optimization problem formulation has the form of Eq. (2).

$$W^k x^{k-1} + \hat{b}^k = x^k - s^k \tag{11a}$$

$$x^k \le M_1 y \tag{11b}$$

$$s^k \le -M_2(1-y) \tag{11c}$$

$$x^k \ge 0 \tag{11d}$$

$$s^k \ge 0 \tag{11e}$$

$$y \in \{0,1\}^n \tag{11f}$$

In (11), y is a vector of binary variables, $s^k \in \mathbb{R}^n$ is the vector of slack variables, and M_i is a large scalar value. To improve the computational performance of identifying the optimal solution, the big M values, M_1 and M_2, are defined to tightly bound the optimization variables x^k and s^k (Grimstad and Andersson, 2019b). If the big M values are well defined, an unnecessary computational burden is avoided. After recasting, the total number of binary variables is equal to the total number of nodes that constitute the hidden layers. Note that the recasted neural network with ReLU activation function is an exact reformulation. The binary variables enable the activation function to output a value of 0 or x, via the constraints (11b) and (11c). Incorporating the recasted neural network into an optimization formulation provides an effective strategy to maintain high accuracy with a surrogate model, and obtaining the global optimum does not require specialized global optimization techniques. Instead, standard mixed-integer linear programming methods are utilized such as cutting planes and branch and bound techniques which are readily available in existing software.

During training, the number of active nodes (nodes that take a value other than 0) can be reduced using regularization techniques. Minimizing the number of active nodes directly corresponds to a reduction in the number of binary variables needed during the recasting procedure. By reducing the number of binary variables needed to define the neural network, the computational burden of solving the final optimization problem is significantly diminished. After training, postprocessing the neural network model also has the potential to minimize unnecessary binary variables. For instance, a node in a hidden layer that is always positive is represented by a linear activation function. Hence, that node does not require a slack variable and a binary variable. Because of the mixed-integer nature of the optimization formulation, any reduction in the number of binary variables made have the potential for impacting the computational performance notably.

5.3.1. Variable aggregation

Following Eq. (11), the neural network is transformed to a system of equality constraints, inequality constraints, binary variables, and slack variables. Because the equality constraints are linear, the intermediate optimization variables can be eliminated via variable aggregation. By eliminating intermediate variables, the number of optimization variables and constraints is reduced, which is important in developing the multiparametric solution.

$$x^1 = W^1 x^0 + \hat{b}^1 + s^1 \tag{12a}$$

$$x^k = \prod_{i=k}^{1} W^i x^0 + \sum_{i=1}^{k-1} \prod_{j=k}^{i+1} W^j (\hat{b}^i + s^i) + \hat{b}^k + s^k, \;\; k = 2, \ldots, K-1 \tag{12b}$$

$$x^K = \prod_{i=K}^{1} W^i x^0 + \sum_{i=1}^{K-1} \prod_{j=K}^{i+1} W^j (\hat{b}^i + s^i) + \hat{b}^K \tag{12c}$$

[1] The final layer, or the output layer, does not require the modification presented in (11) because the activation function in the final layer is linear

where W^k and $\hat{b}^k$ define the weights of the k^{th} hidden layer, K is the output layer, s^k is the vector of slack variables, x^0 is the input vector to the neural network, and x^K is the vector of outputs of the neural network. Eqs. (11) and (12) are combined to provide a set of constraints in the form of Eq. (2).

5.3.2. The importance of sparsity

Introducing sparsity into the neural network has a profound impact on the resulting optimization formulation. Because sparsity aims to minimize the active nodes in the neural network, the number of optimization variables and constraints are reduced. Furthermore, by reducing the size of the final optimization problem, the resulting parametric solution requires less offline computational effort, and maintains fewer defining critical regions. The offline computational performance is improved because the complexity of mpMILPs scales with the number of optimization variables and constraints.

5.4. Final mpMILP formulation

The final multiparametric formulation is in the form of an mpMILP, Eq. (2). The mpMILP contains the inherent linear components of the optimization problem, such as box constraints, and the embedded neural network. By using a neural network to approximate the nonlinear parts of the mpNLP, the optimization formulation is tractable and provides an optimal solution resembling the true optimal. Another key feature of the resulting mpMILP is that the full parametric solution can be developed, unlike the original mpNLP, using the POP toolbox.

6. Results

A chemostat is considered to demonstrate the effectiveness of integrating deep learning models and multiparametric programming. The aim of the chemostat is to maximize the biomass concentration during operation. In this work, the ability of a neural network to accurately capture the nonlinearity of a process is not under scrutiny. Therefore, the illustrative example used is of small scale, and is adapted from (Eaton and Rawlings, 1992). In this process, the chemostat is continuously operated, and the process is substrate inhibited. It is assumed the chemostat (i) is fed with pure substrate, (ii) has constant volume, (iii) and is temperature independent. The differential algebraic equations defining the chemostat are presented in Eq. (13).

$$\frac{dx}{dt} = (\mu - D)x \tag{13a}$$

$$\frac{ds}{dt} = (s_f - s)D - \frac{\mu x}{y} \tag{13b}$$

$$\mu = \frac{\mu_{max} s}{k_m + s + k_1 s^2} \tag{13c}$$

where x is the biomass concentration in the chemostat, s is the substrate concentration in the chemostat, $s_f = 4$ is the substrate concentration in the feed, μ is the specific growth rate, $y = 0.4$ is the yield of cell mass, and D is the dilution rate. The specific growth rate is defined by the parameters $\mu_{max} = 0.53$, $k_m = 0.12$, and $k_1 = 0.4545$.

The objective for the chemostat is to develop a multiparametric model predictive control formulation, whereby the dilution rate is manipulated in real time to maximize the biomass concentration. The difficulty in satisfying the control objective is the nonlinearity present from the specific growth rate. The continuous time control formulation is provided by Eq. (14).

$$\begin{aligned} \min_{D} \quad & x(t_f) \\ \text{s.t.} \quad & \dot{x} = (\mu - D)x, \ x(0) = x_0 \\ & \dot{s} = (s_f - s)D - \frac{\mu x}{y}, \ s(0) = s_0 \\ & \mu(t) = \frac{\mu_{max} s(t)}{k_m + s(t) + k_1 s(t)^2} \\ & \underline{D} \le D \le \bar{D} \\ & x \in X, \ s \in S \end{aligned} \tag{14}$$

Because of the difficulty in solving the continuous time optimal control problem, a discrete time approximate model is developed. By utilizing the presented framework, a neural network consisting of ReLU activation functions is used to approximate the chemostat model.

6.1. Open loop

Input/output data is first generated in open loop for the chemostat process. The input data is generated by perturbing the dilution rate randomly for $5 \cdot 10^4$ time steps. The input profile aims to excite the system to capture the nonlinear dynamics of the process, and to ensure enough data is collected to train the neural network. To improve the training performance, the input and output data is normalized between $[-1, 1]$. During training, the possibility of overfitting is reduced via partitioning the data set into a training, validating, and testing set following a 70%/15%/15% split, respectively. The neural network maintains 3 inputs (the dilution rate and the initial conditions of the process) and 2 outputs (the predicted states of the system). In addition, the neural network has 3 hidden layers with 5 nodes in the first hidden layer, 4 in the second, and 5 in the third.

Fig. 7 presents the open loop response of the process and the output of the neural network for a slice of the total training data for visibility. The neural network has a mean squared error (MSE) of $1.2 \cdot 10^{-5}$ indicating an acceptable open loop performance. The developed neural network is designed to predict one step ahead into the future. Therefore, the explicit controller predicts one step into the future. Further time step prediction can be developed via

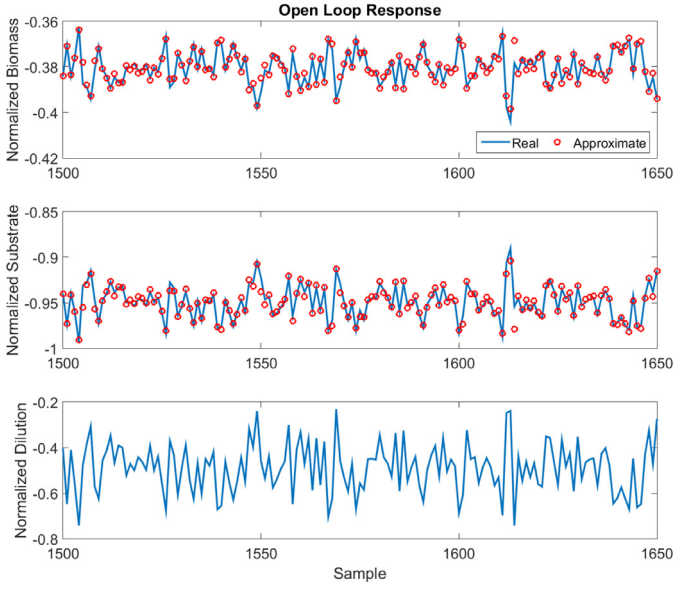

Fig. 7. The open loop performance of the chemostat and neural network.

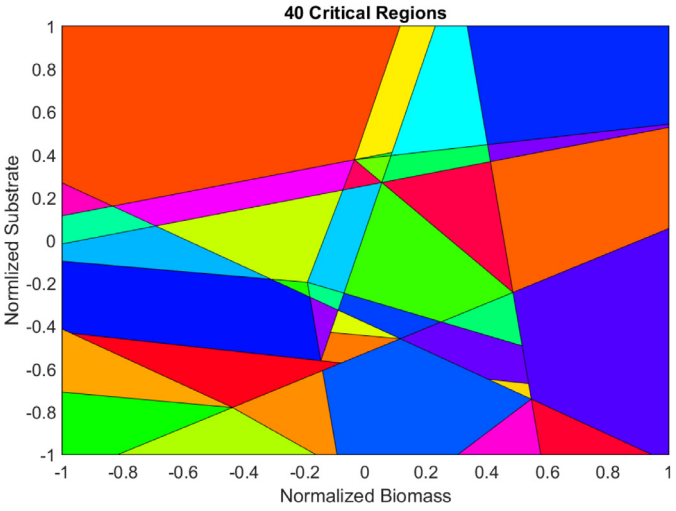

Fig. 8. The parametric solution for the explicit model predictive controller with a deep learning model.

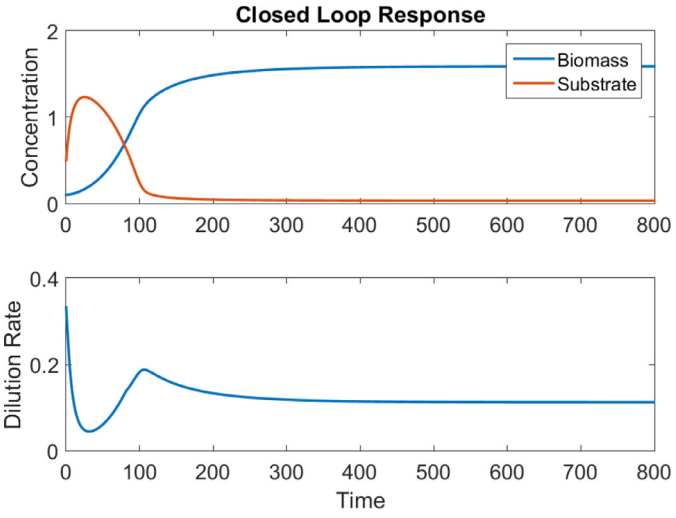

Fig. 9. The closed loop performance of the chemostat with an explicit model predictive controller.

(i) training the neural network on input/output data that represents multiple steps into the future, or (ii) feeding the output of the one step ahead neural network back to itself.

6.2. Multiparametric solution

The neural network developed to approximate the nonlinear dynamics in the optimal control formulation, Eq. (14), is used to formulate the resulting mpMILP. The developed mpMILP has 62 constraints, 15 continuous optimization variables, 14 binary variables, and 2 uncertain parameters. The full solution is identified using the POP toolbox in the Matlab environment (Oberdieck et al., 2016b). The parametric solution has 40 critical regions, calculated in 14.7 seconds and is presented in Fig. 8.

6.3. Closed loop performance

The developed explicit controller is validated in closed loop against the chemostat dynamic model. Fig. 9 demonstrates the ability of the explicit controller to maximize the biomass concentration. From the closed loop response, the process has no fluctuations and meets the target objective. It is important to point out the nonlinearity displayed by the controller is a direct consequence

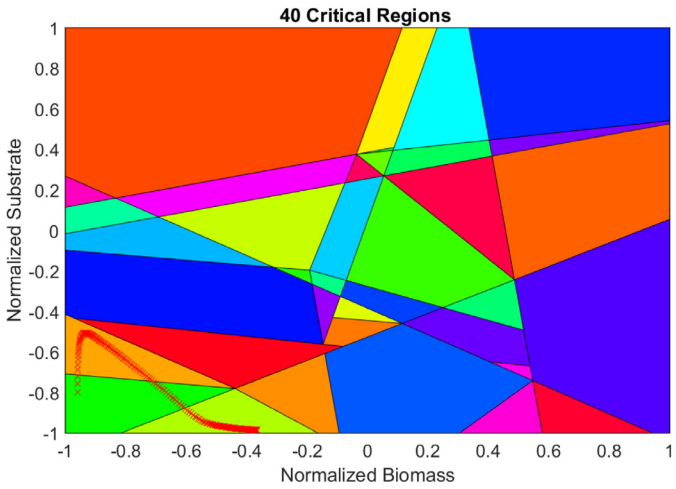

Fig. 10. The states of the chemostat overlayed on the parametric solution during the closed loop response.

of using the neural network model. The input profile is developed in such a way to maximize the biomass concentration to a final value of 1.6, when started from a value of 0.1. The state trajectories are also visualized in Fig. 10, in which the closed loop performance is overlayed on the explicit map of solution.

7. Conclusion

In online applications where time is a critical factor, determining the optimal solution with these embedded deep learning models can be a challenge. Multiparametric programming alleviates the computational burden by developing the offline, explicit solution to the parametric optimization problem. With available online solvers, developing the full parametric solution to the mpMILP is manageable and permits its application for the integration of deep learning models and multiparametric programming.

This manuscript presented the integration of deep learning models and multiparametric programming. The deep learning models are neural networks with a feedforward structure. Rectified linear units defined the activation function of the neural network due to their piecewise linear nature. Therefore, neural networks with ReLU activation functions are suitable candidates to approximate nonlinear functions because of their predictive capabilities and ability to be embedded as a MILP in a general optimization formulation. The ability to integrate neural networks with ReLU activation functions into parametric optimization was demonstrated by the optimal operation of a chemostat.

Integrating accurate surrogate models in optimization formulations is a critical step in many fields. Future work includes (i) using the presented methodology in various optimization based applications that are already formulated as MILPs such as scheduling decisions, (ii) comparisons of different surrogate modeling approaches, and (iii) reducing the time to develop the full multiparametric solution.

Our future work will be focused on incorporating the proposed approach to other multiparametric programming formulations and in using adaptive/alternative neural network structures as a means to enhance the ability of the network to capture nonlinear dynamic behavior (Wu et al., 2019a; Katz, 2020).

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

CRediT authorship contribution statement

Justin Katz: Methodology, Software, Validation, Formal analysis, Writing - original draft. **Iosif Pappas:** Software, Validation, Formal analysis, Writing - review & editing. **Styliani Avraamidou:** Validation, Formal analysis, Writing - review & editing. **Efstratios N. Pistikopoulos:** Conceptualization, Supervision, Project administration, Funding acquisition.

Acknowlgedgments

Financial support from the NSF SusChEM (Grant no. 1705423), NSF INFEWS (Grant no. 1739977), Texas A&M Energy Institute, and the Rapid Advancement in Process Intensification Deployment (RAPID SYNOPSIS Project - DE-EE0007888-09-04) Institute is gratefully acknowledged.

References

Afram, A., Janabi-Sharifi, F., Fung, A.S., Raahemifar, K., 2017. Artificial neural network (ANN) based model predictive control (MPC) and optimization of HVAC systems: a state of the art review and case study of a residential HVAC system. Energy Build. 141, 96–113. doi:10.1016/j.enbuild.2017.02.012.

Ahmadi-Moshkenani, P., Johansen, T.A., Olaru, S., 2018. Combinatorial approach toward multiparametric quadratic programming based on characterizing adjacent critical regions. IEEE Trans. Autom. Control 63 (10), 3221–3231. doi:10.1109/TAC.2018.2791479.

Asher, M.J., Croke, B.F.W., Jakeman, A.J., Peeters, L.J.M., 2015. A review of surrogate models and their application to groundwater modeling. Water Resour. Res. 51 (8), 5957–5973. doi:10.1002/2015WR016967.

Avraamidou, S., Pistikopoulos, E.N., 2019. A multi-parametric optimization approach for bilevel mixed-integer linear and quadratic programming problems. Comput. Chem. Eng. 125, 98–113. doi:10.1016/j.compchemeng.2019.01.021.

Bemporad, A., Morari, M., Dua, V., Pistikopoulos, E.N., 2002. The explicit linear quadratic regulator for constrained systems. Automatica 38, 3–20.

Beykal, B., Boukouvala, F., Floudas, C.A., Pistikopoulos, E.N., 2018. Optimal design of energy systems using constrained grey-box multi-objective optimization. Comput. Chem. Eng. 116, 488–502. doi:10.1016/j.compchemeng.2018.02.017.

Burnak, B., Diangelakis, N.A., Katz, J., Pistikopoulos, E.N., 2019. Integrated process design, scheduling, and control using multiparametric programming. Comput. Chem. Eng. 125, 164–184. doi:10.1016/j.compchemeng.2019.03.004.

Burnak, B., Katz, J., Diangelakis, N.A., Pistikopoulos, E.N., 2018. Simultaneous process scheduling and control: amultiparametric programming-based approach. Ind. Eng. Chem. Res. 57 (11), 3963–3976. doi:10.1021/acs.iecr.7b04457.

Charitopoulos, V.M., Papageorgiou, L.G., Dua, V., 2018. Multi-parametric mixed integer linear programming under global uncertainty. Comput. Chem. Eng. 116, 279–295.

Chen, S., Saulnier, K., Atanasov, N., Lee, D.D., Kumar, V., Pappas, G.J., Morari, M., 2018. Approximating explicit model predictive control using constrained neural networks. In: 2018 Annual American Control Conference (ACC), pp. 1520–1527. doi:10.23919/ACC.2018.8431275.

Chiang, L., Lu, B., Castillo, I., 2017. Big data analytics in chemical engineering. Annu. Rev. Chem. Biomol. Eng. 8 (1), 63–85. doi:10.1146/annurev-chembioeng-060816-101555.

Eaton, J.W., Rawlings, J.B., 1992. Model-predictive control of chemical processes. Chem. Eng. Sci. 47 (4), 705–720. doi:10.1016/0009-2509(92)80263-C.

Eckle, K., Schmidt-Hieber, J., 2019. A comparison of deep networks with relu activation function and linear spline-type methods. Neural Netw. 110, 232–242. doi:10.1016/j.neunet.2018.11.005.

Fischetti, M., Jo, J., 2018. Deep neural networks and mixed integer linear optimization. Constraints 23 (3), 296–309. doi:10.1007/s10601-018-9285-6.

Grimstad, B., Andersson, H., 2019a. Relu networks as surrogate models in mixed-integer linear programs. arXiv:1907.03140.

Grimstad, B., Andersson, H., 2019. Relu networks as surrogate models in mixed-integer linear programs. Comput. Chem. Eng. 106580. doi:10.1016/j.compchemeng.2019.106580.

Gupta, A., Bhartiya, S., Nataraj, P., 2011. A novel approach to multiparametric quadratic programming. Automatica 47 (9), 2112–2117. doi:10.1016/j.automatica.2011.06.019.

Herceg, M., Kvasnica, M., Jones, C., Morari, M., 2013. Multi-Parametric Toolbox 3.0. In: Proc. of the European Control Conference. Zürich, Switzerland, pp. 502–510. http://control.ee.ethz.ch/~mpt.

Himmelblau, D.M., 2000. Applications of artificial neural networks in chemical engineering. Korean J. Chem. Eng. 17 (4), 373–392. doi:10.1007/BF02706848.

Himmelblau, D.M., 2008. Accounts of experiences in the application of artificial neural networks in chemical engineering. Ind. Eng. Chem. Res. 47 (16), 5782–5796. doi:10.1021/ie800076s.

Hough, B.R., Beck, D.A., Schwartz, D.T., Pfaendtner, J., 2017. Application of machine learning to pyrolysis reaction networks: reducing model solution time to enable process optimization. Comput. Chem. Eng. 104, 56–63. doi:10.1016/j.compchemeng.2017.04.012.

Jones, C.N., Morrari, M., 2006. Multiparametric linear complementarity problems. In: Proceedings of the 45th IEEE Conference on Decision and Control, pp. 5687–5692. doi:10.1109/CDC.2006.377797.

Karasuyama, M., Harada, N., Sugiyama, M., Takeuchi, I., 2012. Multi-parametric solution-path algorithm for instance-weighted support vector machines. Mach. Learn. 88 (3), 297–330. doi:10.1007/s10994-012-5288-5.

Katz, J., 2020. Advancing Multiparametric Programming for Model Predictive Control. Texas A&M University.

Katz, J., Burnak, B., Pistikopoulos, E.N., 2018. The impact of model approximation in multiparametric model predictive control. Chem. Eng. Res. Des. 139, 211–223. doi:10.1016/j.cherd.2018.09.034.

Kim, S.H., Boukouvala, F., 2019. Machine learning-based surrogate modeling for data-driven optimization: a comparison of subset selection for regression techniques. Optim. Lett. 1–22.

Kopanos, G.M., Pistikopoulos, E.N., 2014. Reactive scheduling by a multiparametric programming rolling horizon framework: a case of a network of combined heat and power units. Ind. Eng. Chem. Res. 53 (11), 4366–4386. doi:10.1021/ie402393s.

Lee, J., Chang, H.J., 2018. Explicit model predictive control for linear time-variant systems with application to double-lane-change maneuver. PLoS One 13 (12). doi:10.1371/journal.pone.0208071.

Ljung, L., 1999. System Identification Theory for the User. Prentice-Hall, EngleWood Cliffs, NJ.

Montufar, G.F., Pascanu, R., Cho, K., Bengio, Y., 2014. On the number of linear regions of deep neural networks. In: Advances in Neural Information Processing Systems, pp. 2924–2932.

Nagata, Y., Chu, K.H., 2003. Optimization of a fermentation medium using neural networks and genetic algorithms. Biotechnol. Lett. 25 (21), 1837–1842. doi:10.1023/A:1026225526558.

Ning, C., You, F., 2019. Optimization under uncertainty in the era of big data and deep learning: when machine learning meets mathematical programming. Comput. Chem. Eng. 125, 434–448.

Oberdieck, R., Diangelakis, N.A., Nascu, I., Papathanasiou, M.M., Sun, M., Avraamidou, S., Pistikopoulos, E.N., 2016. On multi-parametric programming and its applications in process systems engineering. Chem. Eng. Res. Des. 116, 61–82.

Oberdieck, R., Diangelakis, N.A., Papathanasiou, M.M., Nascu, I., Pistikopoulos, E.N., 2016. POP – parametric optimization toolbox. Ind. Eng. Chem. Res. 55, 8979–8991.

Oberdieck, R., Diangelakis, N.A., Pistikopoulos, E.N., 2017. Explicit model predictive control: a connected-graph approach. Automatica 76, 103–112. doi:10.1016/j.automatica.2016.10.005.

Oberdieck, R., Pistikopoulos, E.N., 2015. Explicit hybrid model-predictive control: the exact solution. Automatica 58, 152–159. doi:10.1016/j.automatica.2015.05.021.

Pfrommer, J., Zimmerling, C., Liu, J., Kärger, L., Henning, F., Beyerer, J., 2018. Optimisation of manufacturing process parameters using deep neural networks as surrogate models. Procedia CIRP 72, 426–431. doi:10.1016/j.procir.2018.03.046.

Pinkus, A., 1999. Approximation theory of the MLP model in neural networks. Acta Numer. 8, 143–195. doi:10.1017/S0962492900002919.

Poole, D.L., Mackworth, A.K., 2010. Artificial Intelligence: Foundations of Computational Agents. Cambridge University Press.

van Ravenzwaaij, D., Cassey, P., Brown, S.D., 2018. A simple introduction to Markov chain monte–carlo sampling. Psychonom. Bull. Rev. 25 (1), 143–154. doi:10.3758/s13423-016-1015-8.

Rister, B., Rubin, D.L., 2017. Piecewise convexity of artificial neural networks. Neural Netw. 94, 34–45. doi:10.1016/j.neunet.2017.06.009.

Schweidtmann, A.M., Huster, W.R., Lüthje, J.T., Mitsos, A., 2019. Deterministic global process optimization: accurate (single-species) properties via artificial neural networks. Comput. Chem. Eng. 121, 67–74. doi:10.1016/j.compchemeng.2018.10.007.

Shang, C., You, F., 2019. Data analytics and machine learning for smart process manufacturing: recent advances and perspectives in the big data era. Engineering.

Shokry, A., Vicente, P., Escudero, G., Pérez-Moya, M., Graells, M., Espuña, A., 2018. Data-driven soft-sensors for online monitoring of batch processes with different initial conditions. Comput. Chem. Eng. 118, 159–179. doi:10.1016/j.compchemeng.2018.07.014.

Steuer, R.E., Qi, Y., Hirschberger, M., 2006. Portfolio optimization: new capabilities and future methods. Z. Betriebswirtschaft 76 (2), 199–220. doi:10.1007/s11573-006-0006-z.

Tran, A., Pont, M., Aguirre, A., Durand, H., Crose, M., Christofides, P.D., 2018. Bayesian model averaging for estimating the spatial temperature distribution in a steam methane reforming furnace. Chem. Eng. Res. Des. 131, 465–487.

Tso, W.W., Demirhan, C.D., Floudas, C.A., Pistikopoulos, E.N., 2019. Multi-scale energy systems engineering for optimal natural gas utilization. Catal. Today doi:10.1016/j.cattod.2019.09.009.

Wittmann-Hohlbein, M., Pistikopoulos, E.N., 2013. On the global solution of multiparametric mixed integer linear programming problems. J. Glob. Optim. 57 (1), 51–73. doi:10.1007/s10898-012-9895-2.

Wu, Z., Rincon, D., Christofides, P.D., 2019. Real-time adaptive machine-learning-based predictive control of nonlinear processes. Ind. Eng. Chem. Res.

Wu, Z., Tran, A., Rincon, D., Christofides, P.D., 2019. Machine learning-based predictive control of nonlinear processes. Part I: theory. AIChE J. 65 (11), e16734. doi:10.1002/aic.16734.

Wu, Z., Tran, A., Rincon, D., Christofides, P.D., 2019. Machine-learning-based predictive control of nonlinear processes. Part II: computational implementation. AIChE J. 65 (11), e16734. doi:10.1002/aic.16734.