Global Optimization of Processes through Machine Learning

Globale Prozessoptimierung durch maschinelles Lernen

Von der Fakultät für Maschinenwesen der Rheinisch-Westfälischen Technischen Hochschule Aachen zur Erlangung des akademischen Grades eines Doktors der Ingenieurwissenschaften genehmigte Dissertation

vorgelegt von

Artur M. Schweidtmann

Berichter: Universitätsprofessor Alexander Mitsos, Ph. D.
Universitätsprofessor Dr. rer. nat. Andreas Schuppert

Tag der mündlichen Prüfung: 8. März 2021

Titel: Global Optimization of Processes through Machine Learning

Autor: Artur M. Schweidtmann

Reihe: Aachener Verfahrenstechnik Series
AVT.SVT - Process Systems Engineering
Band 18 (2021)

Herausgeber: Aachener Verfahrenstechnik
Forckenbeckstraße 51
52074 Aachen
Tel.: +49 (0) 241 80 97717
Fax.: +49 (0) 241 80 92326
E-Mail: secretary.svt@avt.rwth-aachen.de
http://www.avt.rwth-aachen.de/AVT

DOI: https://doi.org/10.18154/RWTH-2021-05536

Preface

This thesis was developed during my time as a research associate at the Aachener Verfahrenstechnik, Chair of Process Systems Engineering (AVT.SVT) of RWTH Aachen University.
I am extremely grateful to my doctoral advisor Prof. Alexander Mitsos, Ph.D. for his exceptional support and academic guidance. I also thank Prof. Andreas Schuppert for the co-examination of my thesis and Prof. Sebastian Trimpe for chairing the examination committee.
My main project partner in the SynErgie project was the Linde Aktiengesellschaft and I thank in particular Anna Ecker, Dr.-Ing. Andreas Peschel, and Florian Schliebitz from Linde for the very pleasant collaboration. I gratefully acknowledge the financial support of the Kopernikus project SynErgie by the Federal Ministry of Education and Research (BMBF) and the project supervision by the project management organization Projektträger Jülich (PtJ).
I would like to extend my deepest gratitude to all colleagues from AVT.SVT for the friendly and cooperative environment as well as for the time spent together inside and outside the office. In particular, I thank Adrian Caspari, Dominik Bongartz, Andreas Bremen, Wolfgang Huster, Chryssa Kappatou, Luise Kaven, Deniz Rall, Pascal Schäfer, and Yannic Vaupel for scientific discussions, great collaborations, joint (business) trips, and many enjoyable evenings at Kaktus. Furthermore, I want to extent my thanks to Adel Mhamndi, Petra Eissa, Wand Frohn, Didem Uslu, Jutta Friedrich, and Sascha Gerhards for their support.
I had the pleasure to work with a large number of students and I would like to thank all of them. In particular, I thank Linus Netze who supported me and my students for many years as a research assistant. Also, I thank Danimir Doncevic, Jan Rittig, Tim Kerkenhoff, and Jannik Lütchje for their exceptional contributions. Moreover, I would like to thank my (external) collaborators. In particular, I thank Eric Bradford, Prof. Alexei Lapkin, and Prof. Matthias Wessling for many fruitful collaborations.
I would like to thank my friends from high school, in particular Marvin and Laura, my Freshers' group, and the Gummibärenbande, in particular Andreas, Sebastian, Tobe, and Thomas. Finally, yet importantly, I am forever grateful to Jana and my family, in particular, my mother Yvonne, my sister Theresa, and my grandmother Helga for their everlasting support.

Den Haag, June 2021 *Artur M. Schweidtmann*

Contents

Abstract

Machine learning models can learn complex relationships from data and have led to breakthrough results in various domains. In chemical engineering, machine learning models have great potential for process optimization when combined with mechanistic model equations. However, machine learning models frequently lead to large-scale nonlinear optimization problems where deterministic global optimization is desirable but often intractable. In this dissertation, the global solution of optimization problems with machine learning models embedded is accelerated by orders of magnitude through the development of reduced-space formulations and tight relaxations. The reduced-space formulations are proposed for optimization problems with trained (deep and shallow) artificial neural networks and Gaussian processes embedded. The approach formulates the machine learning models in their original variable space which reduces the number of variables to be branched on compared to the standard full-space formulation. To obtain convex and concave relaxations, we propagate McCormick relaxations through the models which lead to smaller sizes of subproblems compared to the commonly used auxiliary variable method. Moreover, we develop tight relaxations for activation functions of neural networks, for acquisition functions used in Bayesian optimization, and for covariance functions used in Gaussian processes. Our approach greatly improves computational performance compared to the standard full-space formulations and thus enables global optimization for the rational design of ion-separation membranes, energy processes, and chemical processes using machine learning models. To ensure validity of the machine learning models during optimization, we learn the training data domain and encode it as constraints in process optimization. For this, we perform a topological data analysis using persistent homology identifying potential holes or separated clusters in the training data. Then, we train a one-class classifier on the training data domain or construct the convex hull and encode it as constraints in the subsequent process optimization. The developed methods are available in our open-source "MeLOn - **M**achin**e** **L**earning Models for **O**ptimizatio**n**" toolbox, which is a submodule to our global solver "MAiNGO - **M**cCormick-based **A**lgorithm for mixed-**i**nteger **N**onlinear **G**lobal **O**ptimization".

To further accelerate the learning of complex relationships, this work investigates information and knowledge representations for chemical engineering data. We study complex molecular structures as input data for physicochemical property prediction. In particular, we investigate higher-order physical graph neural networks that enable end-to-end learning of physicochemical properties from the molecular graph. We use the method for the prediction of the ignition quality of biofuels. In light of limited experimental data, a combination of multi-task learning, transfer learning, and ensemble learning is used, which results in competitive performance compared to state-of-the-art QSPR models. Furthermore, we identify physical pooling functions based on the molecular size dependency of physicochemical properties. Integrating this physical knowledge into the model structure can be understood as a hybrid modeling approach that improves generalization capabilities and reduces data requirements.

Kurzfassung

Modelle des maschinellen Lernens können komplexe Zusammenhänge aus Daten lernen und haben in verschiedenen Bereichen zu bahnbrechenden Ergebnissen geführt. In der Verfahrenstechnik hat die Kombination von maschinellem Lernen und mechanistischer Modellierung ein großes Potenzial für die Prozessoptimierung. Allerdings führen maschinelle Lernmodelle häufig zu großen nichtlinearen Optimierungsproblemen, bei denen eine deterministische globale Optimierung wünschenswert, aber oft unlösbar ist. In dieser Arbeit wird die globale Lösung von Optimierungsproblemen mit eingebetteten maschinellen Lernmodellen durch die Entwicklung von Unterraumformulierungen und engen Relaxationen um Größenordnungen beschleunigt. Die Unterraumformulierungen werden für Optimierungsprobleme mit eingebetteten trainierten (tiefen und flachen) künstlichen neuronalen Netzen und Gaußschen Prozessen entwickelt. Der Ansatz formuliert die Modelle in ihrem ursprünglichen Variablenraum, was die Anzahl der zu verzweigenden Variablen im Vergleich zur Standard Vollraumformulierung reduziert. Wir propagieren McCormick Relaxationen durch die Modelle, die im Vergleich zur üblicherweise verwendeten Hilfsvariablenmethode zu einer geringeren Größe der Unterprobleme führen. Darüber hinaus werden enge Relaxationen für die Aktivierungsfunktion von neuronalen Netzen, für Funktionen, die in der Bayes'schen Optimierung verwendet werden, und für Kovarianz Funktionen entwickelt. Unser Ansatz verbessert die Rechenleistung im Vergleich zu den Vollraumformulierungen erheblich und ermöglicht so eine globale Optimierung für das Design von Membranen, Energieprozessen und chemischen Prozessen unter Verwendung von datengetriebenen Modellen. Um die Gültigkeit der datengetriebenen Modelle während der Optimierung sicherzustellen, lernen wir die Trainingsdatendomäne. Dazu führen wir eine topologische Datenanalyse durch, die potenzielle Löcher oder getrennte Cluster in den Trainingsdaten identifiziert. Dann trainieren wir einen Ein-Klassen Klassifikator auf der Trainingsdatendomäne oder konstruieren die konvexe Hülle und kodieren sie als Nebenbedingungen in der nachfolgenden Prozessoptimierung. Die entwickelten Methoden sind in unserer Open-Source Toolbox "MeLOn - **M**achin**e** **L**earning Models for **O**ptimizatio**n**" verfügbar, die ein Submodul von unserem globalen Löser "MAiNGO - **M**cCormick-based **A**lgorithm for mixed-**i**nteger **N**onlinear **G**lobal **O**ptimization" ist.

Um das Lernen komplexer Zusammenhänge weiter zu beschleunigen, werden in dieser Arbeit Informations- und Wissensrepräsentationen für physikochemische Eigenschaften untersucht. Insbesondere untersuchen wir Graph neuronale Netze höherer Ordnung, die ein end-to-end Lernen von physikochemischer Eigenschaften aus dem molekularen Graphen ermöglichen. Wir verwenden die Methode für die Vorhersage der Zündqualität von Biokraftstoffen. Angesichts der begrenzten experimentellen Daten wird eine Kombination aus Multi-Task-Lernen, Transfer-Lernen und Ensemble-Lernen verwendet, was zu einer höheren Genauigkeit im Vergleich zu QSPR-Modellen führt. Darüber hinaus identifizieren wir physikalische Pooling-Funktionen, die auf der molekularen Größenabhängigkeit der physikochemischen Eigenschaften basieren. Die Integration dieses physikalischen Wissens in die Modellstruktur kann als ein hybrider Modellierungsansatz verstanden werden, der die Generalisierungsfähigkeiten verbessert und die Datenanforderungen reduziert.

Publications and copyrights

This thesis originates from the research performed by the author during his time at the Chair of Process Systems Engineering at the Aachener Verfahrenstechnik (AVT.SVT). Parts of this thesis have already been published. The publications are used in the following chapters as described including a description of the contributions of all authors:

- Chapter 1 uses parts of the abstracts published in [1–4] to summarize the chapters of the dissertation. Also, parts of the conference abstract in [5] has been used to describe advances in surrogate-based optimization.
  In addition, Chapter 1 and Chapter 7 are based on parts of [6], which has been submitted for publication. This literature review has been conducted by Artur M. Schweidtmann and Erik Esche. It has been edited by Alexander Mitsos and Jens-Uwe Repke and further adapted for a DFG priority programme proposal on machine learning in chemical engineering by Alexander Mitsos, Asja Fischer, Marius Kloft, Jens-Uwe Repke, Sebastian Sager, Erik Esche, and Artur M. Schweidtmann (c.f. Project call [7]).

- Chapter 2 is based on parts of [1].
  Authorship contribution statement: Artur M. Schweidtmann: Methodology, Software, Investigation, Validation, Formal analysis, Writing - original draft. Alexander Mitsos: Supervision, Resources, Conceptualization, Validation, Formal analysis, Funding acquisition, Writing - review & editing.
  Artur M. Schweidtmann conceptualized and supervised the bachelor theses of Nils Graß on this topic. Artur M. Schweidtmann supervised the student assistant Linus Netze on this topic.

- Chapter 3 is based on parts of [2].
  Authorship contribution statement: Artur M. Schweidtmann: Conceptualization, Methodology, Software, Investigation, Validation, Formal analysis, Writing - original draft, Writing - review & editing. Dominik Bongartz: Methodology, Software, Investigation, Validation, Formal analysis, Writing - review & editing. Daniel Grothe: Software, Investigation, Validation, Formal analysis. Tim Kerkenhoff: Software, Investigation, Validation, Formal analysis. Xiaopeng Lin: Software, Investigation, Validation, Formal analysis. Jaromił Najman: Methodology, Software, Investigation, Validation, Formal analysis, Writing - review & editing. Alexander Mitsos: Supervision, Resources, Validation, Formal analysis, Funding acquisition, Writing - review & editing.
  Artur M. Schweidtmann conceptualized and supervised the master theses of Xiaopeng Lin and Daniel Grothe on this topic. Artur M. Schweidtmann supervised the student assistants Tim Kerkenhoff and Linus Netze on this topic. Artur M. Schweidtmann, Xiaopeng Lin, and Daniel Grothe implemented the GP models and parser. Artur M. Schweidtmann, Dominik Bongartz, Jaromił Najman, Tim Kerkenhoff derived the function relaxations. Jaromił Najman and Dominik Bongartz implemented the function relaxations.

- Chapter 4 is based on parts of [4].
  Authorship contribution statement: Artur M. Schweidtmann: Conceptualization, Methodology, Software, Investigation, Validation, Formal analysis, Writing - original draft, Writing - review & editing. Jana M. Weber: Conceptualization, Methodology, Software, Investigation, Validation, Formal analysis, Writing - review & editing. Christian Wende: Methodology, Investigation, Software, Validation, Formal analysis, Writing - review & editing. Linus Netze: Methodology, Investigation, Software, Validation, Formal analysis. Alexander Mitsos: Supervision, Resources, Methodology, Validation, Formal analysis, Funding acquisition, Writing - review & editing.
  Artur M. Schweidtmann conceptualized and supervised the bachelor thesis of Christian Wende on this topic. Artur M. Schweidtmann and Jana M. Weber designed the research concept. Jana M. Weber run the persistent homology analyzed the persistent plots, and wrote the corresponding method and result sections. Artur M. Schweidtmann, Christian Wende, and Linus Netze run the optimization and implemented the model in the MeLOn tool.

- Chapter 5 is based on parts of [3].
  Authorship contribution statement: Artur M. Schweidtmann: Conceptualization, Methodology, Investigation, Validation, Formal analysis, Writing - original draft, Writing - review & editing. Jan G. Rittig: Methodology, Software, Investigation, Validation, Formal analysis, Writing - original draft, Writing - review & editing. Andrea König: Investigation, Validation, Formal analysis, Writing - review & editing. Martin Grohe: Validation, Formal analysis, Writing - review & editing. Alexander Mitsos: Supervision, Resources, Validation, Formal analysis, Funding acquisition, Writing - review & editing. Manuel Dahmen: Supervision, Resources, Validation, Formal analysis, Writing - original draft, Writing - review & editing.
  Artur M. Schweidtmann conceptualized and supervised the master thesis of Jan G. Rittig on this topic.

- Chapter 6 is based on a draft of a manuscript to be soon submitted [8].
  Authorship contribution statement: Artur M. Schweidtmann: Conceptualization, Methodology, Investigation, Validation, Formal analysis, Writing - original draft, Writing - review & editing. Jan G. Rittig: Methodology, Software, Investigation, Validation, Formal analysis, Writing - original draft, Writing - review & editing. Jana M. Weber: Validation, Formal analysis, Writing - review & editing. Martin Grohe: Methodology. Manuel Dahmen: Validation, Formal analysis, Writing - review & editing. Kai Lenonhard: Methodology. Alexander Mitsos: Supervision, Resources, Validation, Formal analysis, Funding acquisition, Writing - review & editing.
  Artur M. Schweidtmann conceptualized and supervised the master thesis of Jan G. Rittig on this topic.

Additionally, the author published the articles [9, 10] as first author, [11–14] as equally contributed first author, and [15–33] as coauthor during his time at AVT.SVT, which are not part of this thesis.

While at AVT.SVT, the author (co)-supervised the master theses of Pascal Padberg, Luise Bering, Stephan Pilz, Konstantin Schürholt, Haagen Seele, Hermann Graf von West-

erholt, Pascal Neumann, Philipp Lenz, Xiaopeng Lin, Danimir Doncevic, Fabian Lechtenberg, Jing Cui, Daniel Grothe, Justus Evenschor, Jan Rittig, Tim Kerkenhoff, Stefanie Winkler, and Amedeo Ceruti. In addition, the author (co)-supervised the bachelor theses of Nils Graß, Christoph Offermanns, Jannik Lütchje, Linus Netze, Fabian Hübenthal, Hannah Markgraf, Frederik Erkens, Jesse Rejek, Christian Wende, Laurens Lueg, Ye Wang, Max Theisen, Mathis Heyer, Dominik Goldstein, and Dion Jakobs. Furthermore, the author (co)-supervised the project theses of Sebastian Biesek, Frederik Steufmehl, Christian Wende, Kaveh Allahdin, Katrin Ostendorf, Nico Dirkes, Udo Lennart Faulhaber, Friedrich von Bülow, Tim Faruß, and Laurens Lueg, Leonard Schenk, Lukas Polte, Seyit Battal Temizm Toias Neef. Moreover, the author (co)-supervised the following student assistants: Linus Netze, Maximilian Theise, Christian Wende, Tim Kerkenhoff, Laurens Lueg, Nils Graß, Katrin Ostendorf. Finally, the author (co)-supervised the following international research intern: Alexis Dubs, Shaylin Cetegen, Anna Romanov, Fathima Shabnam. The work of all students is gratefully acknowledged.

Results of the bachelor thesis of Nils Graß has been partially used for the publication [1] and is, consequently, used in Chapter 2. Results of the master thesis of Daniel Grothe and Xiaopeng Lin have been partially used for the publication [2] and are, consequently, used in Chapter 3. Results of the master thesis of Jan G. Rittig has been partially used for the publication [3] and is, consequently, used in Chapter 5. In addition, the master thesis of Jan G. Rittig has been used in Chapter 6. Results of the bachelor thesis of Christian Wende has been partially used for the publication [4] and is, consequently, used in Chapter 4.

Copyrights

Chapter 1

Introduction

1.1 Motivation

The chemical industry currently requires about 10% of the total energy utilized by end users. Almost all of this is obtained from fossil resources and is used as carbon feedstock (58%) or process energy (42%) [34]. Hence, a transformation of the chemical industry to renewable energy and feedstock supply is needed for a sustainable future [34–36]. The chemical industry has already reduced greenhouse gas emissions significantly by increasing efficiency but the potential of current measures is limited and new technologies are required [37]. While the consumption of many fossil resources for process energy can be replaced by renewable technologies, the need for fossil feedstock is expected to make chemical production the largest driver of global oil consumption by 2030 [34]. In addition, renewable resources are fluctuating over time and space, requiring dynamic operation and a new paradigm for the design of flexible plants [36]. Measures to develop new process routes based on renewable resources and to design such flexible plants need to be found now as the chemical industry quickly needs to get ready for an emission-free, all-renewable future. At the same time, the chemical companies are facing increased competition and must ensure optimal operation and short development cycles for new processes.

Facilitating this radical change poses difficulties as conventional methods for process synthesis and operation may not be sufficient. To make optimal decisions in complex environments, models are required. These models cover all scales from molecular to enterprise-level and the environment and typically rely on conservation laws for mass, energy, and momentum as well as, e.g., thermodynamic equations of state. However, the development of physicochemical models is expensive and many phenomena cannot be fully described by tractable models.

Machine learning (ML) has the potential to overcome the aforementioned limitations of mechanistic modeling as ML methods can learn complex behaviors, the model development is cheaper, and can be advantageous for optimization.

1.2 Why is now the right time for machine learning?

Chemical engineering has seen two big waves of ML applications between the 1980s and 2008, i.e., expert systems and (shallow) artificial neural networks (ANNs) (c.f. [38, 39]). These waves had limited impact due to several reasons [38]: First of all, the challenges were lack of data, of data accessibility, of computation power, and of programming environments / paradigms. Secondly, there were more successful emerging technologies for chemical engineering at that time, i.e., mechanistic modeling, optimization, and Nonlinear Model-Predictive Control (NMPC), allowing for significant advances.

Today, we have cheap and powerful computing, easy-to-use programming environments

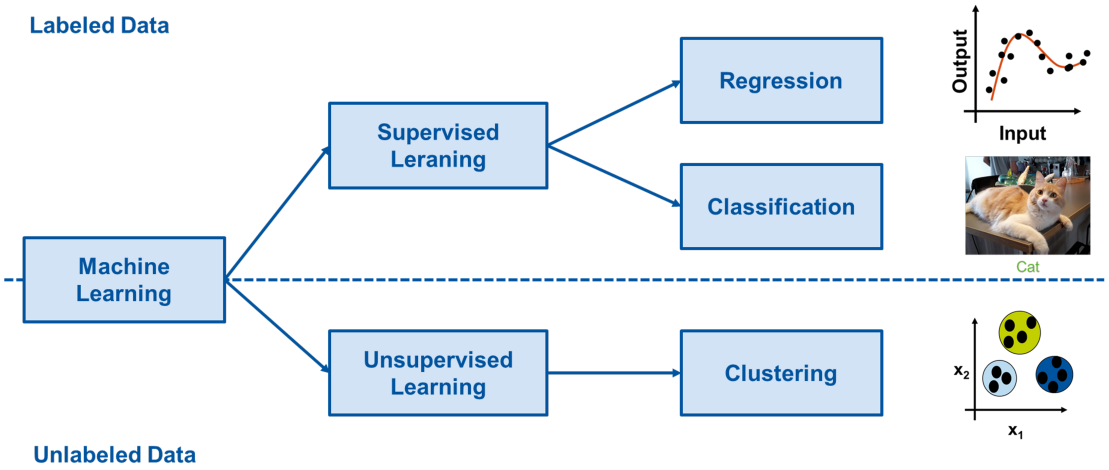


Figure 1.1.: Overview of the general classes of ML methods. ML can be broadly classified into supervised learning and unsupervised learning [42].

(e.g., Python & Tensorflow), and a large open-source community in ML. This led to a technology push in ML and the stunning development of deep learning [40]. In chemical engineering, we currently undergo a transformation towards digitization and full automation of industry and research. This leads to an ever-increasing availability of data and the need for automated optimal decision making based on data, allowing for more sustainable process operations [41]. We thus have a technology push and industry pull situation, where ML opens up new possibilities to overcome pressing challenges in chemical engineering.

1.3 Established machine learning methods in chemical engineering

ML has roots in computer science and mathematics and gives computers the ability to learn without being explicitly programmed. As illustrated in Figure 1.1, ML can be broadly classified into supervised learning and unsupervised learning [42]. Other types of ML are for example reinforcement learning (RL) as well as hybrids such as semi-supervised learning.

First applications of ML methods in chemical engineering were proposed in the 1980s with the advent of expert systems, e.g., for thermophysical properties of complex fluid mixtures [43] and catalyst design [44]. They did not achieve breakthroughs mainly because implementation, training, and maintenance were costly and time-consuming (c.f. Section 1.2). As ML theory, computer hardware, and programming languages advanced, ML was applied in chemical engineering to experimental and simulated data. In the following, we demonstrate that a large part of the initial application of ML techniques in chemical engineering focused on data mining and analytics [45]. Here, ML serves as a computational engine to extract information, to recognize patterns, and to make predictions.

Unsupervised learning describes the collection of techniques that investigate “unlabeled data”, i.e., data with no explicit input-output connection. The main purpose is to find hidden structure in data, e.g., for clustering, feature extraction, compression, or anomaly detection. In chemical engineering, unsupervised learning techniques have been applied to process data for many tasks. Process monitoring and fault detection have seen many

applications over the past decades leading to commercial tools and industrial applications. Process monitoring is mostly based on classical principal component analysis (PCA) [46], while some researchers investigated Independent Component Analysis for non-Gaussian processes [47] and Kernel Density Estimation for applications with unknown distributions, e.g., for data smoothing [48]. Further advances are monitoring platforms, e.g., using Self-Organizing Maps for a wastewater treatment plant [49] and Gaussian Mixture Models for the Tennessee Eastman process [50].

Fault detection is another common application of unsupervised learning to process data. In the previous literature, variations of PCA have been used frequently for fault detection [51–53]. Furthermore, other advanced methods have been applied to distinguish between normal and faulty batches (e.g., Support Vector Data Description [54] and K-Means Clustering [55]). Today, first fault detection tools are commercially available for the process industry [56]. BP & GE monitor the performance of oil-wells to optimize production with an estimated gain of up to $15 trillion [57].

Supervised learning methods are trained on labeled data with an explicit input-output structure and predict their relationship. Regression is a supervised ML tool that is part of the standard repertoire in Process Systems Engineering (PSE) and has long been used for modeling and subsequent optimal design of processes. Regarding soft sensor applications, i.e., online prediction of process qualities, a large variety of different methods has been used including Partial Least Squares [58–60], Principal Component Regression [61, 62], Support Vector Machines (SVMs) [63], ANNs [64], and Gaussian Process (GP) Regression [65–67]. Applications of these include complex large-scale processes such as air separation units [59], injection-molding [61], reverse osmosis of seawater [60], and further chemical production processes [64, 65, 68].

Overall, applications of ML methods for process monitoring, fault detection, and soft sensing are mostly mature and commercially available in chemical engineering.

1.4 Emerging machine learning areas in chemical engineering

Beyond the previously summarized topics deeply rooted in data mining and analytics, we have identified six emerging areas of ML which contribute to the necessary transition of the chemical industry [7]. This dissertation contributes to four of the six emerging areas directly (c.f. Section 1.5).

1.4.1 #1 Optimal decision making

Optimal decision making is a very prominent topic in chemical engineering and PSE with numerous examples including optimal process synthesis, optimal solvent, catalyst, and adsorbent selection, optimal equipment design, optimal design of control structures for processes, as well as optimal control and operation of processes. All of these decisions need to be made based on existing information which can be in the form of data and mechanistic knowledge, e.g., models. As shown in Figure 1.2, optimal decision making based on data requires the training of data-driven models and subsequent optimization with embedded data-driven models [69].

In previous PSE literature, a large variety of data-driven models has been used for re-

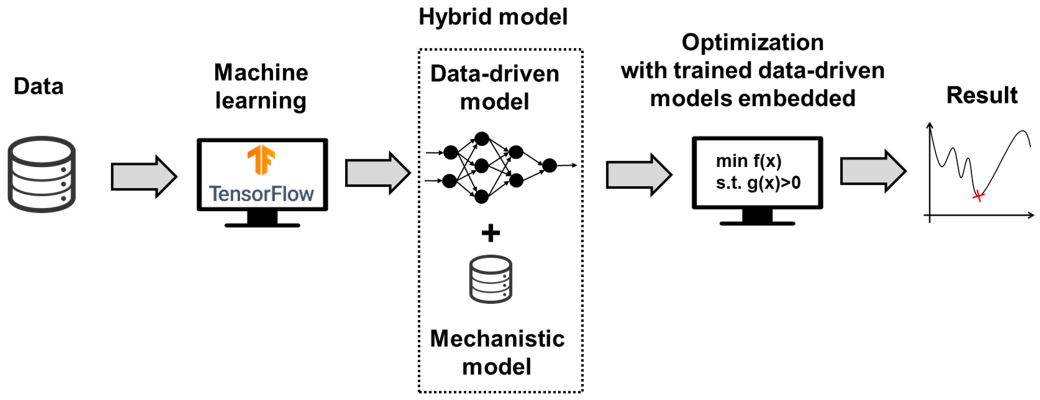


Figure 1.2.: Illustration of the data driven modeling and optimization approach.

gression and subsequent process optimization. As illustrated in Figure 1.3, the complexity of applied data-driven models ranges from linear approximations to deep ANNs. For a long time, the literature focused on linear models to approximate simulation and experimental data [70]. Since the 1990s, shallow ANNs have been used extensively. Shallow ANNs can approximate any nonlinear, smooth function given a sufficient number of neurons to any given positive accuracy [71]. In many PSE applications, ANNs are fitted to complete processes (black-box approach, e.g., [72]) or ANNs are combined with mechanistic model equations (hybrid modeling approach, e.g., [73–75]). Subsequently, the obtained process models can be optimized, e.g., to identify process design [76]. Today, the re-emergence of ML is mostly driven by deep ANNs and big data [40]. The deep ANNs are thought to become more important in PSE because abundant data becomes available, e.g., through smart manufacturing, high throughput experiments, and simulation studies [41]. However, applications of deep ANNs are still limited in the area of process design in PSE [39].

While optimization of problems with linear models can be solved globally on a large scale, e.g., for structural optimization [70], linear models cannot learn high dimensional nonlinear problems accurately. On the other hand, the consideration of more complex data-driven models like ANNs and GPs, which could reflect high dimensional nonlinear problems better, has long been limited to local or stochastic solution approaches [74–78]. Although there are some tailor-made data-driven models that can be solved using state-of-the-art global solvers, these are also limited to low-dimensional problems [79, 80]. Trying to overcome this issue, a few researchers in chemical engineering and ML developed tailormade optimization approaches for problems with ML models embedded: [81] proposed a tailored algorithm for problems with gradient boosted trees embedded. [82] proposed an algorithm for the optimization of piecewise polynomial functions. Similarly, [83] also worked on global optimization of spline functions. Some previous works also used general-purpose global solver to solve optimization problems with complex surrogate models embedded [77, 78]. However, such approaches have mostly been limited by computational burdens. Notably, ANNs with rectified linear unit (ReLU) have recently been reformulated as mixed-integer linear programs (MILPs) through big-M reformulations [84–86]. Similarly, tree models can be reformulated as MILPs [81, 87, 88]. However, the number of integer variables and constraints grows linearly with the model complexity for these formulations.

Overall, emerging area #1 has already seen some work on optimal process synthesis

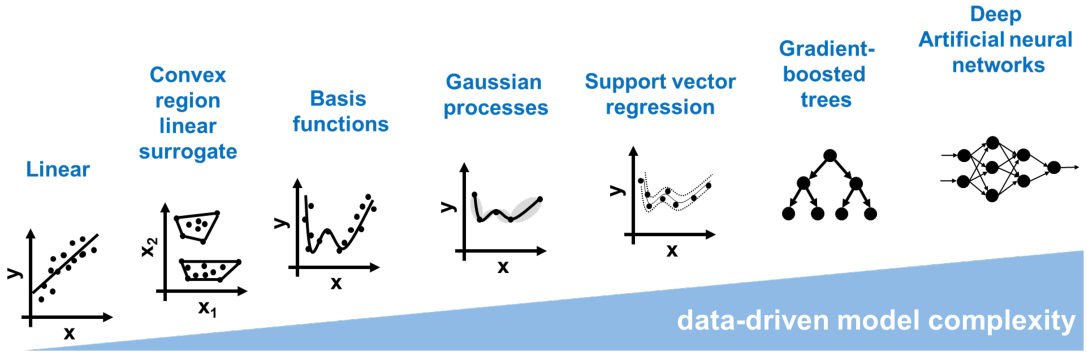


Figure 1.3.: Overview of data-driven models embedded in optimization problems in chemical engineering ordered by the models' abilities to learn complex dependencies: linear [70], convex region linear [79], nonlinear basis functions, e.g., ALAMO [80], piecewise polynomial function [82], spline function [83], Gaussian process regression [77, 78], gradient boosted trees [81, 87], ANNs with ReLU activation [84–86], and other ANNs with more complex activation functions [1, 77].

and optimal process operation. However, there are still severe limitations when it comes to the solution of integrated learning and optimization frameworks that exhibit complex ML models such as ANNs or GPs. In this dissertation, we address this challenge by developing optimization problem formulations that accelerate deterministic global optimization with complex ML models embedded by orders of magnitude.

1.4.2 #2 Introducing and enforcing physics in machine learning

Introducing and enforcing physics in ML is a frequently called-for development and often voiced by chemical engineers. Across many disciplines, supervised learning is directly applied to the input-output data set in a black-box approach. However, black-box approaches have severe drawbacks with respect to interpretability, extrapolation, data demand, and reliability. These drawbacks can limit applications of ML and can even lead to fatal errors when being applied to industry without necessary checks. Mechanistic, physicochemical models that are a priori known on the other hand can provide structural knowledge that can be combined with data-driven models.

The combination of mechanistic and data-driven models is called hybrid (semi-parametric) modeling and promises advantages such as better interpretability, enhanced extrapolation properties, and higher prediction accuracy (Figure 1.4) [89–91]. It has numerous applications in chemical engineering and biotechnology since the early 1990s, e.g., for process [92] and reactor modeling [93], polymerization [94], crystallization, distillation, drying processes [95], and process control [29, 96]. Also, many empirical constitutive equations in chemical engineering can be interpreted as simple data-driven parts in a hybrid model. Hybrid modeling has strong theoretical foundations within the chemical engineering community and is believed to gain importance within and beyond chemical engineering: [73, 97] provided fundamental insight into the identification of hybrid models and extrapolation properties. Furthermore, [98] developed methods for determining a valid input domain for hybrid models.

The ML community has identified the need for the incorporation of a priori knowledge

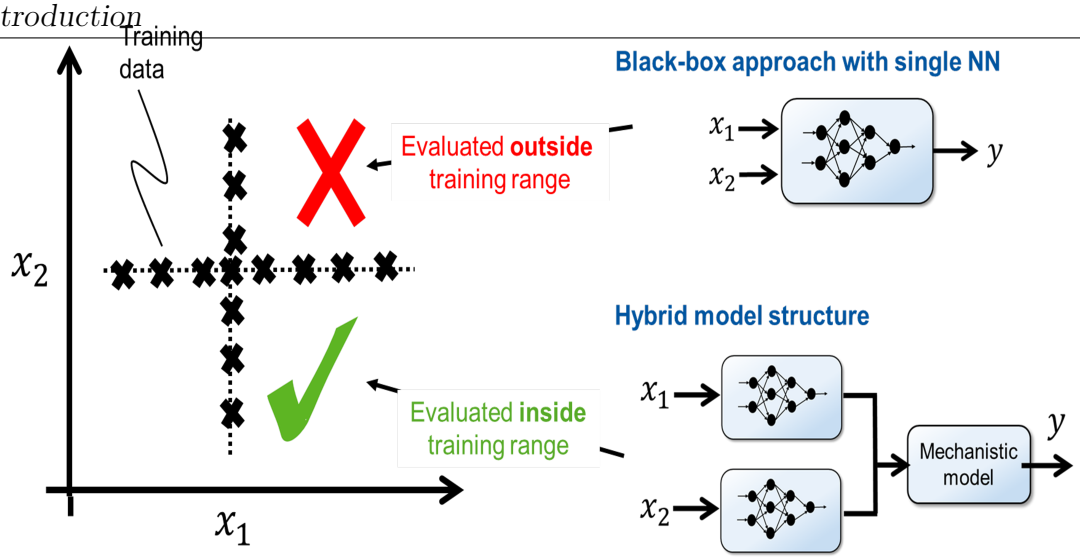


Figure 1.4.: Comparison of hybrid model structure and black-box modeling approach. The figure is adapted from a lecture of Andreas Schuppert on hybrid modeling.

for many applications. For instance, researchers in the ML community incorporate prior knowledge as a penalty term in the training and thus enforce physics-informed ANN [99]. For example, [100] includes known operators in ANNs, which corresponds essentially to a hybrid model. A few works also aim to extract physical knowledge from data or data-driven models. For instance, symbolic regression was used to identify physical laws from kinematic data [101]. Interestingly, advanced optimization formulations for symbolic regression have been developed [102] and applied in chemical engineering [103].

Overall, chemical engineering is experienced in physicochemical modeling and formulating predictive models. Exploiting these capabilities for hybrid modeling is promising to ensure interpretability, extrapolation, reliability, and trust of ML models. In this dissertation, we employ hybrid models for the optimization of chemical and energy processes. In addition, we introduce physical knowledge to deep learning models for molecular property prediction.

1.4.3 #3 Information and knowledge representation

The great surge of ML applications in social media platforms, online shopping, and video on-demand services mostly stems from the great accessibility of huge amounts of structured data. In chemical engineering, however, only a tiny fraction of knowledge and information is accessible for ML methods while the majority is only available in analog or non-standardized digital form. While data from computations, sensors, and measurements can be processed by ML techniques, molecular data, process flow charts, P&IDs, publications, lab books, etc. are not often accessible to standard ML techniques. This is a major hurdle for finding and exploiting more complex relationships by ML techniques.

To overcome this hurdle, specialized representations of chemical engineering information and knowledge are required. For a long time, researches have designed manual features to describe data. For example, molecules can be described through molecular counts or group contribution methods [104, 105]. However, this manual feature design requires expert knowledge and time. In addition, the manual selection of features can lead to a model bias.

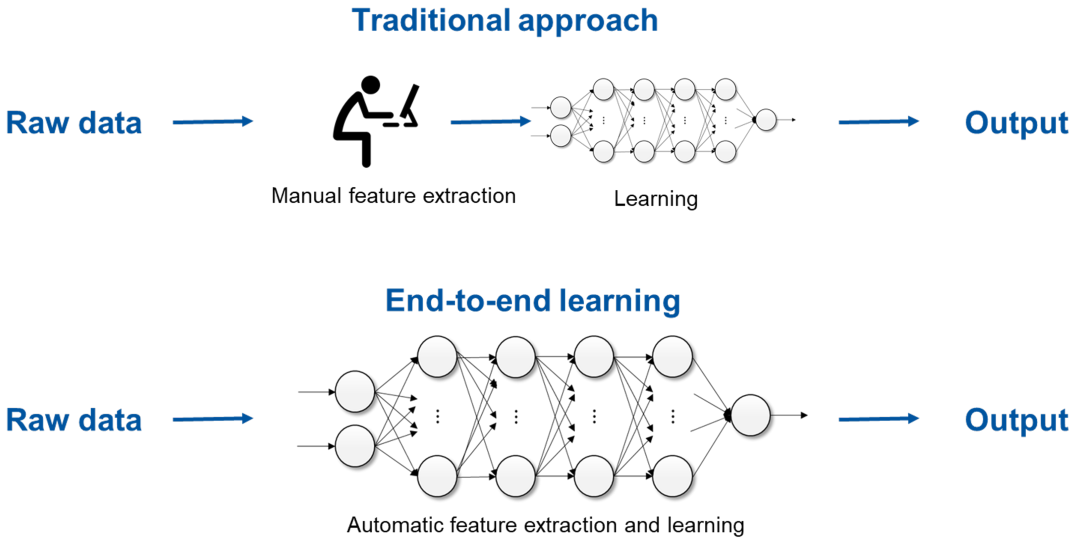


Figure 1.5.: Illustration of the concept of end-to-end learning in comparison to manual feature extraction.

A promising solution is end-to-end learning where gradient-based learning is applied on a complete system from the information representation to the output with differentiability of all model parts [40]. This has lead to breakthrough results in many complex applications including self-driving cars [106] and speech recognition [107]. Recently, molecules and crystals have be represented as graphs and processed by specialized ML algorithms for end-to-end learning [108, 109]. In particular, a ML method called Graph Neural Networks (GNNs) has shown promising results for the prediction of structure-property relationships. GNNs are a type of model that directly operates on graph structure [110]. For instance, GNNs can utilize a graph representation of molecules, where atoms correspond to nodes and bonds to edges containing information about the molecular structure. More specifically, GNNs learn physicochemical properties as a function of the molecular graph in a supervised learning setup using a backpropagation algorithm. As illustrated in Figure 1.5, the end-to-end learning approach eliminates the need for manual feature selection. In particular, the GNN learns optimal molecular fingerprints through graph convolutions and maps the fingerprints to the physicochemical properties by deep learning. [111] applies GNNs for the prediction of quantitative structure-activity and property relationships, e.g., octanol solubility, aqueous solubility, melting point, and toxicity. Further, [112] represent crystal structures by a crystal graph that encodes atomic information and bonding interactions for the prediction of target properties, e.g., formation energy and absolute energy. In addition to these applications, GNNs have also been extended to recognize higher-dimensional features from graphs [113]. There have also been some first efforts to represent reaction networks [114, 115] and flowsheets [116] as graphs and apply ML to this data. Furthermore, researchers in reaction engineering recently developed specific knowledge representations (e.g., Chemputer [117]) to allow for data processing and usage of ML techniques. [38] names models for information representation as promising building blocks to further advance the field of chemical engineering.

Overall, we use only a tiny part of the available information in chemical engineering for computer-aided decision making because the remaining information is inaccessible by

conventional methods. To properly exploit the large scope of data available in chemical engineering, it needs to be structured and represented in a machine-readable form. In this dissertation, we employ advanced GNNs for the end-to-end learning of physicochemical properties based on the molecular graphs.

1.4.4 #4 Heterogeneity of data

Heterogeneity of data has a multitude of sources in chemical engineering (e.g., lab books, measurements, property data, publications, simulation files, flowsheets) and processing heterogeneous data is a major hurdle of many chemical engineering topics, while ML is famous for its handling of large data sets and combination of heterogeneous data. Heterogeneity in chemical engineering stems from (1) multiple scales in time and space, which can also cross several orders of magnitude: e.g., milliseconds to month in control and scheduling, nm to m in pore diffusion and pressure swing adsorption; (2) a variety of data sources, which need to be combined to understand chemical processes, e.g., process data, alarms, property data, equipment specifications; (3) highly different data frequencies; which can be present in data in chemical engineering, e.g., continuous measurement data appears once per ms, while quality data is gathered every other hour or day. All of this is exacerbated by the frequent high dimensionality of data sets in chemical engineering [45].

Unsupervised learning has emerged from ML for treating high-dimensional problems and perform dimensionality reduction in chemical engineering. Recently, an outlier detection algorithm identified strategic molecules for circular supply chains within the "network of organic chemistry" with roughly one million reactions [114]. Breaking down the dimensionality makes huge networks accessible to reaction pathway optimization methods [32, 118–120]. Furthermore, [23] use PCA to design features from a set of molecular descriptors for solvent selection.

To tackle highly different data frequencies or data, which is erroneous or incomplete, [45] suggest to use semi-supervised learning. It can be employed in case of a mismatch between input and output data, e.g., when a data basis consists of labeled and unlabeled data. A small amount of labeled data can be augmented by a larger unlabeled data set [121]. First preliminary applications in chemical engineering use semi-supervised learning for the prediction of missing data for a soft sensor in a penicillin production [122].

When working with experimental and industry data, the success of ML is highly dependent on data quality [123]. To this end, [45] highlight issues in chemical engineering, such as missing data imputation, outlier detection, issues caused by different sampling rates, and the consideration of noise from various measurement devices. These issues are mostly not present during the training of surrogate models on simulated data, but arise when using experimental data.

Overall, the heterogeneity of data is a central challenge in chemical engineering. Despite the aforementioned advances, much needs to be done especially in the field of semi-supervised learning. In this dissertation, the heterogeneity of data is not primarily addressed because the focus of this thesis lies in modeling and optimization.

1.4.5 #5 Safety and trust in machine learning applications

Safety and trust in ML applications are related to the call for the introduction of physical laws into ML techniques but goes well beyond. One example is the validity domain of a

data-driven model. It is well-known that data-driven models cannot extrapolate over their initial training domain. However, when training data-driven models on industry data, defining a validity domain is a major issue. In particular, the data can include holes or can be distributed in separate clusters. Here, standard methods to determine the validity domain such as the convex hull criterion are not suitable. In practical chemical engineering applications, a failure could amount to a runaway reaction causing damage to equipment. Similar issues can arise when applying GNNs to molecular property prediction or when applying Reinforcement Learning (RL) to control chemical processes [124, 125]. Overcoming this hurdle is a relevant issue, where ML and chemical engineering together can generate considerable added value in terms of research and which would open up a huge area for new applications of ML in chemical engineering.

Overall, these developments are of great importance for chemical engineering where safety and trust are decisive criteria for the application of new technology in the industry. In this dissertation, we model the validity domain of ML models through one-class classifiers and obey validity limits in optimization. Thus, this dissertation directly contributes towards this emerging area.

1.4.6 #6 Creativity

Creativity is a feature that ML has become quite famous for (e.g., for creating pictures in artistic styles [126]). Some examples are generating new texts, new images, and new sounds. In chemical engineering, a lot of effort using non-ML techniques is currently going into inverse problems, e.g., finding a new catalyst or a novel solvent for a given application [127]. Also, deriving new process structures or new control structures, which no human ever thought of, is a frequently desired goal. In ML, matrix completion is a semi-supervised technique that generated a lot of attention by its large-scale application for the "Netflix problem" [128]. Here, predictions are made for non-rated movies based on a large (and sparse) matrix of viewers and ratings. Recently, [129] applied this technique to the prediction of activity coefficients for component mixtures, which were never experimentally investigated.

Further progress came with generative adversarial networks (GANs), which are deep neural network architectures consisting of two nets competing against one another ("adversarial") [130]. GANs quickly proved to be highly capable at creative tasks, e.g., in creating new works of art. Another type of promising generative models are Variational Autoencoder (VAE) [131], which are especially used for data generation (image, sound, text) and missing data imputation.

Overall, chemical engineering is a field where novel process designs, products, and materials can currently only be found or discovered by experimental trial and error or by human design. Supporting this with creative techniques from ML might allow for new discoveries as of yet unimaginable and research in this direction is hence highly desirable. In this dissertation, we do not focus on creativity of ML method directly. However, our methods are a first step towards the creative use of ML methods in chemical engineering. For examples, generative models can be combined with our proposed GNN model for molecular design.

1.5 Dissertation outline

We have outlined emerging areas within ML and chemical engineering that can generate considerable added value to solve chemical engineering's pressing issues. In the following, we will detail how this dissertation achieved progress in some of these areas. In particular, this dissertation focuses on the modeling and optimization of chemical engineering problems. Note that the interdisciplinary work in the area is emerging and the dissertation can by no means address all challenges that were outlined in the previous sections. Rather, we consider our work as one contribution to this interdisciplinary research effort. We give an overview of the chapters of this dissertation. The individual chapters are independent and each includes an introduction, a methodology, numerical results, and a conclusion.

In Chapter 2, we propose an efficient method for deterministic global optimization of optimization problems with ANNs embedded. The proposed method is based on relaxations of algorithms using McCormick relaxations in a reduced-space employing the convex and concave envelopes of nonlinear activation functions. The branch-and-bound solver branches only on the free variables and McCormick relaxations are propagated through explicit ANN models. The approach also leads to significantly smaller and computationally cheaper subproblems for lower and upper bounding. The optimization problem is solved using our in-house deterministic global solver **M**cCormick based **A**lgorithm for mixed **i**nteger **N**onlinear **G**lobal **O**ptimization (MAiNGO). The results show that computational solution time is favorable compared to a state-of-the-art global general-purpose optimization solver. The method is ready-to-use and available open-source as part of our "MeLOn - **M**achin**e** **L**earning Models for **O**ptimizatio**n**" toolbox [132]. The methods described in this chapter directly contribute to the emerging area #1 Optimal decision making. Indirectly, this chapter also contributes towards #5 Safety and trust in machine learning applications because the described algorithms are the basis for advanced solution approaches that can consider trust and safety in their formulations (e.g., stochastic programming formulations).

We have applied global optimization with ANNs embedded in a reduced-space to a number of case studies that are only partly described in this dissertation. In Chapter 2, we briefly summarize these applications of our developed method. First, the performance of the proposed method is shown in four optimization examples: an illustrative function, a fermentation process, a compressor plant, and a chemical process. Then, we briefly discuss the aggregation of hybrid process models and superstructure problems based on simulation data from specialized software for organic Rankine cycles. Herein, we substituted equations of state using ANNs for global process optimization. The surrogate models show very good accuracy and accelerate optimization. Moreover, the approach is promising to integrate experimental data via ANNs in global optimization. We used a multi-objective optimization approach to identify design fabrication parameters of ion separation membranes and separation process structures. Moreover, we briefly present applications for scheduling problems and nonlinear model predictive control. As we combine mechanistic model equations with data-driven models, this directly contributes towards #2 Introducing and enforcing physics in machine learning. Indirectly, this also contributes towards #5 Safety and trust in machine learning applications because the hybrid models allow for safe model evaluations.

In Chapter 3, we propose a reduced-space formulation for deterministic global optimization with trained GPs embedded. GPs (Kriging) are interpolating data-driven models that

are frequently applied in various disciplines. Often, GPs are trained on datasets and are subsequently embedded as surrogate models in optimization problems. These optimization problems are nonconvex and global optimization is desired. However, previous literature observed computational burdens limiting deterministic global optimization to GPs trained on few data points. In the proposed approach, we develop a reduced-space formulation for optimization problems with GPs embedded. To further accelerate convergence, we derive envelopes of common covariance functions for GPs and tight relaxations of acquisition functions used in Bayesian optimization including expected improvement, probability of improvement, and lower confidence bound. In total, we reduce computational time by orders of magnitude compared to state-of-the-art methods, thus overcoming previous computational burdens. We demonstrate the performance and scaling of the proposed method and apply it to Bayesian optimization with global optimization of the acquisition function and chance-constrained programming. The method is ready-to-use and available open-source as part of our MeLOn toolbox [132]. The methods described in this chapter directly contribute to the emerging area #1 Optimal decision making.

In Chapter 4, we propose a method to learn this validity domain and encode it as constraints in process optimization. Data-driven models are becoming increasingly popular in engineering, on their own or in combination with mechanistic models. Commonly, the trained models are subsequently used in model-based optimization of design and/or operation of processes. Thus, it is critical to ensure that data-driven models are not evaluated outside their validity domain during process optimization. We first perform a topological data analysis using persistent homology identifying potential holes or separated clusters in the training data. In case clusters or holes are identified, we train a one-class classifier, i.e., a one-class support vector machine, on the training data domain and encode it as constraints in the subsequent process optimization. Otherwise, we construct the convex hull of the data and encode it as constraints. We finally perform deterministic global process optimization with the data-driven models subject to their respective validity constraints. To ensure computational tractability, we develop a reduced-space formulation for trained one-class support vector machines and show that our formulation outperforms common full-space formulations by three orders of magnitude, making it a viable tool for engineering applications. The method is ready-to-use and available open-source as part of our MeLOn toolbox [132]. The methods described in this chapter directly contribute to the emerging area #5 Safety and trust in machine learning applications.

Chapters 2, 3, and 4 use standard ML models to learn from real-valued data. However, as disused in Section 1.4, chemical engineering data is often heterogeneous and not accessible for standard ML methods. For example, the prediction of combustion-related properties of (oxygenated) hydrocarbons is an important but challenging task because the input data comes in the form of molecules. In Chapter 5, we develop GNN models for predicting three fuel ignition quality indicators, i.e., the derived cetane number (DCN), the research octane number (RON), and the motor octane number (MON), of oxygenated and non-oxygenated hydrocarbons. In light of limited experimental data in the order of hundreds, we propose a combination of multi-task learning, transfer learning, and ensemble learning. The results show the competitive performance of the proposed GNN approach compared to state-of-the-art QSPR models making it a promising field for future research. The prediction tool is available via a web front-end at `www.avt.rwth-aachen.de/gnn`. The methods described in this chapter directly contribute to the emerging area #3 Information and knowledge representation.

In Chapter 6, we derive meaningful GNN architectures based on physical knowledge about the learned properties. A key element of GNNs is the pooling function where atom feature vectors are combined into molecular fingerprints. Most previous works use a standard pooling function to predict a variety of properties. However, unsuitable pooling functions can lead to unphysical GNNs that learn through inherent overfitting and that cannot generalize. The impact of physical pooling functions is demonstrated for quantum mechanical properties of molecules and is compared with standard pooling functions and the recent set2set pooling approach. We recommend using sum-pooling for the prediction of properties that are molecular size-dependent and show that physical pooling functions greatly reduce overfitting and enhance generalization. The methods described in this chapter directly contribute to the emerging area #2 Introducing and enforcing physics in machine learning. Also, the methods contribute indirectly to #3 Information and knowledge representation and #5 Safety and trust in machine learning applications.

Finally, Chapter 7 summarizes the key results of the dissertation and outlines a few avenues for future research and applications.

Chapter 2

Deterministic global optimization with artificial neural networks embedded

2.1 Introduction

A feed-forward artificial neural network (ANN) with one hidden layer can approximate any smooth function on a compact domain to an arbitrary degree of accuracy given a sufficient number of neurons [71]. Thus, ANNs are said to be universal approximators. Due to their excellent approximation properties, ANNs have attracted widespread interest in various fields such as chemistry [133], chemical engineering [134], pharmaceutical research [135] and biosorption processes [136] for the black-box modeling of complex systems. Furthermore, various industry applications of ANNs exist, e.g., modeling and identification, optimization and process control [137].

In many applications, a model including ANNs is developed and then used for optimization. In biochemical engineering for example, ANNs are commonly used to model fermentation processes [138] and optimize their output to identify promising operating points for further experiments [139–141]. ANNs are also used as surrogate models for the optimization of chemical processes [72, 74, 75, 142–153].

Most of the previous publications on optimizations with ANNs embedded rely on local optimization techniques. Fernandes [153], for instance, optimized the product concentrations of a Fischer-Tropsch synthesis using the quasi-Newton method and finite-differences. Other authors used local solvers like DICOPT [154] (e.g., [74, 75, 145, 147]), sequential quadratic programming (e.g., [146]) or CONOPT [155] (e.g., [147]). These local methods can yield suboptimal solutions when the learned input-output relation is multi-modal. In addition, the activations functions involved in ANNs are usually nonconvex. A few researchers have addressed this problem by using stochastic global methods such as genetic algorithms (e.g., [139–141, 150, 151, 156]) or brute-force grid search (e.g., [142–144]). However, these methods cannot guarantee global optimality. Deterministic global optimization of an ANN embedded problem was carried out by Smith *et al.* (2013) [72] who used BARON [157] to optimize an ANN with one hidden layer and three neurons that emulates a flooded bed algae bioreactor. As the hyperbolic tangent activation function is currently not available as an intrinsic function with its corresponding envelopes in BARON, Smith *et al.* (2013) [72] used an algebraic formulation including exponential functions which leads to weaker relaxations (cf. Section 2.3). In 2009, de Weerdt *et al.* [158] optimized the output of randomly generated ANNs with one hidden layer and up to 128 neurons using interval analysis [159]. The work considered box-constrained problems with up to three degrees of freedom (DoF). Henao argues in his Ph.D. thesis [147] that with state-of-the-art global optimization solvers such as BARON at the time (i.e., 2012) only small to medium sized problems could be solved. He suggests a piecewise linear approximation of the hy-

perbolic tangent function. The literature shows that there remains a need for an efficient deterministic global optimization algorithm of problems with ANNs embedded.

In deterministic global optimization, convex relaxations are derived and relaxed problems are solved using branch-and-bound (B&B) or related algorithms [157, 160]. In most state-of-the-art deterministic global optimization solvers such as BARON [157], ANTIGONE [160] and SCIP [161] the complete set of constraints and variables is handed to the optimization algorithm that builds convex relaxations. This method is referred to as the full-space (FS) formulation. Another possibility is the reduced-space (RS) formulation. The key idea thereof is to hide some optimization variables and constraints from the optimizer [162–166]. Note that a discussion about similarities between RS formulation and selective branching can be found in Bongartz and Mitsos (2017) [165].

For the deterministic global optimization of ANNs, the FS formulation inherently results in a large-scale optimization problem because the ANN network structure includes multiple (hidden) layers, neurons and network equations. Additionally, auxiliary variables used in state-of-the-art solvers for the relaxations increase the problem size. Solving large-scale optimization problems globally is difficult because of the exponential worst-case runtime of the B&B algorithm. Moreover, in standard solvers the user has to provide tight variable bounds, which are difficult to provide for some network variables. A possible disadvantage of the RS formulation is that the propagation of relaxations through large ANNs may result in weak relaxations.

One possibility to construct relaxations in the RS setup are McCormick relaxations [167] that have shown favorable convergence properties [168, 169] and have been extended to the relaxation of multivariate functions [170, 171]. Recently, a differentiable modification of the relaxations has been proposed as well [172, 173]. The global optimization using McCormick relaxations has been applied to several problems such as flowsheet optimization [165, 166, 174] and the solution of ODE and nonlinear equation systems [164, 175, 176].

In this contribution, ANNs are optimized in a RS, which significantly reduces the dimensionality of the optimization problems. More specifically, the equations that describe the ANN and the corresponding variables are hidden from the B&B solver. In order to construct tight convex and concave relaxations of the ANNs, we utilize the convex and concave envelopes of the activation function (shown in Appendix A.1) and automatic construction of McCormick relaxations [163].

In the remainder of this chapter, first an overview on multilayer perceptron ANNs is provided (Section 2.2). In Section 2.3, the optimization problem formulation and the relaxation of ANNs are proposed. Further, implementation details are provided. In Section 2.4, the proposed method is applied to four numerical examples and compared to the optimization algorithm BARON. In Section 2.6, advantages, limitations and possible applications of the proposed method are discussed.

2.2 Background on multilayer perceptrons

In this section, the multilayer perceptron (MLP), also known as feed-forward ANN, is briefly introduced (see also, e.g., [42]). As depicted in Figure 2.1, the MLP can be illustrated as a directed acyclic graph connecting N layers, numbered from $k = 1$ to N. Layer 1 is imaginary and corresponds to the input of the MLP, whereas the last layer ($k = N$) corresponds to its outputs. The layers $k = 2, .., N - 1$ are the hidden layers. Each layer

Table 2.1.: Mathematical notation for MLPs indicating constant parameters (Param.) and variables (Var.) for both training and optimization problems with MLPs embedded (Emb. Opt.).

Symbol		Training	Emb. Opt.	Meaning
$\mathbf{b}^{(k-1)}$		Var.	Param.	Bias vectors
$\mathbf{W}^{(k)}$		Var.	Param.	Weight matrices
$\mathbf{z}^{(k)}$	$k = 2, ...N-1$	Var.	Var.	Output of the hidden layers
$\mathbf{z}^{(k)}$	$k = 1$	Param.	Var.	Input vector
$\mathbf{z}^{(k)}$	$k = N$	Var.	Var.	Output vector

consists of $D^{(k)}$ neurons.

Let $\mathbf{z}^{(k)} \in \mathbb{R}^{D^{(k)}}$ be the output vector of layer k. Thus, $\mathbf{z}^{(1)}$ is the input vector and $\mathbf{z}^{(N)}$ is the output vector of the MLP. For each layer $k \geqslant 2$, the vector $\mathbf{z}^{(k)}$ is given by

$$\mathbf{z}^{(k)} = h_k\left(\mathbf{W}^{(k-1)}\mathbf{z}^{(k-1)} + \mathbf{b}^{(k-1)}\right) \tag{2.1}$$

where $h_k(\cdot)$ is an activation function, $\mathbf{W}^{(k-1)}$ is a weight matrix and $\mathbf{b}^{(k-1)} \in \mathbb{R}^{D^{(k)}}$ a bias vector. In this chapter, we mainly focus on optimization with (already trained) MLPs embedded. In this case, all weights ($\mathbf{W}^{(k)}$) and bias vectors ($\mathbf{b}^{(k)}$) are given constant parameters (cf. Table 2.1).

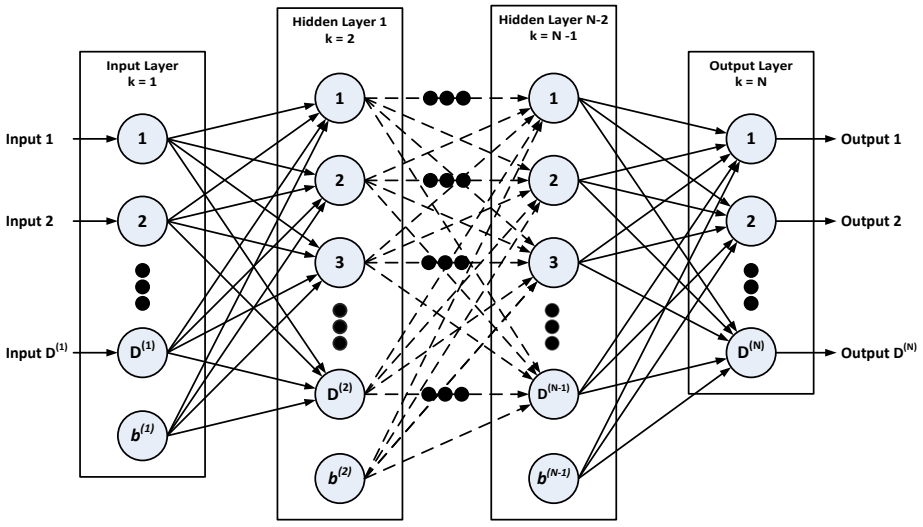


Figure 2.1.: Graphical illustration of a multilayer perceptron artificial neural network with N layers as a directed acyclic graph. Herein, layer 1 corresponds to the input layer and layer N corresponds to the output layer. The layers 2 to N-1 are the hidden layers. The illustrated network has a fully connected feed-forward architecture where all neurons of a layer, k, are connected to all neurons of the following layer, k+1.

A commonly-used activation function of the hidden layers is the hyperbolic tangent function ($h_k(x) = \tanh(x)$) (componentwise) which is discussed herein. The proposed method can be easily applied to other common activation functions like the logistic function.

2.3 Method

In this section, the optimization problem formulations and the McCormick relaxation of MLPs are proposed. Further, details about the implementation are provided.

2.3.1 Optimization problem formulations

Optimization problems, which use MLPs as surrogate models typically have a particular structure. Usually, one or more of the input variables of the MLPs correspond to (scaled) degrees of freedom of the problem ($\mathbf{x}$) and can be chosen within given bounds $D = [\mathbf{x}^L, \mathbf{x}^U]$. Once the input variables are fixed, the dependent variables in the networks, $\mathbf{z}$, can be determined by solving the nonlinear network equations $\mathbf{h}(\mathbf{x}, \mathbf{z}) = \mathbf{0}, \quad \mathbf{h} : D \times \mathbb{R}^{n_z} \to \mathbb{R}^{n_z}$, i.e., Equations (2.1). The outputs $\mathbf{y}$ of the networks can be understood as network variables of the output layers. Thus, the output variables are a subset of the network variables with $n_y < n_z$. The objective function, $f(\mathbf{x}, \mathbf{y}), \quad f : D \times \mathbb{R}^{n_y} \to \mathbb{R}$, usually depends on the networks inputs and outputs but not on the remaining network variables. Similarly, the feasible region is often constrained by inequalities $\mathbf{g}(\mathbf{x}, \mathbf{y}) \leqslant \mathbf{0}, \quad \mathbf{g} : D \times \mathbb{R}^{n_y} \to \mathbb{R}^{n_g}$ which also depend only on the network inputs and outputs.

2.3.1.1 Full-space formulation

In the FS formulation, the input and the network variables are optimization variables and the problem can be formulated as follows:

$$\begin{aligned} \min_{\mathbf{x}\in D, \mathbf{z}\in Z} \quad & f(\mathbf{x}, \mathbf{y}) \\ \text{s.t.} \quad & \mathbf{h}(\mathbf{x}, \mathbf{z}) = \mathbf{0}, \qquad \mathbf{g}(\mathbf{x}, \mathbf{y}) \leqslant \mathbf{0} \end{aligned} \tag{FS}$$

In this case, a global B&B solver requires bounds on $\mathbf{x}$ and $\mathbf{z}$. The FS formulation is utilized by common general-purpose deterministic global solvers.

2.3.1.2 Reduced-space formulation

In MLPs, the network equations ($\mathbf{h}(\mathbf{x}, \mathbf{z})$) can be solved explicitly for one layer after another. Thus, the network equations can be used to substitute the network variables ($\mathbf{z}$) in the FS optimization formulation (cf. [165]):

$$\begin{aligned} \min_{\mathbf{x}\in D} \quad & f(\mathbf{x}, \hat{\mathbf{y}}(\mathbf{x})) \\ \text{s.t.} \quad & \mathbf{g}(\mathbf{x}, \hat{\mathbf{y}}(\mathbf{x})) \leqslant \mathbf{0} \end{aligned} \tag{RS}$$

The RS formulation operates only in the domain D of the optimization variables $\mathbf{x}$. Further, the equality constraints, which describe the network equations, are no longer visible

to the optimization algorithm. This is possible when the equality constraints can be formulated explicitly as is the case for MLPs. However, the complete substitution of equality constraints is not in general possible when optimizing arrangements of several MLPs or hybrid modeling formulations. In these cases, some equality constraints and additional variables might remain in the RS formulation (e.g., for connecting different MLPs with a recycle) similar to, e.g., [165]. As an option, these can also be relaxed using extensions of the McCormick approach to implicit functions [175, 176].

2.3.2 Relaxations of artificial neural networks

In order to solve the RS optimization problem using a B&B algorithm, lower bounds of the MLPs have to be derived. Herein, the lower bounds are calculated by automatic propagation of McCormick relaxations and natural interval extensions [159]. As described in Section 2.2, the only nonlinearities of MLPs are their activation functions, e.g., hyperbolic tangent functions. Thus, the tightness of MLP relaxations is considerably influenced by the relaxations of the hyperbolic tangent function.

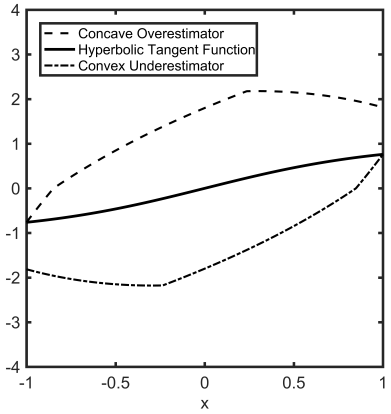
In general, McCormick relaxations do not provide the envelopes. The envelopes of the hyperbolic tangent function are derived in Appendix A.1. They are once continuously differentiable (Proposition A.1) and strictly monotonically increasing (Proposition A.3). In the proposed framework, the envelopes of the hyperbolic tangent activation function are propagated through the network equations to derive concave and convex relaxations of the outputs of MLPs. The resulting relaxations of MLPs are once continuously differentiable as shown in the following proposition.

Proposition 2.1. *(smoothness of MLP relaxations). Consider an MLP using (exclusively) the hyperbolic and identity activation functions. The McCormick relaxations built with the envelopes of the hyperbolic tangent function (Appendix A.1) are once continuously differentiable.*

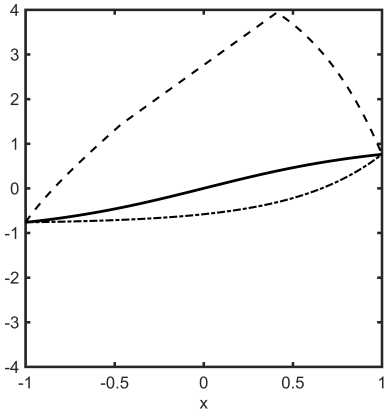
Proof The result follows directly from Proposition A.1, A.3 and the application of Corollary 4 by Khan, Watson and Barton (2017) [172, 173]. □
It should be noted that the relaxations are not C^2 because the envelopes of the hyperbolic tangent function are not C^2 (see Proposition A.1).

Proposition 2.1 can have a beneficial consequence for the global optimization of MLPs. In general, McCormick relaxations are continuous but not differentiable (C^0) necessitating nonsmooth algorithms, e.g., bundle methods [177] or linearization-based methods [163]. However, when optimizing MLPs, a $C^{1,1}$ optimization algorithm can be used which is potentially more efficient. In the numerical results presented in Section 2.4 the McCormick relaxations are linearized and thus the potential benefit of smooth relaxations is not exploited fully.

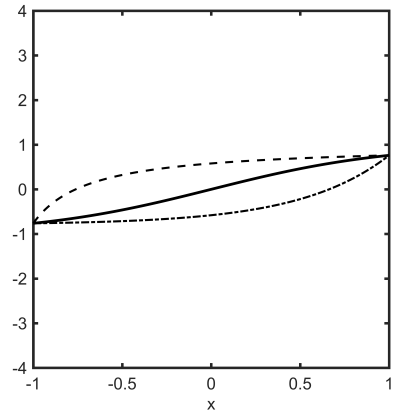
When using the state-of-the-art solvers BARON, ANTIGONE, and SCIP through interfaces like GAMS, the user cannot directly use tanh as it is not directly available. Instead, tanh can be used in its explicit algebraic form ($\tanh = \frac{e^x - e^{-x}}{e^x + e^{-x}}$) which results in weaker relaxations. As the tightness of the McCormick relaxations of different algebraic formulations can differ, the relaxations of four common algebraic formulations $F_1, F_2, F_3, F_4 : \mathbb{R} \to \mathbb{R}$ with $F_1(x) := \frac{e^x - e^{-x}}{e^x + e^{-x}}$, $F_2(x) := \frac{e^{2x}-1}{e^{2x}+1}$, $F_3(x) := 1 - \frac{2}{e^{2x}+1}$, $F_4(x) := \frac{1-e^{-2x}}{1+e^{-2x}}$ are compared throughout this work. As an illustration, the McCormick relaxations obtained from the four algebraic formulations are plotted on the interval $[-1, 1]$ in Figure 2.2. The exemplary



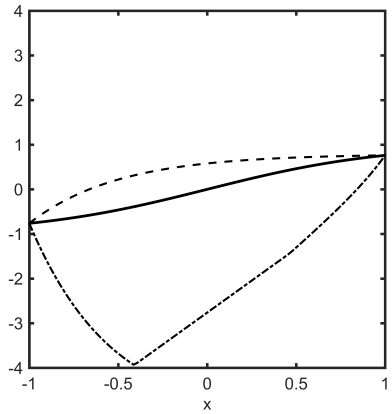
(a) Relaxations of algebraic formulation F_1



(b) Relaxations of algebraic formulation F_2



(c) Relaxations of algebraic formulation F_3



(d) Relaxations of algebraic formulation F_4

Figure 2.2.: Graphical illustration of the McCormick relaxations of algebraic formulations of the hyperbolic tangent function.

plot shows that the McCormick relaxations of F_1, F_2 and F_4 are weaker than the ones of F_3. Further, the McCormick relaxations of F_1, F_2 and F_4 are not differentiable. Finally, it can be observed that relaxations of the algebraic formulations are considerably weaker than the envelopes.

2.3.3 Implementation

The ANN models are implemented in our open-source toolbox MeLOn [132]. For training, MeLOn interfaces the state-of-the-art tools Matlab and Keras. For optimization, MeLOn is integrated into our open-source B&B solver MAiNGO [178] as a submodule. Thus, the ANN models are automatically available within MAiNGO. MAiNOG implements a best-first heuristic and bisection along the longest relative edge for branching. A convex underestimation is found using MC++ v2.0 [179, 180]. The interval extensions are provided by FILIB++ v3.0.2 [181]. The convex relaxations of the constraints and the objective are linearized with the use of subgradients at the centerpoint of each node and the resulting linear program (LP) is solved by CPLEX v12.8 [182]. Further, optimization-based bound tightening is used that is improved by filtering bounds technique with a factor of 0.1 as described in Gleixner *et al.* (2017) [183] and bound tightening based on the dual multipliers returned by CPLEX is used [184, 185]. For upper bounding, the problem is locally optimized using the SLSQP algorithm [186] in the NLopt library v2.4.2 [187]. The derivatives are provided by the FADBAD++ tool for automatic differentiation [188]. Furthermore, recently developed heuristics for tighter McCormick relaxations are used [189]. The convex and concave envelopes of tanh are added to the MC++ library for this work.

2.4 Numerical results

In this section, the numerical results are presented and compared to the solver BARON 17.4.1. using GAMS 24.8.5. All numerical examples were run on one thread of an Intel Xeon CPU E5-2630 v2 with 2.6 GHz, 128 GB RAM and Windows Server 2008 operating system. For the BARON solver we used default solver settings as numerical studies on the illustrative example suggested that selective branching does not give advantages four our case studies.

2.4.1 Illustrative example & scaling of the algorithm

In the first illustrative example, a two-dimensional mathematical test function is learned by a MLP that is subsequently optimized. The peaks function is provided by Matlab (peaks()) and is given by $f_{\text{peaks}} : \mathbb{R}^2 \rightarrow \mathbb{R}$ with

$$f_{\text{peaks}}(x_1, x_2) =$$
$$3(1-x_1)^2 \cdot e^{-x_1^2-(x_2+1)^2} - 10 \cdot (\frac{x_1}{5} - x_1^3 - x_2^5) \cdot e^{-x_1^2-x_2^2} - \frac{e^{-(x_1+1)^2-x_2^2}}{3}$$

The function has multiple local optima on $D = \{x_1, x_2 \in \mathbb{R}\colon\ -3 \leqslant x_1, x_2 \leqslant 3\}$. The known unique global minimizer is -6.551 at $(0.228, -1.626)$.

To learn the peaks function, 500 points are generated on D using Latin hypercube (LHC) sampling. As a MLP with one hidden layer is a universal approximator, this network architecture is chosen with $N = 3,\ D^{(1)} = 2,\ D^{(3)} = 1$. In order to avoid over- and under-fitting, the data is randomly divided into a training, a validation and a test sets with the respective size ratios of $0.7 : 0.15 : 0.15$. The weights of the network are fitted in the Matlab Neural Network Toolbox by minimizing the mean squared error (MSE) on the training set using Levenberg-Marquardt algorithm and the early stopping procedure [42]. To obtain a suitable number of neurons in the hidden layer the training is repeated using different number of neurons. The configuration with the lowest MSE on the test set determines the number of neurons to be $D^{(2)} = 47$. The training results in a MSE of $6.8 \cdot 10^{-5}$, $3.9 \cdot 10^{-4}$ and $4.6 \cdot 10^{-4}$ on the training, validation and test set respectively.

After training, the MLP is optimized using the proposed methods. The absolute optimization tolerance is set to $\epsilon_{\text{tol}} = 10^{-4}$ because a more accurate solution of the optimization problem is not sensible due to the prediction error of the MLP. The relative tolerance is set to its minimum value (10^{-12}) and is thus not active. Further, a time limit of 100,000 seconds is set.

The formulations of the optimization problems are provided and described as follows. The FS formulation of the peaks case study is given by Equations (2.2) - (2.10).

$$\min_{\mathbf{x}\in D, \mathbf{z}\in Z} \quad y \tag{2.2}$$

$$\text{s.t.} \tag{2.3}$$

$$z_1^{(1)} = \frac{x_1 - x_1^L}{x_1^U - x_1^L} \cdot 2 - 1 \tag{2.4}$$

$$z_2^{(1)} = \frac{x_2 - x_2^L}{x_2^U - x_2^L} \cdot 2 - 1 \tag{2.5}$$

$$v_i^{(1)} = \sum_{j=1}^{2} (w_{j,i}^{(1)} z_j^{(1)}) + b_i^{(1)} \qquad \forall i = 1, 2, ..., 47 \tag{2.6}$$

$$z_i^{(2)} = \tanh\left(v_i^{(1)}\right) \qquad \forall i = 1, 2, ..., 47 \tag{2.7}$$

$$v_1^{(2)} = \sum_{j=1}^{47} (w_{j,1}^{(2)} z_j^{(2)}) + b_1^{(2)} \tag{2.8}$$

$$z_1^{(3)} = \left(v_1^{(2)}\right) \tag{2.9}$$

$$y = (z_1^{(3)} + 1) \cdot \frac{y^U - y^L}{2} + y^L \tag{2.10}$$

Herein, the objective (Equation (2.2)) is to minimize the output of the MLP, i.e., y. Equation (2.4) and (2.5) scale the DoFs ($\mathbf{x} = (x_1, x_2)$) onto $[-1, 1]$. This is necessary as MLPs are usually trained on scaled data. Equations (2.6) compute the argument of the activation function ($v_i^{(1)}$) for each neuron in layer 2, i.e., the hidden layer. Equations (2.7) compute the output of each neuron in layer 2 ($z_i^{(2)}$). Equation (2.8) computes the argument of the activation function ($v_1^{(3)}$) for the neuron in layer 3, i.e., the output layer. Equation (2.9) computes the output of the neuron in the output layer ($z_1^{(3)}$). In this example, the equation is simplified because the identity output activation function is used. However,

Table 2.2.: Bounds on the variables of the peaks case study

Variable	x_i	$z_i^{(1)}$	$z_i^{(2)}$	$z_1^{(3)}$	$v_i^{(k)}$	y
Lower bound	-3	-1	-1	-10^6	-10^6	-10^6
Upper bound	3	1	1	10^6	10^6	10^6

in the more general case, Equation (2.9) includes a nonlinear transformation. Finally, Equation (2.10) scales the output of the neuron ($z_1^{(3)}$) back to the actual domain of the training data. The FS optimization problem constitutes 99 equality constraints. The original DoF ($\mathbf{x} = (x_1, x_2)$) have the dimension $n_x = 2$. The additional optimization variables $\mathbf{z} = (z_1^{(1)}, z_2^{(1)}, v_1^{(1)}, v_2^{(1)}, ..., v_{47}^{(1)}, z_1^{(2)}, z_2^{(2)}, ...z_{47}^{(2)}, v_1^{(2)}, z_1^{(3)}, y)$ have a dimension of $n_z = 99$.
The RS formulation of the peaks case study is given by Equation (2.11).

$$\min_{\mathbf{x} \in D} \quad \hat{y}(\mathbf{x}) \tag{2.11}$$

Herein, the objective ($\hat{y}$) is a function of the Dof (x). In the objective function, the complete MLP as well as the scaling of the inputs and outputs is included and effectively hidden to the B&B algorithm. The original DoF ($\mathbf{x} = (x_1, x_2)$) have the dimension $n_x = 2$. The additional optimization variables $\mathbf{z} = (\varnothing)$ have a dimension of $n_z = 0$.
Finally, the bounds on $\mathbf{x}$ and $\mathbf{z}$ have to be provided for the FS formulation (see Table 2.2). For the RS formulation, only bounds on x have to be provided. The variables $\mathbf{x}$ and their bounds have usually a physical meaning. In contrast, the additional optimization variables $\mathbf{z}$ usually do not have a meaning. These bounds are rather lose because we intend to use the same for all optimization problems (see Table 2.2). Tighter bounds can be derived by natural interval extensions and are shrunk by bound tightening techniques in MAiNGO and BARON.

Table 2.3 summarizes the results of the optimization runs. The RS formulation took roughly 0.3% of the CPU time to solve compared to the FS formulation when using the presented solver. In comparison to the FS formulations, the RS formulation reduces the number of the optimization variable from 101 to 2 and substitutes all 99 equality constraints. Further, different McCormick relaxations of the algebraic formulations of the activation function yield different solution times. The utilization of the envelope of the activation function accelerates the optimization significantly compared to the utilization of the McCormick relaxations of the algebraic formulations. When the presented solver is used in the FS formulation, the weak McCormick relaxations of the algebraic formulations even lead to a variable overflow error in the exponential function in FILIB++. The variable overflow error occurs in FILIB++ when a large number is used as an input of the exponential function (i.e., $x > 7.1 \cdot 10^2$). The error occurs in the lower bounding problem when weak relaxations lead to large numbers. These cases would not likely result in competitive CPU times. Further, the utilization of the McCormick relaxations of the algebraic formulation F_3 outperforms the other algebraic formulations in terms of computational efficiency. In general, all RS optimizations converged to the same optimal solution (-6.563). The solution is about 0.18% different from the global optimizer of the underlying peaks function.

In a second step, the peaks function is used to illustrate the scaling of the optimization algorithm with respect to the network size. Therefore, networks of different sizes are

Table 2.3.: Numerical results of the peaks function optimization (Subsection 2.4.1).

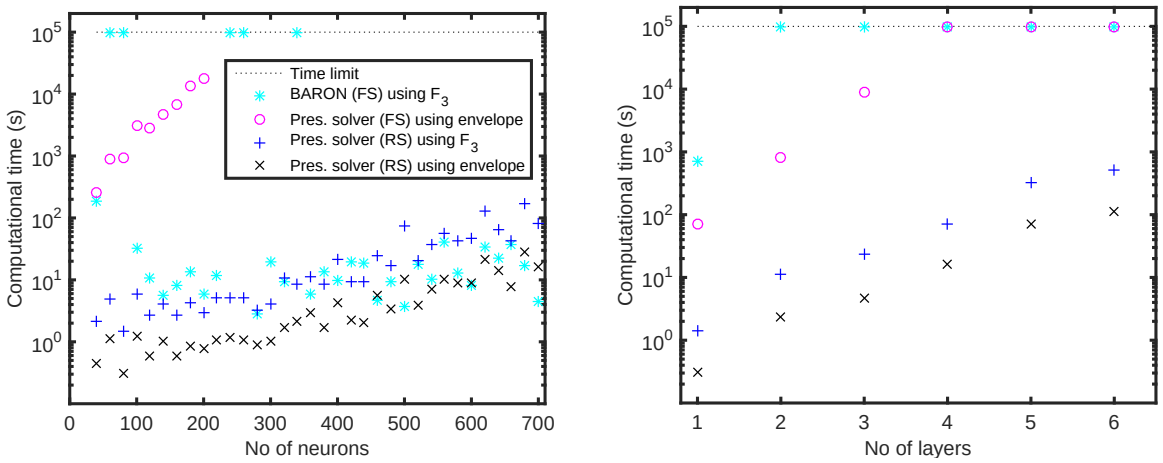
	Problem size	Solver	CPU time [s]	Iterations	Abs. gap
	101 variables	BARON (F_1)	1,727.29	55,047	$< \epsilon_{\text{tol}}$
	99 equalities	BARON (F_2)	387.44	13,329	$< \epsilon_{\text{tol}}$
	0 inequalities	BARON (F_3)	21.08	1,223	$< \epsilon_{\text{tol}}$
		BARON (F_4)	100,000.00	2,732,488	0.0003
(FS)		Presented solver (F_1)	–variable overflow–		
		Presented solver (F_2)	–variable overflow–		
		Presented solver (F_3)	–variable overflow–		
		Presented solver (F_4)	–variable overflow–		
		Presented solver (envelope)	161.35	657	$< \epsilon_{\text{tol}}$
	2 variables	Presented solver (F_1)	10.05	4,857	$< \epsilon_{\text{tol}}$
	0 equalities	Presented solver (F_2)	19.45	11,595	$< \epsilon_{\text{tol}}$
(RS)	0 inequalities	Presented solver (F_3)	1.16	657	$< \epsilon_{\text{tol}}$
		Presented solver (F_4)	13.09	7,469	$< \epsilon_{\text{tol}}$
		Presented solver (envelope)	0.50	137	$< \epsilon_{\text{tol}}$

variable overflow: Crashed due to variable overflow in FILIB++.

trained on a LHC set of 5,000 points and subsequently optimized. In Subfigure 2.3 (a), the number of neurons in the hidden layer of a shallow MLP, i.e., a MLP with one hidden layer, is varied from 40 to 700 and the computational time for solving the resulting optimization problem is depicted. The RS formulation using the envelope and the McCormick relaxations of the algebraic formulation F_3 shows an increase of CPU time with the number of neurons whereas the CPU time of BARON using F_3 does not show a clear trend. For instance, BARON did not converge within 100,000 seconds for networks with 60, 80, 240, 260, and 340 neurons but shows good performance for a network with 700 neurons. Using the FS formulation and the presented solver yields the weakest performance. Due to long CPU times, this optimization was not executed for networks with more than 200 neurons. The presented solver using the RS formulation and the envelope shows consistently one of the best performances. It should be noted that the number of neurons is unreasonably high for this simple function; likely, the high number leads to overfitting. This is irrelevant for the purposes of the chapter because the focus herein is on CPU performance as function of size. It should further be noted that the network sizes (up to 700 neurons) exceed the network size of the case studies by Smith *et al.* (2013) [72] (3 neurons) and de Weerdt *et al.* [158] (128 neurons).

In Subfigure 2.3 (b), the number of neurons in each hidden layer is fixed to 20 and the number of hidden layers is varied from 1 to 6. The results show that the CPU times of the presented solver using the RS and FS formulation increases approximately exponentially with the number of layers. Further, the utilization of the envelopes always has an advantage over the utilization of the McCormick relaxations of the algebraic formulation F_3. The solution of the FS formulation using either BARON or the presented solver does not converge to an optimal solution within the time limit for most cases. In general, it is apparent that deep networks require more CPU time for optimization than shallow networks with the same or even larger number of neurons.

One reason for the increase of CPU time with the ANN size is the tightness of the McCormick relaxations. In Figure 2.4, the McCormick relaxations of some previously used



(a) MLP with one hidden layer and varying number of neurons

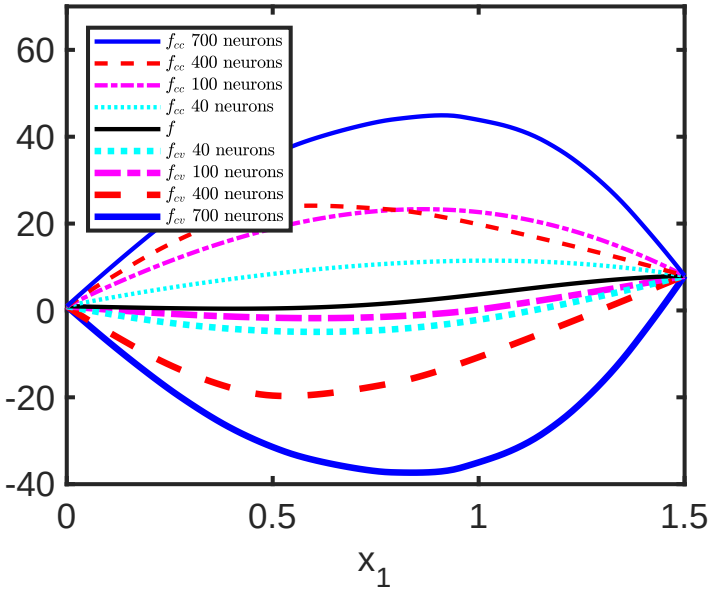
(b) MLP with 20 neurons in each layer and varying number of hidden layers

Figure 2.3.: Graphical illustration of the computational time over the ANN size for the peaks function optimization problem (Subsection 2.4.1).

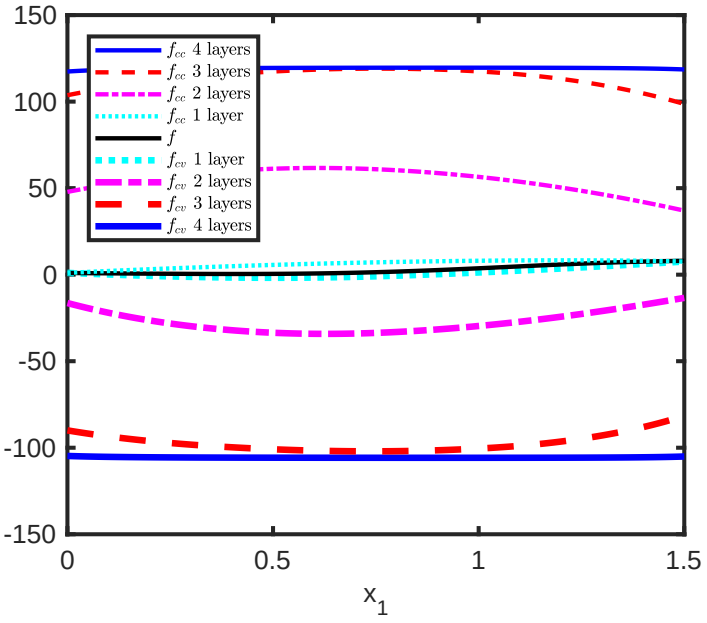
ANNs are illustrated for the interval $x_1 \in [0, 1.5]$ with $x_2 = 0$. Figure 2.3 shows that usually shallow ANNs with fewer numbers of neurons show tighter relaxations. However, the shallow network with 40 neurons shows similar McCormick relaxations as the one with 100 neurons on the depicted interval. This suggests that tightness also depends on the actual realization of the weights. Further, it can be observed that shallow networks show much tighter relaxations than deep networks. For this case study, the scaling of the proposed method using the reduced-space formulation and envelopes in MAiNGO is further analyzed in Table 2.4. Herein, the number of layers and neurons is varied simultaneously. The results support the previous observations that larger ANNs in general lead to longer CPU times. Note that the tolerances used for this comparison are slightly larger than for the actual numerical example mentioned above and that the number of neurons is the same for all hidden layers.

Table 2.4.: Numerical results of the scaling of the peaks optimization (Subsection 2.4.1) using the reduced-space formulation and envelopes of the hyperbolic tangent function in MAiNGO. The table shows CPU times in seconds for each run ($\epsilon_{abs}, \epsilon_{rel} = 10^{-4}$). The network with 4 hidden layers and 100 neurons is not available due to memory limitation during training.

# hidden layers	# neurons				
	20	40	60	80	100
1	0.22	0.62	0.70	1.28	1.53
2	2.03	6.74	22.40	73.60	173.30
3	5.94	22.71	221.44	1012.87	2252.33
4	13.20	295.25	1564.47	3943.05	N/A



(a) Influence of the number of neurons in shallow ANNs



(b) Influence of the number of layers in ANNs with 20 neurons in each hidden layer

Figure 2.4.: Illustration of the influence of the ANN size on the tightness of convex (f_{cv}) and concave relaxations (f_{cc}).

2.4.2 Fermentation process

As a second case study, a fermentation of glucose to gluconic acid is learned from experimental data and optimized. The example is based on and compared to the work of Cheema *et al.* [139] where an MLP is learned from the same experimental data and optimized using stochastic optimization approaches. The numerical example is relevant because mechanis-

tic models of fermentation processes are often not available and surrogate-based optimization can help to identify promising operating conditions.

The inputs of the MLP and degrees of freedom are the glucose concentration (x_1 in [g/L]), the biomass concentration (x_2 in [g/L]) and the dissolved oxygen (x_3 in [mg/L]), whereas the output of the MLP is the concentration of gloconic acid (y in [g/L]). The objective of the optimization problem is to maximize the yield of gluconic acid defined as $y_{gl} = \frac{100y}{1.088x_1}$. As the network weights used by Cheema *et al.* [139] are not available, we trained a MLP using the same structure, training and validation data, and training algorithm.

After the training, we performed deterministic global optimization to maximize the gluconic acid yield. The result of the deterministic global optimization is $x_1 = 156.466$ g/L, $x_2 = 3$ g/L, $x_3 = 57.086$ mg/L, $y = 170.127$ g/L and $y_{gl} = 99.937$ which is similar to the literature result. However, it is important to note that the underlying network is rebuilt based on the data and it is very unlikely that the network is identical to the one from literature. The main difference to the results from literature is that all deterministic methods in this work converge to the same solution whereas the stochastic literature method exhibits considerable variations in their solutions [139].

Due to the small network size, the problem size of the RS is only slightly smaller than the FS formulation (Table 2.5). The CPU time of the presented solver using the envelope and RS formulation took roughly 10% of the CPU time to solve compared to the FS formulation. Further, the CPU time of BARON using different formulations does not seem to directly relate to the tightness of their McCormick relaxations.

Table 2.5.: Numerical results of the fermentation process optimization (Subsection 2.4.2).

	Problem size	Solver	CPU time [s]	Iterations	Abs. gap
	13 variables	BARON (F_1)	14.38	8,233	$< \epsilon_{\text{tol}}$
	10 equalities	BARON (F_2)	0.44	5	$< \epsilon_{\text{tol}}$
	0 inequalities	BARON (F_3)	0.56	7	$< \epsilon_{\text{tol}}$
		BARON (F_4)	0.48	15	$< \epsilon_{\text{tol}}$
(FS)		Presented solver (F_1)	–variable overflow–		
		Presented solver (F_2)	–variable overflow–		
		Presented solver (F_3)	–variable overflow–		
		Presented solver (F_4)	–variable overflow–		
		Presented solver (envelope)	2.42	307	$< \epsilon_{\text{tol}}$
	3 variables	Presented solver (F_1)	40,418.20	18,516,400	$< \epsilon_{\text{tol}}$
	0 equalities	Presented solver (F_2)	100,000.00	1,852,750	10^{19}
(RS)	0 inequalities	Presented solver (F_3)	1.19	881	$< \epsilon_{\text{tol}}$
		Presented solver (F_4)	14.01	11,133	$< \epsilon_{\text{tol}}$
		Presented solver (envelope)	0.23	133	$< \epsilon_{\text{tol}}$

variable overflow: Crashed due to variable overflow in FILIB++.

2.4.3 Compressor plant

In the third numerical example, the operating point of a compressor plant is optimized. This is a relevant case study because air compressors are commonly used in industry and have a high electrical power consumption, e.g., in cryogenic air separation units. In

addition, the power consumption of industrial compressors is often provided in form of data (compressor maps, e.g., Figure 2.6) and not mechanistic models preventing model-based optimization techniques. The considered compressor plant is comprised of two compressors that are connected in parallel as shown in Figure 2.5. The compressors are sized such that a large compressor (compressor 1) is supplemented by a smaller compressor (compressor 2). The intent is to minimize the electrical power consumption of the overall process by optimal operation. Mathematically, the case study is challenging as it combines two MLPs with two hidden layers in one optimization problem.

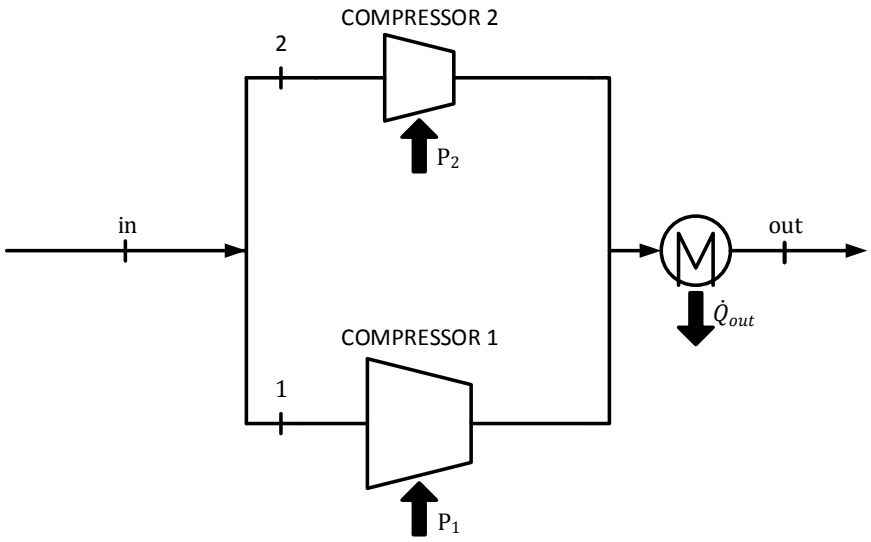
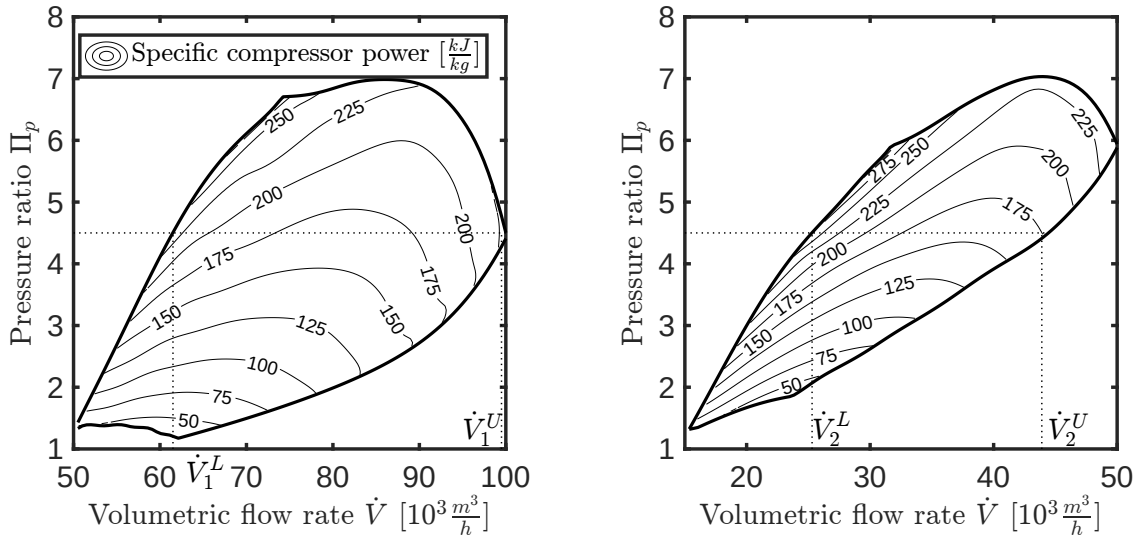


Figure 2.5.: Graphical illustration of the compressor configuration.

The specific power of the compressors is given by compressor maps $M_i : \mathbb{R}^2 \to \mathbb{R}$ (Figure 2.6) that map the volumetric inlet flow rate, $\dot{V}_i$, and compression ratio, $\Pi_i = \frac{p_{\text{out},i}}{p_{\text{in},i}}$, to the specific power, w_i, of the corresponding compressor i. For the MLP training, a set of 405 data points is read out from each compressor map and randomly divided into a training, validation and test set with the respective size ratios of $0.7 : 0.15 : 0.15$. This numerical example utilizes the network structure suggested by Ghorbanian and Gholamrezaei [190], namely a MLP with two hidden layers with 10 neurons each. For training, Levenberg-Marquardt algorithm and early stopping were used on a scaled training data set. The training results in a MSE of 0.17 (0.95), 0.78 (3.87) and 0.39 (1.67) on the training, validation and test set respectively for compressor one (two).

The optimization problem minimizes the total power consumption of the compressor plant, P_{total}. It has one degree of freedom, the split factor (x), that determines the ratio between the volumetric flow rate $\dot{V}_1$ and the total volumetric inlet flow rate, $\dot{V}_{\text{in}}$. The problem is formulated as follows:

$$
\begin{aligned}
\min_{0 \leqslant x \leqslant 1} \quad & P_{\text{total}}(x) = w_{1,\text{MLP}}(\Pi_p, \dot{V}_1) \cdot \frac{\dot{V}_{\text{in}} \cdot x}{v_{\text{in}}} + w_{2,\text{MLP}}(\Pi_p, \dot{V}_2) \cdot \frac{\dot{V}_{\text{in}} \cdot (1-x)}{v_{\text{in}}} \\
\text{s.t.} \quad & \dot{V}_1 - \dot{V}_1^U \leqslant 0, \qquad \dot{V}_1^L - \dot{V}_1 \leqslant 0 \\
& \dot{V}_2 - \dot{V}_2^U \leqslant 0, \qquad \dot{V}_2^L - \dot{V}_2 \leqslant 0
\end{aligned}
$$



(a) Performance map of compressor 1 (b) Performance map of compressor 2

Figure 2.6.: Graphical illustration of the compressors' specific performance maps.

with the constants $v_{\text{in}} = 0.8305\ \frac{\text{m}^3}{\text{kg}}$, $p_{\text{in}} = 100$ kPa, $p_{\text{out}} = 450$ kPa, $\dot{V}_{\text{in}} = 100 \cdot 10^3\ \frac{\text{m}^3}{\text{h}}$, $\dot{V}_1^L = 61.5 \cdot 10^3\ \frac{\text{m}^3}{\text{h}}$, $\dot{V}_1^U = 100 \cdot 10^3\ \frac{\text{m}^3}{\text{h}}$, $\dot{V}_2^L = 25.3 \cdot 10^3\ \frac{\text{m}^3}{\text{h}}$ and $\dot{V}_2^U = 44.4 \cdot 10^3\ \frac{\text{m}^3}{\text{h}}$. Further, $\dot{V}_1 = \dot{V}_{in} \cdot x$ and $\dot{V}_2 = \dot{V}_{\text{in}} \cdot (1 - x)$ are explicit functions of x. The functions $w_{1,\text{MLP}}$ and $w_{2,\text{MLP}}$ describe the specific power predicted by the MLPs. The lower and upper bounds of the volumetric flow rates, $\dot{V}_1^L$, $\dot{V}_2^L$ and $\dot{V}_1^U$, $\dot{V}_2^U$ correspond to the surge and choke lines of the compressors at the specified pressure ratio of 4.5 (left and right hand boundaries in Figure 2.6).

The optimization problem is solved using an absolute optimality gap of 10^{-4} and a relative optimality gap of 10^{-12}. Further, a CPU time limit of 100,000 seconds is set. As shown in Table 2.6, the RS formulation reduces the number of variables from 97 to 1 and substitutes all 96 equality constrains compared to the FS formulation. The RS formulation is solved using the presented solver within 0.19 seconds when using the envelopes. Thus, the RS formulation took about 0.7% of the CPU time to solve compared to the same solver and relaxation in the FS formulation. BARON converges to the pre-defined optimality gap only using formulation F_2 (within 25.02 second). When the other formulations are used, the desired tolerance is not reached within 100,000 seconds.

For this case study, the scaling of the proposed method using the reduced-space formulation and envelopes in MAiNGO is further analyzed. The two performance maps are learned using ANNs of different sizes. For the comparison, both ANNs have the same number of neurons in each hidden layer. Table 2.7 shows that larger ANNs in general lead to longer CPU times.

The optimal solution of the problem is $P_{\text{total}} = 6.20$ MW with a split ratio of $x = 0.683$ that corresponds to $\dot{V}_1 = 68.28\ \frac{m^3}{h}$ and $\dot{V}_2 = 31.72\ \frac{m^3}{h}$. The optimization using learned compressor maps can have some advantage over simple operating heuristics. For instance, if the main air compressor would be operated at its most efficient operating point, the the total power consumption would be 6.2% higher ($P_{\text{total}} = 6.61$ MW) and if the flowrates would be divided according to the maximum volumetric capacity of the compressors, the

Table 2.6.: Numerical results of the compressor plant optimization (Subsection 2.4.3).

	Problem size	Solver	CPU time [s]	Iterations	Abs. gap
(FS)	97 variables 96 equalities 4 inequalities	BARON (F_1)	100,000.00	1,934,517	$> 3 \cdot 10^8$
		BARON (F_2)	25.02	892	$< \epsilon_{\text{tol}}$
		BARON (F_3)	100,000.00	4,224,962	0.6
		BARON (F_4)	100,000.00	3,128,979	0.8
		Presented solver (F_1)	–variable overflow–		
		Presented solver (F_2)	–variable overflow–		
		Presented solver (F_3)	–variable overflow–		
		Presented solver (F_4)	–variable overflow–		
		Presented solver (envelope)	26.74	85	$< \epsilon_{\text{tol}}$
(RS)	1 variables 0 equalities 4 inequalities	Presented solver (F_1)	–variable overflow–		
		Presented solver (F_2)	–variable overflow–		
		Presented solver (F_3)	0.23	107	$< \epsilon_{\text{tol}}$
		Presented solver (F_4)	–variable overflow–		
		Presented solver (envelope)	0.19	55	$< \epsilon_{\text{tol}}$

variable overflow: Crashed due to variable overflow in FILIB++.

Table 2.7.: Numerical results of the scaling of the compressor plant optimization (Subsection 2.4.3) using the reduced-space formulation and envelopes of the hyperbolic tangent function in MAiNGO. The table shows CPU times in seconds for each run ($\epsilon_{abs}, \epsilon_{rel} = 10^{-4}$). Both artificial neural networks of this cases study are scaled in the same way in this comparison.

# hidden layers	# neurons				
	20	40	60	80	100
1	0.09	0.14	0.14	0.19	0.23
2	0.32	2.12	5.32	8.39	14.88
3	0.59	4.96	17.64	42.67	91.21
4	1.79	16.21	66.89	216.69	326.51
5	3.65	32.43	161.56	436.33	783.53

total power consumption would be 9.3% higher ($P_{\text{total}} = 6.84$ MW).

2.4.4 Cumene process

In the fourth numerical example, the operating point of a cumene process is optimized illustrating a complex industrial process. As illustrated in Figure 2.7, the cumene process consists of a plug flow reactor, two rectification columns, one flash, several (integrated) heat exchangers and recycles. A detailed process description including necessary details for model building can be found in Luyben (2017) [191]. For this work, we optimize a hybrid model consisting of 14 MLPs that emulate the process. This hybrid model was provided by Schultz *et al.* (2016) [192] who modeled the cumene process in ASPEN Plus and learned the MLPs using simulated data. The optimization problem minimizes the negative total profit $P(\mathbf{x})$ of the process with $\mathbf{x} = (T_{\text{reactor}}, Q_{\text{rebC1}}, RR_{\text{C1}}, DF_{\text{C2}}, RR_{\text{C2}})$ [192].

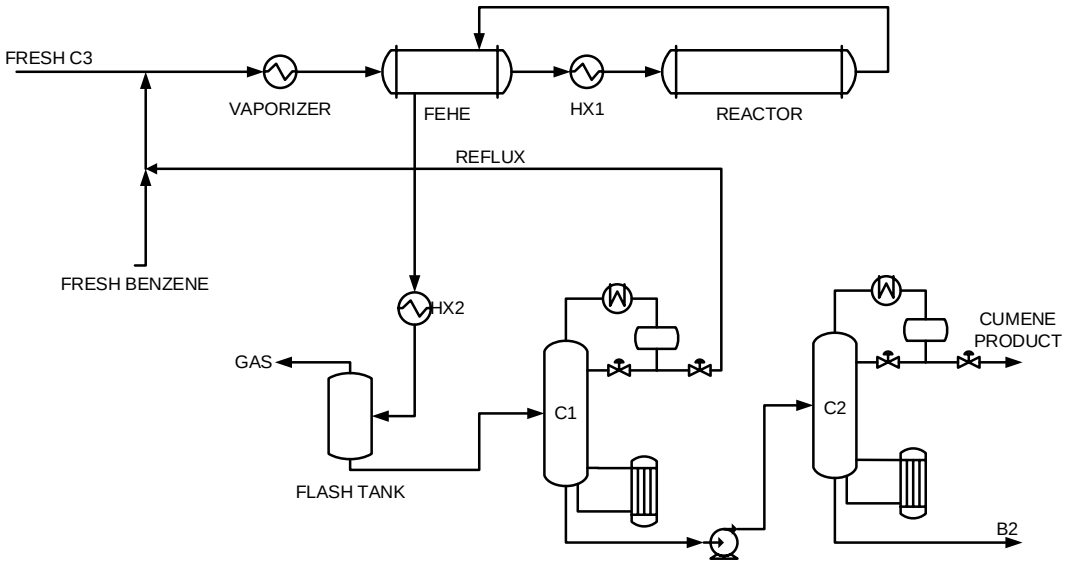


Figure 2.7.: Graphical illustration of the cumene process.

The hybrid model is composed by 14 MLPs, each consisting of 10 to 20 neurons, resulting in a FS optimization problem with 794 variables, 789 equality and 1 inequality constraints. The RS formulation constitutes only 5 variables, 0 equality and 1 inequality constraints. Due to the complexity of the case study, the problem is only implemented in the FS formulation in BARON and the RS formulation in the presented solver. The FS formulation in the presented solver is omitted as it is always outperformed by the RS formulation in the previous examples.

Table 2.8.: Numerical results of the cumene process optimization problem (Subsection 2.4.4).

	Problem size	Solver	CPU time [s]	Iterations	Abs. gap
(FS)	794 variables 789 equalities 1 inequalities	BARON (F_1)	100,000.00	110,453	$1 \cdot 10^{20}$
		BARON (F_2)	100,000.00	120,536	$1 \cdot 10^{20}$
		BARON (F_3)	100,000.00	136,645	$1 \cdot 10^{20}$
		BARON (F_4)	100,000.00	117,400	$1 \cdot 10^{20}$
(RS)	5 variables 0 equalities 1 inequalities	Presented solver (F_1)	–variable overflow–		
		Presented solver (F_2)	–variable overflow–		
		Presented solver (F_3)	100,000.00	4,671,260	2930
		Presented solver (F_4)	–variable overflow–		
		Presented solver (envelope)	100,000.00	6,329,810	444
		Presented solver (envelope)*	100,000.00	10,033,800	328

variable overflow: Crashed due to variable overflow in FILIB++.
*: Adapted setting such that upper bound is obtained by function evaluation, not local search.

The results in Table 2.8 show that none of the tested solution approaches converges to the desired tolerance within the CPU time limit. In order to analyze this, a convergence plot is provided in Figure 2.8 that depicts the lower bounds of the solvers over CPU time. Apparently, BARON does not improve its initial lower bound on the objective at all. In

comparison, the presented solver improves its lower bound steadily but slowly. Due to the high complexity of the case study another optimization run (annotated with the asterisk (*)) is executed using adapted solver options. More precisely, instead of using a local NLP solver in each iteration for upper bounding, the objective function is just evaluated at the center of the current interval. This leads to about 1.6 times more iterations in the same CPU time and further improvements of the lower bound. However, the complex example would still require longer CPU times to converge to the desired tolerance.

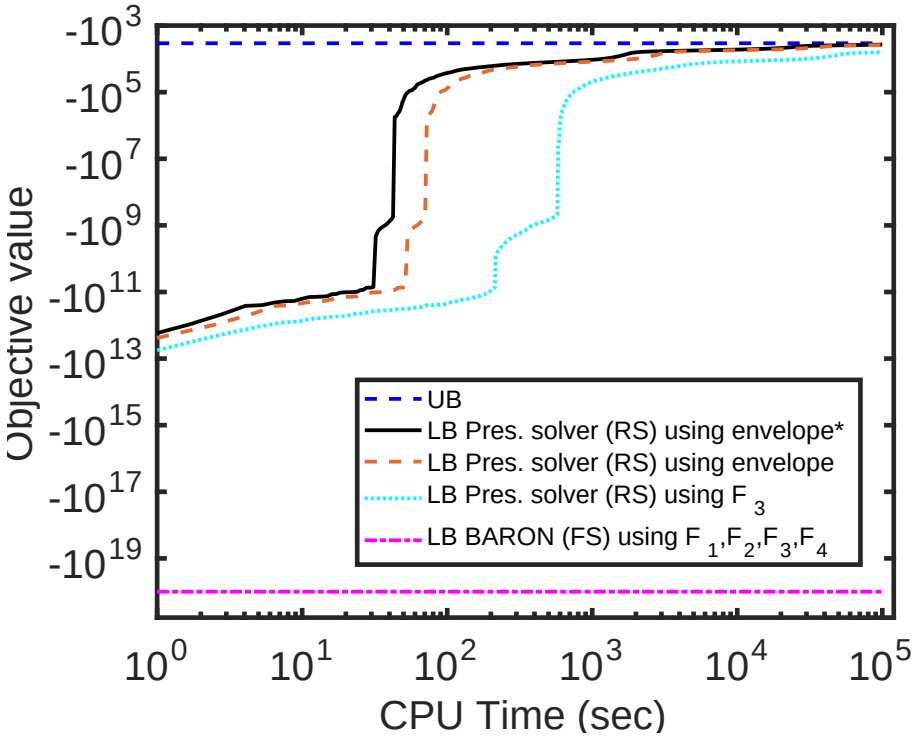


Figure 2.8.: Convergence plot of the cumene process optimization problem (Subsection 2.4.4). Herein, UB refers to the upper bound and LB to the lower bound of the corresponding solvers. The asterisk (*) refers to adapted solver options such that the upper bound is obtained by function evaluation, not local search.

2.5 Engineering applications

The developed method has great potential for many engineering applications. In collaboration with other researchers, we worked on numerous engineering applications of our method.

The deterministic global optimization of organic Rankine cycles (ORCs) is often challenging because accurate thermodynamic models often include nonlinear and implicit equations. This can lead to additional variables and constraints when formulating problems in the reduced space. In addition, we observed that some thermodynamic models lead to weak McCormick relaxations. In [11], we propose to simulate thermodynamic models and train ANNs on the simulated data. We show that the surrogate models can learn single fluid thermodynamic properties accurately. Moreover, the ANNs can reduce the number

of optimization variables and constraints in the optimization and they lead to tighter McCormick relaxations compared to the original model. In [30], we show that the additional effort of including accurate thermodynamic models is important because simplified models can lead to wrong design decisions. In [12, 31], we extend this work to the automatic working fluid selection for ORC processes. Later, we extended the work to the optimization of binary working fluid mixture compositions [28]. Finally, we performed superstructure process optimization for ORC processes with the ANN surrogate models embedded [26].

The design of efficient membrane systems is essential for a safe and sufficient drinking water supply. However, the design of membrane systems is a challenging problem because phenomena on multiple scales need to be considered in the optimal decision-making process. In [21], we proposed a hybrid modeling approach that combines experimental data for the membrane synthesis with known mechanistic transport models. Moreover, we solved the optimal operation and membrane synthesis problem simultaneously [13]. Finally, we also considered a membrane process superstructure optimization simultaneously to the membrane synthesis problem [14].

As renewable energy supply increases demand-side management becomes increasingly relevant for the economic operation of processes and requires processes to be operated more flexible [36]. Large deviations from the design operating point of a process can lead to nonlinear effects on the process efficiency. Thus, it is desirable to include process models into scheduling problems. However, the consideration of complex process models can lead to intractable optimization problems. In [25, 27], we propose to learn process models through ANNs and include the ANN surrogate models in the scheduling. In addition, we used recurrent ANNs models for the nonlinear model-predictive control of processes [33].

2.6 Conclusions

A method for deterministic global optimization of ANN embedded optimization problems is presented that formulates the problem in a RS and computes lower bounds by propagation of McCormick relaxations through the network equations. Thus, variables inside the ANNs and the network equations are not visible to the optimization algorithm and problem size is reduced significantly.

The proposed method is tested on four numerical examples including one illustrative function and three engineering applications. In all examples, the number of optimization variables and equality constraints was reduced and the optimization was accelerated significantly. Further, the results suggest that the combination of the RS formulation and the envelopes yield favorable performance in comparison to BARON.

In terms of problem sizes, the proposed approach optimized single shallow ANNs with up to 700 neurons in under 30 seconds. Further, deep ANNs require more CPU time for optimization than shallow ANNs with the same or even larger number of neurons. In general, the considered problem sizes exceed the examples for deterministic ANN embedded optimization found in the literature in terms of neurons, networks and nonlinear constraints.

This method can be further extended to, for instance, mixed integer problems where a process superstructure is optimized globally [75, 193] allowing the utilization of data from different sources such as experiments, process simulations and life-cycle assessment tools (e.g., [16]). Another possible extension could be deterministic global training of ANNs.

Chapter 3

Deterministic global optimization with Gaussian processes embedded

3.1 Introduction

A Gaussian process (GP) is a stochastic process where any finite collection of random variables has a multivariate Gaussian distribution; they can be understood as an infinite-dimensional generalization of multivariate Gaussian distributions [66]. The predictions of GPs are Gaussian distributions that provide not only an estimate but also a variance. GPs originate from geostatistics [194] and gained popularity for the design and analysis of computer experiments (DACE) since 1989 [195]. Furthermore, GPs are commonly applied as interpolating surrogate models across various disciplines including biotechnology [196–200], chemical engineering [16, 69, 201–205], chemistry [9, 23], and deep-learning [206]. Note that GP regression is also often referred to as Kriging. In many applications, GPs are trained on a data set and are subsequently embedded in an optimization problem, e.g., to identify an optimal operating point of a process. Moreover, many derivative-free solvers for expensive-to-evaluate black-box functions actually train GPs and optimize their predictions (e.g., Bayesian optimization algorithms [18, 197, 207, 208] and other adaptive sampling approaches [202–205, 209–211]). In Bayesian optimization, the optimum of an acquisition function determines the next sampling point [207]. The vast majority of these optimizations have been performed by local solution approaches [212, 213] and a few by stochastic global optimization methods [18]. Our contribution focuses on the deterministic global solution of optimization problems with trained GPs embedded and on applications in process systems engineering.

GPs are commonly used to learn the input-output behavior of unit operations [78, 201, 214–218], complete flowsheets [219], or thermodynamic property relations from data [220]. Subsequently, the trained GPs are often combined with nonlinear mechanistic process models leading to hybrid mechanistic and data-driven models [91, 94, 98, 221] which are optimized. Many of the previous works on optimization with GPs embedded rely on local optimization techniques. Caballero and Grossmann, for instance, train GPs on data obtained from a rigorous divided wall column simulation. Then, they iteratively optimize the operation of the column (modeled by GPs) using SNOPT [222], sample new data at the solution point, and update the GPs [201, 214]. Later, Caballero and co-workers extend this work to distillation sequence superstructure problems [215, 216]. In [216], the authors solve the resulting mixed-integer nonlinear programs (MINLPs) using a local solver in GAMS. Therein, the GP estimate is computed via an external function in Matlab which leads to a reduced optimization problem size visible to the local solver in GAMS. However, all these local methods have the drawback that they can lead to suboptimal solutions, because the resulting optimization problems are nonconvex. This nonconvexity is induced by the co-

variance functions of the GPs as well as often the mechanistic part of the hybrid models.

Deterministic global optimization can guarantee to identify globally optimal solutions within finite time to a given nonzero tolerance [223]. In a few previous studies, deterministic global optimization with GPs embedded was done using general-purpose global solvers. For instance, in the black-box optimization algorithms ALAMO [209] and ARGONAUT [211], GPs are included as surrogate models and are optimized using BARON [157] and ANTIGONE [160], respectively. However, computational burdens were observed that limit applicability, e.g., in terms of the number of training points. Cozad et al. [209] state that GPs are accurate but "difficult to solve using provable derivative-based optimization software". Similarly, Boukouvala and Floudas [211] state that the computational cost becomes a limiting factor because the number of nonlinear terms of GPs equals the product of the number of interpolated points (N) and the dimensionality (D) of the input domain. More recently, Keßler et al. [78, 218] optimized the design of nonideal distillation columns by a trust-region approach with GPs embedded. Therein, optimization problems with GPs embedded are solved globally using BARON [157] within relatively long CPU times ($10^2 - 10^5$ CPU seconds on a personal computer).

As mentioned earlier, Quirante et al. [216] call an external Matlab function to compute GP estimates with a local solver in GAMS. As an alternative approach, they also solve the problem globally using BARON in GAMS by providing the full set of GP equations as equality constraints. This leads to additional intermediate optimization variables besides the degrees of freedom of the problem. Similar to other studies, they observe that their formulation is only practical for a small number of GP surrogates and training data points to avoid large numbers of variables and constraints [216]. We refer to the problem formulation where the GP is described by equality constraints and additional optimization variables as a full-space (FS) formulation. It is commonly used in modeling environments, e.g., GAMS, that interface with state-of-the-art global solvers such as ANTIGONE [160], BARON [157], and SCIP [161].

An alternative to the FS is a reduced-space (RS) formulation where some optimization variables are eliminated using explicit constraints. This reduced problem size leads to a lower number of variables for branching as well as potentially smaller subproblems. The former has some similarity to selective branching [162] (c.f. discussion in [224]). The exact size of the subproblems for lower bounding and bound tightening depends on the method for constructing relaxations. In particular, when constructing relaxations in the RS using McCormick [167], alphaBB [225] or natural interval extensions, the resulting lower bounding problems are much smaller compared to the auxiliary variable method (AVM) [226, 227]. Therefore, any global solver can in principle handle RS but some methods for constructing relaxations appear more promising to benefit from the RS [224]. We have recently released the open-source global solver MAiNGO [178] which uses the MC++ library [180] for automatic propagation of McCormick relaxations through computer code [163]. We have shown that the RS formulation can be advantageous for flowsheet optimization problems [165, 228] and problems with artificial neural networks embedded [1, 21, 229]. In the context of Bayesian optimization, Jones et at. [208] develop valid overestimators of the expected improvement (EI) acquisition function in the RS. However, their relaxations rely on interval extensions and optimization-based relaxations limited to a specific covariance function; they do not derive envelopes. Furthermore, they do not provide convex underestimators which are in general necessary to embed GPs in optimization problems.

The main contribution of this work is the efficient deterministic global optimization of

optimization problems with GPs embedded. We develop a RS formulation for optimization problems with GPs embedded. The performance of the proposed method is analyzed in an extensive computational study by solving about 90,000 optimization problems. The proposed RS outperforms a FS formulation for problems with GPs embedded by speedup factors of several magnitudes. Moreover, this speedup increases with the number of training points. To further accelerate convergence, we derive and implement envelopes of covariance functions for GPs and tight relaxations of acquisition functions, which are commonly used in Bayesian optimization. Finally, we solve a chance-constrained optimization problem with GPs embedded and we perform global optimization of an acquisition function. The GP training methods and models are provided as our open-source toolbox MeLOn under the Eclipse public license [132]. The resulting optimization problems are solved using our global solver MAiNGO [178]. Note that the MeLOn toolbox is also automatically included as a submodule in our new MAiNGO release.

3.2 Optimization problem formulations

In the simplest case, which is common in the literature, the (scaled) inputs of a GP are the free variables of the optimization problem with $\boldsymbol{x} \in \tilde{X} = [\boldsymbol{x}^L, \boldsymbol{x}^U]$. For given $\boldsymbol{x}$, the dependent (or intermediate) variables $\boldsymbol{z}$ can be computed by the solution of $\boldsymbol{h}(\boldsymbol{x}, \boldsymbol{z}) = \boldsymbol{0}$, $\boldsymbol{h} : \tilde{X} \times \mathbb{R}^{n_z} \to \mathbb{R}^{n_z}$. In the case of GP models, we aim to include the estimate ($m_{\mathcal{D}}$) and variance ($k_{\mathcal{D}}$) in the optimization. As will be shown in Section 3.3, we can solve explicitly for $m_{\mathcal{D}}$ and $k_{\mathcal{D}}$ (c.f. Equations (3.1) and (3.2)).

The realization of the objective function f depends on the application. In many applications, it depends on the estimate of the GP, i.e., $f(m_{\mathcal{D}})$ (c.f. Subsection 3.6.1). In Bayesian optimization, the objective function is called the acquisition function and usually depends on the estimate and variance of the GP, i.e., $f(m_{\mathcal{D}}, k_{\mathcal{D}})$ (c.f. Subsection 3.6.3). Finally, additional constraints might depend on the inputs of the GP, its estimate, and variance, i.e., $\boldsymbol{g}(\boldsymbol{x}, m_{\mathcal{D}}, k_{\mathcal{D}}) \leqslant \boldsymbol{0}$. In more complex cases, multiple GPs can be combined in one optimization problem (c.f. Subsection 3.6.2).

In the following, we describe two optimization problem formulations for problems with trained GPs embedded: the commonly used FS formulation in Subsection 3.2.1 and the RS formulation in Subsection 3.2.2. Both problem formulations are exact reformulations in the sense of Liberti et al. [230], meaning that they have the same local and global optima. The equivalence is shown in Appendix A of [224]. However, the formulation significantly affects problem size and performance of global optimization solvers.

3.2.1 Full-space formulation

In the FS formulation, the nonlinear equations $\boldsymbol{h}(\boldsymbol{x}, \boldsymbol{z}) = \boldsymbol{0}$ are provided as equality constraints and the intermediate dependent variables $\boldsymbol{z} \in Z$ are optimization variables. A general FS problem formulation is:

$$\begin{aligned} \min_{\boldsymbol{x}\in\tilde{X},\boldsymbol{z}\in Z} \quad & f(\boldsymbol{x}, \boldsymbol{z}) \\ \text{s.t.} \quad & \boldsymbol{h}(\boldsymbol{x}, \boldsymbol{z}) = \boldsymbol{0}, \qquad \boldsymbol{g}(\boldsymbol{x}, \boldsymbol{z}) \leqslant \boldsymbol{0} \end{aligned} \tag{FS}$$

In general, there exist multiple valid FS formulations for optimization problems. In Section 3 of the electronic supplementary information (ESI), we provide a representative FS formulation for the case where the estimate of a GP is minimized. This is also the FS formulation that we use in our numerical examples (c.f., Section 3.6.1).

3.2.2 Reduced-space formulation

In the RS formulation, the equality constraints are solved for the intermediate variables and substituted in the optimization problem (c.f. [165]). A general RS problem formulation in the context of optimization with a GP embedded is:

$$\begin{aligned} \min_{\boldsymbol{x}\in\tilde{X}} \quad & f(m_{\boldsymbol{\mathcal{D}}}(\boldsymbol{x}), k_{\boldsymbol{\mathcal{D}}}(\boldsymbol{x})) \\ \text{s.t.} \quad & \boldsymbol{g}(\boldsymbol{x}, m_{\boldsymbol{\mathcal{D}}}(\boldsymbol{x}), k_{\boldsymbol{\mathcal{D}}}(\boldsymbol{x})) \leqslant \boldsymbol{0} \end{aligned} \tag{RS}$$

Herein, the Branch-and-Bound (B&B) solver operates only on the free variables $\boldsymbol{x}$ and no equality constraints are visible to the solver. In GPs, the estimate and variance are explicit functions of the input (Equations (3.1) and (3.2)). Thus, we can directly formulate a RS formulation. The RS formulation effectively combines those equations and hides them from the B&B algorithm. This results in a total number of D optimization variables, zero equality constraints, and no additional optimization variables $\boldsymbol{z}$. Thus, the RS formulation requires only bounds on $\boldsymbol{x}$.

Note that the direct substitution of all equality constraints is not always possible when multiple GPs are combined with mechanistic models, e.g., in the presence of recycle streams. Here, a small number of additional optimization variables and corresponding equality constraints can remain in the RS formulation [165]. As an alternative, relaxations for implicit functions can also be derived [175, 176]. Moreover, we have previously observed that a hybrid between RS and FS formulation can be more efficiently solvable for some optimization problems [228]. In this work, we compare the RS and the FS formulation and do not consider any hybrid problem formulations.

3.3 Gaussian processes

In this section, GPs are briefly introduced (c.f. [66]). We first describe the GP prior distribution, i.e., the probability distribution before any data is taken into account. Then, we describe the posterior distribution, which results from conditioning the prior on training data. Finally, we describe how hyperparameters of the GP can be adapted to data by a maximum a posteriori (MAP) estimate.

3.3.1 Prior

A GP prior is fully described by its mean function $m(\boldsymbol{x})$ and positive semi-definite covariance function $k(\boldsymbol{x}, \boldsymbol{x}')$ (also known as kernel function). We consider a noisy observation y from a function $\tilde{f}(\boldsymbol{x})$ with $y(\boldsymbol{x}) \coloneqq \tilde{f}(\boldsymbol{x}) + \varepsilon_{\text{noise}}$, whereby the output noise $\varepsilon_{\text{noise}}$ is independent and identically distributed (i.i.d.) with $\varepsilon_{\text{noise}} \sim \mathcal{N}(0, \sigma^2_{\text{noise}})$. We say y is distributed

as a GP, i.e., $y \sim \mathcal{GP}(m(\boldsymbol{x}), k(\boldsymbol{x}, \boldsymbol{x}'))$ with

$$m(\boldsymbol{x}) \coloneqq \mathbb{E}\big[\tilde{f}(\boldsymbol{x})\big],$$
$$k(\boldsymbol{x}, \boldsymbol{x}') \coloneqq \mathbb{E}\big[\ (\ y(\boldsymbol{x}) - m(\boldsymbol{x})\ )\ (\ y(\boldsymbol{x}') - m(\boldsymbol{x})\ )^{\mathrm{T}}\big].$$

Without loss of generality, we assume that the prior mean function is $m(\boldsymbol{x}) = 0$. This implies that we train the GP on scaled data such that the mean of the training outputs is zero. A common class of covariance functions is the Matérn class.

$$k_{\text{Matérn}}(\boldsymbol{x}, \boldsymbol{x}') \coloneqq \sigma_f^2 \frac{2^{1-\nu}}{\Gamma(\nu)} \left(\sqrt{2\nu} r\right)^{\nu} K_\nu \left(\sqrt{2\nu} r\right),$$

where σ_f^2 is the output variance, $r \coloneqq \sqrt{(\boldsymbol{x} - \boldsymbol{x}')^{\mathrm{T}} \boldsymbol{\Lambda}\ (\boldsymbol{x} - \boldsymbol{x}')}$ is a weighted Euclidean distance, $\boldsymbol{\Lambda} \coloneqq \text{diag}(\lambda_1^2, \cdots, \lambda_i^2, \cdots \lambda_{n_x}^2)$ is a length-scale matrix with $\lambda_i \in \mathbb{R}$, $\Gamma(\cdot)$ is the gamma function, and $K_\nu(\cdot)$ is the modified Bessel function of the second kind. The smoothness of Matérn covariance functions can be adjusted by the positive parameter ν. When ν is a half-integer value, the Matérn covariance function becomes a product of a polynomial and an exponential [66]. Common values for ν are $1/2,\ 3/2,\ 5/2$, and ∞, i.e., the most widely-used squared exponential covariance function, $k'_{SE}(r) \coloneqq \exp\left(-\frac{1}{2}\ r^2\right)$. We derive envelopes of these covariance functions in Section 3.4.1 and implement them within MeLOn [132]. Also, a noise term, $\sigma_{\text{n}}^2 \cdot \delta(\boldsymbol{x}, \boldsymbol{x}')$, can be added to any covariance function where σ_{n}^2 is the noise variance and $\delta(\boldsymbol{x}, \boldsymbol{x}')$ is the Kronecker delta function. The hyperparameters of the covariance function are adjusted during training and are jointly noted as $\boldsymbol{\theta} = [\lambda_1, .., \lambda_d, \sigma_f, \sigma_{\text{n}}]$. Herein, a log-transformation is common to prevent negative values during training.

3.3.2 Posterior

The GP posterior is obtained by conditioning the prior on observations. We consider a set of N training inputs $\boldsymbol{\mathcal{X}} = \{\boldsymbol{x}_1^{(\boldsymbol{\mathcal{D}})}, ..., \boldsymbol{x}_N^{(\boldsymbol{\mathcal{D}})}\}$ where $\boldsymbol{x}_i^{(\boldsymbol{\mathcal{D}})} = [x_{i,1}^{(\boldsymbol{\mathcal{D}})}, ..., x_{i,D}^{(\boldsymbol{\mathcal{D}})}]^T$ is a D-dimensional vector. Note that we use the superscript $(\boldsymbol{\mathcal{D}})$ to denote the training data. The corresponding set of scalar observations is given by $\boldsymbol{\mathcal{Y}} = \{y_1^{(\boldsymbol{\mathcal{D}})}, ..., y_N^{(\boldsymbol{\mathcal{D}})}\}$. Furthermore, we define the vector of scalar observations $\boldsymbol{y} = [y_1^{(\boldsymbol{\mathcal{D}})}, ..., y_N^{(\boldsymbol{\mathcal{D}})}]^T \in \mathbb{R}^N$. The posterior GP is obtained by Bayes' theorem:

$$\tilde{f}(\boldsymbol{x}) \sim \mathcal{GP}(m(\boldsymbol{x}), k(\boldsymbol{x}, \boldsymbol{x}') | \boldsymbol{\mathcal{X}}, \boldsymbol{\mathcal{Y}}) = \mathcal{N}\left(m_{\boldsymbol{\mathcal{D}}}(\boldsymbol{x}), k_{\boldsymbol{\mathcal{D}}}(\boldsymbol{x}, \boldsymbol{x}')\right)$$

with

$$m_{\boldsymbol{\mathcal{D}}}(\boldsymbol{x}) = \boldsymbol{K}_{\boldsymbol{x},\mathcal{X}} \left(\boldsymbol{K}_{\mathcal{X},\mathcal{X}}\right)^{-1} \boldsymbol{y}, \tag{3.1}$$
$$k_{\boldsymbol{\mathcal{D}}}(\boldsymbol{x}) = K_{\boldsymbol{x},\boldsymbol{x}} - \boldsymbol{K}_{\boldsymbol{x},\mathcal{X}} \left(\boldsymbol{K}_{\mathcal{X},\mathcal{X}}\right)^{-1} \boldsymbol{K}_{\mathcal{X},\boldsymbol{x}}, \tag{3.2}$$

where the covariance matrix of the training data is given by $\boldsymbol{K}_{\mathcal{X},\mathcal{X}} \coloneqq [k(\boldsymbol{x}_i, \boldsymbol{x}_j)] \in \mathbb{R}^{N \times N}$, the covariance vector between the candidate point $\boldsymbol{x}$ and the training data is given by $\boldsymbol{K}_{\boldsymbol{x},\mathcal{X}} \coloneqq \left[k(\boldsymbol{x}, \boldsymbol{x}_1^{(\boldsymbol{\mathcal{D}})}), .., k(\boldsymbol{x}, \boldsymbol{x}_N^{(\boldsymbol{\mathcal{D}})})\right] \in \mathbb{R}^{1 \times N}$, $\boldsymbol{K}_{\mathcal{X},\boldsymbol{x}} = \boldsymbol{K}_{\boldsymbol{x},\mathcal{X}}{}^T$, and $K_{\boldsymbol{x},\boldsymbol{x}} \coloneqq k(\boldsymbol{x}, \boldsymbol{x})$. Equations (3.1) and (3.2) describe essentially the predictions of a GP and are implemented within MeLOn.

3.3.3 Maximum a posteriori

In order to find appropriate hyperparameters $\boldsymbol{\theta}$ for a given problem, we use a MAP estimate which is known to be advantageous compared to the maximum likelihood estimation (MLE) on small data sets [231]. Using the MAP estimate, the hyperparameters are identified by maximizing the probability that the GP fits the training data, i.e., $\boldsymbol{\theta}_{\text{opt}} \coloneqq \text{argmax}_{\boldsymbol{\theta}} \mathcal{P}(\boldsymbol{\theta}|\boldsymbol{\mathcal{X}}, \boldsymbol{\mathcal{Y}})$. Analytical expressions for $\mathcal{P}(\boldsymbol{\theta}|\boldsymbol{\mathcal{X}}, \boldsymbol{\mathcal{Y}})$ and its derivatives w.r.t. the hyperparameters can be found in the literature [66]. We provide a Matlab training script in MeLOn that is based on our previous work [18]. Therein, we assume an independent Gaussian distribution as a prior distribution on the log-transformed hyperparameters, i.e., $\theta_i \sim \mathcal{N}(\mu_i, \sigma_i^2)$. The implementation of the training is efficient through the pre-computation of squared distances, the Cholesky decomposition for computing the inverse of the covariance matrix, and a two-step training approach that searches first globally and then locally [18].

3.4 Convex and concave relaxations

The construction of relaxations, i.e., convex function underestimators (F^{cv}) and concave function overestimators (F^{cc}), is essential for B&B algorithms. In our open-source solver MAiNGO, we use the (multivariate) McCormick method [167, 170] to propagate relaxations and their subgradients [163] through explicit functions using the MC++ library [180]. However, the McCormick method often does not provide the tightest possible relaxations, i.e., the envelopes. In this section, we derive tight relaxations or envelopes of functions that are relevant for GPs and Bayesian optimization. The functions and their relaxations are implemented in MC++. When using these intrinsic functions and their relaxations in MAiNGO, the (multivariate) McCormick method is only used for the remaining parts of the model. Note that the derived relaxations are used within MAiNGO while BARON does not allow for implementation of custom relaxations or piecewise defined functions.

3.4.1 Covariance functions

The covariance function is a key element of GPs. When embedding trained GPs into optimization problems, the covariance function occurs N times because it is used in the covariance vector between the candidate point $\boldsymbol{x}$ and the training data, i.e., $\boldsymbol{K}_{\boldsymbol{x},\mathcal{X}} = \left[k(\boldsymbol{x}, \boldsymbol{x}_1^{(\mathcal{D})}), .., k(\boldsymbol{x}, \boldsymbol{x}_N^{(\mathcal{D})})\right] \in \mathbb{R}^{1\times N}$. Note that the covariance matrix $\boldsymbol{K}_{\mathcal{X},\mathcal{X}}$ depends only on training data and is thus a parameter during the optimization. Thus, tight relaxations of the covariance functions are highly desirable. In this subsection, we derive envelopes for common Matérn covariance functions. We consider univariate covariance functions, i.e., $k_\nu : \mathbb{R} \to \mathbb{R}$, with input $d = (\boldsymbol{x} - \boldsymbol{x}')^{\mathrm{T}} \boldsymbol{\Lambda} (\boldsymbol{x} - \boldsymbol{x}') \geqslant 0$. This is possible because we consider stationary covariance functions that are invariant to translations in the input space. Common Matérn covariance functions use $\nu = 1/2,\ 3/2,\ 5/2$ and ∞ and are given by:

$$k_{\nu=1/2}(d) \coloneqq \exp\left(-\sqrt{d}\right), \qquad k_{\nu=3/2}(d) \coloneqq \left(1+\sqrt{3}\,\sqrt{d}\right)\cdot\exp\left(-\sqrt{3}\,\sqrt{d}\right)$$
$$k_{\nu=5/2}(d) \coloneqq \left(1+\sqrt{5}\,\sqrt{d}+\frac{5}{3}\,d\right)\cdot\exp\left(-\sqrt{5}\sqrt{d}\right), \qquad k_{SE}(d) \coloneqq \exp\left(-\frac{1}{2}\,d\right),$$

where k_{SE} is the squared exponential covariance function with $\nu \to \infty$. We find that these four covariance functions are convex because their Hessian is positive semidefinite. Thus, the convex envelope is given by $F^{cv}(d) = k(d)$ and the concave envelope by the secant $F^{cc}(d) = \mathrm{sct}(d)$ where $\mathrm{sct}(d) = \frac{k(d^U)-k(d^L)}{d^U-d^L}d + \frac{d^U k(d^L)-d^L k(d^U)}{d^U-d^L}$ on a given interval $[d^L, d^U]$. As the McCormick composition and product theorems provide weak relaxations of $k_{\nu=3/2}$ and $k_{\nu=5/2}$ (c.f. ESI Section 1), we implement these functions and their envelopes in our library of intrinsic functions in MC++. Furthermore, natural interval extensions are not exact for $k_{\nu=3/2}$ and $k_{\nu=5/2}$. Thus, we also provide exact interval bounds based on the monotonicity.

It should be noted that covariance functions are commonly given as a function of the weighted Euclidean distance $r = \sqrt{d}$. However, we chose to use d instead for three main reasons: (1) $\boldsymbol{x}$ is usually a free variable of the optimization problem. Thus, the computation of r would lead to potentially weaker relaxations for $k_{\nu=5/2}$ and k_{SE}. (2) The derivative of $k_{\nu=3/2}(\cdot)$, $k_{\nu=5/2}(\cdot)$, and $k_{SE}(\cdot)$ is defined at $d = 0$ while the derivative of the square root function is not. (3) The covariance functions $\hat{k}_{v=3/2} : r \mapsto k_{v=3/2}(r^2)$, $\hat{k}_{v=5/2} : r \mapsto k_{v=5/2}(r^2)$, and $\hat{k}_{SE} : r \mapsto k_{SE}(r^2)$ are nonconvex in r, so deriving the envelopes would be nontrivial.

Finally, it can be noted that we did not derive envelopes of $k_{\text{Matérn}}(\boldsymbol{x}, \boldsymbol{x}')$, because the variable input dimensions pose difficulties in implementation and the multidimensionality is a challenge for the derivation of envelopes. Nevertheless, the McCormick composition theorem applied to $k_\nu(d(\boldsymbol{x}, \boldsymbol{x}'))$ yields relaxations that are exact at the minimum of $k_{\text{Matérn}}$ because the natural interval extensions of the weighted squared distance d are exact (c.f. [232]). This means that the relaxations are exact in Hausdorff metric.

3.4.2 Gaussian probability density function

The PDF is used to compute the EI acquisition function and is given by $\phi : \mathbb{R} \to \mathbb{R}$ with

$$\phi(x) := \frac{1}{\sqrt{2\pi}} \cdot \exp\left(\frac{-x^2}{2}\right) \tag{3.3}$$

The Gaussian probability density function (PDF) is a nonconvex function for which the McCormick composition rule does not provide its envelopes. For one-dimensional functions, McCormick [167] also provides a method to construct envelopes. We construct the envelopes of PDF using this method and implement them in our library of intrinsic functions. The envelope of the PDF is illustrated in Figure 3.1 and derived in Appendix A.2.

3.4.3 Gaussian cumulative distribution function

The Gaussian cumulative distribution function (CDF) is given by $\Phi : \mathbb{R} \to \mathbb{R}$ with

$$\Phi(x) := \int_{-\infty}^{x} \phi(t)\, dt = \frac{1 + \mathrm{erf}\left(\frac{\sqrt{2}x}{2}\right)}{2}. \tag{3.4}$$

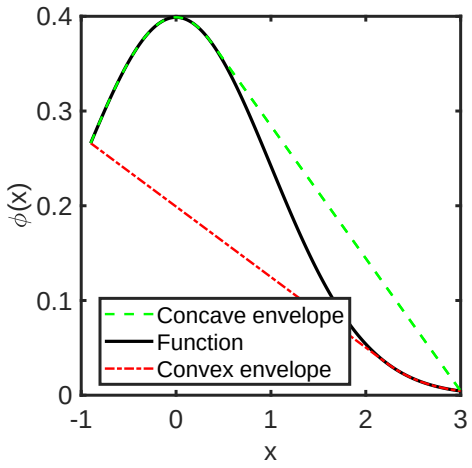


Figure 3.1.: Illustration of the envelope of the Gaussian PDF.

The envelopes of the error function are already available in MC++ as an intrinsic function. The envelopes of the CDF can be computed through the McCormick technique and the envelopes of the error function (see Figure 2a in ESI). In contrast, the error function is not available as an intrinsic function in BARON and a closed-form expression does not exist. Thus, a numerical approximation is required for optimization in BARON. Common numerical approximations of the error function are only valid for $x \geqslant 0$ and use point symmetry of the error function. To overcome this technical difficulty in BARON, a big-M formulation with additional binary and continuous variables is a possible workaround. However, this workaround leads to potentially weaker relaxations (see Section 2 in the ESI).

3.4.4 Lower confidence bound acquisition function

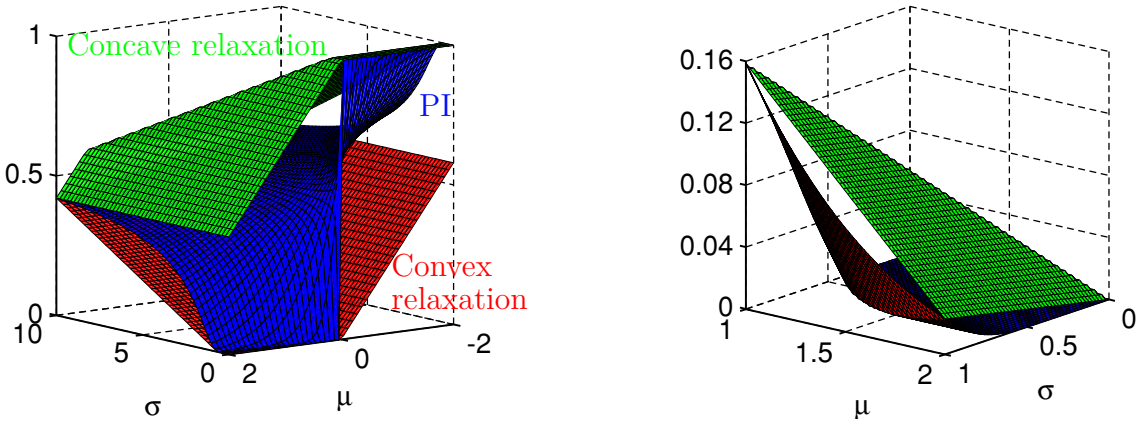
The lower confidence bound (LCB) (upper confidence bound when considering maximization) is an acquisition function with strong theoretical foundation. For instance, a bound on its cumulative regret, i.e., a convergence rate for Bayesian optimization, for relatively mild assumptions on the black-box function is known [233]. It is given by $\mathrm{LCB} : \mathbb{R} \times \mathbb{R}_{\geqslant 0} \to \mathbb{R}$ with

$$\mathrm{LCB}(\mu, \sigma) := \mu - \kappa \cdot \sigma$$

with a parameter $\kappa \in \mathbb{R}_{>0}$. LCB has not been popular in engineering applications as it requires an additional tuning parameter κ and leads to heavy exploration when a rigorous value for κ is chosen [233]. Recently, LCB has gained more popularity through the application as a policy in deep reinforcement learning, e.g., by DeepMind [234]. LCB is a linear function and thus McCormick relaxations are exact.

3.4.5 Probability of improvement acquisition function

Probability of improvement (PI) computes the probability that a prediction at x is below a given target $f_{\min}$, i.e., $\tilde{\mathrm{PI}}(\boldsymbol{x}) = \mathcal{P}\left(f(\boldsymbol{x}) \leqslant f_{\min}\right)$. When the underlying function is distributed as a GP with mean μ and variance σ, the PI is given by $\mathrm{PI} : \mathbb{R} \times \mathbb{R}_{\geqslant 0} \to \mathbb{R}$ with



(a) (b)

Figure 3.2.: Graph of the probability of improvement acquisition function (PI) as in Equation (3.5) for $f_{\min} = 0$ along with the developed convex and concave relaxations. (a) On the interval $[-2, 2] \times [0, 10]$, the relaxations are constructed on the basis of monotonicity properties of PI. (b) On the interval $[1, 2] \times [0, 1]$, the relaxations are constructed on the basis of componentwise convexity properties via the methods of Meyer and Floudas [235] and Najman et al. [236]. Note that the ranges of μ and σ are different in the two subfigures to highlight the individual relaxations that are derived on different intervals. The ranges in Subfigure (a) are such that they overlap with all four sets I_1-I_4 defined in Section A.3, while the ranges in Subfigure (b) lie within the set I_4.

$$\mathrm{PI}(\mu, \sigma) := \begin{cases} \Phi\left(\frac{f_{\min}-\mu}{\sigma}\right), & \sigma > 0, \\ 0, & \sigma = 0,\ f_{\min} \leqslant \mu, \\ 1, & \sigma = 0,\ f_{\min} > \mu. \end{cases} \tag{3.5}$$

The PI acquisition function is neither convex or concave over its entire domain. However, as analyzed in Section A.3, there are parts of its domain over which the function is *componentwise* convex or convex with respect to σ or μ. For componentwise convex or concave functions, there exist methods for constructing tight relaxations. [235] introduced a method for constructing concave relaxations for componentwise convex functions (or vice versa). In particular, the concave envelope of a componentwise convex functions is polyhedral [237], and thus the method of [235] amounts to finding the correct combinations of corners of the considered interval box to construct the facets of the polyhedral concave envelope. [236], in contrast, introduce a method for constructing convex relaxations of componentwise convex functions that satisfy a certain monotonicity condition on their first order partial derivatives (or, in case of twice continuously differentiable functions, have mixed second-order partial derivatives with constant sign over the box). For functions that are componentwise convex with respect to some and concave with respect to other variabels, [236] also show that by taking the secant with respect to the concave (or convex) variables, one can obtain a relaxation that is componentwise convex (or concave) with respect to all variables. Using the aforementioned methods, this function can then be further relaxed to obtain convex (or concave) relaxations.

We use these methods that exploit componentwise convexity along with techniques that exploit monotonicity properties to construct tight relaxations of the PI acquisition func-

tion. The procedure for constructing these relaxations is described in detail in Appendix A.2. Examples for the resulting relaxations on two subsets of the domain of PI are shown in Figure 3.2.

3.4.6 Expected improvement acquisition function

EI is the acquisition function that is most commonly used in Bayesian optimization [208]. It is defined as $\tilde{\mathrm{EI}}(\boldsymbol{x}) = \mathbb{E}\big[\max(f_{\min} - f(\boldsymbol{x}), 0)\big]$. When the underlying function is distributed as a GP, $\mathrm{EI} : \mathbb{R} \times \mathbb{R}_{\geqslant 0} \to \mathbb{R}$ is given by

$$\mathrm{EI}(\mu, \sigma) \coloneqq \begin{cases} (f_{\min} - \mu) \cdot \Phi\left(\frac{f_{\min}-\mu}{\sigma}\right) + \sigma \cdot \phi\left(\frac{f_{\min}-\mu}{\sigma}\right), & \sigma > 0 \\ f_{\min} - \mu, & \sigma = 0, \quad \mu < f_{\min} \\ 0 & \sigma = 0, \quad \mu \geqslant f_{\min} \end{cases} \tag{3.6}$$

As noted by Jones et al. [208], EI is componentwise monotonic and thus, exact interval bounds can easily be derived. In Section A.4, we show that EI is convex and we provide its envelopes. As EI is not available as an intrinsic function in BARON, an algebraic reformulation is necessary that uses Equation (3.6) where Φ is substituted from Equation (3.4) with Equation (1) in ESI and ϕ from Equation (3.3). In addition, some workaround would be necessary for $\sigma = 0$ (e.g., additional binary variable and big-M formulation).

3.5 Implementation

The described methods are implemented in our open-source solver MAiNGO [178] and the MeLOn toolbox [132]. The modeling interfaces of MAiNGO (currently either text-based input or a C++ API) allow a convenient implementation of RS models without having to eliminate variables symbolically. Instead, the sequential evaluation of model equations can be expressed as in procedural programming paradigms.

MAiNGO implements a spatial B&B algorithm enhanced with some features for range reduction [183–185] and a multi-start heuristic. A directed acyclic graph representation of the model is constructed using the MC++ library [180] and evaluated in different arithmetics: We use automatic differentiation via FADBAD++ [188] to obtain first and second derivatives of functions and provide them to the desired local solver. Currently, MAiNGO supports local solvers found in the NLopt package [187], IPOPT and Knitro. These local solvers can be used for pre-processing and solving the upper bounding problems. In the presented manuscript, we use SLSQP [238] in the pre-processing and in the upper bounding. During pre-processing, a simple multistart heuristic initializes the first local search at the center point of the variable ranges. Subsequent local searches are initialized randomly within the variable ranges.

MAiNGO constructs (multivariate) McCormick relaxations of factorable functions [167, 170]. The convex and concave relaxations together with their subgradients [163] are constructed through the MC++ library [180]. The necessary interval extensions are provided through FILIB++ [181]. MAiNGO currently supports CPLEX [182] and CLP [239] as linear programming solvers for lower bounding. In this work, the convex relaxations of the objective and the constraints are linearized at the center point of each node. Subsequently, CPLEX [182] solves the resulting linear problems for lower bounding. MAiNGO can also

be run in parallel on multiple cores through MPI. For a fair comparison, we run all optimizations on a single core in this work.

The GP models, acquisition functions, and training scripts are available open-source within the MeLOn toolbox [132] and the relaxations of the corresponding functions are available through the MC++ library used by MAiNGO.

In order to install MAiNGO, please visit our public git repository at `https://git.rwth-aachen.de/avt.svt/public/maingo`. Our machine learning toolbox MeLOn comes as a submodule of MAiNGO and will be installed with MAiNGO. Currently, MAiNGO can be run using our C++ interface or using our own modeling language called ALE [240]. At the time of writing this manuscript, the authors develop a new Python interface for MAiNGO which will be available soon via pypi.

3.6 Numerical results

We now investigate the numerical performance of the proposed method on one core of an Intel Xeon CPU with 2.60 GHz, 128 GB RAM and Windows Server 2016 operating system. We present three case studies. We use MAiNGO version v0.2.1 and BARON v19.12.7 through GAMS v30.2.0 to solve different optimization problems with GPs embedded on a single core. Note that we use CPLEX as a lower bounding solver in both BARON and MAiNGO. In MAiNGO we use SLSQP [238] in the pre-processing and in the upper bounding. By default, BARON automatically selects NLP solvers and may switch between different NLP solvers.

First, we illustrate the scaling of the method w.r.t. the number of training data points on a representative test function. Herein, the estimate of the GP is optimized. Second, we consider a chemical engineering case study with a chance constraint, which utilizes the variance prediction of a GP. Third, we optimize an acquisition function that is commonly used in Bayesian optimization on a chemical engineering dataset.

3.6.1 Illustrative example & scaling of the algorithm

In the first illustrative example, the peaks function is learned by GPs. Then, the GP predictions are optimized on $\tilde{X} = \{x_1, x_2 \in \mathbb{R}\colon\ -3 \leqslant x_1, x_2 \leqslant 3\}$. The peaks function is given by $f_{peaks} : \mathbb{R}^2 \to \mathbb{R}$ with

$$f_{\text{peaks}}(x_1, x_2) :=$$
$$3\,(1-x_1)^2 \cdot e^{-x_1^2-\,(x_2+1)^2} - 10 \cdot\ (\frac{x_1}{5} - x_1^3 - x_2^5) \cdot e^{-x_1^2-x_2^2} - \frac{e^{-(x_1+1)^2-x_2^2}}{3}$$

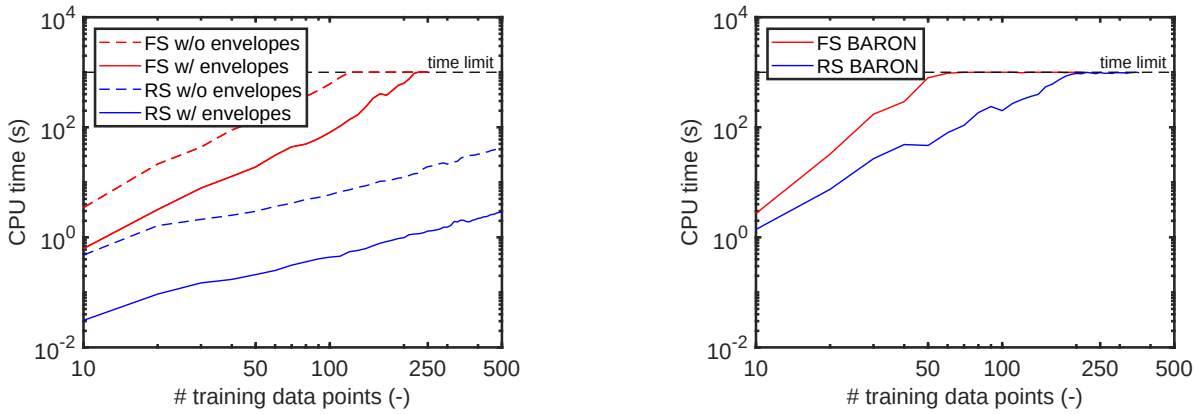
The two-dimensional function has multiple suboptimal local optima and one unique global minimizer at $\boldsymbol{x}^* \approx [0.228, -1.626]^T$ with $f_{\text{peaks}}(\boldsymbol{x}^*) \approx -6.551$.

We generate various training data on $\tilde{X}$ using a Latin hypercube sampling of sizes 10, 20, 30, ..., 500. Then, we train GPs with $k_{\nu=1/2}(d)$, $k_{\nu=3/2}(d)$, $k_{\nu=5/2}(d)$, and $k_{SE}(d)$ covariance functions on the data. The parameters of the trained GPs are saved in individual JSON files. After training, the JSON files are read by the solver and the predictions of the GPs are minimized using the RS and FS formulation to locate an approximation of the minimum of f_{peaks}. We run optimizations in MAiNGO once using the developed en-

velopes and once using standard McCormick relaxations. Due to long CPU times, we run optimizations for the FS formulations only for up to 250 data points in MAiNGO. The whole data generation, training, and optimization procedure are repeated 50 times for each data set. Thus, we train a total of 10,000 GPs and run 90,000 optimization problems in MAiNGO. We also solve the FS and RS formulation in BARON by automatically parsing the problem from our C++ implementation to GAMS. This is particularly important in the RS as equations with several thousand characters are generated. We solve the RS problem for up to 360 and the FS for up to 210 data points in BARON due to the high computational effort. The optimality tolerances are set to $\epsilon_{\text{abs. tol.}} = 10^{-3}$ and $\epsilon_{\text{rel. tol.}} = 10^{-3}$ and the maximum CPU time is set to 1,000 CPU seconds. The reported CPU times do not include any compilation time in MAiNGO and BARON. Note that the MAiNGO code is just compiled once for each problem class because the individual GPs are parameterized by JSON files. Thus, no repeated compilation is necessary. The feasibility tolerances are set to 10^{-6}. The analysis in this section is based on results for the $k_{\nu=5/2}$ covariance function. The detailed results for the other covariance functions show qualitatively similar results (c.f. ESI Section 4). Also, the results in this section are based on the median computational times of the 50 repetitions because the variations are comparably small. Boxplots that illustrate the variance are provided in Section 4 of the ESI.

In the FS, this problem has $D + 2 \cdot N + 2$ equality constraints and $2 \cdot D + 2 \cdot N + 2$ optimization variables while the RS has D optimization variables and no equality constraints. Note that for practical applications the number of training data points is usually much larger than the dimension of the inputs, i.e., $N \gg D$. The full problem formulation is also provided in ESI Section 3.

Figure 3.3 shows a comparison of the CPU time for optimization of GPs. For the



(a) MAiNGO (b) GAMS BARON

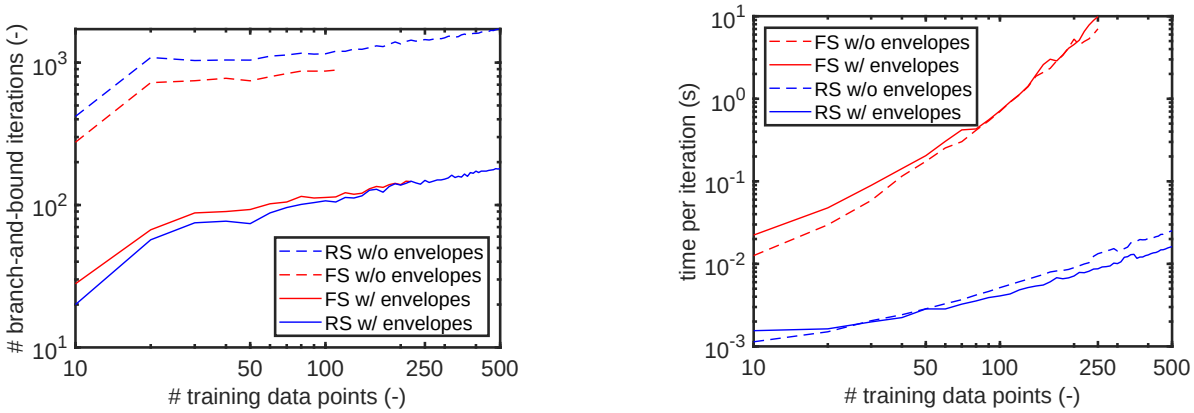
Figure 3.3.: Comparison of the total CPU time for optimization, i.e., the sum of preprocessing time and B&B time, of GPs with $k_{\nu=5/2}$ covariance function. The plots show the median of 50 repetitions of data generation, GP training, and optimization. Note that #points are incremented in steps of 10 and the lines are interpolations between them.

solver MAiNGO, Figure 3.3a shows that RS formulation outperforms the FS formulation by more than one order of magnitude and shows a more favorable scaling with the number of training data points. For example, the speedup increases to a factor of 778 for 250 data points. Notably, the achieved speedup increases drastically with the number

of training data points (c.f. ESI Section 4). This is mainly due to the fact that the CPU times for the FS formulations scale approximately cubically with the data points ($\text{CPU}_{FS\ w/\ env}(N) = 1.053 \cdot 10^{-4} N^{2.958}$ sec with $R^2 = 0.993$) while the ones for the RS scale almost linearly ($\text{CPU}_{RS\ w/\ env}(N) = 0.0022 \cdot N^{1.156}$ sec with $R^2 = 0.995$).

In general, the number of optimization variables can lead to an exponential growth of the worst-case B&B iterations and thus runtime. In this particular case, the number of B&B iterations is very similar for the FS and RS formulation (see Figure 3.4a). Instead, for the present problems the number of B&B iterations is more influenced by the use of tight relaxations. Figure 3.4b shows that the CPU time per iteration increases drastically with problem size in the FS while it increases only moderately in the RS. This indicates that the solution time of the lower bounding, upper bounding, and bound tightening subproblems scales favorably in the RS and that this is the main reason for speedup of the RS formulation in MAiNGO. This is probably due to the smaller subproblem sizes when using McCormick relaxations in the RS formulation (c.f. discussion in Section 3.1).

The use of envelopes of covariance functions also improves computational performance



(a) B&B iterations in MAiNGO

(b) CPU time per B&B iteration in MAiNGO

Figure 3.4.: Comparison of number of B&B iterations of optimization problems with GPs embedded with $k_{\nu=5/2}$ covariance function. The plots show the median of 50 repetitions of data generation, GP training, and optimization. Note that #points are incremented in steps of 10 and the lines are interpolations between them.

(see Figure 3.3a). However, this effect is approximately constant over the problem size (c.f. Figure 3 in ESI Section 4). In other words, the CPU time shows a similar trend for the cases with and without envelopes in Figure 3.3a. In the RS, the CPU time with envelopes takes on average 7.1% of the CPU time without envelopes ($\approx$ 14 times less). In the FS, the impact of the envelopes is less pronounced, i.e., the CPU time w/ envelopes is on average 15.1% of the CPU time w/o envelopes ($\approx$ 6.6 times less). Figure 3.4a shows that the envelopes considerably reduce the number of necessary B&B iterations. However, the relaxations do not show a significant influence on the CPU time per iteration (see Figure 3.4b).

The results of this numerical example show clearly that the development of tight relaxations is more important for the RS formulation than for the FS. As shown in Section 3.4.2 of [224], this effect can be explained by the fact that in RS, it is more likely to

have reoccurring nonlinear factors which can cause the McCormick relaxations to become weaker (c.f. also the relationship to the AVM in this case explored in [170]). However, in this study, the improvement in relaxations is outweighed by the increase of CPU time per iteration when additional variables are introduced in the FS.

The RS formulation also performs favorably compared to the FS formulation in the solver BARON (see Figure 3.3b). However, the differences between the CPU times are less pronounced. In contrast to MAiNGO, the number of B&B iterations in the FS and RS drastically increase with increasing number of training data points when using BARON (c.f. Figure 4 in ESI). Also, the time per B&B iteration is similar between RS and FS. This is probably due to the AVM method for the construction of relaxations. The AVM method introduces auxiliary variables for some factorable terms. Thus, the size of the subproblems in BARON increases with the number of training data points regardless of which of the two formulations is used.

The results of the optimizations also provide information about the ability of GP surrogate models to approximate a function for optimization. The results shows that the solution point of the GP optimization problem approximately converges to the optimum of the learned peaks function for all covariance functions. However, it is clear that some covariance functions lead to more accurate solution for the same number of training data points in this particular case. In the ESI, we provide figures that show the solution point and objective function value over the number of data points for this problem (Figures 8-11). Interestingly, the objective function value is overestimated considerably for all problems.

3.6.2 Chance-constrained programming

Probabilistic constraints are relevant in engineering and science [241] and GPs have been used in the previous literature to formulate chance constraints, e.g., in model predictive control [242] or production planning [243].

As a second case study, we consider the N-benzylation reaction of α methylbenzylamine with benzylbromide to form desired secondary (2°) amine and undesired tertiary (3°) amine. We utilize an experimental data set consisting of 78 data points from a robotic chemical reaction platform [9]. We aim to maximize the expected space-time yield of 2° amine (2°-STY) and ensure that the probability of a product quality constraint satisfaction is above 95%. The 2°-STY and yield of 3° amine impurity (3°-Y) are modeled by individual GPs. Thus, we solve optimization problems with two GPs embedded. The chance-constrained optimization problem is formulated as follows

$$\begin{aligned} \min_{\boldsymbol{x}\in E} \quad & -\mathbb{E}\big[f_{\mathrm{STY}}(\boldsymbol{x})\big] \\ \text{s.t.} \quad & \mathcal{P}\left(f_{\mathrm{impurity}}(\boldsymbol{x}) \leqslant c\right) \geqslant 95\% \end{aligned}$$

Here, the objective is to minimize the negative of the expected STY. This corresponds to minimizing the negative prediction of the GP, i.e., $-m_{\boldsymbol{\mathcal{D}},2^\circ-\mathrm{STY}}$. The chance constraint ensures that the impurity is below a parameter c with a probability of 95%. This corresponds to the constraint $m_{\boldsymbol{\mathcal{D}},3^\circ-Y} + 1.96 \cdot \sqrt{k}_{\boldsymbol{\mathcal{D}},3^\circ-Y} \leqslant c$ with $c = 5$.

The optimization is conducted with respect to four optimization variables: (1) the primary (1°) amine flow rate of the feed varying between 0.2 and 0.4 mL min^{-1}, (2) the ratio between benzyl bromide and 1° amine varying between 1.0 and 5.0, (3) the ratio between

Table 3.1.: Numerical results of the N-benzylation reaction optimization with chance constraint (Subsection 3.6.2). The FS formulation has 330 optimization variables, 326 equality constraints, and 1 inequality constraint. The RS formulation has 4 optimization variables, 0 equality constraints, and 1 inequality constraint. The CPU time limit is 86,400 seconds.

	Solver	CPU [s]	Iter.	UB	LB	Abs. gap
(FS)	MAiNGO w/ Env.	86,400	26,761	N/A	-379,2	N/A
	MAiNGO w/o Env.	86,400	20,795	N/A	-1,704.1	N/A
	BARON	86,400	219,239	-226.5	-53,775.0	53,548.5
(RS)	MAiNGO w/ Env.	86,400	$1.6 \cdot 10^6$	-226.5	-244.8	18.3
	MAiNGO w/o Env.	86,400	$1.1 \cdot 10^6$	-226.5	-573.2	346.7
	BARON	86,400	7,927	-226.5	-56,394.0	56,167.5

solvent and 1° amine varying between 0.5 and 1.0, and (4) the temperature varying between 110 and 150°C.

As this problem is highly multimodal and difficult to solve, we increase the number of local searches in pre-processing in MAiNGO to 500 and increase the maximum CPU time to 24 hours. The computational performance of the different methods is given in Table 3.1. The results show that none of the considered methods converged to the desired tolerance within the time limit. The RS formulation in MAiNGO that uses the proposed envelopes outperforms the other formulations and BARON solver as it yields the smallest optimality gap. Note that the considered SLSQP solver does not find any valid solution point in the FS in MAiNGO while feasible points are found in the RS. This demonstrates that the RS formulation can also be advantageous for local solvers. Note that when using IPOPT [244] with 500 multistart points in the FS formulation in MAiNGO, it identifies a local optimum with $f^* = -226.5$ in the pre-processing. In the ESI, we provide a brief comparison of a few pre-processing settings for this case study.

The best solution of the optimization problem that we found is $x_1 = 0.40$ min^{-1}, $x_2 = 1.0$, $x_3 = 0.5$, and $x_4 = 123.5$°C. At the optimal point, the predicted 2°-STY is 226.5 kg m^{-3} $\text{h}^{-}1$ with a variance of 17.1 while the predicted amine impurity is 4.2 % with a variance of 0.17. The result shows that the probability constraint ensures a safety margin between the predicted impurity and $c = 5$. Note that the chance constraint is active at the optimal solution point.

3.6.3 Bayesian optimization

In the third case study, we consider the synthesis of layer-by-layer membranes. Membrane development is a prerequisite for sustainable supply of safe drinking water. However, synthesis of membranes is often based on try-and-error leading to extensive experimental efforts, i.e., building and measuring a membrane in the development phase usually takes several weeks per synthesis protocol. In this case study, we plan to improve the retention of Na_2SO_4 salt of a recently developed layer-by-layer nanofiltration membrane system. The optimization variables are the sodium chloride concentration in the polyelectrolyte solution $c_{NaCl} \in [0, 0.5]$ $\text{gL}^{-}1$, the deposited polyelectrolyte mass $m_{PE} \in [0, 5]$ gm^{-2}, and the number of layers $N_{layer} \in \{1, 2, 3, .., 10\}$. The detailed description of the setup is given in the literature [21, 245]. Overall, we utilize 63 existing data points from previous

Table 3.2.: Numerical results of the membrane synthesis optimization (Subsection 3.6.3). The FS formulation has 136 optimization variables and 133 equality constraints. The RS formulation has 3 optimization variables and no equality constraints.

	Solver	CPU [s]	Iter.	UB	LB	Abs. gap
(FS)	MAiNGO w/ Env.	3,802	25,995	-2.025	-2.027	0.002
(RS)	MAiNGO w/ Env.	405	14,331	-2.025	-2.027	0.002

literature [245]. We identify a promising synthesis protocol based on the EI acquisition function by solving:

$$\min_{\boldsymbol{x}\in E} \quad -\mathrm{EI}\big(m_{\mathcal{D}}(\boldsymbol{x}), k_{\mathcal{D}}(\boldsymbol{x})\big)$$

with $\boldsymbol{x} = [c_{NaCl}, m_{PE}, N_{layer}]^T$. Thus, this numerical example corresponds to one step of a Bayesian optimization setup for this experiment. Global optimization of the acquisition function is particularly relevant due to inherent multimodality of the acquisition functions [212] and high cost of experiments. Note that the experimental validation of this data point is not within the scope of this work.

The computational performance of the proposed method is summarized in Table 3.2. Using the solver MAiNGO, the RS formulation converges approximately 9 times faster to the desired tolerance compared to the FS formulation. Herein, we use the derived tailored relaxations of the EI acquisition function and envelopes of the covariance functions in both cases. Notably, the FS requires approximately 1.8 times the number of B&B iterations compared to the RS formulation, which is much less than the overall speedup. Thus, the results are in good agreement with the previous examples showing that the reduction of CPU time per iteration in the RS has a major contribution to the overall speedup. For this example, a comparison to BARON is omitted due to necessary workarounds including several integer variables and function approximations for CDF and EI (c.f., Section 3.4.3, 3.4.6).

The optimal solution point of the optimization problem is $c_{NaCl} = 0.362$ gL^{-}1, $m_{PE} = 0$ gm^{-2}, and $N_{layer} = 4$. The expected retention is 85.32 with a standard deviation $\sigma = 14.8$. The expected retention is actually worse than the best retention in the training data of 96.1. However, Bayesian optimization takes also the high variance of the solution into account, i.e., it is also exploring the space. EI identifies an optimal trade-off between exploration and exploitation.

3.7 Conclusions

We propose a RS formulation for the deterministic global solution of problems with trained GPs embedded. Also, we derive envelopes of common covariance functions or tight relaxations of acquisition functions leading to tight overall problem relaxations.

The computational performance is demonstrated on illustrative and engineering case studies using our open-source global solver MAiNGO. The results show that the number of optimization variables and equality constraints are reduced significantly compared to the FS formulation. In particular, the RS formulation results in smaller subproblems

whose size does not scale with the number of training data points when using McCormick relaxations. This leads to tractable solution times and overcomes previous computational limitations. For example, we archive a speedup factor of 778 for a GP trained on 250 data points. The GP training methods and models are provided in the open-source MeLOn toolbox [132].

We thus demonstrate a high potential for future research and industrial applications. For instance, global optimization of the acquisition function can improve the efficiency of Bayesian optimization in various applications. It also allows to easily include integer decisions and nonlinear constraints in Bayesian optimization. Furthermore, the proposed method could be extended to various related concepts such as multi-task GPs [246], deep GPs [247], global model-predictive control with dynamic GPs [199, 248], and Thompson sampling [18, 249]. Finally, the proposed work demonstrates that the RS formulation may be advantageous for a wide variety of problems that have a similar structure, including various machine-learning models, model ensembles, Monte-Carlo simulation, and two-stage stochastic programming problems.

Chapter 4

Obey validity limits of data-driven models

4.1 Introduction

Supervised machine-learning techniques have been re-emerging as a promising avenue for data-driven modeling in various engineering disciplines [38]. In most applications, the overall goal is the optimal decision-making based on available data and *a priori* knowledge. Thus, data-driven models and mechanistic models are often combined to form hybrid models [73, 91, 94, 221]. Subsequently, hybrid models are frequently used in model-based optimization of design and/or operation of processes [1, 69].

A critical issue of data-driven models is their limited extrapolability. Unless strong assumptions are posed on the learned function, data-driven models can only be valid in regions where they have sufficiently dense coverage of training data points. We refer to this as the *validity domain* [250]. Consequently, there is a need to avoid the evaluation of data-driven models outside their validity domain during optimization. Note that we refer to the validity domain of individual data-driven models throughout this work, but the concept can also be applied to hybrid models [98].

The vast majority of previous publications use box constraints (i.e., hyperrectangles) to bound the inputs of data-driven models, i.e., each variable has independent bounds. This approach is practical when the training data is obtained from simulations based on regular grids or Latin hypercubes that are sufficiently dense. It is also advantageous for local and global optimization. However, it requires *a priori* known bounds and the possibility to obtain training data for any input combination. In practice, simulations can fail [251] and industrial process data usually does not cover hyperrectangular spaces [252]. This leads to manual selection of wrong bounds, which may cut off optimal solutions or overestimate the validity domain.

As proposed by [250], a few previous works in process systems engineering (PSE) constructed the convex hull of the training data points to describe the validity domain and integrated it as a set of linear constraints in optimization problems [79, 98, 252]. By definition, the convex hull is the smallest convex set that contains all data points. Commonly, evaluations of data-driven models inside the convex hull of the training data are called interpolation and outside extrapolation. However, roughly speaking, the convex hull cannot distinguish between for potential holes in the training data set, gaps between separated clusters of data, and nonconvex boundaries. Thus, staying within the convex hull seems only a necessary condition for the validity of data-driven models and not sufficient. Identifying if the convex hull is a suitable model for the data domain is very challenging in high dimensions. Notably, [79] extended the convex hull method to the union of multiple polytopes by introducing binary variables and additional constraints to the problem.

However, this algorithm becomes impractical when the number of data points or their dimension is very high. Besides convex hull formulations, there exist also several publications that circumvent extrapolation problems en passant in different ways. For instance, [81] penalize deviations from a training data mean in a space that is parameterized using principal component analysis. [21] constrain the maximal allowed distance from the nearest training data point resulting in a nonsmooth optimization problem. [253] train multiple data-driven models on a design problem and reject designs where the variation between the models is large. Similarly, [254] use bootstrap aggregation to estimate error bounds for hybrid mechanistic/data-driven models. There exist further methods that quantify the variance or confidence interval of predictions such as Bayesian methods and maximum likelihood estimations [255]. However, this leads to chance-constrained programming problems [241, 256]. Likewise, there are a few studies on the adaptive exploration of the design space [257–259] and related works on constrained Bayesian optimization [207]. However, we focus on fixed training data sets in this study while the extension of our method to adaptive sampling is a promising future research.

An alternative to box constrains and convex hull is to use a nonlinear classifier that can also model complicated validity domains. A few previous studies in mechanical engineering [260, 261] used Support Vector Domain Description (SVDD) [262] to model the validity domain of data-driven equipment models. Also, [263] use binary support vector classification to include reliability constraints into model-based design of experiment. As only valid training data points are given in most engineering applications, we consider one-class classification in this work. One-class classification is an unsupervised machine learning technique. There exists a broad variety of one-class classifiers that can be divided into density methods, boundary methods, and reconstruction methods [264]. Also, one-class classification is closely related to novelty, outlier, or anomaly detection [52, 265–268]. The previous literature indicates that one-class support vector machines (SVMs) [269] are common and suitable for the problem at hand. Compared to density models, less training data is required to construct the boundary, since only the boundary is estimated and not a complete density distribution [264]. In addition, the one-class SVM is tolerant to outliers in the training set [52].

Optimization problems with one-class SVMs embedded are nonconvex. Thus, deterministic global optimization is desirable to identify global solutions. However, these models lead to large-scale optimization problems that are difficult to solve. In our previous work, we showed that a reduced-space (RS) formulation and the use of McCormick relaxations are advantageous for the optimization of two other important classes of data-driven models, namely artificial neural networks [1] and Gaussian processes [256]. We propose a similar idea here for one-class SVM.

The global shape of data matters because it often provides important information about the underlying phenomena represented by the data. Especially in high-dimensional data, topological data analysis (TDA) can reveal and quantify objects and features not directly visible to the human eye. In the context of this work, it provides valuable information about typologies in the training data that can be colloquially thought of as holes or separated clusters. TDA was initiated relatively recently [270, 271]. Its roots lie in applied (algebraic) topology and computational geometry [272] and it is commonly used to account for higher-order interactions in data, to comprehend mesoscale structures, or to compare different data spaces [273]. The most common TDA method is persistent homology [274]. So far, there are only a few applications of persistent homology in the fields of

(bio-)chemical engineering and material science [275–279].

We propose a three-step approach to model the validity domain of data-driven models for optimization. We first perform TDA using persistent homology. In case clusters or holes are identified, we train a one-class SVM on the training data domain of the data-driven models and encode it as constraints in the subsequent process optimization. Otherwise, we construct the convex hull of the data and encode it as constraints. We finally perform deterministic global process optimization with the data-driven models and their respective validity constraints. To ensure computational tractability, we develop a RS formulation for trained one-class SVMs. Moreover, we employ convex and concave envelopes of kernel functions to accelerate optimization. We demonstrate the potential of our method on a set of illustrative mathematical case studies and an engineering case study, i.e., the open-loop control of a sulfur recovery unit.

4.2 Methodology

As illustrated in Figure 4.1, we propose a three step approach to obey validity limits of data-driven models during optimization. In the first step, we conduct a TDA of the training data. In the second step, we either construct the convex hull of the data or we train a one-class classifier, i.e., a SVM. In the third step, we embed the trained classifier or convex hull in the optimization problem and solve it to global optimality. The described methods are available open-source. We use the *Ripser.py* toolbox that is available open-source under MIT license in Python for performing the TDA [280]. The training of the one-class SVM is performed by Scikit-learn [281] and the convex hulls are identified using SciPy [282]. We provide the one-class SVM within the MeLOn toolbox under the Eclipse public license [132]. The resulting optimization problems are solved using our open-source global solver MAiNGO [178].

4.2.1 Topological data analysis using persistent homology

In persistent homology, we are interested in so-called topological invariants, i.e., properties that are invariant under homeomorphisms. The topological invariants of interest are homology groups, i.e., H_k of dimension k, with $\beta_k = \dim(H_k)$ being the Betti numbers [283, 284]. "Informally, β_0 is the number of connected components, β_1 is the number of two-dimensional holes or "handles" and β_2 is the number of three-dimensional holes or "voids" etc." [283].

The topological invariants are computed by representing the original dataset, i.e., a point cloud, as a simplicial complex through a simplicial filtration. We utilize the common Vietoris-Rips filtration, where a n-simplex in the simplicial complex is formed if and only if the pairwise distance between all points in the n-simplex is at most ϵ. At the bottom of Figure 4.2, we show a series of simplicial complexes for an illustrative point cloud.

Persistent homology studies topological invariants that persist over multiple length scales (ϵ) in the data [277, 278, 286, 287]. In other words, we examine the lifespan of topological invariants by increasing ϵ incrementally and constructing simplicial complexes. At the bottom of Figure 4.2, we can observe that $\beta_0 = 10$ connected components (H_0) exist at ϵ_1, $\beta_0 = 5$ connected components exist at ϵ_2, $\beta_0 = 1$ connected component and $\beta_1 = 1$ two-dimensional hole (H_1) exist at ϵ_3, and $\beta_0 = 1$ connected component exist at ϵ_4.

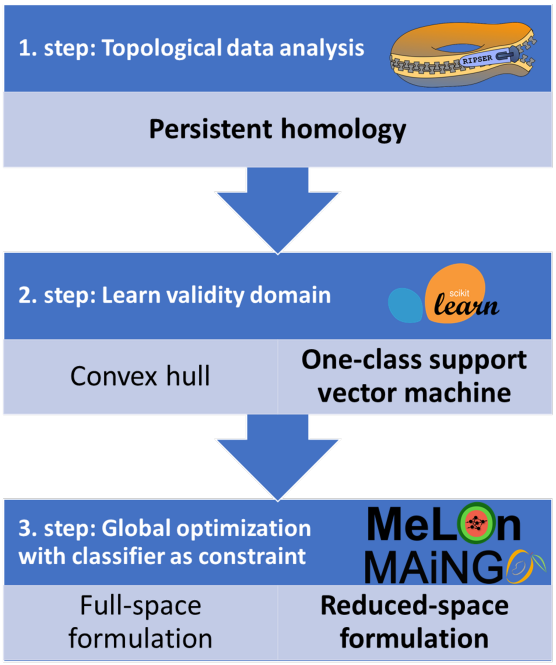


Figure 4.1.: Overview of the proposed three step methodology to obey validity limits of data-driven models in optimization.

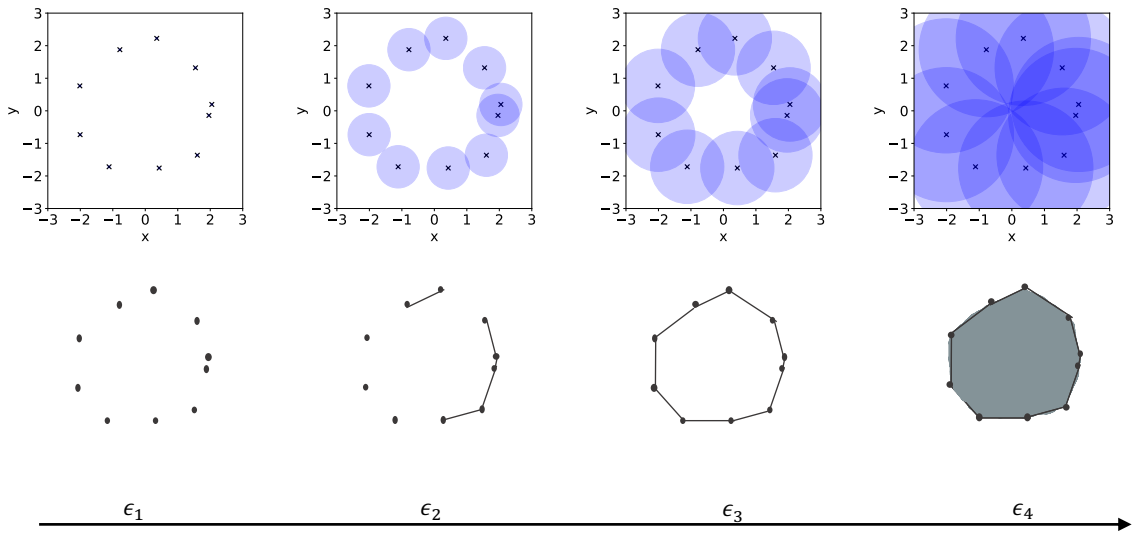


Figure 4.2.: Illustration of a Vietoris-Rips filtration utilized for persistent homology. The upper part shows the data set and circles around the data points with increasing diameter ϵ. The bottom image illustrates the simplicial complexes formed during the filtration. The figure is based on [285].

The results of the persistent homology can be depicted in barcode diagrams or persistent diagrams. We use the more common persistent diagrams in this work. The coordinates of birth and death of the homology groups in the example are shown in the persistent diagram in Figure 4.3. The x-axis represents the ϵ_{birth} while the y-axis the ϵ_{death} distance of H_0 and H_1 homology groups. Features with long lifespan correspond to points far from the diagonal [274]. The blue triangle at the bottom left corner of the plot corresponds to the merge of two very close data points at small ϵ. The blue triangles with ϵ_{death} between 1 and 1.5 in Figure 4.3 resemble the merge of connected components between ϵ_2 and ϵ_3 in Figure 4.2, describing the decrease in Betti number β_0 from 5 to 1. The highest blue triangle illustrates that one connected component exists until infinite. The red circle represents homology group H_1 and demonstrates the birth and death of the two-dimensional hole which is formed around ϵ_3 and dies before ϵ_4 in Figure 4.2.

In this example, the persistent diagram shows that a hole exist providing useful insight

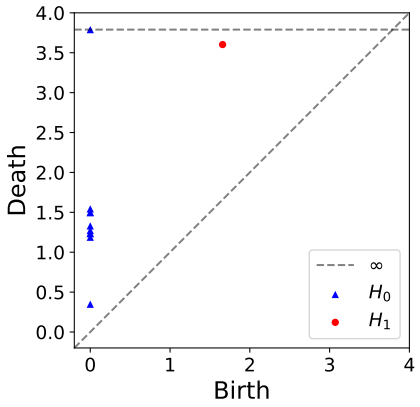


Figure 4.3.: Persistent homology plot of the illustrative point cloud. The x-axis shows the birth and the y-axis the death of the homology groups.

to guide the decision process for model selection. For example, the ϵ_{death} of the H_1 components provide information about the data density. In the example, the maximal distance in the last H_1 component is at most 1.5. This is significantly smaller than the ϵ_{death} of the H_1 hole. In other words, the life span of the hole is characteristic for the dataset.

4.2.2 Learn validity domain using one-class support vector machines

We model the validity domain using the convex hull or one-class SVM approach. The one-class SVMs are trained using the open-source python implementation in Scikit-learn [281] and the convex hulls are identified using the open-source implementation in SciPy [282]. The details of the one-class SVM are described in the following.

SVMs are a popular method for binary classification [288] and regression [289]. One-class SVMs are a modification of these classical SVMs [269] (c.f. [264] on similarity to SVDD). The goal is to learn a boundary of a given set of training points $X = \{\hat{\boldsymbol{x}}^{(1)}, ..., \hat{\boldsymbol{x}}^{(i)}, ..., \hat{\boldsymbol{x}}^{(N)}\}$ with $\hat{\boldsymbol{x}}^{(i)} \in \mathbb{R}^D$. Similar to classical SVMs, the data is mapped to high-dimensional feature space by $\phi : \mathbb{R}^D \mapsto \mathbb{R}^d$ with $d >> D$ and later solved in the dual formulation using the *kernel trick* [290]. In the feature space, a maximum-margin hyperplane is found that

separates the data from the origin by solving:

$$\min_{\boldsymbol{w}\in\mathbb{R}^d,\xi_i\in\mathbb{R},\rho\in\mathbb{R}} \quad \frac{1}{2}\boldsymbol{w}^T\boldsymbol{w} - \rho + \frac{1}{\nu N}\sum_{i=1}^{N}\xi_i, \tag{4.1}$$

$$\text{s.t} \quad \boldsymbol{w}^T\phi(\hat{\boldsymbol{x}}^{(i)}) \geqslant \rho - \xi_i \qquad \forall i = \{1,..,i,..N\}, \tag{4.2}$$

$$\xi_i \geqslant 0 \qquad \forall i = \{1,..,i,..N\}, \tag{4.3}$$

where $\nu \in (0,1)$ is a regularization hyperparameter, $\xi_i \in \mathbb{R}$ are slack variables, and $\boldsymbol{w}$ and ρ are the parameters of the hyperplane in high-dimensional feature space. [269] show that ν is an upper bound on the fraction of outliers and a lower bound on the fraction of support vectors in the training set, which is known as the ν-property. The decision function $f_{\text{DF-P}}(\boldsymbol{x}) = \boldsymbol{w}^T\phi(\boldsymbol{x}) - \rho$ is positive if a candidate point $\boldsymbol{x}$ is classified to be within the training data domain and negative if not. The dual of (4.1)-(4.3) is the quadratic program:

$$\min_{\alpha_i\in[0,\frac{1}{\nu N}]} \quad \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N}\alpha_i\alpha_j K(\hat{\boldsymbol{x}}^{(i)},\hat{\boldsymbol{x}}^{(j)}), \tag{4.4}$$

$$\text{s.t} \quad \sum_{i=1}^{N}\alpha_i = 1, \tag{4.5}$$

where α_i are dual variables and $K(\cdot,\cdot)$ is a kernel function. Often, the radial basis kernel function $K(\boldsymbol{x},\boldsymbol{y}) = \exp(-\gamma\|\boldsymbol{x}-\boldsymbol{y}\|^2)$ with hyperparameter γ is used since it has been shown that it is best able to model the most complex boundaries [264]. It holds that $\alpha_i = 0$ for training samples inside the learned boundary and $\alpha_i > 0$ for samples on or outside the boundaries. Samples for which $\alpha_i > 0$ are called support vectors. The decision function in the dual variables is given by $f_{\text{DF-D}}(\boldsymbol{x}) = \sum_{i\in I_{\text{sv}}}\alpha_i K(\hat{\boldsymbol{x}}^{(i)},\boldsymbol{x}) - \rho$, where I_{sv} denotes the indexes of the support vectors in the training data (i.e., the data points with corresponding $\alpha_i > 0$). To obey validity limits of data-driven models in an optimization problem, the following inequality has to hold:

$$f_{\text{DF-D}}(\boldsymbol{x}) \geqslant 0. \tag{4.6}$$

The parameter ν can be estimated from an outlier fraction by using the aforementioned ν-property. This makes this method more tolerant to outliers in the training data [52]. The hyperparameter γ controls the model complexity when using the radial basis kernel. If a large γ is used, all samples are mapped to a small region in the feature space and the one-class SVM cannot distinguish between the samples well. In other words, the model lacks complexity. If γ is small, pairs of samples become orthogonal in the feature space. This leads to overfitting and a high number of support vectors. A common approach to identify an appropriate γ is to gradually decrease γ until the number of support vectors does not decrease much (e.g., [291]). However, automatically selecting an appropriate γ is challenging (e.g.,[292–294]).

4.2.3 Optimization with classifier as constraint

We consider a global optimization problem where a classifier is used to obey validity limits of a data-driven model. In most cases, the inputs of the classifier model correspond to the degrees of freedom $\boldsymbol{x}$ of the optimization problem. The classifier can determine if a given $\boldsymbol{x}$ is feasible or infeasible by evaluating its decision function $f_{\text{DF-D}}(\cdot)$. To obey validity limits, Inequality (4.6) has to hold. Although the decision function is an explicit function, there exist different ways to formulate it in optimization problems. These problem formulations are equivalent as they have the same solution, but they can have a large impact on the computational performance of global optimization.

In the FS formulation, a set of nonlinear equations is provided as equality constraints while the dependent (or intermediate) variables are optimization variables. Note that there exist multiple valid FS formulations depending on the equality constraints and optimization variables provided to the solver. One representative FS formulation for optimization with one-class SVMs embedded is shown in the following:

$$\min_{\boldsymbol{x}\in\mathbb{R}^D, z_{\text{obj}}\in\mathbb{R}, d_i\in\mathbb{R}, k_i\in\mathbb{R}, \boldsymbol{z}_{\text{dd}}\in\mathbb{R}^Z} \quad z_{\text{obj}}, \tag{4.7}$$

$$\text{s.t} \quad z_{\text{obj}} = f_{\text{obj}}(\boldsymbol{x}, \boldsymbol{z}_{\text{dd}}), \tag{4.8}$$

$$\boldsymbol{h}_{\text{dd}}(\boldsymbol{x}, \boldsymbol{z}_{\text{dd}}) = \boldsymbol{0}, \tag{4.9}$$

$$\sum_{i\in I_{\text{sv}}} \alpha_i k_i \geqslant \rho, \tag{4.10}$$

$$k_i = \exp\left(-\gamma \cdot d_i\right) \qquad \forall i \in I_{\text{sv}}, \tag{4.11}$$

$$d_i = \|\hat{\boldsymbol{x}}^{(i)} - \boldsymbol{x}\|^2 \qquad \forall i \in I_{\text{sv}}. \tag{4.12}$$

Herein, Equation (4.7) minimizes the objective function value z_{obj} that is given by Equation (4.8). Note that the objective depends on the variables of the data driven model $\boldsymbol{z}_{\text{dd}}$ that are given by the solution of Equations (4.9). The decision function of the one-class SVM is given by the inequality constraint (4.10) while its intermediate variables are given by the solution of Equations (4.11),(4.12). This FS formulation has in total $D + 2 \cdot |I_{\text{sv}}| + \dim(\boldsymbol{z}_{\text{dd}}) + 1$ optimization variables, $2 \cdot |I_{\text{sv}}| + \dim(\boldsymbol{z}_{\text{dd}}) + 1$ equality constraints, and one inequality constraint.

The equality constraints of the one-class SVM can be solved explicitly for the intermediate variables. Thus, we can directly formulate a RS formulation of the optimization problem (c.f. [165]):

$$\min_{\boldsymbol{x}\in\mathbb{R}^D} \quad f_{\text{RS}}(\boldsymbol{x}), \tag{4.13}$$

$$\text{s.t} \quad f_{\text{DF-D}}(\boldsymbol{x}) \geqslant 0. \tag{4.14}$$

Herein, $f_{\text{RS}}(\cdot)$ is the RS formulation of the data-driven model and objective function. Thus, Equation (4.13) results from sequential substitutions of Equations (4.7)-(4.9). This is possible as most data-driven models such as ANNs or GPs are explicit functions (c.f. [1, 256]) and as the objective function is a function of the degrees of freedom and the predictions of the data-driven model. Similarly, Equation (4.14) results from the substitution of Equations (4.10)-(4.12). The RS formulation has only D optimization variables and one inequality constraint.

The convex hull of a point cloud with a finite number of points can be formulated as a set of linear inequality constraints $X_{\text{convHull}} = \{\boldsymbol{x} \in \mathbb{R}^D \mid \boldsymbol{A}\boldsymbol{x} + \boldsymbol{b} \leqslant \boldsymbol{0}\}$. Assuming that the convex hull has f facets, the matrix $\boldsymbol{A} \in \mathbb{R}^{f \times D}$ and the vector $\boldsymbol{b} \in \mathbb{R}^f$ [98]. Thus, the FS and RS formulation of the convex hull are identical. Note that the data-driven model can still be formulated in the RS and FS formulation when using the linear convex hull constraints.

The RS formulation has three major advantages for global optimization: First, the problem formulation has a direct influence on the variables to be branched on. In the RS, the B&B solver branches only on the degrees of freedom $\boldsymbol{x}$. In the FS, the B&B solver branches on the degrees of freedom and also on the intermediate variables. This is undesirable given the exponential worst-case runtime of global optimization methods. Note that this issue can also be mitigated by selective branching [162]. Second, the size of the subproblems that are solved during optimization is affected by the problem formulation and the method for constructing relaxations. Our previous work shows that a combination of McCormick relaxations and RS formulation can reduce the time to solve an iteration of the B&B solver significantly [224, 256]. Third, global optimization solvers usually require bounds on all optimization variables. Often, meaningful bounds are known for degrees of freedom but bounds on intermediate variables can be difficult to determine. Note that this problem is mitigated by some state-of-the-art solvers through automatic bound tightening techniques.

The vast majority of previous literature approaches formulate global optimization in the FS because they frequently use modeling environments such as GAMS that essentially require an equation-oriented modeling approach. Recently, [295] developed the Python-based optimization tool Pyomo which allows modeling, implementation of own solvers, and provides access to multiple solvers. Pyomo allows both FS and RS and recently [229] demonstrated RS optimization of ANNs in BARON through Pyomo. However, BARON relies on the auxiliary variable method for relaxations which results in larger subproblems [256]. Thus, a RS formulation in BARON does not take full advantage of the RS formulation. In contrast, our open-source solver MAiNGO [178] relies on McCormick relaxations in the space of the original variables utilizing the library MC++ [163, 180]. Another open-source solver that allows for McCormick relaxations is called EAGO has been released by [296].

The optimization problems in this work are implemented in MeLOn [132] and solved by MAiNGO [178]. For comparison, the problems are additionally exported to GAMS and solved by the commercial solver BARON [157]. We provide the implementation of the one-class SVM in the open-source modeling toolbox MeLOn [132].

Tight convex and concave relaxations are highly desirable in global optimization. Therefore, we use the tightest possible relaxations, i.e., the envelopes, of the radial basis function kernel in our solver MAiNGO. Note that we derived these envelopes in our previous work [256] as the squared exponential covariance function in Gaussian processes is equivalent to the radial basis function kernel in SVMs.

4.3 Illustrative case studies

As high dimensional problems are difficult to visualize, we consider eight two-dimensional data sets for illustration of the proposed method in a first step. Afterwards, we consider an engineering case study in Section 4.4. As shown in Figure 4.5, the illustrative exam-

ples cover a variety of relevant scenarios. All data points are randomly generated within pre-specified bounds and perturbed by noise. Thus, the data sets do not exhibit sharp boundaries, rather they also include noisy outlier data points.

In the next step, we evaluate an adapted peaks function, $f_{\text{Peaks}} : \mathbb{R}^2 \mapsto \mathbb{R}$, on all data sets with $f_{\text{Peaks}}(x_1, x_2) = 3(1-x_1)^2 \cdot \exp\left(-x_1^2 - (x_2+1)^2\right) - 10(\frac{x_1}{5} - x_1^3 - x_2^5) \cdot \exp\left(-x_1^2 - x_2^2\right) - \frac{1}{3} \cdot \exp\left(-(x_1+1)^2 - x_2^2\right) - 1.3x_2$. Then, we train individual ANNs on the eight data sets using Keras [297]. All ANNs exhibit one input layer with 2 neurons, two hidden layers with six and eight neurons, respectively, and an output layer with one neuron. The hidden layers use tanh activation and the output layers use linear activation. For training, the inputs are scaled onto $[-1, 1]$ and the outputs are scaled to zero mean and unit variance. We further use a batch size of 128 and an epoch limit of 4,000. Note that we omit a hyperparameter study for the ANNs because ANN training is not the focus of this work.

All optimization problems are solved one core of an Intel Xeon CPU E5-2630 v3 (2.40GHz) with 192 GB RAM and Windows Server 2016 operating system. We use a 0.001 relative and absolute optimality tolerance, a CPU time limit of 1,000 seconds, and default settings in BARON and MAiNGO.

4.3.1 Topological data analysis

The persistent diagrams of the eight input data sets are shown in Figure 4.4. The Subfigures 4.4d and 4.4e show a H_1 component with a large life span each. These correspond to the holes in the respective data sets "box w/ hole" and "circle w/ hole". Moreover, the corresponding ϵ_{death} provide information about the diameter of the holes. Recall that H_1 components that are close to the diagonal have a short life span and are therefore not relevant for this analysis.

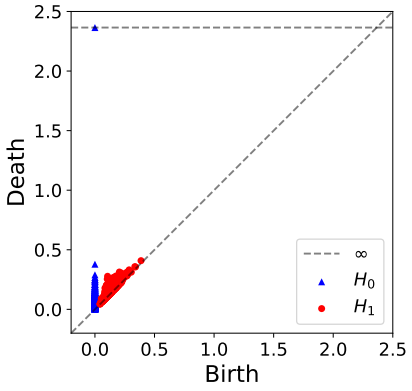
The persistent diagrams also show the existence of disjunct clusters in the data sets "two circles" and "two ovals". In Subfigures 4.4f and 4.4g, the H_0 components with a high ϵ_{death} indicate disjunct clusters and are a measure for the distance between them. Note that the H_0 components at the infinity line persist when ϵ goes to infinity and do not die. They correspond to the H_0 components that include all data points.

The persistent diagrams show no distinct differences between the "box", "oval", "box2", and "banana" case studies. This illustrates that the method cannot distinguish between convex and nonconvex data sets in general. Therefore, the persistent diagrams cannot ensure that the convex hull is sufficient to describe the validity domain. Rather, it can only identify some cases where the convex hull is not sufficient.

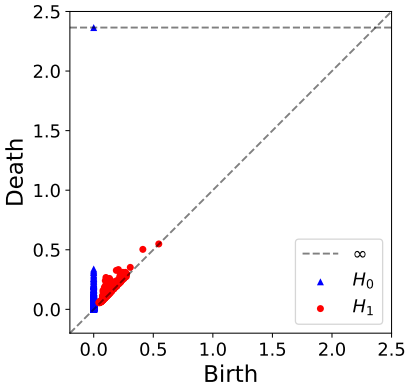
4.3.2 Validity domain modeling

In order to compare the one-class SVM and the convex hull, we model the input data of the eight case studies with both methods. We employ the common radial basis function kernel for the one-class SVM. We set the hyperparameter ν to a low value 0.03 because there are only a few outliers through noise in the data. The hyperparameter γ is identified using the incremental approach described in Section 4.2.2. The selected γ values are summarized in Table 4.1.

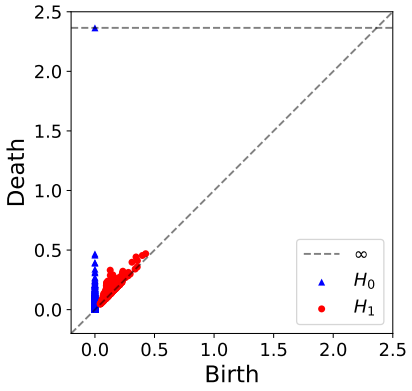
The learned boundaries for the case studies are depicted in Figure 4.5. As expected, the convex hull does not model holes and disjunct data clusters. Instead, the convex hull overestimates the validity domain of the case studies. In contrast, the one-class SVM is



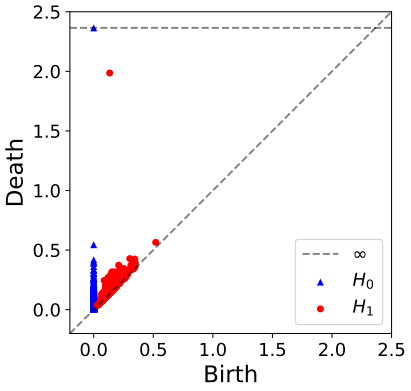
(a) “Box” case study



(b) “Oval” case study

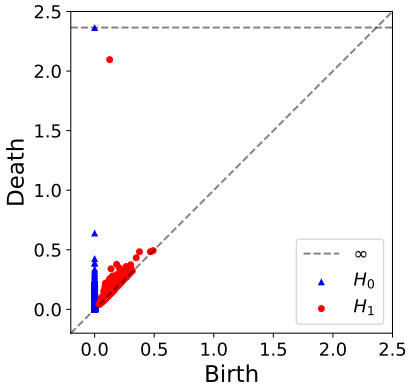


(c) “Box2” case study

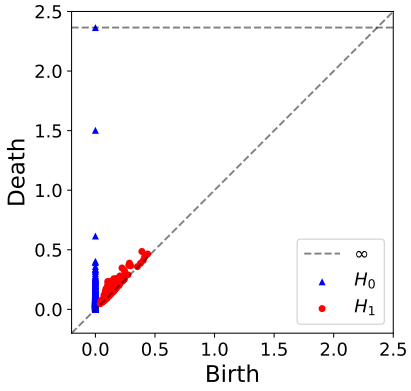


(d) “Box w/ hole” case study

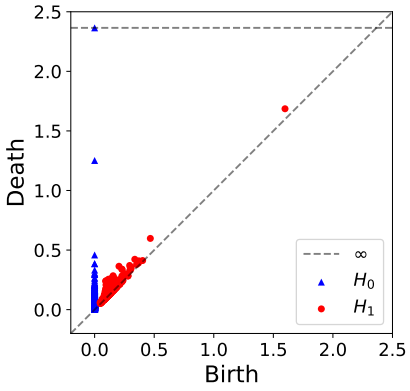
Figure 4.4.: Comparison of the persistent homology plots of the eight illustrative data sets ((a) to (d)).



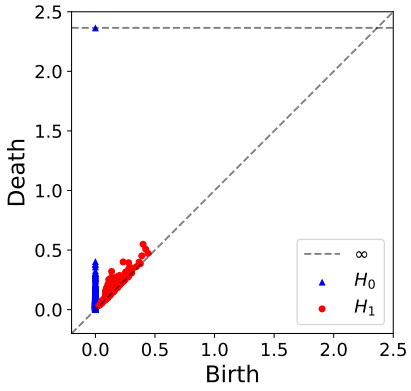
(e) “Circle w/ hole” case study



(f) “Two circles” case study



(g) “Two ovals” case study



(h) “Banana” case study

Figure 4.4.: Comparison of the persistent homology plots of the eight illustrative data sets ((e)-(h)).

Oval	Two circles	Box	Two ovals	Banana	Box2	Box w/ hole	Circle w/ hole
0.31	0.28	0.25	0.35	0.25	0.25	0.23	0.25

Table 4.1.: The values of the hyperparameter γ for the eight case studies. The hyperparameters are selected based on the incremental approach described in Section 4.2.2.

Case study	Convex hull x^*	Convex hull f^*_{ANN}	Convex hull Δ	One-class SVM x^*	One-class SVM f^*_{ANN}	One-class SVM Δ	Reference x^*
Banana	$(0.1, 2.0)$	-5.2	8.52	$(0.3, -1.6)$	-4.3	0.11	$(0.2, -1.6)$
Two circles	$(-1.5, 1.5)$	-5.1	3.65	$(1.0, 3.7)$	-4.8	0.01	$(1.3, 3.7)$
Box	$(0.2, -1.9)$	-5.4	2.14	$(0.3, -1.5)$	-4.5	0.07	$(0.2, -1.5)$
Box w/ hole	$(0.2, 3.6)$	-6.4	1.65	$(0.5, 3.2)$	-4.8	0.70	$(0.2, -1.6)$
Circle w/ hole	$(-1.5, 3.5)$	-5.0	0.41	$(0.1, 3.6)$	-4.6	0.02	$(0.2, 3.6)$
Two ovals	$(-1.3, 0.4)$	-3.5	0.13	$(-1.3, 0.4)$	-3.5	0.13	$(-1.3, 0.4)$
Oval	$(1.3, 3.5)$	-4.6	0.13	$(0.2, -1.6)$	-4.3	0.14	$(0.2, -1.6)$
Box2	$(0.1, -1.7)$	-4.2	0.05	$(2.8, 2.9)$	-3.8	0.05	$(2.9, 2.9)$

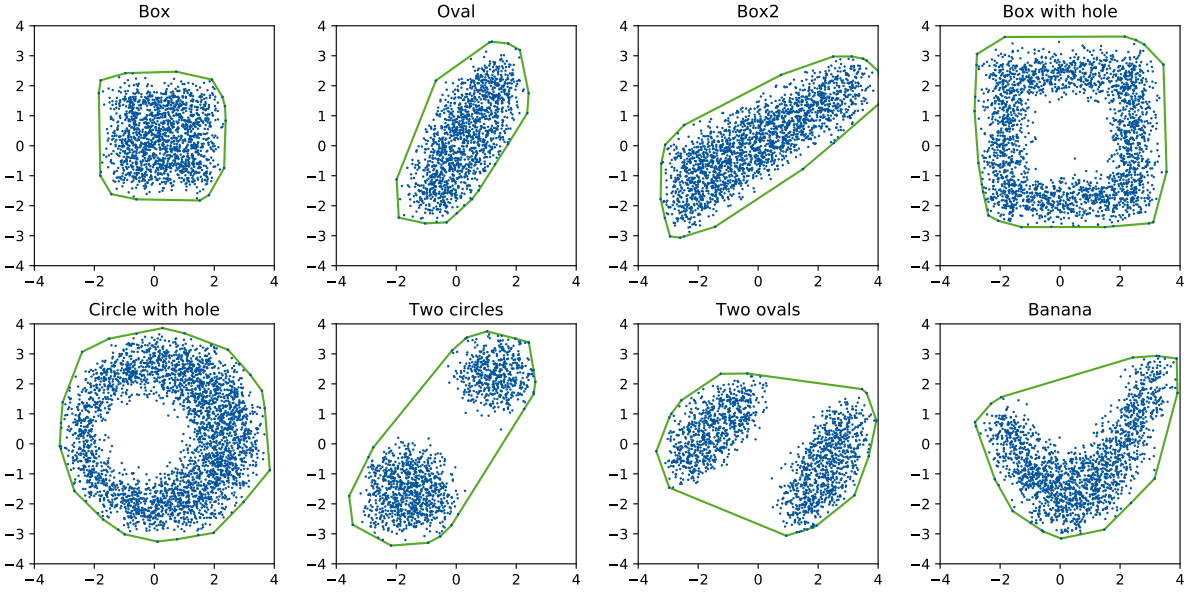
Table 4.2.: The table compares the global optimal solutions with convex hull and one-class SVMs (SVMs) as constraints. The optimal solution point is given by x^* and the objective function value is given by f^*_{ANN}. Also, we provide the error of the data-driven model at the optimal solution $\Delta = |f^*_{\text{ANN}} - f_{\text{Peaks}}(x^*)|$. The reference solution point shows the optimal solution when considering the underlying function f_{Peaks} as the objective with the one-class SVM constraint.

able to model holes and disjunct data clusters. Furthermore, the convex hull includes all data points whereas the one-class SVM also allows for outliers in the data and excludes regions with only small data density from the validity domain (e.g., see the "box" case study).

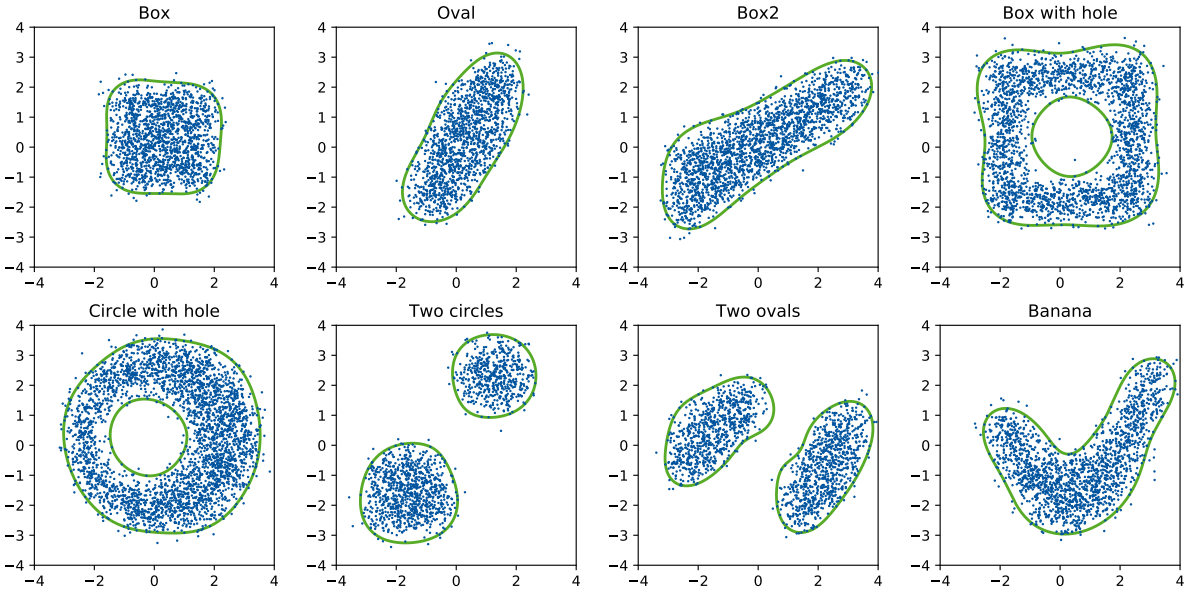
4.3.3 Optimization results

We minimize the prediction of the eight trained ANNs subject to the convex hull or the one-class SVM as constraints. Table 4.2 shows the optimal solution points, $\mathbf{x}^*$, and objective function values, $f_{\text{ANN}}(\mathbf{x}^*)$, for the problem with convex hull and one-class SVM as constraints.

The optimal objective function values of the convex hull approach are lower than the ones with the one-class SVM for all case studies because the convex hull overestimates the validity domain. This overestimation can lead to large errors at the optimal solution points. For the "banana" case study, the optimal solution found by the convex hull approach is outside the validity domain but at the boundary of the convex hull (see Table 4.2). This leads to a wrongly estimated objective values by the ANN of -5.2 with an absolute error of 8.52. In contrast, the one-class SVM models the validity domain accurately and yields an optimal solution of -4.3 with an absolute error of 0.11. Also, the solution point found by the ANN model with the one-class SVM constraint is close to the reference solution where



(a) Convex hulls of the eight illustrative data sets



(b) One-class SVM boundaries of the eight illustrative data sets

Figure 4.5.: Comparison of the convex hull and the boundaries learned by the one-class SVMs

Case study	# Sup. vec.	BARON FS	BARON RS	BARON sp-f	MAiNGO FS	MAiNGO RS	MAiNGO sp-f
Oval	48	158.2 s	35.0 s	**5**	197.2 s	0.20 s	**986**
Two circles	50	489.5 s	36.7 s	**13**	247.0 s	0.25 s	**988**
Box	52	346.6 s	25.8 s	**13**	139.9 s	0.24 s	**583**
Two ovals	58	341.3 s	38.6 s	**9**	433.2 s	0.22 s	**1,969**
Banana	62	1,000.0 s	70.6 s	**>14**	416.7 s	0.19 s	**2,193**
Box2	67	497.1 s	22.9 s	**22**	1,000.0 s	0.31 s	**>3,226**
Box w/ hole	81	1,000.0 s	73.1 s	**>14**	656.5 s	0.36 s	**1,823**
Circle w/ hole	103	1,000.0 s	75.8 s	**>13**	1,000.0 s	0.75 s	**>1,333**

Table 4.3.: CPU times for optimization of the eight case studies with the one-class SVM as a constraint. The table compares the FS and RS formulations for the BARON and MAiNGO solvers. Here, the data-driven model (i.e., the ANN) and the one-class SVM are formulated in the RS and FS. The speed up factor (sp-f) is given as the ratio between the FS and the RS solution times. Also, the number of support vectors (# Sup. vec.) is shown as a measure for the problem complexity.

the learned peaks function is optimized subject to the SVM constraint. Similarly, the one-class SVM leads to more reliable results in the data sets "two circles", "box w/ holes", and "circle w/ holes". Interestingly, the convex hull approach also leads to a substantial prediction error in the "box" case study while the one-class SVM models the validity domain accurately. This highlights the risk of using the convex hull approach in the presence of noise.

In Table 4.3, we provide the CPU times for optimization with one-class SVMs embedded. Using the FS formulation, BARON and MAiNGO perform similarly and solve most problems in the a few hundred CPU seconds. The RS formulation outperforms the FS formulation on all problem instances. In BARON, the speedup factor between the RS and the FS formulation ranges from 5 to over 14. In comparison, the speedup factor between the RS and FS in MAiNGO ranges between 583 to over 3,226. This is in agreement with our previous studies where the McCormick relaxations in the RS lead to smaller subproblems compared to the auxiliary variable method (see Section 4.2.3).

In Table 4.4, we compare the computational performance for optimization with the convex hull embedded. The CPU times with the convex hull are lower compared to the ones with one-class SVMs. MAiNGO is substantially faster than BARON when formulating the problem in the FS. On average, BARON requires about 83 seconds to solve the problem in the FS while MAiNGO requires only 3 seconds. The RS formulation again outperforms the FS formulation for all problems. However, in this case, the speedup factors are in the same order of magnitude for BARON and MAiNGO ranging between 13 and 54. It should be noted that the RS and the FS formulation of the convex hull constrains are identical. Therefore, the difference is only due to the formulation of the data-driven model in the objective function, i.e., the ANN (c.f. [1]).

Case study	# Sup. vec.	BARON FS	BARON RS	BARON sp-f	MAiNGO FS	MAiNGO RS	MAiNGO sp-f
Oval	48	74.9 s	2.8 s	**27**	2.7 s	0.05 s	**54**
Two circles	50	46.4 s	3.5 s	**13**	2.4 s	0.09 s	**27**
Box	52	85.8 s	1.5 s	**57**	1.7 s	0.06 s	**28**
Two ovals	58	103.9 s	3.1 s	**34**	3.8 s	0.08 s	**48**
Banana	62	136.4 s	5.1 s	**27**	3.0 s	0.09 s	**33**
Box2	67	32.3 s	2.3 s	**14**	2.7 s	0.11 s	**25**
Box w/ hole	81	51.3 s	3.5 s	**15**	2.4 s	0.08 s	**30**
Circle w/ hole	103	130.2 s	4.1 s	**32**	3.1 s	0.11 s	**28**

Table 4.4.: CPU times for optimization of the eight case studies with the convex hull as a constraint. The table compares the FS and RS formulations for the BARON and MAiNGO solvers. The speed up factor (sp-f) is given as the ratio between the FS and the RS solution times. Also, the number of support vectors (# Sup. vec.) is shown as a measure for the problem complexity.

4.4 Engineering application

We consider a sulfur recovery unit as a relevant engineering case study for our work because a large data set of industry operating data is available online for this process. The efficient recovery of sulfur in petroleum refineries from tail gas is important for environmental reasons. The sulfur recovery unit process is illustrated in Figure 4.6. The process has two acid gases as inputs: the MAE gas stream is rich in hydrogen sulphide (H_2S) and comes from the gas washing plants. The SWS gas stream is rich in H_2S and ammonia and comes from a sour water stripping plant. In the sulfur recovery unit, the acid gases are burnt via partial reaction with air in a two-chamber reaction furnace. Then, the combustion product is further treated in two subsequent catalytic reactors resulting in a tail gas stream that contains residuals of H_2S and sulfur dioxide (SO_2). A detailed process description can be found in the literature [298].

A key issue of the sulfur recovery unit is the control of the secondary air flow to ensure optimal conditions for the total removal of the sulfur compounds in the catalytic converters. Previous works have investigated soft sensors for the tail gas concentrations of hydrogen sulphide (H_2S) and sulfur dioxide (SO_2) using ANNs and implemented those in industry for monitoring [298–300]. In this case study, we solve an open-loop control problem to find the optimal secondary air flow rate. The objective is to minimize $|c_{H_2S} - 2 \cdot c_{SO_2}|$ such that the two reactants are in stoichiometric proportion. Similar to the previous literature by [298, 300], we also train two ANNs to predict the concentrations:

$$
\begin{aligned}
c_{H_2S,k} &= f_{ANN,H_2S}\left(x_{1,k}, x_{1,k-5}, x_{1,k-7}, x_{1,k-9}, ...x_{5,k}, x_{5,k-5}, x_{5,k-7}, x_{5,k-9}\right), \\
c_{SO_2,k} &= f_{ANN,SO_2}\left(x_{1,k}, x_{1,k-5}, x_{1,k-7}, x_{1,k-9}, ...x_{5,k}, x_{5,k-5}, x_{5,k-7}, x_{5,k-9}\right),
\end{aligned}
$$

where $x_{1,k}$ is the gas flow in the MEA zone, $x_{2,k}$ is the air flow in the MEA zone, $x_{3,k}$ is the secondary air flow in the MEA zone, $x_{4,k}$ is the air flow in the SWS zone, $x_{5,k}$ is the gas flow in the SWS zone at time step k. The SO_2 ANN has one hidden layer with eight neurons and the H_2S ANN has two hidden layers with eight neurons each. The data-driven

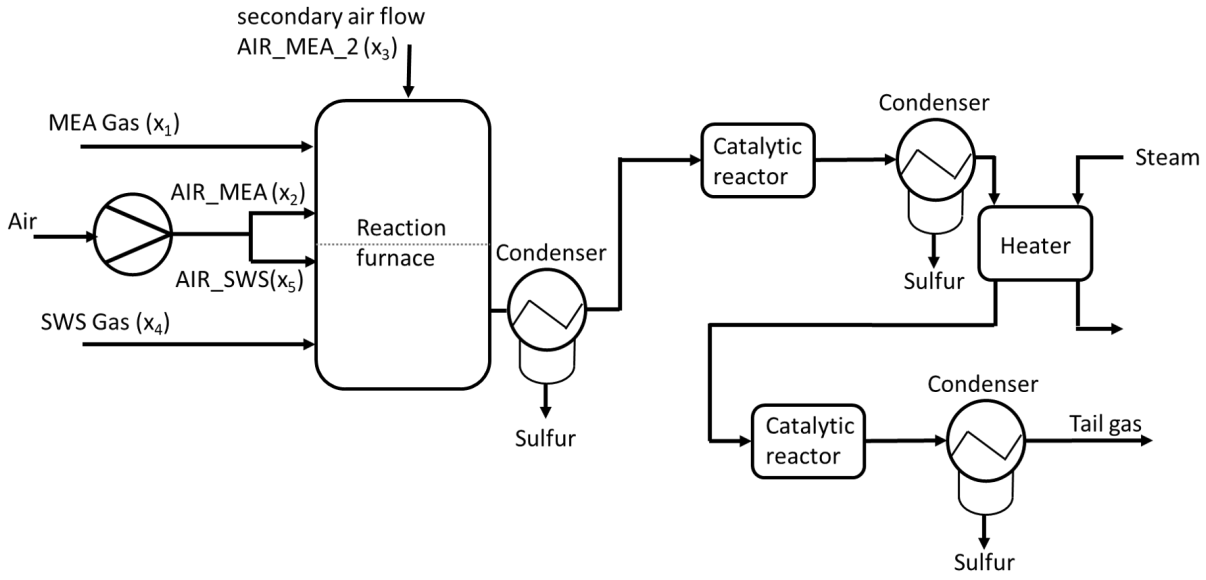


Figure 4.6.: Flowchart of the sulfur recovery unit process.

models are trained on (scaled) industrial data collected at a plant located in Priolo, Italy available at `https://www.openml.org/d/23515`. The data set includes a time series with approximately 10,000 data samples and we use the first 90% of the data for training. The control variable of the NMPC is the secondary air flow $x_{3,k}$ while the other inputs are observable parameters. As the control is critical for process safety, the validity limits of the data-driven model should be considered.

In order to analyze the topology of the 20-dimensional input training data set of ANNs, we perform persistent homology. Due to the large number of data points, the exact computation of the persistent diagram is expensive. We apply approximate sparse filtration instead [301]. The persistent diagram for this case study is shown in Figure 4.7. The diagram shows that there exist a number of holes in the data set that persist over a long time span. Also, a separate cluster can be observed in the data. This motivates the use of one-class SVM to obey validity limits of data-driven models.

As we have no physical model of the process available, the closed-loop performance of the controller is not studied in this example. For illustration, we perform one step of an open-loop controller for the secondary air flow $x_{3,k}$. We select a random operating point from the historic plant data (Table 4.5) and let the solver determine the optimal control action. The problem is solved to global optimality within 0.33 CPU seconds and identifies a control action $x_{3,k} = 0.266$ that results in the desired stoichiometric composition, i.e., $|c_{H_2S} - 2 \cdot c_{SO_2}| = 1.3 \cdot 10^{-5}$. This engineering case study also demonstrates the potential of the proposed method for NMPC. Note that deterministic global NMPC can become computationally expensive for long control horizons and higher dimensional control vectors [33, 302, 303].

4.5 Conclusion

Safety concerns and extrapolation issues often impede industrial applications of machine learning models. We present a three-step approach to obey the validity limits of data-

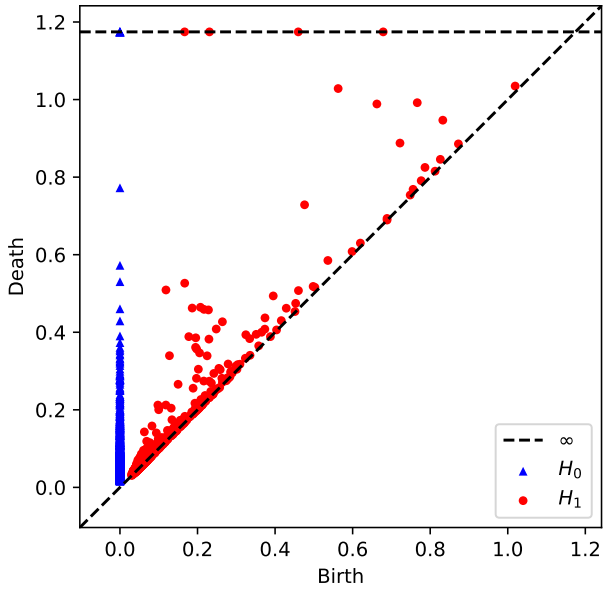


Figure 4.7.: Persistent diagram for of the training data of the engineering case study presented in Section 4.4.

	k	$k-5$	$k-7$	$k-9$
x_1	0.627	0.6215	0.623	0.622
x_2	0.770	0.769	0.754	0.769
x_3	$x_{3,k}$	0.174	0.192	0.198
x_4	0.376	0.399	0.415	0.410
x_5	0.513	0.512	0.511	0.504

Table 4.5.: Operating point of the sulfur recovery unit that is considered for the NMPC optimization step.

driven models. First, we perform a data topology analysis using persistent homology. Second, we model the validity domain of the data-driven model using either the convex hull or a one-class SVM. Third, we perform deterministic global optimization with the validity domain model as a constraint.

All used and developed methods are available open-source. Also, we currently develop a Python interface for our solver MAiNGO. Thus, all methods can be applied and further developed in academia and industry for free.

Our method has the potential to enhance safety, trust, and reliability of machine learning approaches. Moreover, we demonstrate that persistent homology is a valuable method for understanding the topology of data in high dimensional spaces. Besides industry applications, promising future work also includes the application to optimization problems occurring in molecular design where molecules are parameterized through graph neural networks [304] or autoencoders [305]. Also, time-dependent design space descriptions are desired in pharmaceutics [306]. The proposed method can also be extended by considering and comparing other one-class classification methods.

Chapter 5

Graph neural networks for prediction of fuel ignition quality

5.1 Introduction

The worldwide increase in CO_2 emissions and the depletion of fossil resources call for the development of renewable fuels. A wide range of non-oxygenated and oxygenated hydrocarbons derived from renewable resources such as biomass has been investigated as pure-component fuel or blend components for use in internal combustion engines [307–313]. To determine how suited a molecule is for a fuel application, combustion-related properties need to be evaluated. The cetane number (CN) or derived cetane number (DCN), the research octane number (RON), and the motor octane number (MON) are commonly employed to characterize the auto-ignition/knocking behavior of a particular fuel. Fuels with a high RON (MON) exhibit low knocking tendency and are therefore suitable for spark-ignition (SI) engines, whereas fuels with a high (D)CN (approx. above 40) exhibit a short ignition delay which is required in compression-ignition (CI) engines [310, 314, 315]. Experimental RON, MON, and (D)CN values are available for a range of different fuel molecules [316–318], however, for many interesting molecules such data is not readily available. For these molecules predictive models are required that enable rapid estimation of fuel ignition quality [310, 319].

In the past decades, several models have been developed to predict (D)CN [318–334] and RON/MON [318, 322, 335–338] of (oxygenated) hydrocarbons by utilizing quantitative structure-property relationship (QSPR) modeling. In QSPR, the modeling process can be broken down into two steps: First, QSPR models introduce molecular descriptors $\boldsymbol{D} = [d_1, d_2, ..., d_n]^T$ that depend on the structure of a molecule m. Second, a regression model $F(\boldsymbol{D}) : \boldsymbol{D} \mapsto \hat{p}$ is fitted that predicts a property $\hat{p}$ as a function of $\boldsymbol{D}$ [339]. The regression model is either linear or nonlinear, depending on the QSPR. Group contribution methods[340–342] are a particular type of QSPR model where the molecular descriptors $\boldsymbol{D}$ are structural group counts, i.e., the number of occurrences of basic functional groups, e.g., methyl ($-CH_3-$) or methylene ($-CH_2-$), in a molecule m.

QSPR models for DCN, RON, and MON differ in the way they encode the molecular structure. Various descriptors have been used including structural group counts (e.g., in [318–320, 327, 335–338]), the number of aromatic bonds (e.g., in [319, 333]), and topological indices, such as the Wiener Index [343] or branching indices (e.g., in [326, 332, 337]). Previous models have also used a variety of techniques for the regression step, e.g., linear or nonlinear regression [319, 320, 327, 332, 335, 336], or artificial neural networks (ANNs) [318, 323–325, 328, 330, 333, 336–338]. Development of QSPR models, however, highly depends on the choice of informative descriptors, a selection process that requires domain knowledge and intuition.

Deep learning allows to learn representations of data with multiple abstraction levels. This has shown remarkable success for end-to-end learning in various domains surpassing previously performed manual feature selection [40]. In particular, graph neural networks (GNNs) [344, 345] have recently shown promising results for the prediction of structure-property relationships of molecules [113, 346–350]. GNNs utilize graph representations of molecules, where atoms correspond to nodes and bonds correspond to edges. For each atom, its local environment is learned by graph convolutions. These atom environments are then combined into a molecular fingerprint by applying pooling functions [351]. Finally, an ANN maps the fingerprint to the molecular property of interest. Since the graph convolutions and pooling functions are differentiable, the full model can be trained with the backpropagation algorithm. In contrast to QSPRs, the processes of choosing molecular descriptors and performing property regression are thus merged into a simultaneous training step. This enables supervised end-to-end training from the molecular graph to the property. In particular, the molecular fingerprints adapt during training and learn molecular structure information that is important for the property of interest.

We propose the first GNN model for the prediction of DCN, RON, and MON. The model is trained on literature data and is applicable to a wide range of oxygenated and non-oxygenated hydrocarbons. The GNN architecture includes state-of-the-art higher-order molecular graph features [113] and is provided open-source [352]. Furthermore, we provide a web front-end that can be easily accessed online to make predictions. The web front-end takes SMILES strings as input and automatically predicts DCN, RON, and MON (`www.avt.rwth-aachen.de/gnn`).

One of the main challenges in GNN training in the context of fuel ignition quality is the limited availability of training data. To mitigate this issue, we propose three model extensions that reduce data requirements while achieving competitive prediction accuracy: First, we propose a multi-task learning approach where DCN, RON, and MON are trained jointly. This approach shares the graph convolutions and the molecular fingerprint among all prediction tasks and thus takes advantage of correlations between DCN, RON, and MON data sets. Second, we perform a transfer learning approach that utilizes a broader data set from different (D)CN measurement techniques for pre-training of our final model. Third, we perform ensemble learning averaging out random model variations.

The remainder of this chapter is structured as follows: In Section 5.2, we provide a general background on graph representations of molecules and GNNs. Then, we briefly describe the databases of this work in Section 5.3. Afterwards, we propose the GNN architecture in Section 5.4. Furthermore, we briefly describe the considered learning methods: multi-task learning, transfer learning, and ensemble learning. In Section 5.5, we present the results, discuss the different learning methods, and compare our GNN model to state-of-the-art QSPR models. Finally, we conclude our findings and show potentials for future research (Section 5.6).

5.2 Fundamentals of graph neural networks

In this section, we present a brief background on graph representations for molecules and GNNs. Furthermore, we introduce the concept of higher-order GNNs that is fundamental to our modeling approach.

5.2.1 Molecular graphs

Any molecule can be represented as a molecular graph where nodes $w, v \in V$ correspond to atoms. Edges $e_{vw} \in E$ correspond to bonds between two atoms [353, 354]. Furthermore, a feature vector is assigned to each node and each edge that includes information about atom types, e.g., C atom, and bond types, e.g., double bond. The node feature vectors $\boldsymbol{f}^V(v)$ can contain additional atom information such as orbital hybridization. To reduce the size of molecular graphs, hydrogen atoms can be implicitly included in the feature vectors of nodes of heavy atoms by using a hydrogen count, resulting in an H-depleted molecular graph [355]. Similarly, bond feature vectors $\boldsymbol{f}^E(e_{vw})$ can provide additional information, e.g., on ring structures. GNNs operate on graph structures, i.e., they take the molecular graph and its feature vectors as inputs.

5.2.2 Graph neural networks

As illustrated in Figure 5.1, GNNs have two main phases: (i) the message passing phase and (ii) the readout phase [348]. In the message-passing phase, graph convolutional layers are commonly applied with a large variety of layer structures existing [346, 348, 351, 356, 357].

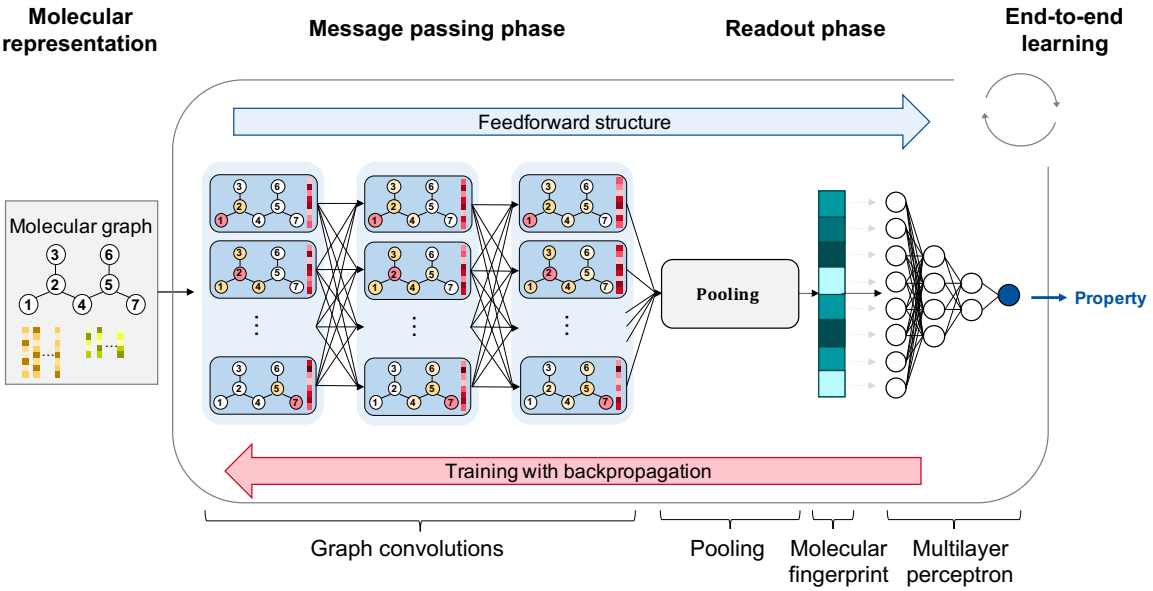


Figure 5.1.: Overview of a graph neural network model for property prediction.

Figure 5.2 illustrates the basic concept of graph convolutional layers. The overall goal of the graph convolutional layer is to combine node information of a considered node (here #2 in red) with node information of its neighbors (here #1,3,4 in yellow) and bond information (green). To this end, node state vectors and edge state vectors are combined through message and update functions as explained in more detail in the following paragraph. The result of the graph convolutional layer is an updated node state vector of the considered node.

Within a graph convolutional layer, information about a node's neighborhood $N(v) = \{w \mid e_{vw} \in E, w \neq v\}$ is aggregated and passed to the respective node [348]. The message

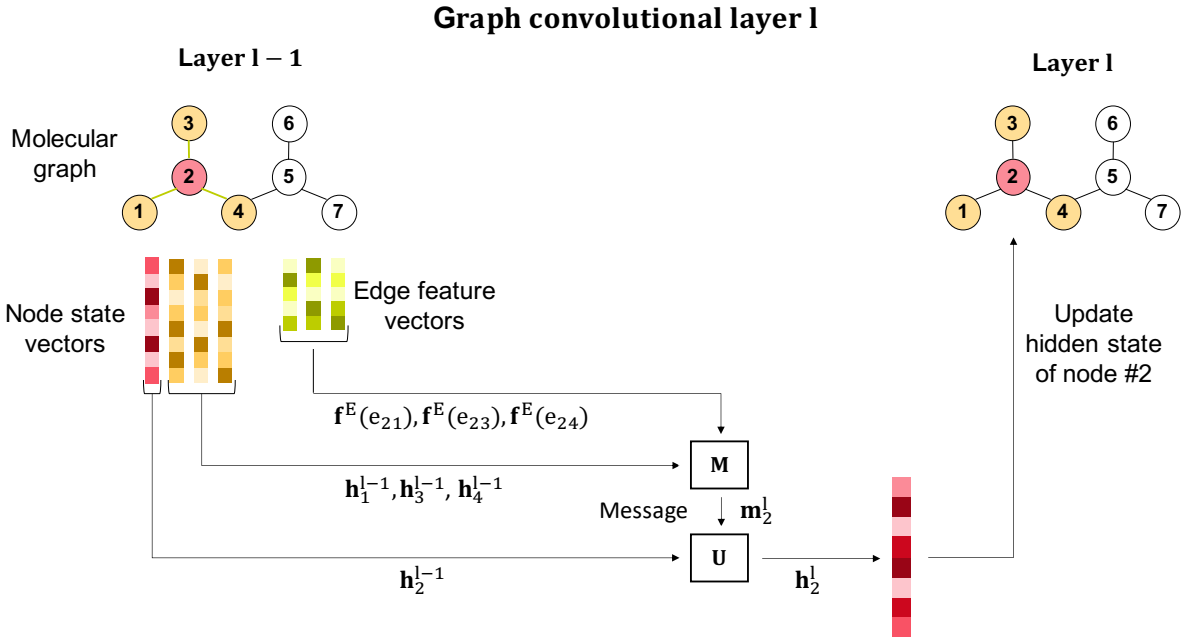


Figure 5.2.: Illustration of a graph convolutional layer. We consider the update of the node state vector of the considered atom (#2 in red). The atom information is given as node state vectors of considered node and its neighbors (#1, 3, 4 in yellow). Furthermore, bond information is given as edge feature vectors (green). The message function $\boldsymbol{M}$ generates the message and the update function $\boldsymbol{U}$ computes the update for new state vector of the considered node.

$\boldsymbol{m}_v^l$ is passed along edges to a node by applying a message function $\boldsymbol{M}_l$, i.e.,

$$\boldsymbol{m}_v^l = \sum_{w \in N(v)} \boldsymbol{M}_l \left(\boldsymbol{h}_w^{l-1}, \boldsymbol{f}^E(e_{vw}) \right) \ , \tag{5.1}$$

where $\boldsymbol{h}_w^{l-1}$ denotes the hidden state of a neighbor in layer $l-1$. The 0-th hidden state vector is initialized with the input feature vector of a node, i.e., $\boldsymbol{h}_v^0 = \boldsymbol{f}^V(v)$. The message function $\boldsymbol{M}_l$ depends on the previous hidden states of the neighbors $\boldsymbol{h}_w^{l-1}$ and the features of the respective edges $\boldsymbol{f}^E(e_{vw})$, with the dimension of the hidden state vector for layers $l > 0$ being a hyperparameter. The hidden state of the considered node $\boldsymbol{h}_v^{l-1}$ is then updated to $\boldsymbol{h}_v^l$ by an update function $\boldsymbol{U}_l$ that combines its hidden state from the previous layer with the received message containing information of its neighbors:

$$\boldsymbol{h}_v^l = \boldsymbol{U}_l \left(\boldsymbol{h}_v^{l-1}, \boldsymbol{m}_v^l \right) \tag{5.2}$$

The message passing and updating is repeated for a fixed number of iterations which results in multiple graph convolutional layers $l \in \{1, 2, ..., L\}$. After L graph convolutions, the hidden states of nodes $\boldsymbol{h}_v^L$ contain local information of environments with a radius of L nodes.

In the readout phase, the hidden node states of the last convolutional layer are combined into a graph representation vector $\boldsymbol{h}_G$, i.e., molecular fingerprint, by using a pooling function $\boldsymbol{h}_G = \boldsymbol{p}(\boldsymbol{h}_1^L, \boldsymbol{h}_2^L, ..., \boldsymbol{h}_{|v|}^L)$ where $\boldsymbol{p}$ is commonly chosen as the mean, sum, or max function [356]. Finally, the molecular fingerprint vector $\boldsymbol{h}_G$ is utilized for regres-

sion of molecular properties of interest, e.g., using a multilayer perceptron (MLP), i.e., $\hat{p} = \mathrm{MLP}(\boldsymbol{h}_G)$.

A strong advantage of this method is that all functions from the molecular graph to the property are explicit and differentiable allowing for supervised training using backpropagation. This enables end-to-end learning of GNNs, whereby the graph convolutions and the molecular fingerprint adapt during training to extract information of the molecular graph that is relevant for the property to be predicted [346, 348, 358].

5.2.3 Higher-order graph neural networks

Morris et al. have recently extended the message passing to higher-order graph features [113]. Here, the message passing step does not only apply to the initial molecular graph but also to modified higher-dimensional molecular graphs. For these higher-dimensional graphs, the nodes are k-dimensional subsets s of the nodes in the initial graph, $s = \{v_1, ..., v_k\} \in [V]^k = \{U \subseteq V \mid |U| = k\}$ for $k > 1$. Hence, any combination of k nodes from the original graph (atoms in the molecular graph) are combined in a separate node.

The neighborhood of a k-dimensional node s is defined as $N(s) = \{t \in [V]^k \mid |s \cap t| = k-1\}$ for $k > 1$. This means that two k-dimensional nodes s and t with k atoms each are adjacent to each other if the cut set of the two nodes consists of $k-1$ atoms. This concept of higher-order neighborhood is illustrated in Figure 5.3, where we consider a red node and its yellow neighbor in the initial molecular graph (1-GNN) as well as higher-order graphs.

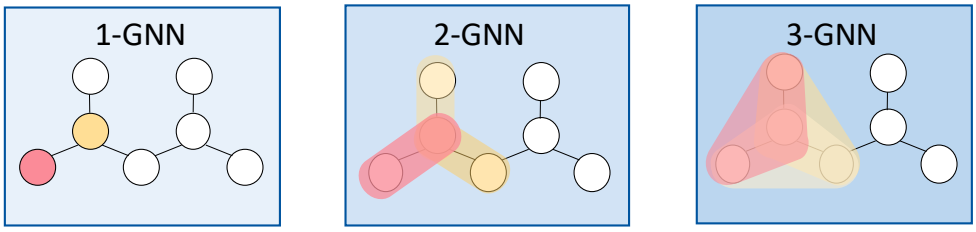


Figure 5.3.: Illustration of the neighborhood in k-dimensional graphs. We highlight the yellow neighbors of a red node.

The message passing phase for the initial molecular graph is referred to as 1-GNN. The message passing phase for higher-dimensional graphs with nodes consisting of k nodes from the original graphs, the structure is called k-GNN [113]. When considering k-dimensional nodes, the initializations of the hidden node states cannot simply be atom types and features, but rather must be combinations of the respective individual atom types and features. At first, the initialization of the hidden node state of a k-dimensional node s contains the isomorphic type $\boldsymbol{f}_{\mathrm{iso}}(s)$ [113]. For example, the initial H-depleted molecular graph ($k = 1$) of oxygenated hydrocarbons, the isomorphic type of the node v corresponds to the atom type {{C},{O}} and is included in the initial feature vector of a node. For the 2-GNN ($k = 2$) of such hydrocarbons, the isomorphic type of the set $s = \{v_1, v_2\}$ corresponds to the 2-set of atom types: {{C,C},{C,O},{O,O}}.

Furthermore, the k-GNN model usually works in a hierarchical manner such that the outputs of the (k-1)-th GNN serve as inputs for the k-th GNN [113]. Therefore, in addition to the isomorphic type, the respective hidden node states of the last graph convolutional

layer L of the preceding GNN are combined by a pooling function, e.g., mean function, and used for initializing the hidden state of a k-dimensional node. Note that this does not refer to the pooling of the molecular fingerprints in the readout phase. Hence for the 2-GNN, the hidden node states of a subset $s = \{v_1, v_2\}$ are initialized with the concatenation ($\|$) of the isomorphic type and the averaged hidden node states of the two respective nodes from the 1-GNN, i.e., $\boldsymbol{h}_s^{2,0} = \boldsymbol{f}_{\text{iso}}(s) \,\|\, \text{mean}(\boldsymbol{h}_{v_1}^L, \boldsymbol{h}_{v_2}^L)$.

5.3 Databases for fuel ignition quality

We collect DCN, RON, and MON data of non-oxygenated and oxygenated hydrocarbons from different literature sources. The number of species per molecular class and fuel ignition quality indicator is shown in Table 5.1. Note that we provide our full data set in the Supporting Information (SI).

The cetane number (CN), an indicator for CI fuel quality, is determined in a cooperative fuel research (CFR) reference engine [359]. In comparison to the CFR engine, testing methods in a variety of constant-volume combustion chambers (CVCCs) require lower fuel quantities and shorter measurement times, but yield so-called derived cetane numbers (DCNs) [317] instead of true CFR CN. Our objective is to predict DCN values determined by a particular CVCC-based experimental setup, i.e., the ignition quality tester (IQT) which is standardized by the ASTM D6980 [360] and widely used to assess diesel fuel ignition quality[319, 361]. To this end, we consider IQT-DCN data from the Compendium of Experimental Cetane Numbers provided by Yanowitz et al. [317] for 236 different species.

Although (D)CN values from non-IQT experiments cannot be expected to closely match IQT-DCN [319], we utilize such data for a transfer learning approach. Specifically, we consider (D)CN values for 479 species from various measurement setups from the Compendium of Experimental Cetane Numbers [317]. We exclude the test set compounds, use the remaining 447 molecules for pre-training, and subsequently refine the resulting model based on IQT-only data. We provide the different data sets created for model development online[352].

RON and MON are used to quantify the knocking tendency of a fuel and are suitable ignition quality indicators for SI engines. They are measured according to the ASTM D2699 [362] and ASTM D2700 [363] standards, respectively. As a database for our model, we take RON (MON) values for 335 (318) species from literature [308, 316, 318, 338, 364–371]. For all reported data (RON, MON, DCN), we take average values whenever multiple values have been reported for a single species.

5.4 Modeling approach

In this section, the GNN model development is described (cf. Figure 5.4). First, the workflow of the molecular representation is explained in Section 5.4.1. Then, the basic architecture of the GNN model is outlined in Section 5.4.2. Finally, the model extensions for multi-task learning (Section 5.4.3), transfer learning (Section 5.4.4), and ensemble learning (Section 5.4.5) are described.

We use PyTorch Geometric [372], an open-source library for deep learning on graphs in Python. The implementation is adapted from our previous work on k-GNNs [113].

Table 5.1.: Databases assembled for this work: Number of species per molecular class and fuel ignition quality indicator. Number of species in our test set is given in parentheses. Details can be found in the Supporting Information.

	IQT-DCN	**(D)CN**	**RON**	**MON**
n-alkanes	9 (0)	18 (0)	7 (0)	7 (0)
iso-alkanes	17 (2)	39 (2)	43 (6)	42(6)
cycloalkanes	20 (0)	34 (0)	74 (7)	65 (7)
alkenes	15 (3)	33 (3)	87 (16)	84 (16)
cycloalkenes	10 (0)	12 (0)	22 (2)	22 (2)
alkynes	1 (0)	1 (0)	8 (0)	4 (0)
aromatics	13 (4)	63 (4)	41 (12)	43 (12)
alcohols	21 (3)	38 (3)	14 (1)	13 (1)
cyclic alcohols	2 (0)	2 (0)	0 (0)	0 (0)
aldehydes	7 (1)	7 (1)	0 (0)	0 (0)
ketones	9 (3)	9 (3)	8 (1)	7 (1)
cyclic ketones	5 (1)	5 (1)	2 (0)	2 (0)
ethers	17 (3)	27 (3)	5 (0)	5 (0)
hydrofurans	7 (1)	7 (1)	3 (1)	3 (1)
other cyclic ethers	4 (0)	4 (0)	2 (0)	2 (0)
esters	39 (4)	133 (4)	12 (3)	12 (3)
lactones	4 (1)	4 (1)	1 (0)	1 (0)
furans	5 (2)	5 (2)	3 (2)	3 (2)
acetals	2 (0)	2 (0)	1 (0)	1 (0)
carboxylic acids	0 (0)	5 (0)	0 (0)	0 (0)
> #1 type of oxygen functionality	29 (4)	31 (4)	2 (0)	2 (0)
Total	**236 (32)**	**479 (32)**	**335 (51)**	**318 (51)**

Our open-source code for training as well as the trained models can be retrieved on [352]. Additionally, for more convenient use, the model can be accessed freely via a web front-end (`www.avt.rwth-aachen.de/gnn`) to make predictions.

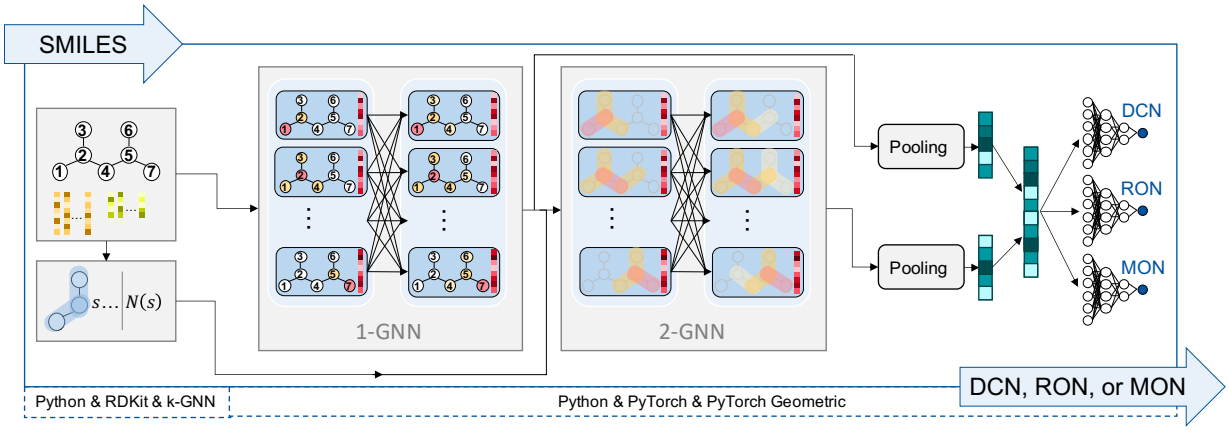


Figure 5.4.: Workflow of the multi-task GNN model for predicting DCN, RON, and MON of hydrocarbons with SMILES strings as input. The model works in a hierarchical manner in which the outputs of the 1-GNN part are used as inputs for the 2-GNN. The outputs of both, the 1-GNN and 2-GNN part, are concatenated and represent the molecular fingerprint. This serves as the input for the three MLP channels predicting the target properties.

5.4.1 Molecular representation

For generating the representation of molecules that serve as an input to the GNN, SMILES strings [373] are transformed into molecular graphs. Each node and each edge is assigned a feature vector. Each feature is represented as a one-hot encoder with the size of the number of possible values for this feature and a single entry with value one at the index corresponding to the value of the feature. The features are selected according to previous literature [346, 348, 349] and are shown in Table 5.2 and 5.3 for nodes and edges, respectively. We use RDKit [374] for SMILES strings transformation and calculation of features. Note that the data set considered in this work does not include any atoms with sp3d or sp3d2 hybridization. Furthermore, we find that RDKit encodes stereochemistry exclusively by means of the more general E/Z notation instead of the cis/trans notation on our data set. The dimensionality of the hybridization and stereo feature vector could thus be reduced without loss of information.

5.4.2 Model architecture

The proposed GNN model combines our higher-dimensional GNNs [113] with recurrent neural network architectures [348, 375, 376]. By using higher-dimensional GNNs, higher-order characteristics of a molecular graph can be extracted. The recurrent neural networks allow us to share parameters within the message passing phase of a GNN. We use gated recurrent units (GRUs) as recurrent neural networks, because GRUs avoid the vanishing gradient problem while having fewer parameters than long short-term memories (LSTMs) and thus have the potential to generalize faster on small data sets [375]. As illustrated in

Table 5.2.: Atomic features used as initial node states, similar to [346, 348, 349]. All features are implemented as one-hot encoders.

Feature	Description	Dimension
atom type	type of atom (C, O)	2
is in ring	whether the atom is part of a ring	1
is aromatic	whether the atom is part of an aromatic system	1
hybridization	sp, sp2, sp3, sp3d, or sp3d2	5
# bonds	number of bonds the atom is involved in	6
# Hs	number of bonded hydrogen atoms	5

Table 5.3.: Bond features used as edge features, similar to [346, 348, 349]. All features are implemented as one-hot encoders.

Feature	Description	Dimension
bond type	single, double, triple, or aromatic	4
conjugated	whether the bond is conjugated	1
is in ring	whether the bond is part of a ring	1
stereo	none, any, E/Z, or cis/trans	6

Figure 5.4, we apply two GNN structures in the message passing phase: (i) 1-GNN and (ii) 2-GNN. Thereby, atom environments in a molecule are first examined locally and then interactions between different atom environments are studied.

First, in the 1-GNN, an edge feature network and a GRU explore local atomic environments within the molecular graph. In particular, the updated hidden state in layer l, i.e., $\boldsymbol{h}_v^l$, is computed as

$$\boldsymbol{h}_v^l = \mathrm{GRU}\left(\boldsymbol{h}_v^{l-1}, \sigma\left(\theta_v \cdot \boldsymbol{h}_v^{l-1} + \boldsymbol{m}_v^l\right)\right) \ , \tag{5.3}$$

where the message $\boldsymbol{m}_v^l$ is given by

$$\boldsymbol{m}_v^l = \sum_{w \in N(v)} \mathrm{ANN}_{\theta_e}\left(\boldsymbol{f}^E(e_{vw})\right) \ \cdot \ \boldsymbol{h}_w^{l-1} \ . \tag{5.4}$$

Herein, the edge feature network is a feedforward ANN, i.e., ANN_{θ_e}, that maps edge features $\boldsymbol{f}^E$ to a parameter matrix θ_e. Then, the parameter matrix θ_e is multiplied with the hidden states of a node's v neighbors, $\boldsymbol{h}_w^{l-1}$ with $w \in N(v)$, to calculate the message $\boldsymbol{m}$. The message is added to the hidden state of the considered node $\boldsymbol{h}_v^{l-1}$ multiplied with a parameter matrix θ_v. This result is transformed with an activation function σ, here rectified linear unit (ReLU). By applying a GRU, the updated hidden state in layer l, i.e., $\boldsymbol{h}_v^l$, is finally computed. Note that in this work the hidden states are based on nodes, whereas Yang et al.[349] consider hidden states based on edges. The initial hidden states $\boldsymbol{h}_v^0 = \boldsymbol{f}^V(v)$ are mapped to the dimension of the following hidden states by a shallow ANN with ReLU activation.

Secondly, a higher-dimensional message passing process is applied to enable interactions between atom environments (cf. Section 5.2.3). By combining the final atom representa-

tions of the 1-GNN into higher-dimensional nodes on which another message passing phase is applied, long-range effects of atom groups within a molecule can be captured. In this work, we found a 1,2-GNN architecture to have a lower mean absolute error (MAE) compared to that of a 1,2,3-GNN or a simple 1-GNN, thus the 1,2-GNN architecture is used to learn higher-dimensional graph features. Accordingly, we call the hierarchical combination of the 1-GNN and 2-GNN structure 1,2-GNN in the remainder of this work. We update the hidden states in the 2-GNN message passing similarly to the previously described 1-GNN, except that the edge feature network is replaced by a simple parameter matrix θ_2^k as there are no features for edges of the higher-order graph [113]. As the 1-GNN is an input to the 2-GNN, the 1,2-GNN indirectly takes the edge features of the original molecular graph into account.

After the message passing process, the model employs sum pooling for aggregating the hidden node states of the 1-GNN and 2-GNN resulting in two graph representation vectors. Sum pooling is applied, since the nature of the contribution of atoms and bonds to DCN, RON, and MON is expected to be additive, similar to in group contribution models [318, 319]. After pooling, the two graph representation vectors are concatenated to the molecular fingerprint, i.e., $\boldsymbol{h}_G = [\boldsymbol{h}_{G_{1-GNN}}^T, \boldsymbol{h}_{G_{2-GNN}}^T]^T$. Finally, the molecular fingerprint is fed into a deep MLP for the prediction of DCN, RON, and MON, $\hat{p} = \mathrm{MLP}(\boldsymbol{h}_G)$.

5.4.3 Single- and multi-task learning

Having several prediction tasks, machine learning models can be trained in single- or multi-task manner [377–379]. In single-task learning, individual models are trained for each task. In multi-task learning, some representation is shared among the different tasks. For ANNs, this means that weights and bias parameters of hidden layers are shared between multiple tasks, i.e., they have equal values. Besides the shared layers, further individual hidden layers are employed for each task. The shared representation captures general information that is relevant to all tasks [378]. In the individual layers, task-specific information is extracted. In this way, the model learns more general input representations in the first layers compared to single-task models and overfitting can be reduced [378]. This is particularly relevant when the data sets are considerably small. Furthermore, multi-task learning can enable knowledge transfer between different prediction tasks [379]. In previous literature, this has been shown to yield superior results to single-task models in multiple molecular applications [346, 380, 381].

In our model, we utilize multi-task learning by sharing the graph convolutional layers to create a general molecular fingerprint on which three individual MLPs (also called channels) are used for predicting DCN, RON, and MON. As cetane and octane numbers are known to correlate negatively [310, 314, 318, 319, 382, 383], multi-task learning is particularly promising in this context.

5.4.4 Transfer learning

Another technique enabling knowledge transfer in machine learning is transfer learning [384, 385]. In transfer learning, knowledge learned in one domain is transferred to another domain, i.e., to the target task [385]. One way to perform transfer learning concerns pre-training of ANNs on a (source) task related to the target task. Afterward, the parameters of the pre-trained model are used to initialize parameters of a model trained

on the target task data. Thus, transfer learning is particularly relevant for problems where the target data basis is small.

Transfer learning has recently been applied in the context of molecular property prediction with GNNs. For example, Grambow et al. pre-trained GNNs for thermophysical property predictions on large data sets from quantum-mechanical calculations and retrained parts of the GNN on a smaller experimental data set [386].

We aim to improve our IQT-DCN prediction by transferring information from additional (D)CN data, i.e., from measurement techniques other than IQT (cf. Section 5.3). Thus, we propose a transfer learning approach, where CN and DCN data from various measurement setups are utilized for pre-training and then models are retrained on IQT-only DCN data.

5.4.5 Ensemble learning

Ensemble learning is a technique in machine learning where multiple models are trained and utilized for a single prediction task [387–389]. In most applications, several individual models are trained independently on a randomly drawn subset of the training data. Then, the predictions of the individual models are averaged to receive a more accurate prediction. This way, prediction can be improved as random model errors are averaged out. Averaging single model predictions is also known as bootstrap aggregating or bagging [387]. This is particularly relevant for models with low bias and high variance which is the case for complex GNNs. Furthermore, small data sets can lead to high variance.

We train independent GNN models with randomly selected training and validation sets. To ensure an unbiased model comparison, all models share the same independent test set. While some advanced ensembling techniques apply weights to models, e.g., boosting [390], we use a standard bagging technique that applies the same weight to all models.

5.5 Results and discussion

In this section, we first briefly summarize the general training settings (Section 5.5.1) and hyperparameter selection (Section 5.5.2). Then, we analyze the prediction accuracy of the proposed model developments: multi-task learning (Section 5.5.3), transfer learning (Section 5.5.4), and ensemble learning (Section 5.5.5). Finally, we compare the proposed model to state-of-the-art QSPR models (Section 5.5.6).

5.5.1 General training settings

As described in Section 5.3, the data set of DCN values extracted from the Compendium of Experimental Cetane Numbers [317] includes DCN measurements of 236 different components measured with the IQT method. We use this high-quality DCN data set and the RON and MON data sets for the training of the single and multi-task models (cf. Section 5.5.3). As typically done in machine learning, the data sets are standardized to zero mean and standard deviation of one for each target property, i.e., DCN, RON, and MON. Then, the data sets are randomly split into a training (85%) and test (15%) set. The test set is separated from the rest of the data and not used until the final testing of the model. For training the model, an internal validation set (15% of the original data set) is separated randomly from the training data and used for early stopping.

For each data point, the molecular graphs are generated as described in Section 5.4.1. Then, the model is trained based on the training set. Here, the mean squared error is used as the loss function. During training, the model performance regarding the internal validation set is measured in each epoch. The learning rate is decreased by a factor of 0.8 after every 3 consecutive epochs in which the error on the internal validation set did not decrease. Training is stopped either after a maximum number of 300 epochs was reached or if the internal validation error did not decrease in the 50 preceding epochs, according to early stopping. The error on the internal validation set is also used for comparison of the different model structures and the selection of hyperparameters. The training and random selection of the internal validation set are repeated 40 times for all models.

5.5.2 Hyperparameter selection

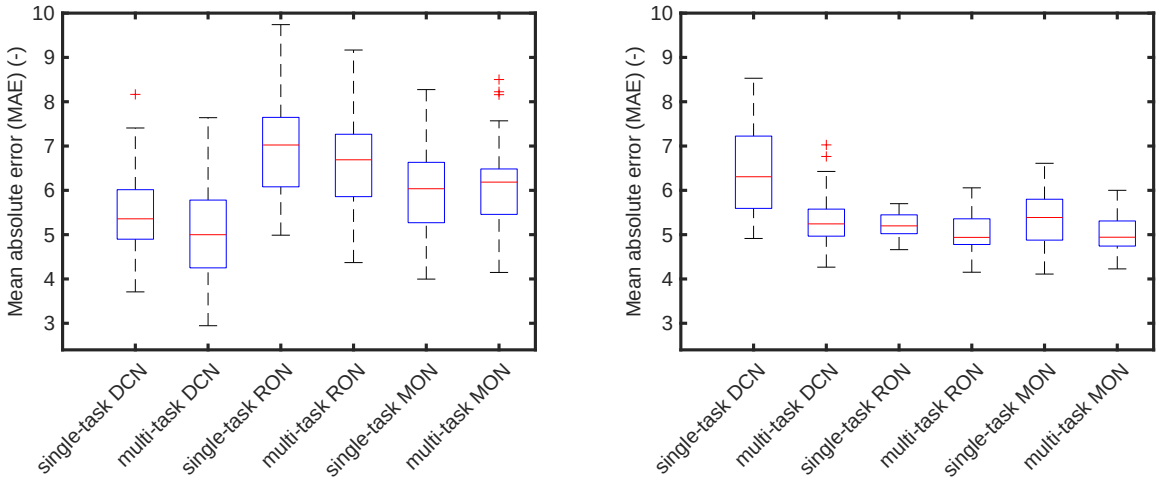
The proposed GNN model exhibits several hyperparameters that need to be chosen. To identify a suitable model architecture, the following hyperparameters are varied within the given ranges: initial learning rate $\in \{0.0005, 0.001, 0.005\}$, hidden states size $\in \{32, 64, 128\}$, number of graph convolutional layers $\in \{1, 2, 3, 4, 5\}$ for the 1-GNN part and number of graph convolutional layers $\in \{1, 2, 3\}$ for the 2-GNN part, and message passing function $\in$ {without GRU, with GRU}. To this end, we performed an extensive hyperparameter study on a preliminary data set with 40 repetitions for each hyperparameter setting. This preliminary data set is 99.7% identical to our final training data set. The only difference is that we have corrected two mistakes in SMILES strings that occurred in the automated data processing and added one new molecule based on new literature that we found. In the hyperparameter study, we considered the total MAE, i.e., RON, MON, and DCN together, on the validation set averaged over 40 runs to decide on the final architecture and parameters. Based on these results, we use message passing with GRU, two graph convolutional layers in the 1-GNN part, two graph convolutional layers in the 2-GNN part, a hidden states size of 64, and an initial learning rate of 0.001 for the final model. We note that the differences between different hyperparameter settings are often similar to the variance of the MAE. Therefore, the sensitivity of the model performance with respect to the hyperparameter selection is rather weak in our study.

The remaining hyperparameters are described in the following and selected based on literature and expert knowledge. Trial and error attempts to change these other hyperparameters did not lead to improved results. We use atom and bond features as described in Section 5.4.1. We apply an edge feature network with three layers and the following number of neurons: #1: 12 (i.e., number of edge features), #2: 128, #3: 4096 (i.e., number of hidden state squared). The MLPs constitute five layers with #1: 128, #2: 64, #3: 32, #4: 16, #5: 1 neurons.

5.5.3 Single- and multi-task learning

The aforementioned model settings were used for single-task and multi-task learning. The mean absolute errors (MAEs) of the two approaches on their validation and test set are displayed in Figure 5.5. The respective box plots illustrate the distribution of MAEs over the 40 individual training runs for each model.

Figure 5.5 shows that the model performance exhibits a high variance. This is mainly caused by the small data size for training, validation, and testing. As the validation sets



(a) MAE on validation set

(b) MAE on test set

Figure 5.5.: Comparison of the MAE of the single-task learning and the multi-task learning approach on the validation and test sets. The box-plots indicate the lowest and largest MAE (excluding outliers), the lower and upper quartile, and the median of the MAE over 40 independent model instances. Note that points that are more than 1.5 times the interquartile range away from the top or bottom of the box are marked as outliers.

of the 40 independent model runs are selected randomly, they show a larger variance of the MAE. In contrast, all 40 independent models share the same test set. Thus, the MAE distribution on the test set is more narrow. One methodology against high model variance is bootstrap aggregation which is performed in Section 5.5.5.

Table 5.4 summarizes the MAEs on the training, internal validation, and independent test set averaged over the 40 training runs for comparison. The averaged results show that the multi-task training approach improves the prediction accuracy on all test sets and for all predicted properties. For instance, the MAE of the DCN on the test set is reduced by about 17% from 6.4 to 5.3.

The results indicate that the simultaneous learning of DCN, RON, and MON leads to a better generalization of the graph convolutional layers and thus molecular fingerprint. One reason for the synergies are believed to be the correlations between DCN, RON, and MON [310, 314, 318, 319, 382, 383].

5.5.4 Transfer learning

For the transfer learning approach, we pre-train the single-task DCN model on data from all different (D)CN measurement methods, i.e., we use (D)CN data of 447 components collected by Yanowitz et al. [317] (cf. Section 5.3). Then, the learned parameters are used to initialize the parameters in the graph convolutions and the MLP of the single-task DCN and also the multi-task model. For the latter, only the parameters of the MLP for predicting the DCN are transferred from the pre-training since RON and MON values are not subject to transfer learning. Finally, we retrain the models by only considering IQT-DCN data.

The results of the transfer learning approach are summarized in Table 5.4. Transfer

Table 5.4.: Mean absolute error (MAE) of training, validation, and test set averaged over 40 training runs. The table includes single-task learning (STL), multi-task learning (MTL), transfer learning (TL), and ensemble learning (EL). Lowest test set errors are highlighted in bold.

	DCN			RON			MON		
	Train.	Val.	Test	Train.	Val.	Test	Train.	Val.	Test
STL	2.7	5.5	6.4	3.7	7.0	5.2	3.1	6.0	5.4
MTL	1.8	5.1	5.3	2.8	6.7	5.0	2.3	6.1	5.0
STL & TL	2.2	4.6	6.0	–	–	–	–	–	–
MTL & TL	1.8	5.2	5.9	3.2	6.6	5.1	2.6	6.0	4.9
MTL & EL	1.8		**4.2**	2.8		**4.5**	2.3		**4.4**

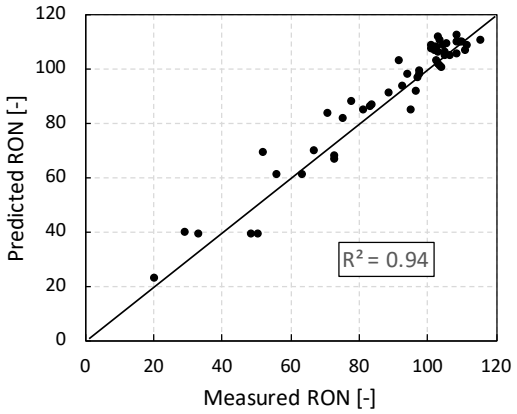
learning improves the MAE of the single-task model for predicting the DCN from 6.4 to 6.0. For the multi-task model, however, transfer learning does not improve prediction accuracy: Test MAEs for RON and MON are almost the same with and without transfer learning. Furthermore, test MAE for DCN even increases from 5.3 to 5.9 with transfer learning. One possible reason for the poor performance of transfer learning in the multi-task learning is that the pre-training is essentially a single-task problem because we use only (D)CN data for pre-training. Thus, the pre-trained model could be biased towards (D)CN which then could lead to poor generalization of the multi-task model. As a consequence, we do not use the transfer learning approach for our final model.

5.5.5 Ensemble learning

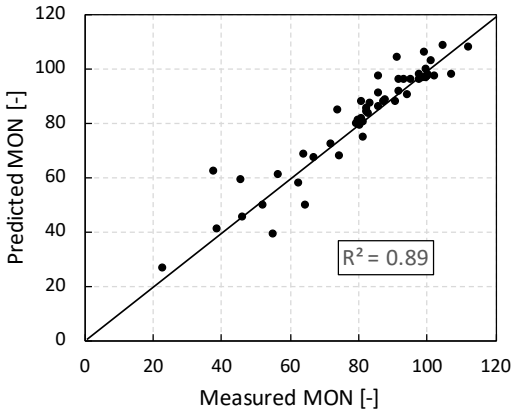
After developing a suitable model architecture, model ensembling is applied to address the observed high variation (cf. discussion in Section 5.5.3). As described in Section 5.4.5, ensemble learning averages the response of multiple models and mitigates random model variations. Herein, we utilize the previously trained 40 model instances. We perform the ensemble learning on the multi-task architecture without transfer learning.

The results are summarized in Table 5.4. They show the averaged MAE on the test set and the combined training and validation set. The error on a validation set is shown as part of the training set because the averaged 40 model instances have individual randomly selected validation sets. Ensemble learning reduces the MAE of the DCN, RON, and MON significantly from 5.3 to 4.2, from 5.0 to 4.5, and from 5.0 to 4.4, respectively. The bootstrap aggregation compensates for the previously identified large model variations.

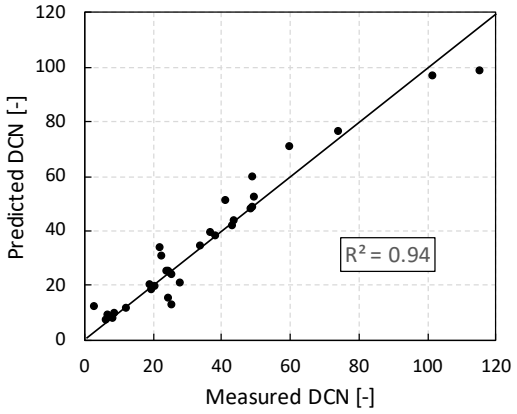
Figure 5.6 illustrates the parity plots for the independent test set of the proposed ensemble model. Herein, every point represents the averaged prediction of 40 multi-task models for a data point in the test set. The plots show high coefficients of determination for all three properties, i.e., $R^2_{\mathrm{DCN}} = 0.94$, $R^2_{\mathrm{RON}} = 0.94$, and $R^2_{\mathrm{MON}} = 0.89$. For the MON, the higher number of outliers causes a slightly weaker coefficient of determination. Note that the parity plots show an uneven distribution of the data in the test set. For instance, there exist few data points with DCN numbers above 100 or RON numbers below 50.



(a) RON test set



(b) MON test set



(c) DCN test set

Figure 5.6.: Multi-task GNN model ensembling: Parity plots for (a) RON, (b) MON, and (c) DCN test sets.

5.5.6 Comparison to recent QSPR models

We compare our GNN model to three recent literature QSPR models for fuel ignition indicator prediction. Our own previous model follows a group contribution and multivariate nonlinear regression approach for predicting IQT-DCN [319]. The model by vom Lehn et al.[337] combines group contributions and a feed-forward artificial neural network (ANN) to predict RON. Finally, the model by Kubic et al.[318] combines group contributions and a multi-task feed-forward ANN for simultaneous regression of CN, RON, and MON values.

A fair comparison between the different models is difficult for multiple reasons. First, models have been developed from different training data sets, leading to different model applicability domains. This is most evident in case of the model proposed by vom Lehn et al.[337], which is applicable to alkanes, alkenes, cyclic alkanes, and alcohols only, whereas molecular diversity of the training data is much higher for our GNN model and the other two models. Second, different performance evaluation methods, i.e., cross-validation or evaluation on an independent test set, and different performance metrics, i.e., mean absolute error (MAE) or coefficient of determination (R^2), have been employed, as can be seen from Table 5.5. Our procedure for GNN hyperparameter tuning (cf. Section 5.5.2) follows a cross-validation (CV) approach, i.e., we repeatedly split the dataset into training and validation sets and use the average validation set MAE to find the optimal hyperparameter settings. Since CV-MAE can be a biased estimator of the true prediction error of a model if model parameters are tuned outside of the CV loop[391], we use an independent test set to estimate the true prediction error. Third, many molecules located in our test set have been used to train the models proposed by Dahmen & Marquardt[319], Kubic et al.[318], and vom Lehn et al.[337]. Simply rerunning these models on our test set would give them an unfair advantage.

Still, a comparative true prediction performance test can be carried out by exclusively examining those compounds from our test set that were not included in the respective training set of a comparison model. This approach results in the four head-to-head comparisons shown in Tables 5.6 to 5.9.

Table 5.6 compares the DCN predictions made by the GNN model to the predictions made by our previous DCN model and the measurement values. While both models exhibit a decent performance, the MAE of the proposed GNN model is lower than the MAE of our previous QSPR model (3.6 instead of 5.2). Table 5.7 shows the comparison between the GNN model and the RON model by vom Lehn et al.[337]. Since the latter model is applicable to alkanes, alkenes, cycloalkanes, and alcohols only, this comparison is based on just four compounds. The resulting MAEs are roughly similar. Finally, Table 5.8 and 5.9 compare the GNN model to the multi-task ANN model by Kubic et al.[318], where it can be noted that the MAEs of the GNN model are far lower. In summary, the different head-to-head comparisons suggest highly competitive performance of the new GNN model. We provide our model, the training scripts, and all data sets open-source[352] as well as a web front-end (`www.avt.rwth-aachen.de/gnn`) so that others can easily perform their own benchmarks.

Development of QSPR and GNN models differs significantly from each other also from a conceptual point of view. Most notably, QSPR modeling requires to choose a set of descriptors, e.g., structural group counts, as potential explanatory variables. This step may facilitate understanding of the prediction problem (the human learns through model development) and can encode physical understanding into a tailored model structure. However,

this also means that QSPR models inherently rely on assumptions about the underlying phenomena, i.e., the descriptors or structural groups of potential value. In contrast, the presented GNN method is trained in an end-to-end learning approach, as it relies on only few atomic and bond features (cf. Tables 5.2 and 5.3), and thus provides a flexible model structure that can possibly learn a broad variety of properties. End-to-end learning with graph convolutions, however, comes at the cost of higher computational effort for training.

Applicability domain (AD) quantification in GNN-based property models is an open research question, which is closely linked with the question of how well GNNs can generalize. Traditional AD methods used in QSPR modeling, e.g., the Williams plot [392, 393], are not directly transferable to GNNs unless a suitable distance metric for molecular graphs is established that can be linked to GNN prediction performance. There are some recent works on autoencoders for molecular graphs[305, 394–398] that might help to derive AD concepts based on a real-valued latent space identified by the autoencoders. When making predictions with our model, we recommend to check from Table 5.1 whether property data was available during model development for the molecular class the compound of interest belongs to. For instance, from Table 5.1 it can be seen that no RON/MON data for aldehydes and cyclic alcohols were available for model training. Whereas it might prove a reasonable assumption that cyclic alcohols behave similar to acyclic alcohols, we abstain from using our model to predict RON/MON of aldehydes.

Table 5.5.: Performance comparison of the proposed model to three recent QSPR models.

	GNN model		[319]		[318]		[337]	
method	test set		test set		cross-validation		cross-validation	
metric	MAE	R^2	MAE	R^2	MAE	R^2	MAE	R^2
DCN	4.2	0.94	5.8	0.84	–	0.90	–	–
RON	4.5	0.94	–	–	–	0.93	4.0	0.92
MON	4.4	0.89	–	–	–	0.91	–	–

5.6 Conclusion

Predictive models for fuel ignition quality play a crucial role in the development of novel fuels. We propose a data-driven graph neural network (GNN) model for the prediction of three important fuel auto-ignition indicators, i.e., the derived cetane number (DCN), the research octane number (RON), and the motor octane number (MON). Our model is applicable to a wide spectrum of non-oxygenated and oxygenated hydrocarbons, shows competitive performance to state-of-the-art models, and can be easily accessed via a web interface.

From the methodological point of view, our GNN-based model offers the advantage that, in contrast to previous works based on QSPR modeling, no molecular descriptors or structural groups, have to be selected, because GNNs achieve end-to-end learning from the molecular structure to the properties of interest. While such a data-driven approach is often believed to require extensively large data sets, this work demonstrates that good model accuracies can indeed be achieved for small data sets (order of hundreds) by using

Table 5.6.: Comparison with the IQT-DCN model by Dahmen & Marquardt[319] based on those molecules from our DCN test set that were not included in the training set used by Dahmen & Marquardt[319].

	true value DCN	GNN model DCN	[319] DCN
1,2-dimethylbenzene	8.3	7.4	7.9
furan	7.0	8.2	9.4
methyl erucate	74.2	75.9	87.6
2-heptanol	25.0	24.1	21.5
4-ethyl guaiacol	19.6	17.4	25.4
1,3,5-triisopropylbenzene	2.8	11.5	10.4
ocimene	28.0	20.1	14.9
1,4-dimethylbenzene	6.2	6.8	7.9
6-undecanone	49.0	59.2	51.8
4-nonanone	43.0	41.3	41.3
mean absolute error (MAE)		3.6	5.2

Table 5.7.: Comparison with the ANN-based RON model by vom Lehn et al.[337] based on those molecules from our RON test set that were not included in the training set used by vom Lehn et al.[337]. Since the model by vom Lehn et al. is applicable to alkanes, alkenes, cycloalkanes, and alcohols only, this comparison is limited to four compounds.

	true value RON	GNN model RON	[337] RON
methylcyclopropane	102.5	107.1	106.0
2,2-dimethyloctane	49.0	38.6	31.7
1,5-hexadiene	71.1	82.9	86.7
2,4-hexadiene	97.1	91.0	98.6
mean absolute error (MAE)		8.2	9.5

Table 5.8.: Comparison with the ANN model by Kubic et al.[318] based on those molecules from our DCN test set that were not included in the training set used by Kubic et al.[318].

	true value DCN	GNN model DCN	[318] DCN
δ-undecalactone	48.6	47.1	38.6
2-heptanol	25.0	24.1	25.2
4-methoxybenzaldehyde	25.8	12.5	49.7
geraniol	19.3	19.4	22.5
4-ethyl guaiacol	19.6	17.4	5.1
ocimene	28.0	20.1	-2.0
dodecyl vinyl ether	101.7	96.0	126.2
propylene glycol monomethyl ether acetate	24.0	24.5	34.2
6-undecanone	49.0	59.2	58.5
mean absolute error (MAE)		4.7	14.0

Table 5.9.: Comparison with the ANN model by Kubic et al. [318] based on those molecules from our RON & MON test set that were not included in the training set used by Kubic et al. [318].

	true value		GNN model		[318]	
	RON	MON	RON	MON	RON	MON
methylcyclopropane	102.5	81.2	107.1	87.5	101.1	90.2
furan	108.6	91.6	111.6	103.3	98.9	97.4
aceticacid-2-methylpropylester	108.7	112.3	109.3	107.3	123.4	105.6
4-methylcyclohexene	84.1	67.0	85.8	66.3	59.2	54.2
tetrahydrofuran	72.9	64.8	66.3	49.1	92.8	87.9
2-octene	56.3	56.5	60.3	60.1	42.3	39.1
1-methylcyclohexene	89.2	72.1	90.1	71.6	64.3	58.6
2-phenylpentane	103.5	92.1	101.2	91.2	99.8	97.7
2-methylpropanoicacidmethylester	103.6	104.7	109.8	108.1	138.9	108.9
1,5-hexadiene	71.1	37.6	82.9	61.7	95.9	81.7
3-methyl-2-butanone	108.9	102.2	104.7	96.9	111.6	103.0
methyl-2-methylbutanoate	110.5	99.1	108.9	105.1	119.9	104.8
4-octene (trans)	73.3	74.3	67.5	67.2	89.1	88.7
2,4-hexadiene	97.1	80.7	91.0	78.4	103.3	88.4
allylcyclopentane	52.1	45.6	68.5	58.3	86.0	73.0
mean absolute error (MAE)			5.1	7.0	16.1	13.2

multi-task and ensemble learning. Given the expected future increase in measurement data available for training, we expect further potential for GNNs in fuel ignition quality prediction. We provide the corresponding training code and the final model open-source making it a viable tool for further development. Finally, this work may constitute a prototype for rapid, versatile property prediction beyond DCN, RON and MON and thus for property prediction in various disciplines.

Chapter 6

Physical graph neural networks

6.1 Introduction

Graph neural networks (GNNs) are emerging for end-to-end learning of molecular properties [350, 351, 358, 399] with a broad variety of applications including (quantum) chemistry [348, 349, 356, 400], physical [346], and crystal properties [112, 401]. In some of these applications, GNNs predict properties that have previously been computed through expensive quantum mechanical calculations. Although GNNs are flexible models for end-to-end learning, we show that their architecture needs to be carefully selected because wrong decisions can lead to unphysical GNNs that can only learn through overfitting. Note that the term overfitting is often used in the narrow context of poor validation set performance of ML models but it has a more general meaning. A more general definition that we find useful and follow in this work is: Overfitting is "the production of an analysis which corresponds too closely or exactly to a particular set of data, and may therefore fail to fit additional data or predict future observations reliably" [402].

GNNs take molecular graphs as inputs and represent atoms by nodes and bonds by edges. In addition, atoms and nodes are characterized by corresponding feature vectors. Most commonly, GNN architectures are based on message passing neural networks (MPNNs) [348]. In MPNNs, the node feature vectors are updated through a series of message passing procedures of neighboring nodes. Each of these sequential message passings corresponds to the graph convolutional layers of the GNN. Then, the resulting node feature vectors are combined into a molecular fingerprint vector through pooling. This molecular fingerprint is finally mapped to molecular properties of interest by feedforward artificial neural networks (ANNs). We show in this work that the selection of the pooling function, which combines the feature vectors of all atoms into the molecular fingerprint, is critical for the GNNs performance.

The vast majority of previous works does not select pooling functions based on physical understanding. In the previous literature, common pooling functions are mean, sum, and max pooling [403]. Most previous works use sum pooling [346, 349, 404, 405], while a few use mean-pooling [113, 406]. However, using the same pooling function for various properties can be error-prone, because wrong pooling functions can lead to unphysical GNNs. An illustrative example is the molecular weight that is given by the sum of the atom's weights. In this case, mean pooling would lead to an unphysical GNN architecture that cannot learn the underlying physics because the molecular mass cannot be computed as an average of atom weights. Thus, a GNN with mean pooling can only learn through overfitting which results in poor generalization. In contrast, selecting the sum pooling function according to the underlying physics enables the GNN to learn a meaningful model (c.f. Section 6.3.2).

Some researchers circumvent the issue of selecting pooling functions by introducing flexible models for pooling such as set2set [407], DiffPool [408], or SortPool [409]. For example,

the set2set approach employs an LSTM desgined for unordered and size-variant input sets [407]. The authors of DiffPool propose a hierarchical GNN structure that progressively coarsens the input graph in each layer by aggregating clusters of nodes until a single graph representation is obtained [408]. SortPool arranges the learned node representation in a consistent ordered tensor which is then truncated or extended to a user-defined fixed size [409]. Advanced pooling methods have also been applied in molecular property prediction. For instance, Gilmer et al. (2017) [348] applied the set2set method for learning various molecular properties from the QM9 dataset [410, 411] and achieved state-of-the-art accuracies on all target properties compared to other GNN models at the time of publication. However, the additional flexibility typically results in larger data requirements, higher model variance, and the risk of overfitting. In other words, the selection of physical pooling functions over flexible model architectures for pooling can be understood as enforcing a hybrid model structure, which is known to reduce the data demand [89, 97].

A few recent studies emphasize the importance of pooling functions for property prediction. Xu et al. (2018) examine sum, mean, and max pooling and conclude that sum pooling is more powerful than mean and max pooling since it can better distinguish different graph structures [404]. Pronobis et al. (2018) state that decomposition of molecules into atom-wise contributions combined with a property-suitable pooling function works better for "extensive properties" [412]. Other works use property-specific pooling functions, where mean or sum pooling is applied to "intensive" or "extensive" properties, respectively [413–416]. Overall, the physical selection of pooling functions is partly contradictory in the previous literature and the terms "intensive" or "extensive" have been used colloquially and not in their thermodynamic sense (cf. Section 6.2.3). Also, the influence of the choice of the pooling function on the generalization of GNNs has not been discussed yet and there is no guide for selecting suitable pooling functions in the literature.

We evaluate GNN pooling functions against the underlying physical nature of the learned properties. We analyze the impact of physical pooling functions on molecular properties learned from the common QM9 dataset [410, 411] and demonstrate their superior performance.

6.2 Materials and methods

We first describe the graph representation of molecules and then briefly introduce the general GNN architecture used for property prediction. Finally, we provide physical insight into the learned properties and use the insight to design physical GNN architectures.

6.2.1 Molecular graph

Molecules can be described as molecular graphs with nodes $v, w \in V$ representing atoms and edges $e_{vw} \in E$ representing bonds. Each atom is described by a feature vector $\mathbf{f}^V(v)$, containing atom information, e.g., atom mass or orbital hybridization. Similarly, each bond is descibed by a bond feature vector $\mathbf{f}^E(e_{vw})$ that contains information on the bond type, e.g., single or double bond. Commonly, the complexity of molecular graphs is reduced by omitting H atoms in the graph and replacing them by including the hydrogen count as a node feature.

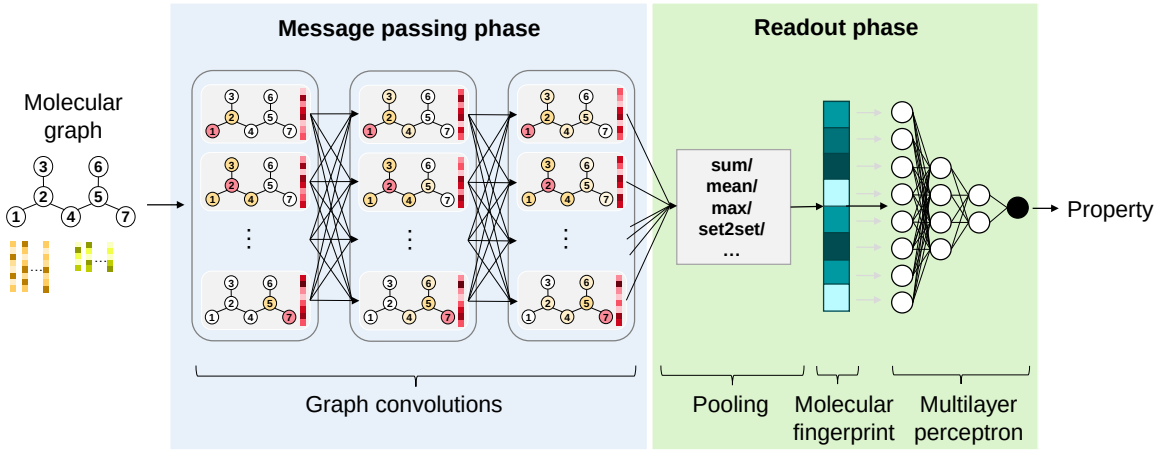


Figure 6.1.: Illustration of the GNN structure highlighting the message passing and readout phases.

6.2.2 Graph neural network

GNNs exhibit two phases [348] as shown in Figure 6.1: (i) message passing phase and (ii) readout phase.

To initialize the message passing phase, each node $v \in V$ is assigned a hidden state vector $\mathbf{h}_v^{l=0}$ initialized by the respective node feature vector [348]. Then, the hidden state vector of the nodes in layer l are updated with information from their neighboring nodes $w \in N(v)$ along edges e_{vw}:

$$\mathbf{h}_v^l = U_l \left(\mathbf{h}_v^{l-1}, \sum_{w \in N(v)} M_l \left(\mathbf{h}_v^{l-1}, \mathbf{h}_w^{l-1}, \mathbf{f}^E(e_{vw}) \right) \right),$$

where $U_l(\cdot)$ and $M_l(\cdot)$ respectively denote the hidden state update function and the message function in layer l. This message passing procedure is repeated L-times until each node state vector $\mathbf{h}_v^L$ includes information of its local environment. This iterative message passing corresponds to the stacking of L graph convolutional layers.

We consider two alternative message passing approaches. The first alternative is a standard GNN including edge features also known as 1-GNN [113, 417]. The 1-GNN uses the following message passing function:

$$\mathbf{h}_v^l = \sigma \left(\theta_v^l \cdot \mathbf{h}_v^{l-1} + \sum_{w \in N(v)} \mathrm{ANN}_{\theta_e^l}(\mathbf{f}^E(e_{vw})) \cdot \mathbf{h}_w^{l-1} \right),$$

where σ indicates an activation function, θ_v^l denotes a parameter matrix, and $\text{ANN}_{\theta_e^l}$ denotes a feedforward ANN mapping the respective feature vectors $\mathbf{f}^E(e_{vw})$ of the edges e_{vw} connecting node v with its neighbors to a parameter matrix θ_e^l, referred to as edge feature network. As second alternative, we consider the more complex MPNN message passing approach that operates on fully-connected molecular graphs, i.e., virtual edges associated with spatial distance features are added to the graph for unconnected atom pairs [348]. In the message passing phase, the model combines an edge feature network with a gated recurrent unit (GRU) [375]:

$$\mathbf{h}_v^l = \text{GRU}\left(\mathbf{h}_v^{l-1}, \sigma\left(\theta_v \cdot \mathbf{h}_v^{l-1} + \sum_{w \in N(v)} \text{ANN}_{\theta_e}\left(\mathbf{f}^E(e_{vw})\right) \cdot \mathbf{h}_w^{l-1}\right)\right).$$

In the readout phase, the final hidden state vectors of the nodes $\mathbf{h}_v^L$ are combined into a graph state vector $\mathbf{h}_G$ by a pooling function. The pooling function is necessary for molecular property prediction because the number of atoms can differ between different molecules. Thus, the pooling function combines a varying number of final hidden state vector for the nodes into a graph state vector. In the context of molecular property prediction, we refer to $\mathbf{h}_G$ as the molecular fingerprint. This molecular fingerprint $\mathbf{h}_G$ is finally fed into a feed-forward ANN for the prediction of molecular properties, $\hat{\mathbf{p}} = \text{MLP}(\mathbf{h}_G)$.

The molecular fingerprint is given by the pooling function $f_p(\cdot)$ that depends on the final hidden state vectors of the nodes $\mathbf{h}_v^L$ with $v \in V$:

$$\mathbf{h}_G = f_p\left(\{\mathbf{h}_v^L \mid v \in V\}\right)$$

Common choices for f_p are the sum, mean, and max functions. An alternative pooling function is the set2set method [407] which is designed to operate on input sets with an invariance for the order of the objects [348, 407]. The method is able to capture more complex relationships between different atomic contributions [348, 414] by employing a long short-term memory (LSTM) model [407]. After T steps of following iterative computation, the molecular fingerprint is obtained by $\mathbf{h}_G = \mathbf{q}_{t=T}^*$ with

$$\begin{aligned}
\mathbf{q}_t^* &= \mathbf{q}_t \parallel \mathbf{r}_t \\
\mathbf{q}_t &= \text{LSTM}\left(\mathbf{q}_{t-1}^*\right) \\
\mathbf{r}_t &= \sum_v \mathbf{a}_{v,t} \cdot \mathbf{h}_v^L \\
\mathbf{a}_{v,t} &= \text{softmax}\left(\mathbf{h}_v^L \cdot \mathbf{q}_t\right)
\end{aligned}$$

where $\mathbf{q}_t$ is a query vector for iteration t providing information about the previous attention readout vector $\mathbf{r}_t$ from the memories, $\mathbf{a}_{v,t}$ is an attention vector resulting from averaging the attention of a node v by applying the softmax function, $\mathbf{r}_t$ is the attention readout, similar to the simple pooling method with sum, and $\mathbf{q}_t$ is a concatenation ($\parallel$) of the current query vector and the attention readout.

6.2.3 Physical insight

The prediction of molecular properties by decomposing molecules into atomic contributions has a long history in chemical research. According to Bonchev (1991) [353], the first investigations into properties of molecules with additive characteristics in terms of atomic contributions were carried out in the 1850s. Later, quantitative structure-property relationships (QSPR) and group additivity methods were also developed based on the additive character of atoms or functional groups within a molecule [340–342, 418]. Yet, not every molecular property exhibits additive effects.

In thermodynamics, properties are considered intensive or extensive [419]. A system property is extensive if it increases linearly with the mass (and as such "extent") of the system; examples are the mass or volume. In contrast, intensive properties do not change with the system mass. Note that sometimes thermodynamicists distinguish between intensive (e.g., temperature and pressure) and specific properties (e.g., density), but we will not. We transfer these concepts to molecules and distinguish between *molecular size-independent* and *molecular size-dependent* properties. Molecular size-dependent properties scale with the number of atoms in a molecule. For example, the molecular weight is determined by how many atoms of which type are present in a molecule. In contrast, there exist molecular size-independent properties that do not scale with the number of atoms in a molecule, e.g., the highest occupied molecular energy level (HOMO) [414]. In addition, some properties of a substance, e.g., activity or toxicity, are mostly influenced by certain functional groups.

Note that this molecular size dependency does not necessarily correspond to the formal definition of intensive or extensive properties in a thermodynamic sense because they are considered at a microscopic atom-based molecular level, not at a macroscopic mass-based system level. For example, the molar enthalpy with the unit J/mol is an intensive property. On a molecular level, the enthalpy of atomization $H_{298,atom}$ with the unit J/mol, is a molecular size-dependent property describing the amount of energy needed to break up a molecule into all of its single atoms at room temperature and fixed pressure [348].

Schütt et al. (2018) consider the QM9 properties dipole moment (μ), isotropic polarizability (α), electronic spatial extent (R^2), zero point vibrational energy (ZPVE), heat capacity at 298.15K ($C_{v,298}$), atomization energy at 0K ($U_{0,atom}$), atomization energy at 298.15K ($U_{298,atom}$), enthalpy of atomization at 298.15K ($H_{298,atom}$), free energy of atomization at 298.15K ($G_{298,atom}$) as "extensive", except highest occupied molecular orbital (ϵ_{HOMO}), lowest unoccupied molecular orbital energy level (ϵ_{HOMO}), and HOMO-LUMO gap ($\Delta\epsilon$) [414].

6.3 Results and discussion

In order to demonstrate the relevance of the pooling function in the readout phase, two case studies are conducted and discussed below. First, we consider the illustrative prediction of the molecular weight in Section 6.3.2. Then, we consider the prediction of twelve quantum mechanical properties collected in the QM9 dataset in Section 6.3.3.

6.3.1 Hyperparameters and implementation

Our implementations are based on the models in PyTorch Geometric developed by Fey & Lenssen [372]. For our case study, we combine each mean-, sum-, and max- as well as set2set-pooling with both the 1-GNN and MPNN. The hyperparameters of the models are selected based on our experience from a related previous work [304]. Both the 1-GNN and MPNN constitute three graph convolutional layers with a hidden dimension size of 64. To map the molecular fingerprint ($\mathbf{h}_G$) to the property ($\hat{p}$) of interest, we use multilayer perceptrons (MLPs), $\hat{p} = \text{MLP}(\mathbf{h}_G)$. The MLPs constitute four layers with #1: 64, #2: 32 #3: 16, #4: 1 neurons when mean-, sum-, or max-pooling is used. When the set2set pooling is used, we use a three-layer MLP with #1: 128, #2: 64, #3: 1 neurons, because the output vector of the set2set method is twice its input size. For the set2set method, we set the number of processing steps T to 3.

6.3.2 Illustrative case study: Molecular weight

In this case study we learn the molecular weight of alkanes. To compose the dataset, we obtain about 3,000 alkanes from C_1H_4 to $C_{60}H_{122}$ from the PubChem database [420] and compute their molecular weight using RDKit [374].

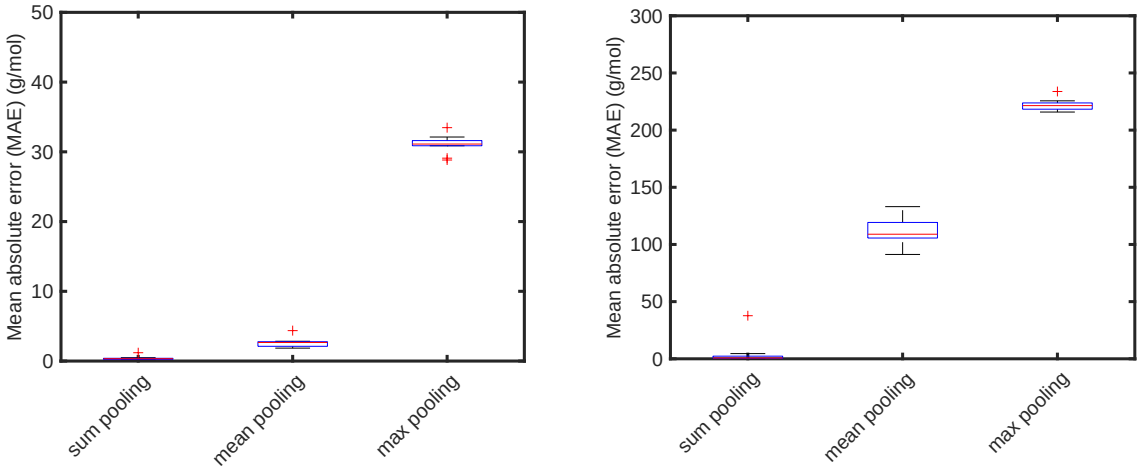
For illustration, we split this case study into two steps. Firstly, 1-GNNs with sum, mean, and max pooling functions are trained, validated, and tested on corresponding datasets with alkanes with up to 30 C-atoms. Secondly, the trained GNNs are tested against an external dataset containing alkanes with more than 30 C-atoms. Thus, the generalization capabilities of the 1-GNNs are tested against extrapolated data.

The training of the 1-GNNs is repeated ten times for each pooling function and the initial dataset with up to 30 C-atoms is randomly split in 80% training, 10% validation, and 10% test sets for each run. Early stopping is applied with a patience of 30 epochs and a maximum number of 300 epochs.

Figure 6.2 shows the test set performance of the 1-GNN with different pooling functions. Figure 6.2 (a) illustrates the test results for the dataset of alkanes with up to 30 C-atoms. As expected, the sum pooling leads to the best performance on the test data set with an average MAE of 0.41 g/mol as it captures the molecular size-dependent character of the molecular weights. lIn contrast, mean pooling leads to an average mean absolute error 2.6 g/mol. Max pooling even leads to an average absolute error in the order of 31 g/mol.

Figure 6.2 (b) shows the mean absolute error on the test datasets of alkanes with more than 30 C-atoms. The 1-GNN with sum pooling leads to an average absolute error of 4.8 g/mol. In contrast, the 1-GNN with mean pooling leads to an average error of 110 g/mol and the 1-GNN with max-pooling leads to an average error of 222 g/mol.

The results clearly shows that the sum pooling function, which was selecteced based on our physical insight, performs better in extrapolation than the unphysical mean and max pooling functions for the prediction of the molecular weight. In particular, the sum pooling outperforms the unphysical pooling functions significantly when extrapolating the model. These results support our theoretical findings that GNNs with unphysical pooling functions can only learn through inherent overfitting. Notably, the MSE of the GNN with mean pooling is much smaller on the test set with alkanes with less than 30 C-atoms compared to the test set with alkanes with more than 30 C-atoms. This result indicates that the selection of pooling functions based on the performance on a standard validation



(a) (b)

Figure 6.2.: Mean absolute error in g/mol for test of the 1-GNN with different pooling functions, sum, mean, max: (a) test dataset of alkanes with up to 30 C-atoms, (b) external dataset of alkanes with 35 up to 60 C-atoms. Results are for ten independent training runs, each with 300 periods.

or test set could also be error prone.

6.3.3 Physicochemical properties

The computation of many physicochemical properties is possible through quantum mechanical calculations but requires large computational time. We analyze the importance of physical pooling functions for a variety of relevant properties. In addition, we compare our results to a more complex set2set readout function and explore a MPNN GNN architecture.

We use the QM9 dataset to train our models [410, 411]. The experimental setup is twofold. First, the 1-GNNs with sum, mean, and max pooling functions are trained, validated, and tested on randomly selected subsets of the whole QM9 dataset. This approach assesses the interpolation capabilites of our models (Section 6.3.3.1). Second, all models are trained, validated, and tested on the QM9 data excluding molecules with exactly 9 heavy atoms. These models are then tested against molecules with 9 heavy atoms from the QM9 dataset. This approach is chosen to asses the generalization capabilities of our models (Section 6.3.3.2). For each training run, the dataset is randomly split into 80% training, 10% validation, and 10% test sets. The training is stopped after 300 periods. The mean absolute error for the test set is reported based on the period with the lowest validation error.

6.3.3.1 Interpolation

The results for testing sum, mean, and max pooling function on the whole QM9 dataset are summarized in Table 6.1. Overall, the results indicate that physically meaningful pooling functions lead to favorable performances on the QM9 dataset for interpolation. It can be observed that for all molecular size-dependent properties except the dipole moment (μ),

Table 6.1.: Mean absolute errors - median over ten training runs - for testing 1-GNN with different pooling functions, sum, mean, max against QM9 target properties. Properties are categorized into molecular size-independent ("m. size-ind.") and molecular size-dependent ("m. size-dep.") according to [414]. Errors of best pooling function are bold type.

	Target		Pooling		
			sum	mean	max
m. size-dep.	μ	Debye	0.496	0.491	**0.489**
	α	a_0^3	**0.486**	0.739	0.761
	R^2	a_0^2	**29.3**	39.2	39.0
	ZPVE	10^{-4} E_h	**5.13**	11.87	16.86
	$C_{v,298}$	cal/mol K	**0.194**	0.327	0.338
	$U_{0,atom}$	kcal/mol	**3.14**	10.21	12.15
	$U_{298,atom}$	kcal/mol	**3.63**	11.96	11.85
	$H_{298,atom}$	kcal/mol	**3.48**	11.38	11.54
	$G_{298,atom}$	kcal/mol	**3.15**	9.13	11.71
m. size-ind.	ϵ_{HOMO}	10^{-3} E_h	3.79	**3.60**	3.67
	ϵ_{LUMO}	10^{-3} E_h	3.94	**3.78**	3.82
	$\Delta\epsilon$	10^{-3} E_h	5.48	5.20	**5.09**

the 1-GNN with sum pooling significantly outperforms the 1-GNNs with mean and max pooling. For the dipole moment, all pooling function lead to a similar performance. For the molecular size-independent properties, HOMO and LUMO, mean pooling yields slightly better prediction accuracy compared to sum and max pooling. For the molecular size-independent property HOMO-LUMO-gap, the 1-GNN with max pooling performs slightly better than the ones with sum and mean pooling.

6.3.3.2 Generalization with 1-GNN architecture

In order to analyze the generalization capability of the pooling functions on the QM9 dataset, we train the 1-GNNs only on molecules with up to 8 heavy atoms, i.e., a maximum number of 8 C, N, O, F atoms in total. Then, we test the prediction accuracy on an internal testset, i.e., containing molecules with up to 8 heavy atoms, and also on a dataset with remaining QM9 molecules that have exactly 9 heavy atoms. It should be noted that this case study has a limited view on the generalization because the molecules in the external test sets (9 atoms) are only slightly larger than the ones in the training set (up to 8 atoms). The results of the extrapolation are summarized in Table 6.2.

As expected, the interpolation performance of the models (indicated in black in Table 6.2) is similar to the previous models trained on the whole QM9 dataset (c.f. Table 6.1). Notably, the absolute errors increase for some properties, e.g., atomization energy, as the training set is much smaller. The QM9 dataset contains about 110,000 molecules with 9 atoms and about 22,000 molecules with 1 to 8 heavy atoms.

For extrapolation (indicated in blue in Table 6.2), we find that sum-pooling performs much better compared to mean, max, and set2set pooling on almost all tested molecular size-dependent properties. Only the dipol moment μ shows similar performance for all

Table 6.2.: Mean absolute errors averaged over ten independent training runs for testing 1-GNN with different pooling functions, sum, mean, max, and set2set, against: (black) QM9 dataset excluding molecules with 9 heavy atoms, (blue) only molecules with 9 heavy atoms of the QM9 dataset. Error of best pooling function are bold type. Units are equivalent to those in Table 6.1.

Target	1-GNN			
	sum	mean	max	set2set
μ	0.476	0.459	**0.457**	0.562
	0.593	0.573	**0.570**	0.653
α	**0.503**	1.297	0.998	1.731
	1.257	8.015	8.346	7.841
R^2	**22.9**	39.2	34.8	64.1
	81.3	229.2	207.5	247.2
ZPVE	**0.56**	2.18	2.26	2.70
	0.81	17.09	15.98	13.45
$C_{v,298}$	**0.272**	0.465	0.399	0.751
	0.616	3.532	3.391	3.317
$U_{0,atom}$	**6.02**	26.49	20.46	28.67
	23.42	218.04	201.05	214.76
$U_{298,atom}$	**6.92**	25.57	20.55	22.65
	24.93	229.62	195.30	218.40
$H_{298,atom}$	**5.99**	25.65	20.74	23.32
	16.60	227.69	201.30	204.17
$G_{298,atom}$	**5.73**	21.78	20.64	23.91
	24.74	209.94	185.37	208.89
ϵ_{HOMO}	4.20	**3.86**	4.01	4.77
	5.41	**4.98**	5.07	5.87
ϵ_{LUMO}	4.07	**3.74**	3.84	4.73
	5.55	**5.29**	5.35	6.12
$\Delta\epsilon$	5.65	5.37	**5.32**	6.37
	7.43	7.39	**7.28**	8.26

pooling function. For the molecular size-independent properties we observe much smaller performance differences between different pooling functions. For ϵ_{HOMO} and ϵ_{LUMO}, the mean-pooling function outperforms the others. For $\Delta\epsilon$, the max-pooling function demonstrates a favorable performance.

In addition to the common sum-, mean-, and max-pooling functions, we also consider the set2set method [407] as a pooling method for the 1-GNN. Notably, the set2set method does not improve the accuracy compared to sum, mean, and max pooling for any property. In the case of molecular size-independent properties, the set2set methods even perform unfavorably.

6.3.3.3 Generalization with MPNN architecture

We also compare our results to the MPNN GNN architecture. The MPNN architecture operates on fully-connected graphs with a GRU in the message passing phase combined with the respective pooling functions is applied. The results are summarized in Table 6.3.

The interpolation performance of the MPNN (indicated in black in Table 6.3) is favorably compared to the 1-GNN architecture but is still in the same order of magnitude. Again, the MPNN with sum-pooling outperforms the other pooling approaches for almost all molecular size-dependent properties. Only for μ, all pooling approaches lead to comparable performances. For the molecular size-independent properties, we observe that sum-, mean, and set2set-pooling perform very similarly. Only max-pooling shows slightly worse performance for these properties. Interestingly, the sum-pooling function slightly outperforms the other approaches for the UMO and HOMO-LUMO-gap.

Notably, the generalization performance of the MPNN architecture (indicated in blue in Table 6.3) also outperforms the 1-GNN architecture. For some properties, the MPNN architecture even reduces the MSE by a factor of about 2. The generalization results follow the same pattern as our previous observations. Again, we observe a significant advantage of sum-pooling for almost all molecular size-dependent properties. Interestingly, sum-pooling is also slightly favorably for the molecular size-independent properties. However, the conclusions we can draw from these observations are very limited because the performance differences are rather small and our extrapolation test set is very similar to our training set in this case study. Although for the MPNN the set2set approach performs better than mean and max-pooling when testing on familiar data, the methods do not generalize well on the data set with 9 heavy atoms.

6.4 Conclusion

The selection of pooling functions in GNNs for property prediction should be based on physical knowledge because incorrect pooling functions lead to inherent overfitting and prevent generalization. The key property for the selection of the pooling function is the molecular size-dependency of the learned property. Our results support one clear rule that says: When a property is molecular size-dependent, the sum pooling function should be used. Our computational results support this hypothesis showing that physical GNN architectures generalize better than unphyscial architectures. In future research, the physical selection of pooling functions should always be considered when predicting molecular properties with GNNs.

Table 6.3.: Mean absolute errors averaged over ten independent training runs for testing MPNN with different pooling functions, sum, mean, max, and set2set, against: (black) QM9 dataset excluding molecules with 9 heavy atoms, (blue) only molecules with 9 heavy atoms of the QM9 dataset. Error of best pooling function are bold type. Units are equivalent to those in Table 6.1.

Target	**MPNN**			
	sum	mean	max	set2set
μ	0.468	**0.466**	0.480	0.491
	0.580	**0.571**	0.592	0.590
α	**0.318**	0.958	1.059	0.619
	0.741	8.284	8.052	8.324
R^2	**18.9**	29.8	28.0	21.1
	51.6	225.3	213.4	245.4
ZPVE	**0.22**	2.87	2.83	1.62
	0.46	15.02	13.48	14.58
$C_{v,298}$	**0.130**	0.362	0.356	0.212
	0.264	3.491	3.398	3.687
$U_{0,atom}$	**2.66**	24.36	22.81	13.02
	10.38	194.04	188.07	196.83
$U_{298,atom}$	**2.64**	24.10	24.65	15.60
	13.29	193.52	189.35	193.95
$H_{298,atom}$	**2.69**	23.19	25.32	15.93
	12.92	197.35	192.04	194.12
$G_{298,atom}$	**2.61**	24.56	21.72	13.50
	10.15	178.20	174.94	181.33
ϵ_{HOMO}	**3.73**	3.89	4.23	3.86
	4.77	4.88	5.25	4.94
ϵ_{LUMO}	**3.54**	3.66	4.13	3.75
	4.97	5.27	5.49	5.21
$\Delta\epsilon$	5.22	5.40	5.47	**5.20**
	6.95	7.34	7.43	7.24

Chapter 7

Conclusion

7.1 Summary

Chemical engineering is at a crossroad. Worldwide, the chemical industry is responsible for 10% of the total energy utilization and relies almost entirely on fossil sources. A transformation of the chemical industry to renewable energy and feedstock supply is of great importance.

To tackle the transformation of chemical production, we identified six areas of interdisciplinary research that open up new methods for chemical engineering, formulate new types of problems for ML, and jointly generate advances for methods in both ML and chemical engineering. These emerging areas are:

- # 1 Optimal decision making,
- # 2 Introducing and enforcing physics in ML,
- # 3 Information and knowledge representation
- # 4 Heterogeneity of data,
- # 5 Safely and trust in ML applications,
- # 6 Creativity.

Under the umbrella of these areas, this dissertation integrated advanced ML algorithms into the optimal decision-making process in chemical engineering and addressed relevant challenges in modeling, optimization formulations, and optimization algorithms.

One major challenge for optimal decision making based on data (area #1) has been the complexity of global optimization problems with ML models embedded. Our developed methods accelerated global optimization with ML models embedded by orders of magnitude and thus, can solve problems that were previously intractable. In particular, we proposed a RS formulation for ANNs, GPs, and one-class SVMs. In addition, we derived and implemented tight function relaxations for nonlinear functions that occur repeatedly in ML models and related problems. This includes activations functions of ANNs, covariance functions of GPs, kernel functions of SVMs, and acquisition functions for Bayesian optimization.

Safety concerns are a major challenge for the application of data-driven models in industry (area #5). We developed a method to model the validity domain of data-driven models and obey these limits in global optimization. First, we analyze the training data of a data-driven model using topological data analysis. Then, we train an unsupervised one-class SVM to learn the validity domain of the data-driven model. This one-class SVM can be embedded as a constraint during optimization to ensure valid model evaluations.

To accelerate global optimization, we formulate it in the RS and use envelopes of the kernel functions of the SVM.

An additional hurdle for the actual implementation of ML models for optimal decision making (area # 1) has been the lack of a model interface. While the ML community relies mostly on open-source developments in Python, the PSE community often uses specialized modeling and optimization environments. For example, the common optimization environment GAMS uses a text input for modeling. This complicates the use of ML models which often include a large number of parameters. We developed the open-source tool MeLOn [132] that interfaces state-of-the-art ML software, e.g., Keras, scikit-learn Learn, and Matlab, to our open-source solver MAiNGO [178]. This simplifies the workflow for training and optimization with trained ML models embedded significantly.

Chemical engineering data often comes in different formats (area # 4) that are not accessible by standard methods (area # 3). For example, the prediction of physicochemical properties is a relevant task where molecular structures are the data input and real-valued properties are the desired output. We utilized GNNs to learn the ignition quality of biofuels from molecular structures that are represented by graphs. This representation of information allows end-to-end learning and overcomes the need for manual feature selection like in previous QSPR models. Above all, the developed model contributes to the transformation of the chemical industry, in particular to more sustainable feedstocks.

Although the architecture of GNNs is highly flexible, we recognized that some common pooling functions are unphysical and can lead to inherent overfitting. We introduce physical knowledge to GNNs by a meaningful selection of pooling functions (area # 2). We show on a number of examples that the physical selection of pooling functions can reduce overfitting considerably and enhance model generalization.

7.2 Outlook

While this dissertation covers several interdisciplinary research topics between ML and chemical engineering, there are many more promising avenues for future work within the six emerging areas described in Section 1.4. In the following, we point out a few directions for future research related to the optimal decision making based on data.

One promising research direction is the further integration of the developed optimization methods with experimental data for the rational design of material properties (c.f. [13, 14, 21, 253]) and with automated experimentation platforms for optimization with hardware in the loop (also known as "self-optimization" [9, 23, 24, 421, 422]). In particular, the global maximization of the acquisition function in Bayesian optimization with the B&B algorithm allows solving mixed-integer nonlinear optimization problems. Thus, integer variables and nonlinear constraints can be considered in Bayesian optimization approaches. We are currently exploring this topic and our preliminary results demonstrate that it enables the optimal selection of operating points (e.g., temperatures, pressures, concentrations) together with the optimal selection of solvents, catalysts, and co-catalysts.

Another promising application of our method in chemical engineering is the optimization of expensive to evaluate simulations. For example, the fouling behavior of microreactors can be modeled through computational fluid dynamic simulations. Bayesian optimization can be used to optimize the design of the microreactor to identify an optimal trade-off between reducing the fouling risk, enhancing mixing, and reducing pressure drop. We are

currently exploring this topic and our preliminary results show that Bayesian optimization is well-suited for this task and can identify good designs in few simulation runs.

This dissertation focuses mostly on hybrid models for stationary processes but many (bio-)chemical processes are dynamically operated and the need for transient operations is expected to increase [36]. Thus, the integration of ML models into dynamic process models is promising future research. For example, our preliminary results show that discrete-time GP models can learn the state of health of Lithium-Ion batteries. Similarly, most previous research in this area rely on discrete-time representations (e.g., [20, 33]) but these often do not capture the long-term dynamic behavior [423]. Thus, the training of continuous-time dynamic models is desirable. We are currently exploring this topic and our preliminary results show that the combination of an integration solver into a state-of-the-art ML training environment allows the training of continuous-time dynamic hybrid models.

Surrogate models can learn simulated data and replace simulations to accelerate computations, e.g., for control. Although ML models can theoretically learn simulations to high accuracy, they could also lead to considerable prediction errors. Thus, enhancing reliability and trust in surrogate models is important. In our previous work [10], we used global optimization to compute a guaranteed worst-case accuracy of surrogate models. In future work, one could integrate this concept into a bilevel training algorithm. To solve the bilevel training rigorously, the global solution of the training problem is necessary. Thus, SVMs would be a promising first step for this research. We are currently exploring this topic and our preliminary results show that the bilevel training with SVMs and ANNs can improve the guaranteed worst-case accuracy of surrogate models.

The bilevel training to guaranteed accuracy is one example where global training of ANNs is desirable but currently intractable. Future research could focus on new methods and formulations for global training of ANNs. One challenge are the inherent symmetries of the training problem. We are currently exploring this topic and our preliminary results show that breaking these symmetries through constraints is possible and can reduce computational time for training. However, further advances are necessary to allow training of relevant problem sizes.

As shown in Chapter 2, the computational time for optimizing problems with ANNs embedded depends on the ANN architecture, i.e., the number of neurons and layers. This behavior is undesirable. Ideally, we want the computational time to scale with the complexity of the learned function and the accuracy of the ANN. One promising future work is to reduce the ANN complexity through pruning. We are currently exploring this topic and our preliminary results show that pruning can reduce computational time for optimization considerably while achieving high model accuracies.

We experienced that some hyperparameters of the global optimization solver have a considerable effect on its performance. For instance, we find that branching priorities have a strong influence on the solution time. One way to tune such parameters for relevant problem classes could be a Bayesian optimization approach that aims to change the branching priorities to reduce computational time. We are currently exploring this topic and our preliminary results show that Bayesian optimization can be used to tune such hyperparameters but the transfer of optimal branching priorities to other problems is challenging. However, we expect that the transfer of good branching priorities to structural similar problems is promising. For example, one could identify good branching priorities for 2-stage stochastic programming problems with few scenarios and transfer them to cases with a large number of scenarios. Similarly, this could be applied to reaction network flux

analysis problems, often utilized for the evaluation of novel reaction routes from renewable feedstocks.

This dissertation focuses on nonlinear programming formulation for ML models. However, some ML models such as ReLU ANNs or tree models can be formulated as mixed-integer linear problems (MILPs). These formulations can be solved by efficient linear solvers but the number of binary variables scales linearly with the model complexity (e.g., number of neurons for the ReLU ANNs) [81, 84–87]. An alternative is to formulate these models in the RS which results in nonsmooth and nonlinear problems (NLPs). In this case, the problem size scales only with the degrees of freedom and not with the model complexity. A comparison between the MILP and RS NLP formulation is a relevant future work. We are currently exploring this topic and our preliminary results show competitive performances of both formulations.

Finally, the integration of GNNs into optimal decision making is a promising research direction that combines all aspects of this thesis. Here, the RS optimization method can be applied to the molecular fingerprint of GNNs for molecular design. For example, the GNNs for ignition quality can be combined with a (surrogate) engine model to design biofuels. Also, GNNs can learn other relevant fuel properties such as toxicity. When optimizing the GNNs, it is important to obey the validity limits of the GNN models during optimization. The first step in this research direction could be the developed one-class classification method which could be applied to the molecular fingerprint space. Moreover, the use of generative models for molecular design is emerging. A comparison between the optimization-based approach and the generative models would be a promising future research.

Appendix A

Convex and concave function relaxations

For the sake of simplicity, we use the same symbols in each subsection for the corresponding convex (F^{cv}) and concave (F^{cc}) relaxations. To solve the one-dimensional nonlinear equations that arise multiple times in the following, we use Newton's method with 100 iterations and a tolerance of 10^{-9}. If this is not successful, we run a golden section search as a backup.

A.1 Hyperbolic tangent activation function

In this section, the envelopes of the hyperbolic tangent function are derived on a compact interval $D = [x^L, x^U]$. As the hyperbolic tangent function is one-dimensional, McCormick (1976) [167] gives a method to construct its envelopes. More specifically, as the hyperbolic tangent function is convex on $]-\infty, 0]$ and concave on $[0, +\infty[$, its convex envelope, $F^{cv} : \mathbb{R} \to \mathbb{R}$, and concave envelope, $F^{cc} : \mathbb{R} \to \mathbb{R}$, are given:

$$F^{cv}(x) = \begin{cases} \tanh(x), & x^U \leqslant 0, \\ \mathrm{sct}(x), & 0 \leqslant x^L, \\ F_3^{cv}(x), & \text{otherwise}, \end{cases} \tag{A.1}$$

$$F^{cc}(x) = \begin{cases} \mathrm{sct}(x), & x^U \leqslant 0, \\ \tanh(x), & 0 \leqslant x^L, \\ F_3^{cc}(x), & \text{otherwise}, \end{cases} \tag{A.2}$$

where the secant is given as $\mathrm{sct}(x) = \frac{\tanh(x^U)-\tanh(x^L)}{x^U-x^L}x + \frac{x^U\tanh(x^L)-x^L\tanh(x^U)}{x^U-x^L}$. For $x^L < 0 < x^U$, the hyperbolic tangent function is nonconvex and nonconcave. The convex envelope, $F_3^{cv} : \mathbb{R} \to \mathbb{R}$, for this case is:

$$F_3^{cv}(x) = \begin{cases} \tanh(x), & x \leqslant x_c^u, \\ \frac{\tanh(x^U)-\tanh(x_c^u)}{x^U-x_c^u} \cdot (x - x_c^u) + \tanh(x_c^u), & x > x_c^u, \end{cases} \tag{A.3}$$

where $x_c^u = \max(x_c^{u*}, x^L)$ and x_c^{u*} is the solution of:

$$1 - \tanh^2(x) = \frac{\tanh(x^U) - \tanh(x)}{x^U - x}, \quad x \leqslant 0 \tag{A.4}$$

Similarly, $F_3^{cc} : \mathbb{R} \to \mathbb{R}$ is given by:

$$F_3^{cc}(x) = \begin{cases} \frac{\tanh(x_c^o)-\tanh(x^L)}{x_c^o-x^L} \cdot (x - x^L) + \tanh(x^L), & x < x_c^o, \\ \tanh(x), & x \geqslant x_c^o, \end{cases} \tag{A.5}$$

where $x_c^o = \min(x_c^{o*}, x^U)$ and x_c^{o*} is the solution of:

$$1 - \tanh^2(x) = \frac{\tanh(x) - \tanh(x^L)}{x - x^L}, \quad x \geqslant 0 \tag{A.6}$$

In the following, we show that the convex and concave envelopes of the hyperbolic tangent function are smooth (C^1) and strictly monotonically increasing.

Proposition A.1 *(smoothness of hyperbolic tangent relaxations). The convex and concave envelopes of the hyperbolic tangent function, $F^{cv}(x)$ and F^{cc}, are once continuously differentiable (C^1) and in general not C^2.*

Proof The envelopes are at least C^1 because they are derived by matching the derivatives of the hyperbolic tangent function and the secant line at the inflection point for convexity. They are at most C^1 because $\frac{\mathrm{d}^2(F_3^{cv})}{\mathrm{d}x^2}|_x = -2\,\mathrm{sech}^2(x)\tanh(x) \neq 0$ for $x < x_c^u$ and $\frac{\mathrm{d}^2(F_3^{cc})}{\mathrm{d}x^2}\Big|_x = -2\,\mathrm{sech}^2(x)\tanh(x) \neq$ for $x \geqslant x_c^o$, where $\mathrm{sech}(x)$ is the hyperbolic secant function. □

As shown in Proof A.1, the first derivative of the convex and concave envelopes of the hyperbolic tangent function, $\frac{\mathrm{d}(F^{cv})}{\mathrm{d}x}$ and $\frac{\mathrm{d}(F^{cc})}{\mathrm{d}x}$, are continuous but not continuously differentiable. The following proof shows that the first derivative of the convex and concave envelopes of the hyperbolic tangent function are Lipschitz continuous.

Proposition A.2 *(Lipschitz continuity of 1st derivative of hyperbolic tangent relaxations). The second derivative of the convex and concave envelopes of the hyperbolic tangent function are bounded. This implies that the first derivative of the convex and concave envelopes of the hyperbolic tangent function, $\frac{d(F_3^{cv})}{dx}$ and $\frac{d(F_3^{cc})}{dx}$, are at least Lipschitz continuous.*

Proof For $x^U \leqslant 0$, $F^{cv}(x) = \tanh(x)$ and $F^{cc}(x) = \mathrm{sct}(x)$ which are C^∞. Similarly, for $0 \leqslant x^L$, $F^{cc}(x) = \tanh(x)$ and $F^{cv}(x) = \mathrm{sct}(x)$ which are C^∞. For $x^L < 0 < x^U$, the second derivative of the convex and concave envelopes of the hyperbolic tangent function can be bounded by

$$\left|\frac{\mathrm{d}^2(F_3^{cv}(x))}{\mathrm{d}x^2}\right| \leqslant 2\left|\mathrm{sech}^2(\tilde{x}^{cv})\tanh(\tilde{x}^{cv})\right| = 2\left|\frac{\sinh(\tilde{x}^{cv})}{\cosh^3(\tilde{x}^{cv})}\right| \leqslant 2\sinh(\left|x^L\right|) \tag{A.7}$$

with $x^L \leqslant x \leqslant x^U$, $x^L \leqslant \tilde{x}^{cv} \leqslant x_c^u$, and $\cosh(x) \geqslant 1$. Thus, $\frac{\mathrm{d}(F^{cv})}{\mathrm{d}x}$ is at least Lipschitz continuous with a Lipschitz constant of at most $L^{cv} = 2\sinh(\left|x^L\right|)$. Similarly, it holds that $\frac{\mathrm{d}(F^{cc})}{\mathrm{d}x}$ is at least Lipschitz continuous with a Lipschitz constant of at most $L^{cc} = 2\sinh(\left|x^U\right|)$. □

Proposition A.3 *(monotonicity of hyperbolic tangent relaxations). The convex and concave envelopes of the hyperbolic tangent function are strictly monotonically increasing.*

Proof For $x^U \leqslant 0$, $F^{cv}(x) = \tanh(x)$ and $F^{cc}(x) = \mathrm{sct}(x)$. Similarly, for $0 \leqslant x^L$, $F^{cc}(x) = \tanh(x)$ and $F^{cv}(x) = \mathrm{sct}(x)$. As $\frac{\mathrm{d}(\tanh(x))}{\mathrm{d}x} = 1 - \tanh^2(x) > 0$, $\tanh(x)$ is strictly monotonically increasing. As $x^U > x^L$, $\mathrm{sct}(x)$ is strictly monotonically increasing. For $x^L < 0 < x^U$, the envelopes are given by (A.3) and (A.5). These are again strictly

monotonically increasing because $x_c^o > x^L$ and $x^U > x_c^u$. □

A.2 Probability density function of Gaussian distribution

In this subsection, the envelopes of the PDF are derived on a compact interval $D = [x^L, x^U]$. As the probability density function is one-dimensional, McCormick [167] gives a method to construct its envelopes. The PDF is convex on $]-\infty, -1]$ and $[1, \infty[$ and it is concave on $[-1, 1]$. Its convex envelope, $F^{cv} : \mathbb{R} \to \mathbb{R}$, and concave envelope, $F^{cc} : \mathbb{R} \to \mathbb{R}$, are given by

$$F^{cv}(x) = \begin{cases} \phi(x), & x^U \leqslant -1, \\ F_2^{cv}(x), & x^L \leqslant -1, \quad -1 \leqslant x^U \leqslant 1, \\ \mathrm{sct}(x), & -1 \leqslant x^L, \quad x^U \leqslant 1, \\ F_4^{cv}(x), & -1 \leqslant x^L, \quad x^U \geqslant 1, \\ F_5^{cv}(x), & x^L \leqslant -1, \quad x^U \geqslant 1, \\ \phi(x), & x^L \geqslant 1 \end{cases}$$

$$F^{cc}(x) = \begin{cases} \mathrm{sct}(x), & x^U \leqslant -1, \\ F_2^{cc}(x), & x^L \leqslant -1, \quad -1 \leqslant x^U \leqslant 1, \\ \phi(x), & -1 \leqslant x^L, \quad x^U \leqslant 1, \\ F_4^{cc}(x), & -1 \leqslant x^L \leqslant 1, \quad x^U \geqslant 1, \\ F_5^{cc}(x), & x^L \leqslant -1, \quad x^U \geqslant 1, \\ \mathrm{sct}(x), & x^L \geqslant 1 \end{cases}$$

where $\mathrm{sct}(x) = \frac{\phi(x^U)-\phi(X^L)}{x^U-x^L} \cdot (x - x^L) + \phi(x^L)$. $F_2^{cc} : \mathbb{R} \to \mathbb{R}$ is given by:

$$F_2^{cc}(x) = \begin{cases} \frac{\phi(x_{c,2}^U)-\phi(x^L)}{x_{c,2}^U-x^L} \cdot (x - x^L) + \phi(x^L), & x \leqslant x_{c,2}^U, \\ \phi(x), & x > x_{c,2}^U, \end{cases}$$

where $x_{c,2}^U = \min(x_{c,2}^{U*}, x^U)$ and $x_{c,2}^{U*}$ is the solution of $\frac{d\phi}{dx}\big|_x = \frac{\phi(x)-\phi(x^L)}{x-x^L}, \quad x \in [-1, 0]$. $F_2^{cv} : \mathbb{R} \to \mathbb{R}$ is given by:

$$F_2^{cv}(x) = \begin{cases} \phi(x), & x \leqslant x_{c,2}^L, \\ \frac{\phi(x^U-\phi(x_{c,2}^L))}{x^U-x_{c,2}^L} \cdot (x - x_{c,2}^L) + \phi(x_{c,2}^L), & x > x_{c,2}^L, \end{cases}$$

where $x_{c,2}^L = \max(x_{c,2}^{L*}, x^L)$ and $x_{c,2}^{L*}$ is the solution of $\frac{d\phi}{dx}\big|_x = \frac{\phi(x)-\phi(x^L)}{x-x^L}, \quad x \in [x^L, -1]$. $F_4^{cc} : \mathbb{R} \to \mathbb{R}$ is given by:

$$F_4^{cc}(x) = \begin{cases} \phi(x), & x < x_{c,4}^L, \\ \frac{\phi(x^U)-\phi(x_{c,4}^L)}{x^U-x_{c,4}^L} \cdot (x - x_{c,4}^L) + \phi(x_{c,4}^L), & x \geqslant x_{c,4}^L, \end{cases}$$

where $x_{c,4}^L = \max(x_{c,4}^{L*}, x^L)$ and $x_{c,4}^{L*}$ is the solution of $\frac{d\phi}{dx}\big|_x = \frac{\phi(x)-\phi(x^L)}{x-x^L}$, $x \in [0, x^U]$. $F_4^{cv} : \mathbb{R} \to \mathbb{R}$ is given by:

$$F_4^{cv}(x) = \begin{cases} \frac{\phi(x_{c,4}^U)-\phi(x_L)}{x_{c,4}^U - x_L} \cdot (x - x_L) + \phi(x_L), & x \leqslant x_{c,4}^U, \\ \phi(x), & x > x_{c,4}^U, \end{cases}$$

where $x_{c,4}^U = \min(x_{c,4}^{U*}, x^U)$ and $x_{c,4}^{U*}$ is the solution of $\frac{d\phi}{dx}\big|_x = \frac{\phi(x)-\phi(x^L)}{x-x^L}$ on $[x^L, -1]$ if $x^L + x^U < 0$ or $[1, x^U]$ if $x^L + x^U > 0$. The case $x^L + x^U = 0$ is symmetrical and handled separately to avoid numerical issues in Newton. $F_5^{cc} : \mathbb{R} \to \mathbb{R}$ is given by a combination of F_2^{cc} and F_4^{cc}:

$$F_5^{cc}(x) = \begin{cases} \frac{\phi(x_{c,2}^U)-\phi(x^L)}{x_{c,2}^U - x^L} \cdot (x - x^L) + \phi(x^L), & x \leqslant x_{c,2}^U, \\ \phi(x), & x_{c,2}^U < x < x_{c,4}^L, \\ \frac{\phi(x^U)-\phi(x_{c,4}^L)}{x^U - x_{c,4}^L} \cdot (x - x_{c,4}^L) + \phi(x_{c,4}^L), & x \geqslant x_{c,4}^L, \end{cases}$$

$F_5^{cv} : \mathbb{R} \to \mathbb{R}$ is given by:

$$F_5^{cv}(x) = \begin{cases} \frac{\phi(x_{c,5}^U)-\phi(x^L)}{x_{c,5}^U - x^L} \cdot (x - x^L) + \phi(x^L), & x^L + x^U \geqslant 0, \quad x \leqslant x_{c,5}^U, \\ \phi(x), & x^L + x^U \geqslant 0, \quad x > x_{c,5}^U, \\ \phi(x), & x^L + x^U < 0, \quad x \leqslant x_{c,5}^L, \\ \frac{\phi(\phi(x^U - x_{c,5}^L))}{x^U - x_{c,5}^L} \cdot (x - x_{c,5}^L) + \phi(x_{c,5}^L), & x^L + x^U < 0, \quad x > x_{c,5}^L, \end{cases}$$

where $x_{c,5}^U = \min(x_{c,5}^{U*}, x^U)$ and $x_{c,5}^{U*}$ is the solution of $\frac{d\phi}{dx}\big|_x = \frac{\phi(x)-\phi(x^L)}{x-x^L}$ on $[x^L, 0]$. Further, $x_{c,5}^L = \max(x_{c,5}^{L*}, x^L)$ and $x_{c,5}^{L*}$ is the solution of $\frac{d\phi}{dx}\big|_x = \frac{\phi(x)-\phi(x^L)}{x-x^L}$ on $[0, x^U]$.

A.3 Probability of improvement acquisition function

In this section, a tight relaxation of the PI acquisition function is derived. PI is continuous for all $(\mu, \sigma) \in \mathbb{R} \times [0, \infty[\setminus \{(0, 0)\}$, since $\lim_{x \to +\infty} \Phi(x) = 1$ and $\lim_{x \to -\infty} \Phi(x) = 0$.

A.3.0.1 Monotonicity

From the gradient of PI on $\mathbb{R} \times (0, \infty)$,

$$\nabla \mathrm{PI}(\mu, \sigma) = -\frac{1}{\sigma^2 \cdot \sqrt{2\pi}} \cdot \exp\left(-\frac{(f_{\min} - \mu)^2}{2 \cdot \sigma^2}\right) \begin{bmatrix} \sigma \\ (f_{\min} - \mu) \end{bmatrix},$$

where $f_{\min}$ is a given target, we identify the following monotonicity properties:

- PI is monotonically decreasing with respect to μ.
- If $\mu < f_{\min}$ then PI is monotonically decreasing with respect to σ.

- If $\mu \geqslant f_{\min}$ then PI is monotonically increasing with respect to σ (recall that $\mathrm{PI}(f_{\min}, 0) = 0$, and $\mathrm{PI}(f_{\min}, \sigma) = 0.5 \ \forall \sigma \in]0, \infty[$).

These properties can be used to obtain exact interval bounds on the function values of PI. Furthermore, they can be exploited to construct relaxations as described in Section A.3.2.

A.3.1 Componentwise convexity

The Hessian of PI on $\mathbb{R} \times (0, \infty[$ is given by

$$\nabla^2 \mathrm{PI}(\mu, \sigma) = \begin{bmatrix} -\frac{f_{\min}-\mu}{\sigma^3} & -\frac{(f_{\min}-\mu)^2-\sigma^2}{\sigma^4} \\ -\frac{(f_{\min}-\mu)^2-\sigma^2}{\sigma^4} & \frac{(f_{\min}-\mu)\cdot\left(\left(2\sigma^2-f_{\min}-\mu\right)^2\right)}{\sigma^5} \end{bmatrix} \cdot \frac{1}{\sqrt{2\pi}} \cdot e^{-\frac{(f_{\min}-\mu)^2}{2\sigma^2}}.$$

The Hessian is indefinite and PI is therefore neither convex nor concave on its whole domain. However, we find *componentwise convexity* properties on certain parts of the domain, i.e., convexity with respect to one variable when the other is fixed. To this end, we divide the domain into the following four sets:

- $I_1 := \{(\mu, \sigma) \mid \mu \leqslant f_{\min} \ \wedge \ \mu - f_{\min} \geqslant -\sqrt{2}\sigma\}$,
- $I_2 := \{(\mu, \sigma) \mid \mu \geqslant f_{\min} \ \wedge \ \mu - f_{\min} \leqslant +\sqrt{2}\sigma\}$,
- $I_3 := \{(\mu, \sigma) \mid \mu \leqslant f_{\min} \ \wedge \ \mu - f_{\min} \leqslant -\sqrt{2}\sigma\}$,
- $I_4 := \{(\mu, \sigma) \mid \mu \geqslant f_{\min} \ \wedge \ \mu - f_{\min} \geqslant +\sqrt{2}\sigma\}$.

On these sets, PI has the componentwise convexity properties listed in Table A.1.

Table A.1.: componentwise convexity properties of PI over subsets of its domain

Subset	componentwise property	
I_1	concave w.r.t. μ	convex w.r.t. σ
I_2	convex w.r.t. μ	concave w.r.t. σ
I_3	concave w.r.t. μ	concave w.r.t. σ
I_4	convex w.r.t. μ	convex w.r.t. σ

A.3.2 Relaxations

We construct relaxations of PI over a given subset $\mathcal{X} = [\mu^{\mathrm{L}}, \mu^{\mathrm{U}}] \times [\sigma^{\mathrm{L}}, \sigma^{\mathrm{U}}]$ of its domain depending on which of the four sets I_1-I_4 contains the set $\mathcal{X}$. If $\mathcal{X} \subset I_1 \cup I_2$, we use the McCormick relaxations obtained by applying the multivariate composition theorem [170] to the composition of the rational function $\frac{f_{\min}-\mu}{\sigma}$ with Φ (c.f. Equation (3.5)), since these are already very tight. If $\mathcal{X}$ does not fully lie within $I_1 \cup I_2$, the McCormick relaxations get increasingly weaker and we thus resort to other methods as described in the following.

If $\mathcal{X} \subset I_4$, PI is componentwise convex with respect to both variables. Therefore, the concave envelope of PI over $\mathcal{X}$ consists of two planes anchored at the four corner points of $\mathcal{X}$ and can be calculated as described by [235]. A tight convex relaxation can be obtained using the method by [236]. Since the off-diagonal entries of the Hessian have a constant

sign over I_4, a sufficient condition for this method is fulfilled (c.f. Corollary 1 in [236]). An example for the resulting relaxation is shown in Figure 3.2b. Similarly, if $\mathcal{X} \subset I_3$, PI is componentwise concave and we obtain its convex envelope using the method by [235] and a tight concave relaxation using the method by [236].

If $\mathcal{X} \subset I_2 \cup I_4$, we construct relaxations exploiting the monotonicity properties of PI. Since for all $(\mu, \sigma) \in I_2 \cup I_4$ we have $\mu \geqslant f_{\min}$, PI is thus monotonically decreasing in μ and increasing in σ over $\mathcal{X}$. Therefore, we can construct a convex relaxation $\mathrm{PI}^{\mathrm{cv}}_{2,4} : \mathcal{X} \to [0,1]$ as

$$\mathrm{PI}^{\mathrm{cv}}_{2,4}(\mu, \sigma) \coloneqq \max\left(f^{\mathrm{cv}}_{\sigma^{\mathrm{L}}}(\mu), f^{\mathrm{cv}}_{\mu^{\mathrm{U}}}(\sigma)\right), \tag{A.8}$$

where $f^{\mathrm{cv}}_{\sigma^{\mathrm{L}}}$ and $f^{\mathrm{cv}}_{\mu^{\mathrm{U}}}$ are the convex envelopes of the univariate functions

$$f_{\sigma^{\mathrm{L}}} : [\mu^{\mathrm{L}}, \mu^{\mathrm{U}}] \to [0,1], \mu \mapsto \mathrm{PI}(\mu, \sigma^{\mathrm{L}})$$

and

$$f_{\mu^{\mathrm{U}}} : [\sigma^{\mathrm{L}}, \sigma^{\mathrm{U}}] \to [0,1], \sigma \mapsto \mathrm{PI}(\mu^{\mathrm{U}}, \sigma), \tag{A.9}$$

respectively, i.e., they correspond to the function PI restricted to one-dimensional facets of $\mathcal{X}$ at σ^{L} and μ^{U}. Both $f^{\mathrm{cv}}_{\sigma^{\mathrm{L}}}(\mu)$ and $f^{\mathrm{cv}}_{\mu^{\mathrm{U}}}(\sigma)$ are valid relaxations of PI because of the monotonicity of PI over $I_2 \cup I_4$. By taking the pointwise maximum in (A.8), we obtain a tighter relaxation while preserving convexity. To compute $f^{\mathrm{cv}}_{\sigma^{\mathrm{L}}}$ and $f^{\mathrm{cv}}_{\mu^{\mathrm{U}}}$, we can use the method described in Section 4 of [167] because they are one-dimensional functions with a known inflection point. To apply this method, we typically need to solve a one-dimensional nonlinear equation, which we do via Newton's method. A concave relaxation can be obtained analogously using concave envelopes of PI over one-dimensional facets of $\mathcal{X}$ at σ^{U} and μ^{L}. If $\mathcal{X} \subset I_1 \cup I_3$, an analogous method can be used since PI is monotonically increasing in both μ and σ.

In the most general case, $\mathcal{X}$ contains parts of all four sets I_1-I_4. In this case, we can still obtain relaxations by exploiting monotonicity properties. In particular, we compute a convex relaxation $\mathrm{PI}^{\mathrm{cv}}_{1-4} : \mathcal{X} \to [0,1]$ as

$$\mathrm{PI}^{\mathrm{cv}}_{1-4}(\mu, \sigma) \coloneqq \max\left(\tilde{f}^{\mathrm{cv}}_{\sigma^{\mathrm{L}}}(\mu), f^{\mathrm{cv}}_{\mu^{\mathrm{U}}}(\sigma)\right), \tag{A.10}$$

where $f^{\mathrm{cv}}_{\mu^{\mathrm{U}}}$ is again the convex relaxation of the univariate function $f_{\mu^{\mathrm{U}}}$ as in (A.9), which is still valid because PI is decreasing with respect to μ on its entire domain. In contrast, the convex relaxation of the univariate function at σ^{L} as in (A.8) is not valid because PI is not monotonic with respect to σ. Instead, in (A.10) it is replaced by the convex relaxation $\tilde{f}^{\mathrm{cv}}_{\sigma^{\mathrm{L}}}$ of the univariate function $\tilde{f}_{\sigma^{\mathrm{L}}} : [\mu^{\mathrm{L}}, \mu^{\mathrm{U}}] \to [0,1]$ with

$$\tilde{f}_{\sigma^{\mathrm{L}}}(\mu) \coloneqq \begin{cases} \mathrm{PI}(\mu, \sigma^{\mathrm{L}}), & \mu \geqslant f_{\min}, \\ \mathrm{PI}(f_{\min}, \sigma^{\mathrm{L}}) + \frac{\mathrm{PI}(f_{\min}, \sigma^{\mathrm{L}}) - \mathrm{PI}(\mu^{\mathrm{L}}, \sigma^{\mathrm{U}})}{f_{\min} - \mu^{\mathrm{L}}} (\mu - f_{\min}), & \text{otherwise.} \end{cases} \tag{A.11}$$

To see that $\tilde{f}^{\mathrm{cv}}_{\sigma^{\mathrm{L}}}$ is a valid relaxation of PI, we first note that by definition it is a relaxation of $\tilde{f}_{\sigma^{\mathrm{L}}}$, so it suffices to show that $\tilde{f}_{\sigma^{\mathrm{L}}}$ is in turn a relaxation of PI. The latter is established in the following Lemma.

Lemma A.1. *Let* PI *be defined as in* (3.5) *and* $\tilde{f}_{\sigma^{\mathrm{L}}}$ *as in* (A.11). *Then* $\tilde{f}_{\sigma^{\mathrm{L}}}(\mu) \leqslant \mathrm{PI}(\mu, \sigma)$ $\forall (\mu, \sigma) \in \mathcal{X} \coloneqq [\mu^{\mathrm{L}}, \mu^{\mathrm{U}}] \times [\sigma^{\mathrm{L}}, \sigma^{\mathrm{U}}]$.

Proof. Consider first any fixed $\hat{\mu}$ such that $\hat{\mu} \geqslant f_{\min}$. In this case, we have $\tilde{f}_{\sigma^{\mathrm{L}}}(\hat{\mu}) = \mathrm{PI}(\hat{\mu}, \sigma^{\mathrm{L}}) \leqslant \mathrm{PI}(\hat{\mu}, \sigma)\ \forall \sigma \in [\sigma^{\mathrm{L}}, \sigma^{\mathrm{U}}]$ because of the monotonicity w.r.t σ (c.f. Section A.3.0.1). Next, consider any $\tilde{\mu}$ such that $\tilde{\mu} < f_{\min}$. Note that this implies $\mu^{\mathrm{L}} < f_{\min}$. In this case, we have

$$\begin{aligned}
\tilde{f}_{\sigma^{\mathrm{L}}}(\tilde{\mu}) &= \mathrm{PI}(f_{\min}, \sigma^{\mathrm{L}}) + \frac{\mathrm{PI}(f_{\min}, \sigma^{\mathrm{L}}) - \mathrm{PI}(\mu^{\mathrm{L}}, \sigma^{\mathrm{U}})}{f_{\min} - \mu^{\mathrm{L}}} (\tilde{\mu} - f_{\min}) \\
&= \mathrm{PI}(f_{\min}, \sigma^{\mathrm{L}}) \frac{\tilde{\mu} - \mu^{\mathrm{L}}}{f_{\min} - \mu^{\mathrm{L}}} + \mathrm{PI}(\mu^{\mathrm{L}}, \sigma^{\mathrm{U}}) \frac{f_{\min} - \tilde{\mu}}{f_{\min} - \mu^{\mathrm{L}}} \\
&\leqslant \mathrm{PI}(f_{\min}, \sigma^{\mathrm{U}}) \frac{\tilde{\mu} - \mu^{\mathrm{L}}}{f_{\min} - \mu^{\mathrm{L}}} + \mathrm{PI}(\mu^{\mathrm{L}}, \sigma^{\mathrm{U}}) \frac{f_{\min} - \tilde{\mu}}{f_{\min} - \mu^{\mathrm{L}}} \\
&\leqslant \mathrm{PI}(\tilde{\mu}, \sigma^{\mathrm{U}}) \\
&\leqslant \mathrm{PI}(\tilde{\mu}, \sigma)\ \forall \sigma \in [\sigma^{\mathrm{L}}, \sigma^{\mathrm{U}}],
\end{aligned}$$

where the inequalities follow, in this order, from the monotonicity of PI with respect to σ for $\mu \geqslant f_{\min}$, its componentwise concavity with respect to μ for $\mu < f_{\min}$, and its monotonicity with respect to σ for $\mu < f_{\min}$. □

A.4 Expected improvement acquisition function

We now show that the EI acquisition function is convex. From the Hessian matrix of EI on $\mathbb{R} \times (0, \infty)$

$$\nabla^2 \mathrm{EI}(\mu, \sigma) = \begin{bmatrix} \frac{1}{\sigma} & -\frac{\mu - f_{\min}}{\sigma^2} \\ -\frac{\mu - f_{\min}}{\sigma^2} & \frac{(\mu - f_{\min})^2}{\sigma^3} \end{bmatrix} \cdot \phi\left(-\frac{\mu - f_{\min}}{\sigma}\right),$$

we find the eigenvalues 0 and $\frac{(\mu - f_{\min})^2 + \sigma^2}{\sigma^3} \cdot \phi\left(-\frac{\mu - f_{\min}}{\sigma}\right)$. As $\sigma \geqslant 0$, $\mathrm{EI}(\cdot, \cdot)$ is convex and the envelopes can be constructed directly.

Bibliography

[1] A. M. Schweidtmann and A. Mitsos, "Deterministic global optimization with artificial neural networks embedded," *Journal of Optimization Theory and Applications*, vol. 180, no. 3, pp. 925–948, 2019.

[2] A. M. Schweidtmann, D. Bongartz, D. Grothe, T. Kerkenhoff, X. Lin, J. Najman, and A. Mitsos, "Deterministic global optimization with Gaussian processes embedded," *In Press: Mathematical Programming Computation*, 2021.

[3] A. M. Schweidtmann, J. G. Rittig, A. König, M. Grohe, A. Mitsos, and M. Dahmen, "Graph neural networks for prediction of fuel ignition quality," *Energy & Fuels*, vol. 34, no. 9, pp. 11395–11407, 2020.

[4] A. M. Schweidtmann, J. M. Weber, C. Wende, L. Netze, and A. Mitsos, "Obey validity limits of data-driven models through topological data analysis and one-class classification," *In Press: Optimization and Engineering*, 2021.

[5] A. M. Schweidtmann, D. Bongartz, W. R. Huster, J. Najman, D. Rall, P. Schäfer, M. Wessling, and A. Mitsos, "Global optimization with neural networks embdedded: theory, applications, and future work," *9th International Conference on Foundations of Computer-Aided Process Design*, 2019.

[6] A. M. Schweidtmann, E. Esche, A. Fischer, M. Kloft, J.-U. Repke, S. Sager, and A. Mitsos, "Machine learning in chemical engineering: A perspective," *Submitted: Chemie Ingenieur Technik*, 2021.

[7] A. Mitsos, A. Fischer, M. Kloft, J.-U. Repke, and S. Sager, "Priority programme machine learning in chemical engineering. knowledge meets data: Interpretability, extrapolation, reliability, trust (SPP 2331)." `https://www.dfg.de/foerderung/info_wissenschaft/info_wissenschaft_20_42/index.html`, 2020.

[8] A. M. Schweidtmann, J. G. Rittig, J. M. Weber, M. Grohe, M. Dahmen, K. Leonhard, and A. Mitsos, "Physical graph neural networks." in preparation, 2020.

[9] A. M. Schweidtmann, A. D. Clayton, N. Holmes, E. Bradford, R. A. Bourne, and A. A. Lapkin, "Machine learning meets continuous flow chemistry: Automated optimization towards the Pareto front of multiple objectives," *Chemical Engineering Journal*, vol. 352, pp. 277–282, 2018.

[10] A. M. Schweidtmann, D. Bongartz, W. R. Huster, and A. Mitsos, "Deterministic global process optimization: Flash calculations via artificial neural networks," in *Computer Aided Chemical Engineering*, vol. 46, pp. 937–942, Elsevier, 2019.

[11] A. M. Schweidtmann, W. R. Huster, J. T. Lüthje, and A. Mitsos, "Deterministic global process optimization: Accurate (single-species) properties via artificial neural networks," *Computers & Chemical Engineering*, vol. 121, pp. 67–74, 2019.

[12] W. R. Huster, A. M. Schweidtmann, and A. Mitsos, "Working fluid selection for organic rankine cycles via deterministic global optimization of design and operation," *Optimization and Engineering*, pp. 1–20, 2019.

[13] D. Rall, A. M. Schweidtmann, B. M. Aumeier, J. Kamp, J. Karwe, K. Ostendorf, A. Mitsos, and M. Wessling, "Simultaneous rational design of ion separation membranes and processes," *Journal of Membrane Science*, vol. 600, p. 117860, 2020.

[14] D. Rall, A. M. Schweidtmann, M. Kruse, E. Evdochenko, A. Mitsos, and M. Wessling, "Multi-scale membrane process optimization with high-fidelity ion transport models through machine learning," *Journal of Membrane Science*, p. 118208, 2020.

[15] U. Lee, J. Burre, A. Caspari, J. Kleinekorte, A. M. Schweidtmann, and A. Mitsos, "Techno-economic optimization of a green-field post-combustion CO2 capture process using superstructure and rate-based models," *Industrial & Engineering Chemistry Research*, vol. 55, no. 46, pp. 12014–12026, 2016.

[16] D. Helmdach, P. Yaseneva, P. K. Heer, A. M. Schweidtmann, and A. A. Lapkin, "A multiobjective optimization including results of life cycle assessment in developing biorenewables-based processes," *ChemSusChem*, vol. 10, no. 18, pp. 3632–3643, 2017.

[17] S. M. Aworinde, A. M. Schweidtmann, and A. A. Lapkin, "The concept of selectivity control by simultaneous distribution of the oxygen feed and wall temperature in a microstructured reactor," *Chemical Engineering Journal*, vol. 331, pp. 765–776, 2018.

[18] E. Bradford, A. M. Schweidtmann, and A. Lapkin, "Efficient multiobjective optimization employing Gaussian processes, spectral sampling and a genetic algorithm," *Journal of Global Optimization*, vol. 71, no. 2, pp. 407–438, 2018.

[19] E. Bradford, A. M. Schweidtmann, A. Lapkin, *et al.*, "Correction to: Efficient multiobjective optimization employing Gaussian processes, spectral sampling and a genetic algorithm," *Journal of Global Optimization*, vol. 71, no. 2, pp. 439–440, 2018.

[20] E. Bradford, A. M. Schweidtmann, D. Zhang, K. Jing, and E. A. del Rio-Chanona, "Dynamic modeling and optimization of sustainable algal production with uncertainty using multivariate gaussian processes," *Computers & Chemical Engineering*, vol. 118, pp. 143–158, 2018.

[21] D. Rall, D. Menne, A. M. Schweidtmann, J. Kamp, L. von Kolzenberg, A. Mitsos, and M. Wessling, "Rational design of ion separation membranes," *Journal of Membrane Science*, vol. 569, pp. 209–219, 2019.

[22] P. Schäfer, H. G. Westerholt, A. M. Schweidtmann, S. Ilieva, and A. Mitsos, "Model-based bidding strategies on the primary balancing market for energy-intense processes," *Computers & Chemical Engineering*, vol. 120, pp. 4–14, 2019.

[23] Y. Amar, A. M. Schweidtmann, P. Deutsch, L. Cao, and A. Lapkin, "Machine learning and molecular descriptors enable rational solvent selection in asymmetric catalysis," *Chemical science*, vol. 10, no. 27, pp. 6697–6706, 2019.

[24] A. D. Clayton, A. M. Schweidtmann, G. Clemens, J. A. Manson, C. J. Taylor, C. G. Niño, T. W. Chamberlain, N. Kapur, A. J. Blacker, A. A. Lapkin, *et al.*, "Automated self-optimisation of multi-step reaction and separation processes using machine learning," *Chemical Engineering Journal*, vol. 384, p. 123340, 2020.

[25] P. Schäfer, A. M. Schweidtmann, P. H. Lenz, H. M. Markgraf, and A. Mitsos, "Wavelet-based grid-adaptation for nonlinear scheduling subject to time-variable electricity prices," *Computers & Chemical Engineering*, vol. 132, p. 106598, 2020.

[26] W. R. Huster, A. M. Schweidtmann, J. T. Lüthje, and A. Mitsos, "Deterministic global superstructure-based optimization of an organic rankine cycle," *Computers & Chemical Engineering*, vol. 141, p. 106996, 2020.

[27] P. Schäfer, A. M. Schweidtmann, and A. Mitsos, "Nonlinear scheduling with time-variable electricity prices using sensitivity-based truncations of wavelet transforms," *AIChE Journal*, vol. 66, no. 10, p. e16986, 2020.

[28] W. R. Huster, A. M. Schweidtmann, and A. Mitsos, "Globally optimal working fluid mixture composition for geothermal power cycles," *Energy*, vol. 212, p. 118731, 2020.

[29] P. Schäfer, A. Caspari, A. M. Schweidtmann, Y. Vaupel, A. Mhamdi, and A. Mitsos, "The potential of hybrid mechanistic/data-driven approaches for reduced dynamic modeling: Application to distillation columns," *Chemie Ingenieur Technik*, vol. 92, no. 12, pp. 1910–1920, 2020.

[30] W. R. Huster, A. M. Schweidtmann, and A. Mitsos, "Impact of accurate working fluid properties on the globally optimal design of an organic Rankine cycle," in *Proceedings of the 9th International Conference on Foundations of Computer-Aided Process Design*, vol. 47 of *Computer Aided Chemical Engineering*, Elsevier, 2019.

[31] W. R. Huster, A. M. Schweidtmann, and A. Mitsos, "Hybrid mechanistic data-driven modeling for the deterministic global optimization of a transcritical organic rankine cycle," in *Computer Aided Chemical Engineering*, vol. 48, pp. 1765–1770, Elsevier, 2020.

[32] J. M. Weber, A. M. Schweidtmann, E. Nolasco, and A. A. Lapkin, "Modelling circular structures in reaction networks: Petri nets and reaction network flux analysis," in *Computer Aided Chemical Engineering*, vol. 48, pp. 1843–1848, Elsevier, 2020.

[33] D. T. Doncevic, A. M. Schweidtmann, Y. Vaupel, P. Schäfer, A. Caspari, and A. Mitsos, "Deterministic global nonlinear model predictive control with neural networks embedded," *IFAC-PapersOnLine*, vol. 53, no. 2, pp. 5273–5278, 2020. 21th IFAC World Congress.

[34] A. Kätelhön, R. Meys, S. Deutz, S. Suh, and A. Bardow, "Climate change mitigation potential of carbon capture and utilization in the chemical industry," *Proceedings of the National Academy of Sciences*, vol. 116, no. 23, pp. 11187–11194, 2019.

[35] A. A. Lapkin, "Chemical engineering science and green chemistry–the challenge of sustainability," *Handbook of Green Chemistry*, pp. 1–16, 2018.

[36] A. Mitsos, N. Asprion, C. A. Floudas, M. Bortz, M. Baldea, D. Bonvin, A. Caspari, and P. Schäfer, "Challenges in process optimization for new feedstocks and energy sources," *Computers & Chemical Engineering*, vol. 113, pp. 209–221, 2018.

[37] J. Artz, T. E. Müller, K. Thenert, J. Kleinekorte, R. Meys, A. Sternberg, A. Bardow, and W. Leitner, "Sustainable conversion of carbon dioxide: an integrated review of catalysis and life cycle assessment," *Chemical reviews*, vol. 118, no. 2, pp. 434–504, 2017.

[38] V. Venkatasubramanian, "The promise of artificial intelligence in chemical engineering: Is it here, finally," *AIChE J*, vol. 65, no. 2, pp. 466–78, 2019.

[39] J. H. Lee, J. Shin, and M. J. Realff, "Machine learning: Overview of the recent progresses and implications for the process systems engineering field," *Computers & Chemical Engineering*, vol. 114, pp. 111–121, 2018.

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

[41] S. J. Qin, "Process data analytics in the era of big data," *AIChE Journal*, vol. 60, no. 9, pp. 3092–3100, 2014.

[42] C. M. Bishop, *Pattern recognition and machine learning*. Information science and statistics, New York, NY: Springer, 8 ed., 2009.

[43] R. Banares-Alcantara, A. Westerberg, and M. Rychener, "Development of an expert system for physical property predictions," *Computers & Chemical Engineering*, vol. 9, no. 2, pp. 127–142, 1985.

[44] R. Banares-Alcantara, E. I. Ko, A. W. Westerberg, and M. D. Rychener, "DECADE – a hybrid expert system for catalyst selection – II. Final architecture and results," *Computers & Chemical Engineering*, vol. 12, no. 9, pp. 923–938, 1988.

[45] Z. Ge, Z. Song, S. X. Ding, and B. Huang, "Data mining and analytics in the process industry: The role of machine learning," *IEEE Access*, vol. 5, 2017.

[46] B. D. Ketelaere, M. Hubert, and E. Schmitt, "Overview of pca-based statistical process-monitoring methods for time-dependent, high-dimensional data," *Journal of Quality Technology*, vol. 47, no. 4, pp. 318–335, 2015.

[47] C.-C. Hsu, M.-C. Chen, and L.-S. Chen, "A novel process monitoring approach with dynamic independent component analysis," *Control Engineering Practice*, vol. 18, no. 3, pp. 242 – 253, 2010.

[48] M. He, C. Yang, X. Wang, W. Gui, and L. Wei, "Nonparametric density estimation of froth colour texture distribution for monitoring sulphur flotation process," *Minerals Engineering*, vol. 53, pp. 203 – 212, 2013.

[49] M. Liukkonen, I. Laakso, and Y. Hiltunen, "Advanced monitoring platform for industrial wastewater treatment: Multivariable approach using the self-organizing map," *Environmental Modelling & Software*, vol. 48, pp. 193 – 201, 2013.

[50] J. Yu, "A particle filter driven dynamic Gaussian mixture model approach for complex process monitoring and fault diagnosis," *Journal of Process Control*, vol. 22, no. 4, pp. 778 – 788, 2012.

[51] A. Prieto-Moreno, O. Llanes-Santiago, and E. GarcÃa-Moreno, "Principal components selection for dimensionality reduction using discriminant information applied to fault diagnosis," *Journal of Process Control*, vol. 33, pp. 14 – 24, 2015.

[52] M. A. Pimentel, D. A. Clifton, L. Clifton, and L. Tarassenko, "A review of novelty detection," *Signal Processing*, vol. 99, pp. 215–249, 2014.

[53] L. H. Chiang, R. J. Pell, and M. B. Seasholtz, "Exploring process data with the use of robust outlier detection algorithms," *Journal of Process Control*, vol. 13, no. 5, pp. 437 – 449, 2003.

[54] M. Yao, H. Wang, and W. Xu, "Batch process monitoring based on functional data analysis and support vector data description," *Journal of Process Control*, vol. 24, no. 7, pp. 1085 – 1097, 2014.

[55] Z. Lv, X. Yan, and Q. Jiang, "Batch process monitoring based on just-in-time learning and multiple-subspace principal component analysis," *Chemometrics and Intelligent Laboratory Systems*, vol. 137, pp. 128 – 139, 2014.

[56] P. S. Corporation, "Data RPM – Anomaly Detection and Prediction." `https://www.progress.com/datarpm/how-it-works`, July 2019.

[57] T. Kellner, "Deep machine learning: Ge and bp will connect thousands of subsea oil wells to the industrial internet," tech. rep., General Electric, July 2015.

[58] J. V. Kresta, T. E. Marlin, and J. F. MacGregor, "Development of inferential process models using PLS," *Computers & Chemical Engineering*, vol. 18, no. 7, pp. 597 – 611, 1994. An International Journal of Computer Applications in Chemical Engineering.

[59] J. Liu, "Developing a soft sensor based on sparse partial least squares with variable selection," *Journal of Process Control*, vol. 24, no. 7, pp. 1046 – 1056, 2014.

[60] S. S. Kolluri, I. J. Esfahani, P. S. N. Garikiparthy, and C. Yoo, "Evaluation of multivariate statistical analyses for monitoring and prediction of processes in an seawater reverse osmosis desalination plant," *Korean Journal of Chemical Engineering*, vol. 32, pp. 1486–1497, Aug 2015.

[61] Y. Yang and F. Gao, "Injection molding product weight: online prediction and control based on a nonlinear principal component regression model," *Polymer Engineering & Science*, vol. 46, no. 4, pp. 540–548, 2006.

[62] Z. Ge, F. Gao, and Z. Song, "Mixture probabilistic PCR model for soft sensing of multimode processes," *Chemometrics and Intelligent Laboratory Systems*, vol. 105, no. 1, pp. 91 – 105, 2011.

[63] P. Jain, I. Rahman, and B. D. Kulkarni, "Development of a soft sensor for a batch distillation column using support vector regression techniques," *Chemical Engineering Research and Design*, vol. 85, no. 2, pp. 283 – 287, 2007.

[64] J. C. B. Gonzaga, L. A. C. Meleiro, C. Kiang, and R. Maciel-Filho, "ANN-based soft-sensor for real-time process monitoring and control of an industrial polymerization process," *Computers & Chemical Engineering*, vol. 33, no. 1, pp. 43 – 49, 2009.

[65] Z. Ge, T. Chen, and Z. Song, "Quality prediction for polypropylene production process based on CLGPR model," *Control Engineering Practice*, vol. 19, no. 5, pp. 423–432, 2011.

[66] C. E. Rasmussen, "Gaussian processes in machine learning," in *Advanced lectures on machine learning*, pp. 63–71, Springer, 2004.

[67] C. K. Williams and C. E. Rasmussen, "Gaussian processes for regression," in *Advances in neural information processing systems*, pp. 514–520, 1996.

[68] H. Kaneko, M. Arakawa, and K. Funatsu, "Novel soft sensor method for detecting completion of transition in industrial polymer processes," *Computers & Chemical Engineering*, vol. 35, no. 6, pp. 1135 – 1142, 2011.

[69] K. McBride and K. Sundmacher, "Overview of surrogate modeling in chemical process engineering," *Chemie Ingenieur Technik*, vol. 91, no. 3, pp. 228–239, 2019.

[70] S. A. Papoulias and I. E. Grossmann, "A structural optimization approach in process synthesis—i: Utility systems," *Computers & Chemical Engineering*, vol. 7, no. 6, pp. 695–706, 1983.

[71] K. Hornik, M. Stinchcombe, and H. White, "Multilayer feedforward networks are universal approximators," *Neural Networks*, vol. 2, no. 5, pp. 359–366, 1989.

[72] J. D. Smith, A. A. Neto, S. Cremaschi, and D. W. Crunkleton, "CFD-based optimization of a flooded bed algae bioreactor," *Industrial & Engineering Chemistry Research*, vol. 52, no. 22, pp. 7181–7188, 2013.

[73] O. Kahrs and W. Marquardt, "Incremental identification of hybrid process models," *Computers & Chemical Engineering*, vol. 32, no. 4-5, pp. 694–705, 2008.

[74] I. Fahmi and S. Cremaschi, "Process synthesis of biodiesel production plant using artificial neural networks as the surrogate models," *Computers & Chemical Engineering*, vol. 46, pp. 105–123, 2012.

[75] C. A. Henao and C. T. Maravelias, "Surrogate-based superstructure optimization framework," *AIChE Journal*, vol. 57, no. 5, pp. 1216–1232, 2011.

[76] C. Nentwich and S. Engell, "Application of surrogate models for the optimization and design of chemical processes," in *2016 International Joint Conference on Neural Networks (IJCNN)*, pp. 1291–1296, IEEE, 2016.

[77] F. Boukouvala, M. F. Hasan, and C. A. Floudas, "Global optimization of general constrained grey-box models: new method and its application to constrained pdes for pressure swing adsorption," *Journal of Global Optimization*, vol. 67, no. 1-2, pp. 3–42, 2017.

[78] T. Keßler, C. Kunde, K. McBride, N. Mertens, D. Michaels, K. Sundmacher, and A. Kienle, "Global optimization of distillation columns using explicit and implicit surrogate models," *Chemical Engineering Science*, vol. 197, pp. 235–245, 2019.

[79] Q. Zhang, I. E. Grossmann, A. Sundaramoorthy, and J. M. Pinto, "Data-driven construction of convex region surrogate models," *Optimization and Engineering*, vol. 17, no. 2, pp. 289–332, 2016.

[80] Z. T. Wilson and N. V. Sahinidis, "The alamo approach to machine learning," *Computers & Chemical Engineering*, vol. 106, pp. 785–795, 2017.

[81] M. Mistry, D. Letsios, G. Krennrich, R. M. Lee, and R. Misener, "Mixed-integer convex nonlinear optimization with gradient-boosted trees embedded," *arXiv preprint arXiv:1803.00952*, 2018.

[82] B. Grimstad and B. R. Knudsen, "Mathematical programming formulations for piecewise polynomial functions," *Journal of Global Optimization*, pp. 1–32, 2020.

[83] B. Grimstad and A. Sandnes, "Global optimization with spline constraints: a new branch-and-bound method based on b-splines," *Journal of Global Optimization*, vol. 65, no. 3, pp. 401–439, 2016.

[84] B. Grimstad and H. Andersson, "Relu networks as surrogate models in mixed-integer linear programs," *Computers & Chemical Engineering*, vol. 131, p. 106580, 2019.

[85] R. Anderson, J. Huchette, W. Ma, C. Tjandraatmadja, and J. P. Vielma, "Strong mixed-integer programming formulations for trained neural networks," *Mathematical Programming*, pp. 1–37, 2020.

[86] J. Katz, I. Pappas, S. Avraamidou, and E. N. Pistikopoulos, "Integrating deep learning models and multiparametric programming," *Computers & Chemical Engineering*, p. 106801, 2020.

[87] A. Thebelt, J. Kronqvist, R. M. Lee, N. Sudermann-Merx, and R. Misener, "Global optimization with ensemble machine learning models," in *Computer Aided Chemical Engineering*, vol. 48, pp. 1981–1986, Elsevier, 2020.

[88] A. Thebelt, J. Kronqvist, M. Mistry, R. M. Lee, N. Sudermann-Merx, and R. Misener, "Entmoot: A framework for optimization over ensemble tree models," *arXiv preprint arXiv:2003.04774*, 2020.

[89] D. C. Psichogios and L. H. Ungar, "A hybrid neural network-first principles approach to process modeling," *AIChE Journal*, vol. 38, no. 10, pp. 1499–1511, 1992.

[90] A. A. Schuppert, "Extrapolability of structured hybrid models: a key to optimization of complex processes," in *Equadiff 99: (In 2 Volumes)*, pp. 1135–1151, World Scientific, 2000.

[91] M. Von Stosch, R. Oliveira, J. Peres, and S. F. de Azevedo, "Hybrid semi-parametric modeling in process systems engineering: Past, present and future," *Computers & Chemical Engineering*, vol. 60, pp. 86–101, 2014.

[92] H. Te Braake, H. Van Can, and H. Verbruggen, "Semi-mechanistic modeling of chemical processes with neural networks," *Engineering Applications of Artificial Intelligence*, vol. 11, no. 4, pp. 507–515, 1998.

[93] M. Dors, R. Simutis, and A. Lübbert, *Hybrid process modeling for advanced process state estimation, prediction, and control exemplified in a production-scale mammalian cell culture*, ch. 14, pp. 144–154. ACS Publications, 1995.

[94] G. Mogk, T. Mrziglod, and A. Schuppert, "Application of hybrid models in chemical industry," in *Computer Aided Chemical Engineering*, vol. 10, pp. 931–936, Elsevier, 2002.

[95] M. von Stosch, S. Davy, K. Francois, V. Galvanauskas, J.-M. Hamelink, A. Luebbert, M. Mayer, R. Oliveira, R. O'Kennedy, P. Rice, *et al.*, "Hybrid modeling for quality by design and PAT-benefits and challenges of applications in biopharmaceutical industry," *Biotechnology journal*, vol. 9, no. 6, pp. 719–726, 2014.

[96] P. Schäfer, A. Caspari, K. Kleinhans, A. Mhamdi, and A. Mitsos, "Reduced dynamic modeling approach for rectification columns based on compartmentalization and artificial neural networks," *AIChE Journal*, vol. 65, no. 5, 2019.

[97] B. Fiedler and A. Schuppert, "Local identification of scalar hybrid models with tree structure," *IMA Journal of Applied Mathematics*, vol. 73, no. 3, pp. 449–476, 2008.

[98] O. Kahrs and W. Marquardt, "The validity domain of hybrid models and its application in process optimization," *Chemical Engineering and Processing: Process Intensification*, vol. 46, no. 11, pp. 1054–1066, 2007.

[99] M. Raissi, P. Perdikaris, and G. E. Karniadakis, "Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations," *Journal of Computational Physics*, vol. 378, pp. 686–707, 2019.

[100] A. K. Maier, C. Syben, B. Stimpel, T. Würfl, M. Hoffmann, F. Schebesch, W. Fu, L. Mill, L. Kling, and S. Christiansen, "Learning with known operators reduces maximum error bounds," *Nature machine intelligence*, vol. 1, no. 8, pp. 373–380, 2019.

[101] M. Schmidt and H. Lipson, "Distilling free-form natural laws from experimental data," *Science*, vol. 324, no. 5923, pp. 81–85, 2009.

[102] A. Cozad and N. V. Sahinidis, "A global MINLP approach to symbolic regression," *Mathematical Programming*, vol. 170, no. 1, pp. 97–119, 2018.

[103] P. Neumann, L. Cao, D. Russo, V. S. Vassiliadis, and A. A. Lapkin, "A new formulation for symbolic regression to identify physico-chemical laws from experimental data," *Chemical Engineering Journal*, vol. 387, p. 123412, 2020.

[104] L. Constantinou and R. Gani, "New group contribution method for estimating properties of pure compounds," *AIChE Journal*, vol. 40, no. 10, pp. 1697–1710, 1994.

[105] A. Fredenslund, *Vapor-liquid equilibria using UNIFAC: a group-contribution method*. Elsevier, 2012.

[106] M. Bojarski, D. Del Testa, D. Dworakowski, B. Firner, B. Flepp, P. Goyal, L. D. Jackel, M. Monfort, U. Muller, J. Zhang, *et al.*, "End to end learning for self-driving cars," *arXiv preprint arXiv:1604.07316*, 2016.

[107] D. Amodei, S. Ananthanarayanan, R. Anubhai, J. Bai, E. Battenberg, C. Case, J. Casper, B. Catanzaro, Q. Cheng, G. Chen, *et al.*, "Deep speech 2: End-to-end speech recognition in english and mandarin," in *International conference on machine learning*, pp. 173–182, 2016.

[108] T. Gärtner, P. Flach, and S. Wrobel, "On graph kernels: Hardness results and efficient alternatives," in *Learning Theory and Kernel Machines* (B. Schölkopf and M. K. Warmuth, eds.), (Berlin, Heidelberg), pp. 129–143, Springer Berlin Heidelberg, 2003.

[109] D. Oglic, S. A. Oatley, S. J. F. Macdonald, T. Mcinally, R. Garnett, J. D. Hirst, and T. Gärtner, "Active search for computer-aided drug design," *Molecular Informatics*, vol. 37, no. 1-2, p. 1700130, 2018.

[110] D. K. Duvenaud, D. Maclaurin, J. Iparraguirre, R. Bombarell, T. Hirzel, A. Aspuru-Guzik, and R. P. Adams, "Convolutional networks on graphs for learning molecular fingerprints," in *Advances in neural information processing systems*, pp. 2224–2232, 2015.

[111] C. W. Coley, R. Barzilay, W. H. Green, T. S. Jaakkola, and K. F. Jensen, "Convolutional embedding of attributed molecular graphs for physical property prediction," *Journal of chemical information and modeling*, vol. 57, no. 8, pp. 1757–1772, 2017.

[112] T. Xie and J. C. Grossman, "Crystal graph convolutional neural networks for an accurate and interpretable prediction of material properties," *Physical review letters*, vol. 120, no. 14, p. 145301, 2018.

[113] C. Morris, M. Ritzert, M. Fey, W. Hamilton, J. Lenssen, G. Rattan, and M. Grohe, "Weisfeiler and leman go neural: Higher-order graph neural networks," in *Proceedings of the 33rd AAAI Conference on Artificial Intelligence, 27.01.-01.02.2019, Honolulu, Hawaii, United States*, vol. 4602-4609, AAAI Press, 2019.

[114] J. M. Weber, P. Lió, and A. A. Lapkin, "Identification of strategic molecules for future circular supply chains using large reaction networks," *Reaction Chemistry & Engineering*, vol. 4, no. 11, pp. 1969–1981, 2019.

[115] P.-M. Jacob and A. Lapkin, "Statistics of the network of organic chemistry," *Reaction Chemistry & Engineering*, vol. 3, no. 1, pp. 102–118, 2018.

[116] T. Zhang, N. V. Sahinidis, and J. J. Siirola, "Pattern recognition in chemical process flowsheets," *AIChE Journal*, vol. 65, no. 2, pp. 592–603, 2019.

[117] P. S. Gromski, J. M. Granda, and L. Cronin, "Universal chemical synthesis and discovery with 'the chemputer'," *Trends in Chemistry*, 2019.

[118] K. Ulonska, M. Skiborowski, A. Mitsos, and J. Viell, "Early-stage evaluation of biorefinery processing pathways using process network flux analysis," *AIChE Journal*, vol. 62, no. 9, pp. 3096–3108, 2016.

[119] K. Ulonska, A. König, M. Klatt, A. Mitsos, and J. Viell, "Optimization of multiproduct biorefinery processes under consideration of biomass supply chain management and market developments," *Industrial & Engineering Chemistry Research*, vol. 57, no. 20, pp. 6980–6991, 2018.

[120] A. Koönig, K. Ulonska, A. Mitsos, and J. Viell, "Optimal applications and combinations of renewable fuel production from biomass and electricity," *Energy & fuels*, vol. 33, no. 2, pp. 1659–1672, 2019.

[121] X. Zhu and A. B. Goldberg, *Introduction to semi-supervised learning*. Morgen & Claypool, 2009.

[122] J. Ji, H. Wang, K. Chen, Y. Liu, N. Zhang, and J. Yan, "Recursive weighted kernel regression for semi-supervised soft-sensing modeling of fed-batch processes," *Journal of the Taiwan Institute of Chemical Engineers*, vol. 43, no. 1, pp. 67 – 76, 2012.

[123] S. Xu, B. Lu, M. Baldea, T. F. Edgar, W. Wojsznis, T. Blevins, and M. Nixon, "Data cleaning in the process industries," *Reviews in Chemical Engineering*, vol. 31, no. 5, pp. 453–490, 2015.

[124] P. Petsagkourakis, I. O. Sandoval, E. Bradford, D. Zhang, and E. A. del Rio-Chanona, "Reinforcement learning for batch bioprocess optimization," *Computers & Chemical Engineering*, vol. 133, p. 106649, 2020.

[125] J. H. Lee and W. Wong, "Approximate dynamic programming approach for process control," *Journal of Process Control*, vol. 20, no. 9, pp. 1038–1048, 2010.

[126] L. A. Gatys, A. S. Ecker, and M. Bethge, "A neural algorithm of artistic style," *arXiv preprint arXiv:1508.06576*, 2015.

[127] A. Bardow, K. Steur, and J. Gross, "Continuous-molecular targeting for integrated solvent and process design," *Industrial & Engineering Chemistry Research*, vol. 49, no. 6, pp. 2834–2840, 2010.

[128] E. J. Candès and B. Recht, "Exact matrix completion via convex optimization," *CoRR*, vol. abs/0805.4471, 2008.

[129] F. Jirasek, R. A. Alves, J. Damay, R. A. Vandermeulen, R. Bamler, M. Bortz, S. Mandt, M. Kloft, and H. Hasse, "Machine learning in thermodynamics: Prediction of activity coefficients by matrix completion," *The Journal of Physical Chemistry Letters*, vol. 11, no. 3, pp. 981–985, 2020.

[130] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," in *Advances in neural information processing systems*, pp. 2672–2680, 2014.

[131] D. P. Kingma and M. Welling, "An introduction to variational autoencoders," *CoRR*, vol. abs/1906.02691, 2019.

[132] A. M. Schweidtmann, L. Netze, and A. Mitsos, "Melon: Machine learning models for optimization." `https://git.rwth-aachen.de/avt.svt/public/MeLOn/`, 2020.

[133] J. Gasteiger and J. Zupan, "Neural networks in chemistry," *Angewandte Chemie International Edition in English*, vol. 32, no. 4, pp. 503–527, 1993.

[134] M. Azlan Hussain, "Review of the applications of neural networks in chemical process control — simulation and online implementation," *Artificial Intelligence in Engineering*, vol. 13, no. 1, pp. 55–68, 1999.

[135] S. Agatonovic-Kustrin and R. Beresford, "Basic concepts of artificial neural network modeling and its application in pharmaceutical research," *Journal of Pharmaceutical and Biomedical Analysis*, vol. 22, no. 5, pp. 717–727, 2000.

[136] A. Witek-Krowiak, K. Chojnacka, D. Podstawczyk, A. Dawiec, and K. Pokomeda, "Application of response surface methodology and artificial neural network methods in modelling and optimization of biosorption process," *Bioresource technology*, vol. 160, pp. 150–160, 2014.

[137] M. Meireles, P. Almeida, and M. G. Simoes, "A comprehensive review for industrial applicability of artificial neural networks," *IEEE Transactions on Industrial Electronics*, vol. 50, no. 3, pp. 585–601, 2003.

[138] E. A. Del Rio-Chanona, F. Fiorelli, D. Zhang, N. R. Ahmed, K. Jing, and N. Shah, "An efficient model construction strategy to simulate microalgal lutein photoproduction dynamic process," *Biotechnology and Bioengineering*, vol. 114, no. 11, pp. 2518–2527, 2017.

[139] J. J. S. Cheema, N. V. Sankpal, S. S. Tambe, and B. D. Kulkarni, "Genetic programming assisted stochastic optimization strategies for optimization of glucose to gluconic acid fermentation," *Biotechnology progress*, vol. 18, no. 6, pp. 1356–1365, 2002.

[140] K. M. Desai, S. A. Survase, P. S. Saudagar, S. S. Lele, and R. S. Singhal, "Comparison of artificial neural network and response surface methodology in fermentation media optimization: Case study of fermentative production of scleroglucan," *Biochemical Engineering Journal*, vol. 41, no. 3, pp. 266–273, 2008.

[141] Y. Nagata and K. H. Chu, "Optimization of a fermentation medium using neural networks and genetic algorithms," *Biotechnology Letters*, vol. 25, no. 21, pp. 1837–1842, 2003.

[142] C. A. O. Nascimento, R. Giudici, and R. Guardani, "Neural network based approach for optimization of industrial chemical processes," *Computers & Chemical Engineering*, vol. 24, no. 9-10, pp. 2303–2314, 2000.

[143] C. A. O. Nascimento and R. Giudici, "Neural network based approach for optimisation applied to an industrial nylon-6,6 polymerisation process," *Computers & Chemical Engineering*, vol. 22, pp. 595–S600, 1998.

[144] M. Chambers and C. A. Mount-Campbell, "Process optimization via neural network metamodeling," *International Journal of Production Economics*, vol. 79, no. 2, pp. 93–100, 2002.

[145] C. A. Henao and C. T. Maravelias, "Surrogate-based process synthesis," in *20th European Symposium on Computer Aided Process Engineering* (S. Pierucci and G. B. Ferraris, eds.), vol. 28 of *Computer Aided Chemical Engineering*, pp. 1129–1134, Milano, Italy: Elsevier, 2010.

[146] H. R. Sant Anna, A. G. Barreto, F. W. Tavares, and M. B. de Souza, "Machine learning model and optimization of a psa unit for methane-nitrogen separation," *Computers & Chemical Engineering*, vol. 104, pp. 377–391, 2017.

[147] C. A. Henao, *A superstructure modeling framework for process synthesis using surrogate models*. Dissertation, University of Wisconsin, Madison, 2012.

[148] O. T. Kajero, T. Chen, Y. Yao, Y.-C. Chuang, and D. S. H. Wong, "Meta-modelling in chemical process system engineering," *Journal of the Taiwan Institute of Chemical Engineers*, vol. 73, pp. 135–145, 2017.

[149] J. Lewandowski, N. O. Lemcoff, and S. Palosaari, "Use of neural networks in the simulation and optimization of pressure swing adsorption processes," *Chemical Engineering & Technology*, vol. 21, no. 7, pp. 593–597, 1998.

[150] C. Gutiérrez-Antonio, "Multiobjective stochastic optimization of dividing-wall distillation columns using a surrogate model based on neural networks," *Chemical and Biochemical Engineering Quarterly*, vol. 29, no. 4, pp. 491–504, 2016.

[151] C. R. Chen and H. S. Ramaswamy, "Modeling and optimization of variable retort temperature (vrt) thermal processing using coupled neural networks and genetic algorithms," *Journal of Food Engineering*, vol. 53, no. 3, pp. 209–220, 2002.

[152] M. Dornier, M. Decloux, G. Trystram, and A. Lebert, "Interest of neural networks for the optimization of the crossflow filtration process," *LWT-Food Science and Technology*, vol. 28, no. 3, pp. 300–309, 1995.

[153] F. A. N. Fernandes, "Optimization of fischer-tropsch synthesis using neural networks," *Chemical Engineering & Technology*, vol. 29, no. 4, pp. 449–453, 2006.

[154] I. E. Grossmann, J. Viswanathan, A. Vecchietti, R. Raman, and E. Kalvelagen, "GAMS/DICOPT: A discrete continuous optimization package," *GAMS Corporation Inc*, 2002.

[155] A. S. Drud, "Conopt—a large-scale grg code," *ORSA Journal on Computing*, vol. 6, no. 2, pp. 207–216, 1994.

[156] S. Nandi, S. Ghosh, S. S. Tambe, and B. D. Kulkarni, "Artificial neural-network-assisted stochastic process optimization strategies," *AIChE Journal*, vol. 47, no. 1, pp. 126–141, 2001.

[157] M. Tawarmalani and N. V. Sahinidis, "A polyhedral branch-and-cut approach to global optimization," *Mathematical Programming*, vol. 103, no. 2, pp. 225–249, 2005.

[158] E. de Weerdt, Q. P. Chu, and J. A. Mulder, "Neural network output optimization using interval analysis," *IEEE transactions on neural networks*, vol. 20, no. 4, pp. 638–653, 2009.

[159] R. E. Moore and F. Bierbaum, *Methods and applications of interval analysis*. SIAM studies in applied mathematics, Philadelphia: Soc. for Industrial and Applied Mathematics, 2 ed., 1979.

[160] R. Misener and C. A. Floudas, "ANTIGONE: Algorithms for continuous / integer global optimization of nonlinear equations," *Journal of Global Optimization*, vol. 59, no. 2, pp. 503–526, 2014.

[161] S. J. Maher, T. Fischer, T. Gally, G. Gamrath, A. Gleixner, R. L. Gottwald, G. Hendel, T. Koch, M. E. Lübbecke, M. Miltenberger, B. Müller, M. E. Pfetsch, C. Puchert, D. Rehfeldt, S. Schenker, R. Schwarz, F. Serrano, Y. Shinano, D. Weninger, J. T. Witt, and J. Witzig, "The SCIP optimization suite (version 4.0)."

[162] T. G. W. Epperly and E. N. Pistikopoulos, "A reduced space branch and bound algorithm for global optimization," *Journal of Global Optimization*, vol. 11, no. 3, pp. 287–311, 1997.

[163] A. Mitsos, B. Chachuat, and P. I. Barton, "McCormick-based relaxations of algorithms," *SIAM Journal on Optimization*, vol. 20, no. 2, pp. 573–601, 2009.

[164] J. K. Scott, M. D. Stuber, and P. I. Barton, "Generalized McCormick relaxations," *Journal of Global Optimization*, vol. 51, no. 4, pp. 569–606, 2011.

[165] D. Bongartz and A. Mitsos, "Deterministic global optimization of process flowsheets in a reduced space using McCormick relaxations," *Journal of Global Optimization*, vol. 20, no. 9, p. 419, 2017.

[166] W. R. Huster, D. Bongartz, and A. Mitsos, "Deterministic global optimization of the design of a geothermal organic rankine cycle," *Energy Procedia*, vol. 129, pp. 50–57, 2017.

[167] G. P. McCormick, "Computability of global solutions to factorable nonconvex programs: Part I — convex underestimating problems," *Mathematical Programming*, vol. 10, no. 1, pp. 147–175, 1976.

[168] A. Bompadre and A. Mitsos, "Convergence rate of McCormick relaxations," *Journal of Global Optimization*, vol. 52, no. 1, pp. 1–28, 2012.

[169] J. Najman and A. Mitsos, "Convergence analysis of multivariate McCormick relaxations," *Journal of Global Optimization*, vol. 66, no. 4, pp. 597–628, 2016.

[170] A. Tsoukalas and A. Mitsos, "Multivariate McCormick relaxations," *Journal of Global Optimization*, vol. 59, no. 2-3, pp. 633–662, 2014.

[171] J. Najman, D. Bongartz, A. Tsoukalas, and A. Mitsos, "Erratum to multivariate McCormick relaxations," *Journal of Global Optimization*, vol. 68, no. 1, pp. 219–225, 2017.

[172] K. A. Khan, H. A. J. Watson, and P. I. Barton, "Differentiable McCormick relaxations," *Journal of Global Optimization*, vol. 67, no. 4, pp. 687–729, 2017.

[173] K. A. Khan, M. Wilhelm, M. D. Stuber, H. Cao, H. A. J. Watson, and P. I. Barton, "Corrections to differentiable McCormick relaxations," *Journal of Global Optimization*, vol. 70, no. 3, pp. 705–706, 2018.

[174] D. Bongartz and A. Mitsos, "Infeasible path global flowsheet optimization using McCormick relaxations," in *27th European Symposium on Computer Aided Process Engineering* (A. Espuna, ed.), vol. 40 of *Computer Aided Chemical Engineering*, San Diego: Elsevier, 2017.

[175] A. Wechsung, J. K. Scott, H. A. J. Watson, and P. I. Barton, "Reverse propagation of McCormick relaxations," *Journal of Global Optimization*, vol. 63, no. 1, pp. 1–36, 2015.

[176] M. D. Stuber, J. K. Scott, and P. I. Barton, "Convex and concave relaxations of implicit functions," *Optimization Methods and Software*, vol. 30, no. 3, pp. 424–460, 2015.

[177] D. P. Bertsekas, A. Nedic, and A. E. Ozdaglar, *Convex analysis and optimization*, vol. 1 of *Athena Scientific optimization and computation series*. Belmont, Mass.: Athena Scientific, 2003.

[178] D. Bongartz, J. Najman, S. Sass, and A. Mitsos, "MAiNGO: McCormick-based Algorithm for mixed integer Nonlinear Global Optimization," *Technical report, Process Systems Engineering (AVT.SVT), RWTH Aachen University*, 2018.

[179] B. Chachuat, "MC++ (version 2.0): A toolkit for bounding factorable functions," 2014.

[180] B. Chachuat, B. Houska, R. Paulen, N. Peric, J. Rajyaguru, and M. E. Villanueva, "Set-theoretic approaches in analysis, estimation and control of nonlinear systems," *IFAC-PapersOnLine*, vol. 48, no. 8, pp. 981–995, 2015.

[181] W. Hofschuster and W. Krämer, "FILIB++ interval library (version 3.0.2)," 1998.

[182] International Business Machies, "IBM ilog CPLEX (version 12.1)," 2009.

[183] A. M. Gleixner, T. Berthold, B. Müller, and S. Weltge, "Three enhancements for optimization-based bound tightening," *Journal of Global Optimization*, vol. 67, no. 4, pp. 731–757, 2017.

[184] H. S. Ryoo and N. V. Sahinidis, "Global optimization of nonconvex NLPs and MINLPs with applications in process design," *Computers & Chemical Engineering*, vol. 19, no. 5, pp. 551–566, 1995.

[185] M. Locatelli and F. Schoen, eds., *Global optimization: Theory, algorithms, and applications*. MOS-SIAM series on optimization, Philadelphia, Pa.: Mathematical Programming Society, 2013.

[186] D. Kraft, *A Software Package for Sequential Quadratic Programming*. Deutsche Forschungs- und Versuchsanstalt für Luft- und Raumfahrt Köln: Forschungsbericht, Köln: Wiss. Berichtswesen d. DFVLR, 1988.

[187] S. G. Johnson, "The NLopt nonlinear-optimization package (version 2.4.2)," 2016.

[188] C. Bendtsen and O. Stauning, "Fadbad++ (version 2.1): a flexible C++ package for automatic differentiation," 2012.

[189] J. Najman and A. Mitsos, "Tighter McCormick relaxations through subgradient propagation," *In Revision*, 2017.

[190] K. Ghorbanian and M. Gholamrezaei, "An artificial neural network approach to compressor performance prediction," *Applied Energy*, vol. 86, no. 7-8, pp. 1210–1221, 2009.

[191] W. L. Luyben, "Design and control of the cumene process," *Industrial & Engineering Chemistry Research*, vol. 49, no. 2, pp. 719–734, 2010.

[192] E. S. Schultz, J. O. Trierweiler, and M. Farenzena, "The importance of nominal operating point selection in self-optimizing control," *Industrial & Engineering Chemistry Research*, vol. 55, no. 27, pp. 7381–7393, 2016.

[193] U. Lee, J. Burre, A. Caspari, J. Kleinekorte, A. M. Schweidtmann, and A. Mitsos, "Techno-economic optimization of a green-field post-combustion CO 2 capture process using superstructure and rate-based models," *Industrial & Engineering Chemistry Research*, vol. 55, no. 46, pp. 12014–12026, 2016.

[194] D. G. Krige, "A statistical approach to some basic mine valuation problems on the witwatersrand," *Journal of the Southern African Institute of Mining and Metallurgy*, vol. 52, no. 6, pp. 119–139, 1951.

[195] J. Sacks, W. J. Welch, T. J. Mitchell, and H. P. Wynn, "Design and analysis of computer experiments," *Statistical science*, pp. 409–423, 1989.

[196] L. Freier, J. Hemmerich, K. Schöler, W. Wiechert, M. Oldiges, and E. von Lieres, "Framework for Kriging-based iterative experimental analysis and design: Optimization of secretory protein production in corynebacterium glutamicum," *Engineering in Life Sciences*, vol. 16, no. 6, pp. 538–549, 2016.

[197] D. Ulmasov, C. Baroukh, B. Chachuat, M. P. Deisenroth, and R. Misener, "Bayesian optimization with dimension scheduling: Application to biological systems," in *Computer Aided Chemical Engineering*, vol. 38, pp. 1051–1056, Elsevier, 2016.

[198] M. Mehrian, Y. Guyot, I. Papantoniou, S. Olofsson, M. Sonnaert, R. Misener, and L. Geris, "Maximizing neotissue growth kinetics in a perfusion bioreactor: An in silico strategy using model reduction and bayesian optimization," *Biotechnology and bioengineering*, vol. 115, no. 3, pp. 617–629, 2018.

[199] E. Bradford, A. M. Schweidtmann, D. Zhang, K. Jing, and E. A. del Rio-Chanona, "Dynamic modeling and optimization of sustainable algal production with uncertainty using multivariate Gaussian processes," *Computers & Chemical Engineering*, vol. 118, pp. 143–158, 2018.

[200] E. A. Del Rio-Chanona, X. Cong, E. Bradford, D. Zhang, and K. Jing, "Review of advanced physical and data-driven models for dynamic bioprocess simulation: Case study of algae–bacteria consortium wastewater treatment," *Biotechnology and bioengineering*, vol. 116, no. 2, pp. 342–353, 2019.

[201] J. A. Caballero and I. E. Grossmann, "An algorithm for the use of surrogate models in modular flowsheet optimization," *AIChE Journal*, vol. 54, no. 10, pp. 2633–2650, 2008.

[202] E. Davis and M. Ierapetritou, "A Kriging method for the solution of nonlinear programs with black-box functions," *AIChE Journal*, vol. 53, no. 8, pp. 2001–2012, 2007.

[203] E. Davis and M. Ierapetritou, "A kriging-based approach to MINLP containing black-box models and noise," *Industrial & Engineering Chemistry Research*, vol. 47, no. 16, pp. 6101–6125, 2008.

[204] E. Davis and M. Ierapetritou, "A centroid-based sampling strategy for Kriging global modeling and optimization," *AIChE journal*, vol. 56, no. 1, pp. 220–240, 2010.

[205] J. P. Eason and L. T. Biegler, "A trust region filter method for glass box/black box optimization," *AIChE Journal*, vol. 62, no. 9, pp. 3124–3136, 2016.

[206] J. Snoek, H. Larochelle, and R. P. Adams, "Practical bayesian optimization of machine learning algorithms," in *Advances in neural information processing systems*, pp. 2951–2959, 2012.

[207] B. Shahriari, K. Swersky, Z. Wang, R. P. Adams, and N. de Freitas, "Taking the human out of the loop: A review of bayesian optimization," *Proceedings of the IEEE*, vol. 104, no. 1, pp. 148–175, 2016.

[208] D. R. Jones, M. Schonlau, and W. J. Welch, "Efficient global optimization of expensive black-box functions," *Journal of Global optimization*, vol. 13, no. 4, pp. 455–492, 1998.

[209] A. Cozad, N. V. Sahinidis, and D. C. Miller, "Learning surrogate models for simulation-based optimization," *AIChE Journal*, vol. 60, no. 6, pp. 2211–2227, 2014.

[210] F. Boukouvala, R. Misener, and C. A. Floudas, "Global optimization advances in mixed-integer nonlinear programming, MINLP, and constrained derivative-free optimization, CDFO," *European Journal of Operational Research*, vol. 252, no. 3, pp. 701–727, 2016.

[211] F. Boukouvala and C. A. Floudas, "Argonaut: Algorithms for global optimization of constrained grey-box computational problems," *Optimization Letters*, vol. 11, no. 5, pp. 895–913, 2017.

[212] J. Kim and S. Choi, "On local optimizers of acquisition functions in bayesian optimization," *arXiv preprint arXiv:1901.08350*, 2019.

[213] J. Wilson, F. Hutter, and M. Deisenroth, "Maximizing acquisition functions for bayesian optimization," in *Advances in Neural Information Processing Systems*, pp. 9884–9895, 2018.

[214] J. A. Caballero and I. E. Grossmann, "Rigorous flowsheet optimization using process simulators and surrogate models," in *Computer Aided Chemical Engineering*, vol. 25, pp. 551–556, Elsevier, 2008.

[215] N. Quirante, J. Javaloyes, R. Ruiz-Femenia, and J. A. Caballero, "Optimization of chemical processes using surrogate models based on a Kriging interpolation," in *Computer Aided Chemical Engineering*, vol. 37, pp. 179–184, Elsevier, 2015.

[216] N. Quirante, J. Javaloyes, and J. A. Caballero, "Rigorous design of distillation columns using surrogate models based on Kriging interpolation," *AIChE Journal*, vol. 61, no. 7, pp. 2169–2187, 2015.

[217] Z. Lin, J. Wang, V. Nikolakis, and M. Ierapetritou, "Process flowsheet optimization of chemicals production from biomass derived glucose solutions," *Computers & Chemical Engineering*, vol. 102, pp. 258–267, 2017.

[218] T. Keßler, C. Kunde, N. Mertens, D. Michaels, and A. Kienle, "Global optimization of distillation columns using surrogate models," *SN Applied Sciences*, vol. 1, no. 1, p. 11, 2019.

[219] M. F. Hasan, R. C. Baliban, J. A. Elia, and C. A. Floudas, "Modeling, simulation, and optimization of postcombustion CO_2 capture for variable feed concentration and flow rate. 2. pressure swing adsorption and vacuum swing adsorption processes," *Industrial & Engineering Chemistry Research*, vol. 51, no. 48, pp. 15665–15682, 2012.

[220] K. McBride, N. M. Kaiser, and K. Sundmacher, "Integrated reaction–extraction process for the hydroformylation of long-chain alkenes with a homogeneous catalyst," *Computers & Chemical Engineering*, vol. 105, pp. 212–223, 2017.

[221] J. Glassey and M. Von Stosch, *Hybrid Modeling in Process Industries*. CRC Press, 2018.

[222] P. E. Gill, W. Murray, and M. A. Saunders, "SNOPT: An SQP algorithm for large-scale constrained optimization," *SIAM review*, vol. 47, no. 1, pp. 99–131, 2005.

[223] R. Horst and H. Tuy, *Global Optimization: Deterministic Approaches*. Berlin, Heidelberg: Springer, 3 ed., 1996.

[224] D. Bongartz, *Deterministic Global Flowsheet Optimization for the Design of Energy Conversion Processes*. PhD thesis, RWTH Aachen University, 2020.

[225] I. P. Androulakis, C. D. Maranas, and C. A. Floudas, "αBB: A global optimization method for general constrained nonconvex problems," *Journal of Global Optimization*, vol. 7, no. 4, pp. 337–363, 1995.

[226] E. M. Smith and C. C. Pantelides, "Global optimisation of nonconvex MINLPs," *Computers & Chemical Engineering*, vol. 21, pp. S791–S796, 1997.

[227] M. Tawarmalani, N. V. Sahinidis, and P. Pardalos, *Convexification and Global Optimization in Continuous and Mixed-Integer Nonlinear Programming: Theory, Algorithms, Software, and Applications*, vol. 65 of *Nonconvex Optimization and Its Applications*. Boston, MA: Springer, 2002.

[228] D. Bongartz and A. Mitsos, "Deterministic global flowsheet optimization: Between equation-oriented and sequential-modular methods," *AIChE Journal*, vol. 65, no. 3, pp. 1022–1034, 2019.

[229] G. Hüllen, J. Zhai, S. H. Kim, A. Sinha, M. J. Realff, and F. Boukouvala, "Managing uncertainty in data-driven simulation-based optimization," *Computers & Chemical Engineering*, p. 106519, 2019.

[230] L. Liberti, S. Cafieri, and F. Tarissan, "Reformulations in mathematical programming: A computational approach," in *Foundations of Computational Intelligence Volume 3. Studies in Computational Intelligence, vol 203* (A. Abraham, A. Hassanien, P. Siarry, and A. Engelbrecht, eds.), pp. 153–234, Berlin & Heidelberg, Germany: Springer, 2009.

[231] S. Sundararajan and S. S. Keerthi, "Predictive approaches for choosing hyperparameters in Gaussian processes," in *Advances in neural information processing systems*, pp. 631–637, 2000.

[232] J. Najman and A. Mitsos, "On tightness and anchoring of McCormick and other relaxations," *Journal of Global Optimization*, pp. 1–27, 2017.

[233] N. Srinivas, A. Krause, S. M. Kakade, and M. Seeger, "Gaussian process optimization in the bandit setting: No regret and experimental design," *arXiv preprint arXiv:0912.3995*, 2009.

[234] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. Riedmiller, "Playing atari with deep reinforcement learning," *arXiv preprint arXiv:1312.5602*, 2013.

[235] C. A. Meyer and C. A. Floudas, "Convex envelopes for edge-concave functions," *Mathematical programming*, vol. 103, no. 2, pp. 207–224, 2005.

[236] J. Najman, D. Bongartz, and A. Mitsos, "Convex relaxations of componentwise convex functions," *Computers & Chemical Engineering*, vol. 130, p. 106527, 2019.

[237] F. Tardella, "On the existence of polyhedral convex envelopes," in *Frontiers in global optimization* (C. A. Floudas and P. Pardalos, eds.), pp. 563–573, Kluwer Academic Publishers, Dordrecht, The Netherlands, 2004.

[238] D. Kraft, "Algorithm 733: TOMP–fortran modules for optimal control calculations," *ACM Transactions on Mathematical Software (TOMS)*, vol. 20, no. 3, pp. 262–281, 1994.

[239] C.-O. CLP, "Linear programming solver: An open source code for solving linear programming problems," 2011.

[240] H. Djelassi and A. Mitsos, "libALE – a library for algebraic logical expression trees," 2019. https://git.rwth-aachen.de/avt.svt/public/libale. Accessed Novermber 8, 2019.

[241] A. Charnes and W. W. Cooper, "Chance-constrained programming," *Management science*, vol. 6, no. 1, pp. 73–79, 1959.

[242] E. Bradford, L. Imsland, D. Zhang, and E. A. d. R. Chanona, "Stochastic data-driven model predictive control using Gaussian processes," *arXiv preprint arXiv:1908.01786*, 2019.

[243] J. Wiebe, I. Cecílio, J. Dunlop, and R. Misener, "A robust approach to warped gaussian process-constrained optimization," *arXiv preprint arXiv: 2006.08222*, 2020.

[244] A. Wächter and L. T. Biegler, "On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming," *Mathematical programming*, vol. 106, no. 1, pp. 25–57, 2006.

[245] D. Menne, J. Kamp, J. E. Wong, and M. Wessling, "Precise tuning of salt retention of backwashable polyelectrolyte multilayer hollow fiber nanofiltration membranes," *Journal of membrane science*, vol. 499, pp. 396–405, 2016.

[246] E. V. Bonilla, K. M. Chai, and C. Williams, "Multi-task Gaussian process prediction," in *Advances in neural information processing systems*, pp. 153–160, 2008.

[247] A. Damianou and N. Lawrence, "Deep Gaussian processes," in *Artificial Intelligence and Statistics*, pp. 207–215, 2013.

[248] J. Wang, A. Hertzmann, and D. J. Fleet, "Gaussian process dynamical models," in *Advances in neural information processing systems*, pp. 1441–1448, 2006.

[249] O. Chapelle and L. Li, "An empirical evaluation of Thompson sampling," in *Advances in Neural Information Processing Systems 24* (J. Shawe-Taylor, R. S. Zemel, P. L. Bartlett, F. Pereira, and K. Q. Weinberger, eds.), pp. 2249–2257, Curran Associates, Inc., 2011.

[250] P. Courrieu, "Three algorithms for estimating the domain of validity of feedforward neural networks," *Neural Networks*, vol. 7, no. 1, pp. 169–174, 1994.

[251] N. Asprion, "Modeling, simulation, and optimization 4.0 for a distillation column," *Chemie Ingenieur Technik*, vol. 92, no. 7, pp. 879–889, 2020.

[252] N. Asprion, R. Böttcher, R. Pack, M.-E. Stavrou, J. Höller, J. Schwientek, and M. Bortz, "Gray-box modeling for the optimization of chemical processes," *Chemie Ingenieur Technik*, vol. 91, no. 3, pp. 305–313, 2019.

[253] J. N. Kumar, Q. Li, K. Y. Tang, T. Buonassisi, A. L. Gonzalez-Oyarce, and J. Ye, "Machine learning enables polymer cloud-point engineering via inverse design," *npj Computational Materials*, vol. 5, no. 1, pp. 1–6, 2019.

[254] J. Pinto, C. R. de Azevedo, R. Oliveira, and M. von Stosch, "A bootstrap-aggregated hybrid semi-parametric modeling framework for bioprocess development," *Bioprocess and biosystems engineering*, vol. 42, no. 11, pp. 1853–1865, 2019.

[255] G. Papadopoulos, P. J. Edwards, and A. F. Murray, "Confidence estimation methods for neural networks: A practical comparison," *IEEE transactions on neural networks*, vol. 12, no. 6, pp. 1278–1287, 2001.

[256] A. M. Schweidtmann, D. Bongartz, D. Grothe, T. Kerkenhoff, X. Lin, J. Najman, and A. Mitsos, "Global optimization of Gaussian processes," *arXiv preprint arXiv:2005.10902*, 2020.

[257] B. J. Larson and C. A. Mattson, "Design space exploration for quantifying a system model's feasible domain," *Journal of Mechanical Design*, vol. 134, no. 4, 2012.

[258] Q. Chen, R. Paulavičius, C. S. Adjiman, and S. García-Muñoz, "An optimization framework to combine operable space maximization with design of experiments," *AIChE Journal*, vol. 64, no. 11, pp. 3944–3957, 2018.

[259] N. Knudde, I. Couckuyt, K. Shintani, and T. Dhaene, "Active learning for feasible region discovery," in *2019 18th IEEE International Conference On Machine Learning And Applications (ICMLA)*, pp. 567–572, IEEE, 2019.

[260] R. J. Malak and C. J. Paredis, "Using support vector machines to formalize the valid input domain of predictive models in systems design problems," *Journal of Mechanical Design*, vol. 132, no. 10, 2010.

[261] E. Roach, R. R. Parker, and R. J. Malak Jr, "An improved support vector domain description method for modeling valid search domains in engineering design problems," in *International Design Engineering Technical Conferences and Computers and Information in Engineering Conference*, vol. 54822, pp. 741–751, 2011.

[262] D. M. Tax and R. P. Duin, "Data domain description using support vectors.," in *ESANN*, vol. 99, pp. 251–256, 1999.

[263] M. Quaglio, E. S. Fraga, E. Cao, A. Gavriilidis, and F. Galvanin, "A model-based data mining approach for determining the domain of validity of approximated models," *Chemometrics and Intelligent Laboratory Systems*, vol. 172, pp. 58–67, 2018.

[264] D. M. J. Tax, *One-class classification: Concept learning in the absence of counter-examples.* PhD thesis, Delft University of Technology, 2001.

[265] V. Chandola, A. Banerjee, and V. Kumar, "Anomaly detection: A survey," *ACM computing surveys (CSUR)*, vol. 41, no. 3, pp. 1–58, 2009.

[266] S. S. Khan and M. G. Madden, "A survey of recent trends in one class classification," in *Irish conference on artificial intelligence and cognitive science*, pp. 188–197, Springer, 2009.

[267] S. S. Khan and M. G. Madden, "One-class classification: taxonomy of study and review of techniques," *The Knowledge Engineering Review*, vol. 29, no. 3, pp. 345–374, 2014.

[268] X. Ding, Y. Li, A. Belatreche, and L. P. Maguire, "An experimental evaluation of novelty detection methods," *Neurocomputing*, vol. 135, pp. 313–327, 2014.

[269] B. Schölkopf, R. C. Williamson, A. J. Smola, J. Shawe-Taylor, and J. C. Platt, "Support vector method for novelty detection," in *Advances in neural information processing systems*, pp. 582–588, 2000.

[270] H. Letscher, D. Edelsbrunner, and A. Zomorodian, "Topological persistence and simplification," *Discrete & Computational Geometry*, vol. 28, pp. 511–533, 2002.

[271] A. Zomorodian and G. Carlsson, "Computing persistent homology," *Discrete & Computational Geometry*, vol. 33, no. 2, pp. 249–274, 2005.

[272] F. Chazal and B. Michel, "An introduction to topological data analysis: fundamental and practical aspects for data scientists," *arXiv preprint arXiv:1710.04019*, 2017.

[273] A. Patania, F. Vaccarino, and G. Petri, "Topological analysis of data," *EPJ Data Science*, vol. 6, pp. 1–6, 2017.

[274] L. Wasserman, "Topological data analysis," *Annual Review of Statistics and Its Application*, vol. 5, pp. 501–532, 2018.

[275] Y. Hiraoka, T. Nakamura, A. Hirata, E. G. Escolar, K. Matsue, and Y. Nishiura, "Hierarchical structures of amorphous solids characterized by persistent homology," *Proceedings of the National Academy of Sciences*, vol. 113, no. 26, pp. 7035–7040, 2016.

[276] M. Saadatfar, H. Takeuchi, V. Robins, N. Francois, and Y. Hiraoka, "Pore configuration landscape of granular crystallization," *Nature communications*, vol. 8, no. 1, pp. 1–11, 2017.

[277] K. Xia, "Persistent homology analysis of ion aggregations and hydrogen-bonding networks," *Physical Chemistry Chemical Physics*, vol. 20, no. 19, pp. 13448–13460, 2018.

[278] K. Xia, D. V. Anand, S. Shikhar, and Y. Mu, "Persistent homology analysis of osmolyte molecular aggregation and their hydrogen-bonding networks," *Physical Chemistry Chemical Physics*, vol. 21, no. 37, pp. 21038–21048, 2019.

[279] A. D. Smith, P. Dlotko, and V. M. Zavala, "Topological data analysis: Concepts, computation, and applications in chemical engineering," *arXiv preprint arXiv:2006.03173*, 2020.

[280] C. Tralie, N. Saul, and R. Bar-On, "Ripser.py: A lean persistent homology library for python," *The Journal of Open Source Software*, vol. 3, p. 925, Sep 2018.

[281] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, *et al.*, "Scikit-learn: Machine learning in python," *the Journal of machine Learning research*, vol. 12, pp. 2825–2830, 2011.

[282] P. Virtanen, R. Gommers, T. E. Oliphant, M. Haberland, T. Reddy, D. Cournapeau, E. Burovski, P. Peterson, W. Weckesser, J. Bright, *et al.*, "Scipy 1.0: fundamental algorithms for scientific computing in python," *Nature methods*, vol. 17, no. 3, pp. 261–272, 2020.

[283] J. Binchi, E. Merelli, M. Rucco, G. Petri, and F. Vaccarino, "jholes: A tool for understanding biological complex networks via clique weight rank persistent homology.," *Electron. Notes Theor. Comput. Sci.*, vol. 306, pp. 5–18, 2014.

[284] M. K. Chung, J. L. Hanson, J. Ye, R. J. Davidson, and S. D. Pollak, "Persistent homology in sparse regression and its application to brain morphometry," *IEEE Transactions on Medical Imaging*, vol. 34, no. 9, pp. 1928–1939, 2015.

[285] Y. Kimura and K. Imai, "Quantification of lss using the persistent homology in the sdss fields," *Advances in Space Research*, vol. 60, no. 3, pp. 722–736, 2017.

[286] E. W. Chambers and D. Letscher, "Persistent homology over directed acyclic graphs," in *Research in Computational Topology*, pp. 11–32, Springer, 2018.

[287] N. Otter, M. A. Porter, U. Tillmann, P. Grindrod, and H. A. Harrington, "A roadmap for the computation of persistent homology," *EPJ Data Science*, vol. 6, no. 1, p. 17, 2017.

[288] C. Cortes and V. Vapnik, "Support-vector networks," *Machine learning*, vol. 20, no. 3, pp. 273–297, 1995.

[289] A. J. Smola and B. Schölkopf, "A tutorial on support vector regression," *Statistics and computing*, vol. 14, no. 3, pp. 199–222, 2004.

[290] B. Schölkopf, "The kernel trick for distances," in *Advances in neural information processing systems*, pp. 301–307, 2001.

[291] S. Dreiseitl, M. Osl, C. Scheibböck, and M. Binder, "Outlier detection with one-class svms: an application to melanoma prognosis," in *AMIA Annual Symposium Proceedings*, vol. 2010, p. 172, American Medical Informatics Association, 2010.

[292] P. F. Evangelista, M. J. Embrechts, and B. K. Szymanski, "Some properties of the Gaussian kernel for one class learning," in *International Conference on Artificial Neural Networks*, pp. 269–278, Springer, 2007.

[293] Y. Xiao, H. Wang, and W. Xu, "Parameter selection of Gaussian kernel for one-class svm," *IEEE transactions on cybernetics*, vol. 45, no. 5, pp. 941–953, 2014.

[294] Y. Xiao, H. Wang, L. Zhang, and W. Xu, "Two methods of selecting Gaussian kernel parameters for one-class svm and their application to fault detection," *Knowledge-Based Systems*, vol. 59, pp. 75–84, 2014.

[295] W. E. Hart, C. D. Laird, J.-P. Watson, D. L. Woodruff, G. A. Hackebeil, B. L. Nicholson, and J. D. Siirola, *Pyomo-optimization modeling in python*, vol. 67. Springer, 2017.

[296] M. Wilhelm and M. Stuber, "Eago. jl: easy advanced global optimization in julia," *Optimization Methods and Software*, pp. 1–26, 2020.

[297] F. Chollet *et al.*, "Keras." Accessed May 2020 `https://keras.io`, 2015.

[298] L. Fortuna, S. Graziani, A. Rizzo, and M. G. Xibilia, *Soft sensors for monitoring and control of industrial processes*. Springer Science & Business Media, 2007.

[299] C. Quek, R. Balasubramanian, and G. Rangaiah, "Consider using soft analyzers to improve SRU control," *Hydrocarbon processing*, vol. 79, no. 1, pp. 101–106, 2000.

[300] L. Fortuna, A. Rizzo, M. Sinatra, and M. Xibilia, "Soft analyzers for a sulfur recovery unit," *Control Engineering Practice*, vol. 11, no. 12, pp. 1491–1500, 2003.

[301] N. J. Cavanna, M. Jahanseir, and D. R. Sheehy, "A geometric perspective on sparse filtrations," *arXiv preprint arXiv:1506.03797*, 2015.

[302] B. Chachuat, A. B. Singer, and P. I. Barton, "Global methods for dynamic optimization and mixed-integer dynamic optimization," *Industrial & Engineering Chemistry Research*, vol. 45, no. 25, pp. 8373–8392, 2006.

[303] C. D. Kappatou, D. Bongartz, J. Najman, S. Sass, and A. Mitsos, "Global dynamic optimization with hammerstein-wiener models embedded." Pre-Print `http://www.optimization-online.org/DB_HTML/2020/09/8018.html`, 2020.

[304] A. M. Schweidtmann, J. G. Rittig, A. König, M. Grohe, A. Mitsos, and M. Dahmen, "Graph neural networks for prediction of fuel ignition quality," *ChemRxiv preprint ChemRxiv:12280325*, 2020.

[305] W. Jin, R. Barzilay, and T. Jaakkola, "Junction tree variational autoencoder for molecular graph generation," in *Proceedings of the 35th International Conference on Machine Learning (ICML 2018), 10.-15.06.2018* (J. Dy and A. Krause, eds.), vol. 80 of *Proceedings of Machine Learning Research*, (Stockholmsmässan, Stockholm, Sweden), pp. 2323–2332, PMLR, 10–15 Jul 2018.

[306] M. von Stosch, R. Schenkendorf, G. Geldhof, C. Varsakelis, M. Mariti, S. Dessoy, A. Vandercammen, A. Pysik, and M. Sanders, "Working within the design space: Do our static process characterization methods suffice?," *Pharmaceutics*, vol. 12, no. 6, p. 562, 2020.

[307] W. Leitner, J. Klankermayer, S. Pischinger, H. Pitsch, and K. Kohse-Höinghaus, "Advanced biofuels and beyond: Chemistry solutions for propulsion and production," *Angewandte Chemie International Edition*, vol. 56, no. 20, pp. 5412–5452, 2017.

[308] J.-P. Lange, R. Price, P. M. Ayoub, J. Louis, L. Petrus, L. Clarke, and H. Gosselink, "Valeric biofuels: A platform of cellulosic transportation fuels," *Angewandte Chemie International Edition*, vol. 49, no. 26, pp. 4479–4483, 2010.

[309] J.-P. Lange, E. Van Der Heide, J. van Buijtenen, and R. Price, "Furfural—a promising platform for lignocellulosic biofuels," *ChemSusChem*, vol. 5, no. 1, pp. 150–166, 2012.

[310] M. Dahmen and W. Marquardt, "Model-based design of tailor-made biofuels," *Energy & Fuels*, vol. 30, no. 2, pp. 1109–1134, 2016.

[311] D. Gschwend, P. Soltic, A. Wokaun, and F. Vogel, "Review and performance evaluation of fifty alternative liquid fuels for spark-ignition engines," *Energy & Fuels*, vol. 33, no. 3, pp. 2186–2196, 2019.

[312] A. König, K. Ulonska, A. Mitsos, and J. Viell, "Optimal applications and combinations of renewable fuel production from biomass and electricity," *Energy & Fuels*, vol. 33, no. 2, pp. 1659–1672, 2019.

[313] J. R. Regalbuto, "Engineering. cellulosic biofuels–got gasoline?," *Science*, vol. 325, no. 5942, pp. 822–824, 2009.

[314] G. T. Kalghatgi, "Auto-ignition quality of practical fuels and implications for fuel requirements of future si and hcci engines," tech. rep., 2005. SAE Technical Paper 2005-01-0239.

[315] G. Kalghatgi, "Developments in internal combustion engines and implications for combustion science and future transport fuels," *Proceedings of the Combustion Institute*, vol. 35, no. 1, pp. 101–115, 2015.

[316] J. M. Derfer, C. E. Boord, F. C. Burk, R. E. Hess, W. G. Lovell, R. A. Randall, and J. R. Sabina, *Knocking Characteristics of Pure Hydrocarbons*. 100 Barr Harbor Drive, PO Box C700, West Conshohocken, PA 19428-2959: ASTM International, 1958.

[317] J. Yanowitz, M. A. Ratcliff, R. L. McCormick, J. D. Taylor, and M. J. Murphy, "Compendium of experimental cetane numbers (technical report nrel/tp-5400-67585)," tech. rep., National Renewable Energy Laboratory (NREL), Golden, CO, United States, 2017. National Renewable Energy Laboratory (NREL), Golden, CO, United States.

[318] W. L. Kubic, R. W. Jenkins, C. M. Moore, T. A. Semelsberger, and A. D. Sutton, "Artificial neural network based group contribution method for estimating cetane and octane numbers of hydrocarbons and oxygenated organic compounds," *Industrial & Engineering Chemistry Research*, vol. 56, no. 42, pp. 12236–12245, 2017.

[319] M. Dahmen and W. Marquardt, "A novel group contribution method for the prediction of the derived cetane number of oxygenated hydrocarbons," *Energy & Fuels*, vol. 29, no. 9, pp. 5781–5801, 2015.

[320] T. H. DeFries, R. V. Kastrup, and D. Indritz, "Prediction of cetane number by group additivity and carbon-13 nuclear magnetic resonance," *Industrial & Engineering Chemistry Research*, vol. 26, no. 2, pp. 188–193, 1987.

[321] H. Yang, C. Fairbridge, and Z. Ring, "Neural network prediction of cetane numbers for isoparaffins and diesel fuel," *Petroleum Science and Technology*, vol. 19, no. 5-6, pp. 573–586, 2001.

[322] A. Lapidus, E. Smolenskii, V. Bavykin, T. Myshenkova, and L. Kondrat'ev, "Models for the calculation and prediction of the octane and cetane numbers of individual hydrocarbons," *Petroleum Chemistry*, vol. 48, no. 4, pp. 277–286, 2008.

[323] R. C. Santana, P. T. Do, M. Santikunaporn, W. E. Alvarez, J. D. Taylor, E. L. Sughrue, and D. E. Resasco, "Evaluation of different reaction strategies for the improvement of cetane number in diesel fuels," *Fuel*, vol. 85, no. 5-6, pp. 643–656, 2006.

[324] T. Sennott, C. Gotianun, R. Serres, M. Ziabasharhagh, J. Mack, and R. Dibble, "Artificial neural network for predicting cetane number of biofuel candidates based on molecular structure," in *ASME 2013 Internal Combustion Engine Division Fall Technical Conference, 13.-16.10.2013, Dearborn, Michigan, USA*, American Society of Mechanical Engineers Digital Collection, 2013.

[325] T. Kessler, E. R. Sacia, A. T. Bell, and J. H. Mack, "Artificial neural network based predictions of cetane number for furanic biofuel additives," *Fuel*, vol. 206, pp. 171–179, 2017.

[326] C. Guan, J. Zhai, and D. Han, "Cetane number prediction for hydrocarbons from molecular structural descriptors based on active subspace methodology," *Fuel*, vol. 249, pp. 1–7, 2019.

[327] H. Ogawa, H. Nishimoto, A. Morita, and G. Shibata, "Predicted diesel ignitability index based on the molecular structures of hydrocarbons," *International Journal of Engine Research*, vol. 17, no. 7, pp. 766–775, 2016.

[328] A. Baghban, M. N. Kardani, and A. H. Mohammadi, "Improved estimation of cetane number of fatty acid methyl esters (fames) based biodiesels using tlbo-nn and pso-nn models," *Fuel*, vol. 232, pp. 620–631, 2018.

[329] A. Baghban and M. Adelizadeh, "On the determination of cetane number of hydrocarbons and oxygenates using adaptive neuro fuzzy inference system optimized with evolutionary algorithms," *Fuel*, vol. 230, pp. 344–354, 2018.

[330] S. M. Miraboutalebi, P. Kazemi, and P. Bahrami, "Fatty acid methyl ester (fame) composition used for estimation of biodiesel cetane number employing random forest and artificial neural networks: A new approach," *Fuel*, vol. 166, pp. 143–151, 2016.

[331] E. Smolenskii, V. Bavykin, A. Ryzhov, O. Slovokhotova, I. Chuvaeva, and A. Lapidus, "Cetane numbers of hydrocarbons: Calculations using optimal topological indices," *Russian Chemical Bulletin*, vol. 57, no. 3, pp. 461–467, 2008.

[332] B. Creton, C. Dartiguelongue, T. de Bruin, and H. Toulhoat, "Prediction of the cetane number of diesel compounds using the quantitative structure property relationship," *Energy & Fuels*, vol. 24, no. 10, pp. 5396–5403, 2010.

[333] Z. Guo, K. H. Lim, M. Chen, B. J. R. Thio, and B. L. W. Loo, "Predicting cetane numbers of hydrocarbons and oxygenates from highly accessible descriptors by using artificial neural networks," *Fuel*, vol. 207, pp. 344–351, 2017.

[334] D. A. Saldana, L. Starck, P. Mougin, B. Rousseau, L. Pidol, N. Jeuland, and B. Creton, "Flash point and cetane number predictions for fuel compounds using quantitative structure property relationship (qspr) methods," *Energy & Fuels*, vol. 25, no. 9, pp. 3900–3908, 2011.

[335] R. Meusinger and R. Moros, "Determination of quantitative structure–octane rating relationships of hydrocarbons by genetic algorithms," *Chemometrics and Intelligent Laboratory Systems*, vol. 46, no. 1, pp. 67–78, 1999.

[336] T. A. Albahri, "Structural group contribution method for predicting the octane number of pure hydrocarbon liquids," *Industrial & Engineering Chemistry Research*, vol. 42, no. 3, pp. 657–662, 2003.

[337] F. vom Lehn, B. Brosius, D. Brust, L. Cai, and H. Pitsch, "Using machine learning in model development for global fuel auto-ignition quantities," in *29. Deutscher Flammentag, 17.-18.09.2019, Bochum, Germany* (F. A. Vom Lehn, B. Brosius, D. Brust, L. Cai, and H. G. Pitsch, eds.), 2019.

[338] A. G. Abdul Jameel, V. van Oudenhoven, A.-H. Emwas, and S. M. Sarathy, "Predicting octane number using nuclear magnetic resonance spectroscopy and artificial neural networks," *Energy & Fuels*, vol. 32, no. 5, pp. 6309–6329, 2018.

[339] A. R. Katritzky, M. Kuanar, S. Slavov, C. D. Hall, M. Karelson, I. Kahn, and D. A. Dobchev, "Quantitative correlation of physical and chemical properties with chemical structure: Utility for prediction," *Chemical Reviews*, vol. 110, no. 10, pp. 5714–5789, 2010.

[340] S. W. Benson, F. R. Cruickshank, D. M. Golden, G. R. Haugen, H. E. O'Neal, A. S. Rodgers, R. Shaw, and R. Walsh, "Additivity rules for the estimation of thermochemical properties," *Chemical Reviews*, vol. 69, no. 3, pp. 279–324, 1969.

[341] K. G. Joback and R. C. Reid, "Estimation of pure-component properties from group-contributions," *Chemical Engineering Communications*, vol. 57, no. 1-6, pp. 233–243, 1987.

[342] R. Gani, B. Nielsen, and A. Fredenslund, "A group contribution approach to computer-aided molecular design," *AIChE Journal*, vol. 37, no. 9, pp. 1318–1332, 1991.

[343] H. Wiener, "Structural determination of paraffin boiling points," *Journal of the American Chemical Society*, vol. 69, no. 1, pp. 17–20, 1947.

[344] M. Gori, G. Monfardini, and F. Scarselli, "A new model for learning in graph domains," in *Proceedings of the IEEE International Joint Conference on Neural Networks (IJCNN), 31.07.-04.08.2005*, (Montreal, Quebec, Canada), pp. 729–734, 2005.

[345] F. Scarselli, M. Gori, A. C. Tsoi, M. Hagenbuchner, and G. Monfardini, "The graph neural network model," *IEEE Transactions on Neural Networks*, vol. 20, no. 1, pp. 61–80, 2009.

[346] C. W. Coley, R. Barzilay, W. H. Green, T. S. Jaakkola, and K. F. Jensen, "Convolutional embedding of attributed molecular graphs for physical property prediction," *Journal of Chemical Information and Modeling*, vol. 57, no. 8, pp. 1757–1772, 2017.

[347] C. W. Coley, W. Jin, L. Rogers, T. F. Jamison, T. S. Jaakkola, W. H. Green, R. Barzilay, and K. F. Jensen, "A graph-convolutional neural network model for the prediction of chemical reactivity," *Chemical Science*, vol. 10, no. 2, pp. 370–377, 2019.

[348] J. Gilmer, S. S. Schoenholz, P. F. Riley, O. Vinyals, and G. E. Dahl, "Neural message passing for quantum chemistry," *arXiv preprint arXiv:1704.01212v2*, 2017. arXiv.

[349] K. Yang, K. Swanson, W. Jin, C. Coley, P. Eiden, H. Gao, A. Guzman-Perez, T. Hopper, B. Kelley, M. Mathea, A. Palmer, V. Settels, T. Jaakkola, K. Jensen, and R. Barzilay, "Analyzing learned molecular representations for property prediction," *Journal of Chemical Information and Modeling*, vol. 59, no. 8, pp. 3370–3388, 2019.

[350] S. Kearnes, K. McCloskey, M. Berndl, V. Pande, and P. Riley, "Molecular graph convolutions: Moving beyond fingerprints," *Journal of Computer-Aided Molecular Design*, vol. 30, no. 8, pp. 595–608, 2016.

[351] D. K. Duvenaud, D. Maclaurin, J. Iparraguirre, R. Bombarell, T. Hirzel, A. Aspuru-Guzik, and R. P. Adams, "Convolutional networks on graphs for learning molecular fingerprints," in *Advances in Neural Information Processing Systems 28 (NIPS 2015)* (C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett, eds.), pp. 2224–2232, Curran Associates, Inc, 2015.

[352] A. M. Schweidtmann, J. G. Rittig, G. M., and A. Mitsos, "Open-source graph neural network for prediction of fuel ignition quality." `https://git.rwth-aachen.de/avt.svt/public/graph_neural_network_for_fuel_ignition_quality`, 2020. accessed on 05.04.2020.

[353] D. Bonchev and D. Rouvray, *Chemical Graph Theory: Introduction and Fundamentals*. Gordon and Breach Science Publishers, New York, United States, 1991.

[354] V. I. Minkin, "Glossary of terms used in theoretical organic chemistry," *Pure and Applied Chemistry*, vol. 71, no. 10, pp. 1919–1981, 1999.

[355] R. Todeschini and V. Consonni, *Handbook of Molecular Descriptors*. Weinheim, Germany: Wiley-VCH Verlag GmbH, 2000.

[356] Z. Wu, B. Ramsundar, E. N. Feinberg, J. Gomes, C. Geniesse, A. S. Pappu, K. Leswing, and V. Pande, "Moleculenet: A benchmark for molecular machine learning," *Chemical Science*, vol. 9, no. 2, pp. 513–530, 2018.

[357] J. Zhou, G. Cui, Z. Zhang, C. Yang, Z. Liu, L. Wang, C. Li, and M. Sun, "Graph neural networks: A review of methods and applications," *arXiv preprint arXiv:1812.08434v4*, 2018. arXiv.

[358] W. Hamilton, Z. Ying, and J. Leskovec, "Inductive representation learning on large graphs," in *Advances in Neural Information Processing Systems 30 (NIPS 2017)*, pp. 1024–1034, 2017.

[359] American Society for Testing and Materials, "Astm d 613: Standard test method for cetane number of diesel fuel oil," 2015.

[360] American Society for Testing, "Astm d 6890: Standard test method for determination of ignition delay and derived cetane number (dcn) of diesel fuel oils by combustion in a constant volume chamber," 2011.

[361] G. T. Kalghatgi, "The outlook for fuels for internal combustion engines," *International Journal of Engine Research*, vol. 15, no. 4, pp. 383–398, 2014.

[362] American Society for Testing and Materials, "Astm d 2699: Standard test method for research octane number of spark-ignition engine fuel," 2018.

[363] American Society for Testing and Materials, "Astm d 2700: Standard test method for motor octane number of spark-ignition engine fuel," 2019.

[364] G. Egloff and P. Van Arsdell, "Octane rating relationships of aliphatic, alicyclic, mononuclear aromatic hydrocarbons, alcohols, ethers, and ketones," *Journal of the Institute of Petroleum (London)*, vol. 27, pp. 121–138, 1941.

[365] J. Yanowitz, E. Christensen, and R. L. McCormick, "Utilization of renewable oxygenates as gasoline blending components (technical report nrel/tp-5400-50791)," tech. rep., 2011. National Renewable Energy Laboratory (NREL), Golden, CO, United States.

[366] R. L. McCormick, G. Fioroni, L. Fouts, E. Christensen, J. Yanowitz, E. Polikarpov, K. Albrecht, D. J. Gaspar, J. Gladden, and A. George, "Selection criteria and screening of potential biomass-derived streams as fuel blendstocks for advanced spark-ignition engines," *SAE International Journal of Fuels and Lubricants*, vol. 10, no. 2, pp. 442–460, 2017.

[367] W. R. Leppard, "The autoignition chemistries of octane-enhancing ethers and cyclic ethers: A motored engine study," *SAE Transactions*, pp. 589–604, 1991.

[368] B. G. Harvey, W. W. Merriman, and R. L. Quintana, "Renewable gasoline, solvents, and fuel additives from 2,3-butanediol," *ChemSusChem*, vol. 9, no. 14, pp. 1814–1819, 2016.

[369] D. W. Naegeli, D. M. Yost, D. S. Moulton, E. C. Owens, and G. K. Chui, "The measurement of octane numbers for methanol and reference fuels blends," *SAE Transactions*, pp. 712–722, 1989.

[370] J. Szybist and B. West, "Update on co-optima light-duty spark-ignition research," 2017. accessed on 05.04.2020.

[371] Römpp Online, "Octan-zahl," 2002. accessed on 05.04.2020.

[372] M. Fey and J. E. Lenssen, "Fast graph representation learning with pytorch geometric," *arXiv preprint arXiv:1903.02428v3*, 2019. arXiv.

[373] D. Weininger, "Smiles, a chemical language and information system. 1. introduction to methodology and encoding rules," *Journal of Chemical Information and Modeling*, vol. 28, no. 1, pp. 31–36, 1988.

[374] Greg Landrum, "Rdkit: Open-source cheminformatics." accessed on 05.04.2020.

[375] K. Cho, B. van Merrienboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio, "Learning phrase representations using rnn encoder-decoder for statistical machine translation," *arXiv preprint arXiv:1406.1078v3*, 2014. arXiv.

[376] Y. Li, D. Tarlow, M. Brockschmidt, and R. Zemel, "Gated graph sequence neural networks," *arXiv preprint arXiv:1511.05493v4*, 2015. arXiv.

[377] R. Caruana, "Multitask learning," *Machine Learning*, vol. 28, no. 1, pp. 41–75, 1997.

[378] S. Ruder, "An overview of multi-task learning in deep neural networks," *arXiv preprint arXiv:1706.05098*, 2017. arXiv.

[379] Y. Zhang and Q. Yang, "A survey on multi-task learning," *arXiv preprint arXiv: 1707.08114*, 2017.

[380] G. E. Dahl, N. Jaitly, and R. Salakhutdinov, "Multi-task neural networks for qsar predictions," *arXiv preprint arXiv: 1406.1231*, 2014.

[381] B. Ramsundar, S. Kearnes, P. Riley, D. Webster, D. Konerding, and V. Pande, "Massively multitask networks for drug discovery," *arXiv preprint arXiv:1502.02072*, 2015.

[382] T. W. Ryan III and A. C. Matheaus, "Fuel requirements for hcci engine operation," *SAE Transactions*, pp. 1143–1152, 2003.

[383] P. L. Perez and A. L. Boehman, "Experimental investigation of the autoignition behavior of surrogate gasoline fuels in a constant-volume combustion bomb apparatus and its relevance to hcci combustion," *Energy & Fuels*, vol. 26, no. 10, pp. 6106–6117, 2012.

[384] L. Torrey and J. Shavlik, "Transfer learning," in *Handbook of Research on Machine Learning Applications and Trends: Algorithms, Methods, and Techniques*, pp. 242–264, IGI Global, 2010.

[385] S. J. Pan and Q. Yang, "A survey on transfer learning," *IEEE Transactions on Knowledge and Data Engineering*, vol. 22, no. 10, pp. 1345–1359, 2009.

[386] C. A. Grambow, Y.-P. Li, and W. H. Green, "Accurate thermochemistry with small data sets: A bond additivity correction and transfer learning approach," *The Journal of Physical Chemistry A*, vol. 123, no. 27, pp. 5826–5835, 2019.

[387] L. Breiman, "Bagging predictors," *Machine Learning*, vol. 24, no. 2, pp. 123–140, 1996.

[388] L. Breiman, "Stacked regressions," *Machine Learning*, vol. 24, no. 1, pp. 49–64, 1996.

[389] T. G. Dietterich, "Ensemble methods in machine learning," in *Multiple Classifier Systems: First International Workshop (MCS 2000), Lecture Notes in Computer Science, 21.06.-23.06.2000*, (Cagliari, Italy), pp. 1–15, 2000.

[390] Y. Freund and R. E. Schapire, "Experiments with a new boosting algorithm," in *Proceedings of the Thirteenth International Conference on Machine Learning, 03.07.-06.07.1996*, ICML'96, (Bari, Italy), pp. 148–156, Morgan Kaufmann, 1996.

[391] S. Varma and R. Simon, "Bias in error estimation when using cross-validation for model selection," *BMC Bioinformatics*, vol. 7, no. 1, p. 91, 2006.

[392] P. Gramatica, "Principles of qsar models validation: internal and external," *QSAR & Combinatorial Science*, vol. 26, no. 5, pp. 694–701, 2007.

[393] P. Gramatica, "On the development and validation of qsar models," in *Computational Toxicology*, pp. 499–526, Springer, 2013.

[394] N. De Cao and T. Kipf, "Molgan: An implicit generative model for small molecular graphs," *arXiv preprint arXiv:1805.11973*, 2018. arXiv.

[395] Q. Liu, M. Allamanis, M. Brockschmidt, and A. Gaunt, "Constrained graph variational autoencoders for molecule design," in *Advances in Neural Information Processing Systems 31 (NIPS 2018)*, pp. 7795–7804, 2018.

[396] M. Simonovsky and N. Komodakis, "Graphvae: Towards generation of small graphs using variational autoencoders," in *27th International Conference on Artificial Neural Networks (ICANN 2018), 04.-07.10.2018*, (Rhodes, Greece), pp. 412–422, Springer, 2018.

[397] M. Krenn, F. Häse, A. Nigam, P. Friederich, and A. Aspuru-Guzik, "Self-referencing embedded strings (selfies): A 100% robust molecular string representation," *arXiv preprint arXiv:1905.13741*, 2019. arXiv.

[398] H. Kajino, "Molecular hypergraph grammar with its application to molecular optimization," in *Proceedings of the 36th International Conference on Machine Learning (ICML 2019), 9.-15.06.2019* (K. Chaudhuri and R. Salakhutdinov, eds.), vol. 97 of *Proceedings of Machine Learning Research*, (Long Beach, California, USA), pp. 3183–3191, PMLR, 09–15 Jun 2019.

[399] M. Niepert, M. Ahmed, and K. Kutzkov, "Learning convolutional neural networks for graphs," 2016.

[400] K. T. Schütt, F. Arbabzadah, S. Chmiela, K. R. Müller, and A. Tkatchenko, "Quantum-chemical insights from deep tensor neural networks," *Nature Communications*, vol. 8, no. 1, p. 13890, 2017.

[401] C. Chen, W. Ye, Y. Zuo, C. Zheng, and S. P. Ong, "Graph networks as a universal machine learning framework for molecules and crystals," *Chemistry of Materials*, vol. 31, no. 9, pp. 3564–3572, 2019.

[402] "Meaning of overfitting in english."

[403] Z. Wu, S. Pan, F. Chen, G. Long, C. Zhang, and P. S. Yu, "A comprehensive survey on graph neural networks," 2019.

[404] K. Xu, W. Hu, J. Leskovec, and S. Jegelka, "How powerful are graph neural networks?," *arXiv preprint arXiv:1810.00826*, 2018.

[405] C. Lu, Q. Liu, C. Wang, Z. Huang, P. Lin, and L. He, "Molecular property prediction: A multilevel quantum interactions modeling perspective."

[406] H. Shindo and Y. Matsumoto, "Gated graph recursive neural networks for molecular property prediction."

[407] O. Vinyals, S. Bengio, and M. Kudlur, "Order matters: Sequence to sequence for sets."

[408] Z. Ying, J. You, C. Morris, X. Ren, W. Hamilton, and J. Leskovec, "Hierarchical graph representation learning with differentiable pooling," in *Advances in Neural Information Processing Systems*, pp. 4800–4810, 2018.

[409] M. Zhang, Z. Cui, M. Neumann, and Y. Chen, "An end-to-end deep learning architecture for graph classification," in *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

[410] L. Ruddigkeit, R. van Deursen, L. C. Blum, and J.-L. Reymond, "Enumeration of 166 billion organic small molecules in the chemical universe database gdb-17," *Journal of chemical information and modeling*, vol. 52, no. 11, pp. 2864–2875, 2012.

[411] R. Ramakrishnan, P. O. Dral, M. Rupp, and O. A. von Lilienfeld, "Quantum chemistry structures and properties of 134 kilo molecules," *Scientific Data*, vol. 1, pp. 140022 EP –, 2014.

[412] W. Pronobis, K. T. Schütt, A. Tkatchenko, and K.-R. Müller, "Capturing intensive and extensive dft/tddft molecular properties with machine learning," *The European Physical Journal B*, vol. 91, no. 8, p. 338, 2018.

[413] K. T. Schütt, A. Tkatchenko, and K.-R. Müller, "Learning representations of molecules and materials with atomistic neural networks."

[414] K. T. Schütt, H. E. Sauceda, P.-J. Kindermans, A. Tkatchenko, and K.-R. Müller, "Schnet - a deep learning architecture for molecules and materials," *The Journal of Chemical Physics*, vol. 148, no. 24, p. 241722, 2018.

[415] K. Gubaev, E. V. Podryabinkin, and A. V. Shapeev, "Machine learning of molecular properties: locality and active learning," *The Journal of Chemical Physics*, vol. 148, no. 24, p. 241727, 2018.

[416] S. Ye, J. Liang, R. Liu, and X. Zhu, "Symmetrical graph neural network for quantum chemistry, with dual r/k space."

[417] W. L. Hamilton, R. Ying, and J. Leskovec, "Representation learning on graphs: Methods and applications."

[418] A. R. Katritzky, V. S. Lobanov, and M. Karelson, "Qspr: the correlation and quantitative prediction of chemical and physical properties from structure," *Chemical Society Reviews*, vol. 24, no. 4, p. 279, 1995.

[419] T. Renner, *Quantities, Units and Symbols in Physical Chemistry*. The Royal Society of Chemistry, 2007.

[420] S. Kim, P. A. Thiessen, E. E. Bolton, J. Chen, G. Fu, A. Gindulyte, L. Han, J. He, S. He, B. A. Shoemaker, J. Wang, B. Yu, J. Zhang, and S. H. Bryant, "Pubchem substance and compound databases," *Nucleic acids research*, vol. 44, no. D1, pp. D1202–13, 2016.

[421] S. V. Ley, D. E. Fitzpatrick, R. J. Ingham, and R. M. Myers, "Organic synthesis: march of the machines," *Angewandte Chemie International Edition*, vol. 54, no. 11, pp. 3449–3464, 2015.

[422] A. Echtermeyer, Y. Amar, J. Zakrzewski, and A. Lapkin, "Self-optimisation and model-based design of experiments for developing a c-h activation flow process," *Beilstein journal of organic chemistry*, vol. 13, pp. 150–163, 2017.

[423] R. Rico-Martines, I. Kevrekidis, M. Kube, and J. Hudson, "Discrete-vs. continuous-time nonlinear signal processing: Attractors, transitions and parallel implementation issues," in *1993 American Control Conference*, pp. 1475–1479, IEEE, 1993.