Computational Materials Science 246 (2025) 113432

Contents lists available at ScienceDirect

Computational Materials Science

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

Full length article

Exploring design space: Machine learning for multi-objective materials design optimization with enhanced evaluation strategies

Felix Conrad [a,*], Julien Philipp Stöcker [b], Cesare Signorini [c], Isabela de Paula Salgado [c,d], Hajo Wiemer [a], Michael Kaliske [b], Steffen Ihlenfeldt [a]

[a] TUD - Dresden University of Technology, Institute of Mechatronic Engineering, Dresden, Germany
[b] TUD - Dresden University of Technology, Institute for Structural Analysis, Dresden, Germany
[c] TUD - Dresden University of Technology, Institute of Construction Materials, Dresden, Germany
[d] United Nations University, Institute for Integrated Management of Material Fluxes and of Resources, Dresden, Germany

ARTICLE INFO

Dataset link: https://github.com/fc-tud/material-optimization

Keywords:
Machine learning in materials design
Automated machine learning
Multi-objective optimization
Splitting methods for performance evaluation

ABSTRACT

Discovering optimal material designs in the design space can be significantly accelerated by leveraging machine learning (ML) models for screening candidates. However, the quality of these designs depends on the prediction accuracy of the ML models and the efficiency of the optimization algorithms used. This study comprehensively compares different ML modeling strategies, optimization algorithms and evaluation strategies. Thereby, automated ML, tree-based ML models and neural networks were compared. Various optimization algorithms were analyzed, including random search, evolutionary and swarm-based methods. In addition, different strategies for evaluating the predictive performance of the ML models were investigated, which is particularly important as these models are expected to predict design parameters that deviate significantly from the known designs in the training data throughout the optimization. Our results highlight the capability of the proposed workflow to discover material designs that significantly outperform those within the training database and approach theoretical optima. Overall, this research contributes to advancing the field of material design optimization by providing a versatile and practical workflow that introduces automated ML into material design optimization and new model error assessment strategies tailored explicitly to optimization tasks.

1. Introduction

In the development of materials, the pursuit of optimized designs is crucial to meet the ever-increasing demands for efficiency, robustness and sustainability. The use of machine learning (ML) models to predict material properties is steadily increasing [1,2], especially for concrete [3,4] and polymers [5]. ML models that capture material behavior are particularly well suited to material optimization as they can quickly predict many different variants from the feature space. Thus, integrating machine learning (ML) techniques has proven to be a promising strategy for optimizing material designs, known as the "inverse material design" problem [6].

1.1. Multi-objective optimization in materials-design

Solving the inverse material design problem with multiple objectives yields many optimal solutions, as the best design parameters for the different properties are different. These solutions, regarded to as "Pareto-optimal solutions", represent interlinked design parameters where enhancing one target property inevitably compromises another, which is schematically illustrated in Fig. 1(b). Numerous research studies have investigated multi-criteria optimization in materials design. Here, we adopt the classification proposed by Hanaoka [7] to classify these approaches.

The strategy, termed *"few solution inverse designs"*, does not determine the entire Pareto front. This method often uses scalarization functions that convert multiple objective properties into a single objective value that can be optimized using any single objective method. Among these functions, the weighted summation of multiple objective properties is the most widely used. An alternative strategy referred to as *"many solution inverse designs"* aims to obtain the entire Pareto front. From this, the most suitable design for the application in question is selected. In the *"find goal design"* approach, optimization targets (multiple for multi-objective scenarios) are defined, and the optimization process concludes as soon as the goal is reached, even if the Pareto optimum has not yet been achieved.

In the scope of "few solution inverse designs", Sun et al. [8] utilized scalarization with a weighted sum of objectives to transform the multi-objective task into a single objective task. They optimized

* Corresponding author.
E-mail address: felix.conrad@tu-dresden.de (F. Conrad).

https://doi.org/10.1016/j.commatsci.2024.113432
Received 8 July 2024; Received in revised form 26 September 2024; Accepted 30 September 2024
Available online 17 October 2024

concrete mixtures to minimize costs and CO_2 emissions, integrating the Analytic Hierarchy Process (AHP) to determine the weights in the scalarization of the optimization objectives. Wang et al. [9] also used a multi-objective approach via a weighted scalarization function and evolutionary optimization for three material properties of polystyrene/polyacrylonitrile fibers. Diao et al. [10] optimized the elongation and tensile strength of carbon steels via scalarization by multiplying both target properties for a single metric. Their study utilized feed-forward neural networks (FFNN), random forests (RF), and support vector regression (SVR) models, with optimization conducted through the efficient global optimization method.

Various methodologies have been employed for "many solution inverse designs". In several studies, Zhang et al. explored various swarm-based optimization techniques to optimize concrete formulations in terms of mechanical properties and sustainability. In particular, they proposed the multi-objective particle swarm optimization (MOPSO) [11], the multi-objective firefly algorithm (MOFA) [12] and the multi-objective bat algorithm (MOBAS) [13]. From the resulting Pareto front, they selected the best design using the technique for order preference by similarity to an ideal solution (TOPSIS). Likewise, Huang et al. [14] optimized steel fiber-reinforced concrete mixtures for cost and compressive or flexural strength using support vector regression and MOFA optimization. They employed an efficiency ratio that divided the cost increase by the strength improvement to obtain the best design from the Pareto front. Feng et al. [15] optimized the mechanical and corrosion properties of aluminum alloys using various ML models, including FFNN, SVR and GBM, as well as evolutionary algorithms for the optimization. Moreover, Motoyama et al. introduced PHYSBO, i.e. a Bayesian optimization tool tailored for multi- and single-output optimization tasks [16]. Built upon Gaussian Processes regression, PHYSBO identifies inputs with the highest probability of yielding improved output values, thus facilitating many solution inverse design endeavors. This methodology was applied to maximize band-gap (E_g) and the dielectric constant of semiconductors.

In "find goal design", Hanaoka [7] demonstrated the effectiveness of Bayesian optimization using Gaussian processes as ML models to optimize the material properties of polystyrene/polyacrylonitrile (PS/PAN) fibers. Remarkably, the search for the optimal material parameters was more than 1000 times faster than the random search. A hybrid strategy combines aspects of "find goal design" and "many solution inverse designs". By setting constraints on all but one of the optimization objectives and optimizing the remaining objective, the Pareto front can be determined incrementally. Golafshani et al. [17] conducted a thorough comparison of various tree-based models, such as XGBoost, CatBoost and Random Forest, combined with the swarm-based optimization algorithms grey wolf optimizer [17]. Their approach focuses on a single optimization goal while restricting the others to predefined objectives. Similarly, Asadishamsabadi et al. [18] studied the optimization of concrete formulations including recycled aggregates, targeting cost and environmental impact indicators while complying with specific strength requirements. They achieved an efficient multi-objective optimization by employing extreme gradient boosting machines as ML models and evolutionary algorithms.

These studies collectively embody the diverse methodologies and considerations in optimizing materials and structural designs, emphasizing the integration of ML models for efficient multi-objective optimization. They employ various ML models, predominantly FFNN and tree-based models, alongside various optimization algorithms, mainly from swarm-based, evolutionary and Bayesian optimization classes. These findings align with the recent review by Stergiou et al. [19] on predicting and optimizing process and material designs. Common to all studies is the manual approach to model selection and hyperparameter optimization, with the best ML model chosen based on randomly splitting the data for training and testing. However, this conventional approach to evaluating model error can be misleading for optimization purposes, as the samples considered in the optimization process may differ significantly from the known designs in the training data.

1.2. Assessment of the prediction error of ML models

Prediction accuracy in materials design relies heavily on the available data volume [20,21]. However, it is essential to acknowledge that the prediction accuracy may vary across the feature space, particularly concerning small datasets. Consequently, understanding and addressing the inherent uncertainty of ML models is essential, especially in scenarios with limited data.

Various methods for deep learning approaches, such as those based on generative adversarial models [22,23], DiscoNets [24] or the deep ensemble method [25], offer promising results. However, these methodologies are not considered within the context of this study, which instead focuses on experiment-intensive research with relatively small data volumes (< 1000 data points) due to cost and resource constraints.

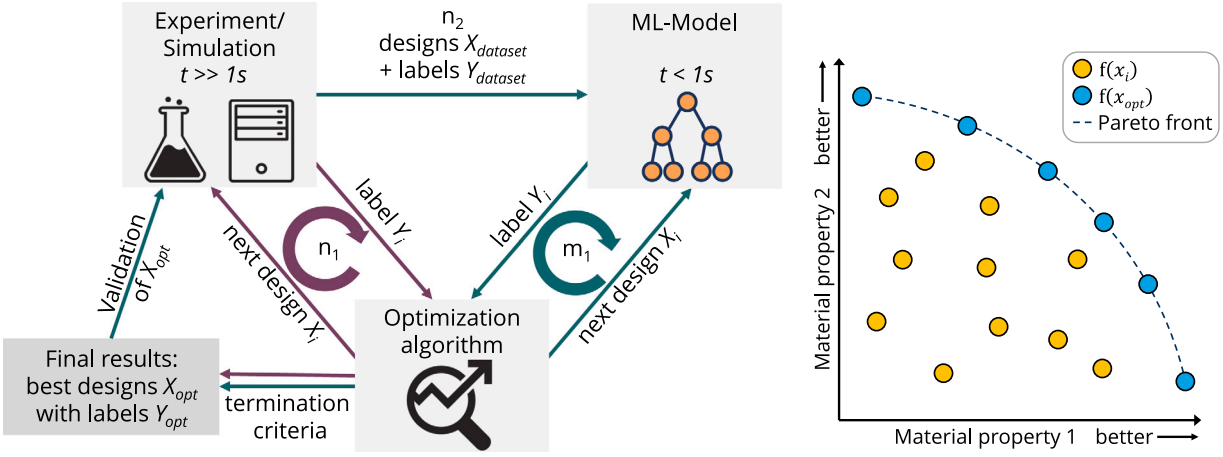
Gaussian Process Regressor (GPR) can effectively represent the uncertainty of its predictions [26]. Vasseur et al. [27] conducted a comparative analysis of various methods for quantile regression, highlighting the superior performance of tree-based Gradient Boosting Machines (GBM) with pinball loss. However, the "No Free Lunch" theorem [28] states that no single model universally outperforms others, and a tailored evaluation for each ML model is needed to identify the most suitable one for optimization and error estimation purposes.

Traditionally, test data is randomly drawn from the entire dataset to estimate model prediction errors using global error metrics. Nonetheless, global prediction metrics might not be optimal for discerning which model excels in predicting promising new design variants [29]. Alternative methods, such as cluster sampling [30] or density-biased sampling [31], attempt to maintain equal representation in training and test datasets. However, these methods often overestimate model performance [32] and are unsuitable for comprehensive evaluation. To enhance the assessment of predictive accuracy for optimized solutions, Rosario et al. [29] proposed a split of training and test data based on the Pareto front, facilitating a more robust evaluation of model errors in predicting new optimized designs.

1.3. Overview of approach for ML-based optimization

This paper addresses two identified gaps. Firstly, it conducts a comprehensive comparative analysis of ML models, optimization algorithms, and performance evaluation strategies. This analysis aims to identify the most effective methods for materials design optimization and provides an in-depth discussion of their implications. Thereby, it introduces novel strategies for performance evaluation of ML models specifically designed for the performance evaluation in an optimizing task. Secondly, incorporating automated machine learning (AutoML) into optimization processes for materials design represents an approach that aims to improve usability in line with the principles of usable AI proposed by Wiemer et al. [33]. The integration of AutoML techniques aims to streamline and automate the optimization process, making it more accessible and usable.

The general approach to materials design optimization presented in this paper is illustrated in Fig. 1(a). While running through a step n can take hours for simulations and often even days for experiments, a step m for ML models can be run through in less than 1 s. Initially, a database of material designs and their properties is established, either by experimental methods or simulations or an existing database can be used. Subsequently, this database is utilized to train an ML model, which is then employed to optimize the optimal design parameters. Validation of the found design parameters is essential since the optimization results rely solely on the ML model's predictions. These must be confirmed with the real-world results of the found design parameters. Once the ML model is obtained, exploring the entire Pareto front becomes feasible, as querying the ML model for the properties of a material design requires very few resources. Therefore, this paper emphasizes a hybrid of "find goal design" and "many solution inverse designs" to achieve the entire Pareto front in a predefined region and step size. The final design



(a) Overview of optimization approaches in inverse materials development. In purple, the classic workflow with direct optimization of the experiment or simulation is shown, in turquoise, the workflow is based on an ML model representing the material behavior.

(b) Illustration of a multi-objective solution space, with the resulting Pareto front.

Fig. 1. Workflow for multi-objective optimization and candidate ranking in a multi-objective optimization setting. Two points cannot outperform each other if they are alternately dominant for different objectives. Pareto-optimal points are not strictly dominated by any other and form the Pareto front.

can then be selected based on the entire Pareto front, considering the prioritization of individual optimization objectives for the given use case.

The subsequent chapters describe the details in the following order: First, the implementation of the workflow is presented, which includes ML modeling, evaluation strategies and multi-objective optimization. Additionally, the applied use cases of a simulation of polystyrene/polyacrylonitrile (PS/PAN) fiber properties and a simulation of a cantilever beam made of textile-reinforced concrete (TRC) are described in detail. Second, a performance comparison for the optimization workflow is presented. For this purpose, the optimization workflow is applied to the use cases. Third, the observations and their implications are discussed, and lastly, conclusions and prospects are presented. Information about data and code availability is appended.

2. Methods

2.1. ML models

A comparative approach for ML modeling was implemented by evaluating different modeling methods. Following the benchmark of Conrad et al. [21], we use AutoSklearn in version 0.15.0 [34], which has shown the best performance of the analyzed automatic ML tools in predicting material properties. Autosklearn outperformed manually created models from the reference studies by up to 20% in terms of prediction performance. In addition, we use the tree-based algorithms CatBoost in version 1.2.3 [35] and XGBoost in version 2.0.3 [36]. Tree-based methods have proven to be high-performing models in many different application areas, often outperforming FFNN for tabular data [37]. XGBoost and CatBoost, in particular, performed well in the study of Bentejca et al. [38], where they were the two best-performing gradient-boosting algorithms. Furthermore, we use PyTorch version 2.2.0 to create FFNN. This covers a wide range of conventional ML modeling techniques for the comparison of different ML models in this study.

2.2. Error estimation

It is intended to enable a more accurate estimation of model performance when points from outside the known distribution are to be predicted, as is usual in optimization tasks. In addition to random splitting into train and test data, the following three splitting methods, ‘Euclidean-distance’ (E-dist), ‘extrapolation’ and ‘Pareto-front’, were used. An example of the train-test split for each of the four methods is shown in Fig. A.1.

The splitting strategy referred to as ‘Euclidean-distance’ uses principal component analysis (PCA) to reduce dimensionality. This involves converting the original variables into a new set of orthogonal variables, called principal components, which capture the maximum variance in the data. In this way, complex datasets can be simplified while retaining the most important information and allowing the data to be represented in fewer dimensions. This representation is used to split the data so that the training data represents the known distribution of the dataset and the test set represents the out-of-distribution data. The procedure is presented in Algorithm 1.

Algorithm 1: Splitting dataset into training and test data with the Euclidean-distance method.

$X\_data \leftarrow dataset[features]$
$X' \leftarrow \text{PCA}(X\_data)$
for x_i ***in*** X' **do**
  for x_j ***in*** X' **do**
    $E[j] \leftarrow \text{euclidean\_distance}(x_i, x_j)$
  end
  $D[i] \leftarrow \text{sum}(\text{sorted}(E)[:10])$
end
$train\_indices,\ test\_indices \leftarrow \text{arg}(\text{sorted(D)}[: n_{\text{train\_data}}]),$
$all\_indices \setminus train\_indices$
$train\_data,\ test\_data \leftarrow dataset[train\_indices],\ dataset[test\_indices]$

The splitting strategy called ‘extrapolation’ involves dividing the dataset based on the quantiles of the label. Specifically, the x percent lowest and x percent highest values of the label are allocated as test data, while the remaining values constitute the training data. We divide the dataset into training and test data:

$$train\_data = \{dataset \mid label \in [q_{\text{low}}, q_{\text{high}}]\}$$
$$test\_data = \{dataset \mid label \notin [q_{\text{low}}, q_{\text{high}}]\} \quad (1)$$

, where q_{low} and q_{high} represent the x percent quantile and $(100 - x)$ percent quantile of the label, respectively. This splitting strategy allows for evaluating extrapolation quality and assessing the model’s performance beyond the range of observed data.

The methodology proposed by del Rosario et al. [29] is used for the Pareto-front splitting. In this scheme, the test data comprise the Pareto frontiers, while the training data comprises the remaining points. When

dealing with a group of candidates denoted as $Y \subseteq \mathbb{R}^D$, we define the Pareto frontier as the collection of points within Y that are not dominated by any other point in Y. This frontier, denoted as $P(Y)$, is mathematically expressed as:

$$P(Y) = \{y' \in Y \mid \{y'' \in Y \mid y'' \succ y', y' \neq y''\} = \emptyset\} \tag{2}$$

This definition implies that each point y in the Pareto frontier represents a unique combination of objective values that cannot be improved by any other point y' in Y without worsening at least one of the objectives. We used the concept of levels to expand the test set to include additional points near the Pareto frontier. These levels are recursively defined as follows: The first level, $L1$, corresponds to the Pareto frontier, as described by $P(Y)$. Subsequent levels, denoted as L_s, are determined iteratively by considering the set Y without the points in the previous level L_{s-1} and then calculating the Pareto frontier of this remaining set. This process continues until the desired amount of test data is reached.

$$\begin{aligned} L_1 &= P(Y) \\ L_s &= P(Y \setminus L_{s-1}) \quad \text{for } s = 2, 3, \ldots \end{aligned} \tag{3}$$

To robustly evaluate the performance of our models, we used nested cross-validation as recommended by Conrad et al. [32], which is presented in Algorithm 2. This approach involved performing a 10-fold inner cross-validation for the hyperparameter optimization (HPO) to ensure the robustness and performance of the HPO process as recommended in [32]. For all models except AutoSklearn, hyperparameters were optimized with Optuna version 3.5.0 using the Tree-structured Parzen Estimator (TPE) algorithm with 200 runs, which allowed a thorough exploration of the hyperparameter space. The specifications of this space, described for CatBoost in Table A.1, for XGBoost in Table A.2 and for FFNN in Table A.3, aimed to ensure comprehensive coverage of potential configurations. AutoSklearn automatically solves the *combined algorithm selection and hyperparameter optimization* (CASH) problem [39]. For this purpose, 10 splits are also performed for the inner cross-validation to solve the CASH problem. The budget for solving the CASH problem was set to 15 min of training time with 8 CPU cores (Intel(R) Xeon(R) Gold 6136 CPU @3.00 GHz). Finally, the model with the best hyperparameters is retrained on the entire dataset.

Algorithm 2: Nested cross-validation

```
generate_outer_cv_splits(data = dataset, mode= 'random', ' E-dist',
  'extrapolation' or 'Pareto-front')
foreach outer_cv_split in outer_splits do
    generate_inner_cv_splits(data = train_data_outer_split,
      mode='random')
    for n in HPO_Budget do
        model.define_hyperparameter()
        foreach inner_cv_split in inner_splits do
            model.train(data = train_data_inner_split)
            model.predict(data = val_data_inner_split)
        end
    end
    best_model = get_best_model(scoring=RMSE)
    best_model.predict(test_data_outer_split)
end
best_model.retrain(data = dataset)
```

2.3. Multi-objective optimization

The aim of systematic material property optimization is to determine the Pareto front uniformly in a predefined region. The following transformation is employed to tailor the multi-objective optimization (MOO) towards a step-wise single-objective optimization (SOO), facilitating the customization of the region and steps within the Pareto front. This adaptable transformation designates one target variable as the objective for optimization while the remaining target variables serve as constraints. Thus, if these constraints are not met, the performance is determined by the aggregated deviation of the constrained variables from their respective boundaries. Let M represent the model, Y_t represent the model output corresponding to the objective variable, and Y_{bn} denote the other outputs of the model. B_n denotes the corresponding boundary conditions for the model output Y_{bn}. The negation of both is used for boundary conditions that represent a maximum acceptable value. The scoring function S is defined as follows:

$$S = \begin{cases} Y_t & \text{if } Y_{bi} \geq B_i \text{ for all } i = n, \\ \sum_{i \in I \mid Y_{bi} < B_i} (Y_{bi} - B_i) & \text{otherwise} \end{cases} \tag{4}$$

The negation of the scoring function was used for the implementation, leading to:

$$\text{Minimize } f(x) = -S(M_n(x), B_n) \tag{5}$$

Nevergrad in version 1.0.2, [40] and skopt in version 0.9.0 were used to perform the optimization tasks with different optimization algorithms. Within Nevergrad, the following four optimizers were utilized: particle swarm optimization (PSO), differential evolution algorithm (DE), covariance matrix adaptation evolution strategy (CMA) and random optimization. The PSO [41] maintains a population of candidate solutions (particles) that move through the search space, adjusting their positions based on their own best-known position and the global best-known position found by any particle. The DE [42] iteratively improves a population of candidate solutions. It consists of four phases. The first is the initialization of the population, which is a one-time process. Afterward, mutation plus recombination of the current population and selection of the best solutions are repeated. The CMA is an evolutionary algorithm that dynamically adapts a multivariate normal distribution to model the search space. It iteratively updates the mean and covariance matrix of the distribution based on the performance of candidate solutions [43]. Furthermore, random optimization, where the candidate solutions are randomly selected, is used for the comparative evaluation of optimization algorithms. Additionally, for the optimization of the Sim-TRC, the Bayesian optimization implemented with skopt, was employed due to its efficacy in low iteration scenarios. This choice was made because each simulation required 2–12 CPU hours.

2.4. Use cases

To demonstrate the application of inverse design in this study for realistic materials design problems, virtual materials design experiments were conducted using two simulation models as a substitute for time-consuming real experiments, as outlined in Fig. 1(a).

The first use case, named "Sim-PAN" in the following, is based on the investigations by Wang et al. [9] on multi-criteria optimization of materials, in which electrospun polystyrene/polyacrylonitrile (PS/PAN) fibers are investigated. These fibers can be used as a potential absorbent for marine oil spills to minimize the ecological damage caused by oil spills, especially in aquatic environments such as the sea. For optimal performance, materials with high water contact angles (WCA), excellent oil sorption capacity (S) and high tensile strength (σ_t) are considered favorable. Seven design parameters, displayed in Table 1, can be adjusted to achieve these goals. These design parameters include three factors related to the material composition, which are PS:PAN ratio, mass fraction of solute (ω solute), and SiO_2 nanoparticle mass fraction in the solute (ω SiO_2 in solute) and four factors related to the manufacturing process, namely feed rate, pick-up distance, applied voltage, and needle inner diameter. σ_tThe simulation can lead to nonphysical results, so the negative S and σ_t values were cut off at 0.

The second use case, referred to as "Sim-TRC", is based on the simulation of a concrete cantilever beam. A steel plate is added on top

Table 1
Description of features and labels from the design space of the PS/PAN fibers simulation, labels are based on the dataset with size 200.

	Features							Labels		
	PS:PAN ratio	ω solute	ω SiO_2 in solute	feed rate	pick-up distance	Applied voltage	Inner diameter	WCA	S	σ_t
Unit	[–]	[–]	[–]	[mL/h]	[cm]	[kV]	[mm]	[°]	[g/g]	[MPa]
Min	3/7	2/3	0	0.35	0.5	4/7	0.311	85.5	0	0
Max	1	1	1	1	1	1	1	178.1	174.0	6.54

Table 2
Description of features and labels from the design space of the TRC cantilever beam simulation.

	Features						Labels	
	x_1	x_2	x_3	y_1	y_2	y_3	f_{res}	A_{beam}
Unit	[mm]	[mm]	[mm]	[mm]	[mm]	[mm]	[kN]	[mm^2]
Min	1	1	1	0	1	0	1.50	1
Max	9	9	9	3	3	3	111.71	61

of the cantilever beam for better load introduction. The optimization goal is maximal load-bearing capacity (f_{res}) with minimal material utilization (A_{beam}). Six variable design parameters related to the beam's cross-section are introduced for this purpose. The cross-section is divided into three rectangular elements, each possessing its own width $x_\square$ and height $y_\square$, where the subscript denotes the cross-section element number. A schematic depiction of the beam and its cross-section is provided in Fig. 2(a). The ranges for the design parameters from the set of natural numbers $\mathbb{N}_0$ are provided in Table 2. It should be noted that the cross-section element denoted by subscript 2 necessarily exists. The center of this block is always aligned with the center of the top plate. Elements 1 and 3 are always connected to element 2. However, those elements might not exist if their height is set to 0. The $y_\square$ values are also set to 0 if $x_\square$ is 0 during ML modeling and optimization. This results in the following number of possible variations

$$\underbrace{9^3 \cdot 3^3}_{\text{3 blocks present}} + \underbrace{2 * 9^2 \cdot 3^2}_{\text{2 blocks present}} + \underbrace{9 \cdot 3}_{\text{1 block present}} = 21168. \tag{6}$$

According to the maximum possible dimensions of the cross-section, the dimensions of the top steel plate are set to 9 mm in the x- and y-direction and 0.5 mm in the z-direction. Overall, the beam, including the plate, has a length of l_{beam} = 18.5 mm. A fixation in all spatial dimensions is applied at z = 0 mm, while an eccentric displacement load is applied at the steel plate at z = 18.5 mm. The eccentricity, given as distances in x- and y-direction from the center of gravity of the steel plate in Fig. 2(a), are set to $x_e = y_e = 2.5$ mm.

The textile-reinforced concrete (TRC) constitutive model of the cantilever beam is obtained from Platen et al. [44], utilizing a microplane material model to capture the induced anisotropy of the concrete. To avoid localization phenomena when material degradation, i.e., softening behavior, is encountered, a nonlocal field with implicit gradient enhancement is utilized. Table A.4 presents the material parameters utilized in this study. Their purpose and implementation within the material model are beyond the scope of this study but can be obtained from Platen et al. [44]. The steel plate is assumed to behave linearly elastic.

The load is applied as a prescribed displacement at the aforementioned load introduction point on the steel plate. In order to take into account the different stress distributions in cross-sections of different sizes and the resulting different displacements at the start of degradation, the maximum applied displacement is defined as a function of the cross-sectional area. The function is obtained as a linear interpolation between the displacements required for 90% material degradation for the smallest and largest possible cross-sectional area, reading

$$d_\square = l_{beam} \cdot 0.003 \cdot (-0.01 \cdot A_{beam} + 1.01). \tag{7}$$

Here, A_{beam} denotes the cross-sectional area of the cantilever beam, computed as the sum of the areas of the elements 1 through 3. The displacement is applied in all three coordinate directions with the scaling factors $d_x = d_y = d_z \cdot 10$. It linearly increased from 0 to the maximum displacement $d_\square$ throughout 100 simulation time steps. An exemplary force–displacement relation is shown in Fig. 2(b), obtained from a simulation with the design parameters randomly chosen as $x_1 = x_2 = 6.0$ mm, $x_3 = y_1 = y_2 = 3.0$ mm and $y_3 = 1.0$ mm, according to their denomination in Fig. 2(a). The displacements are measured at the load introduction point, whereas the reaction forces are summed over the fixed boundary. To calculate f_{res}, the reaction forces in all 3 directions are aggregated, and the maximum is taken.

3. Results

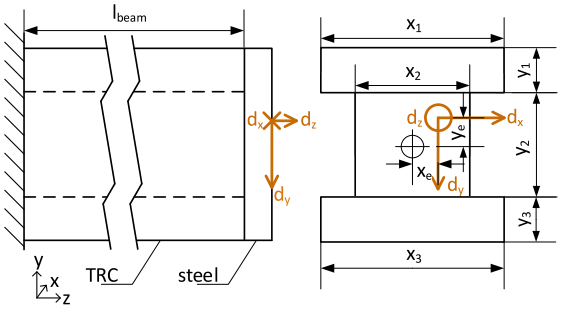
3.1. Dataset creation and optimization targets

The Sim-PAN dataset was created using Latin Hypercube Sampling (LHS) to extract 200 data points from the parameter space described in Table 1. This sample size was determined following the recommendations in the benchmarking study [21], which suggests that around 30 samples per feature are required to ensure robust performance for machine learning models in materials design. Latin hypercube sampling was chosen due to its proven robustness in various scenarios, as shown in [45]. In addition, a reduced dataset with 50 data points using LHS is created for comparative analysis with the work of Hanaoka [7]. The optimal performance of PAN fibers is characterized by a high WCA, maximum oil sorption capacity and high mechanical strength. Using Eq. (4), the objective σ_t served as the optimization target O_t, with the oil sorption capacity set as the first constraint, evaluated in 40 steps from 80 g/g to 200 g/g, while the WCA maintained a constant value of 160°, resulting in 40 individual SOO tasks. Similarly, for Sim-TRC, using LHS, 100 data points were selected from the parameter space described in Table 2. The reduction in dataset size compared to the benchmark recommendation of 30 samples per feature is justified by the nature of the design space, in which only natural numbers are valid, limiting the possible combinations, as shown in Eq. (6). In the simulation of the TRC cantilever beam, the optimization objective is to maximize the load-bearing capacity while minimizing the material consumption to reduce the environmental impact. Using Eq. (4), the objective f_{res} served as the optimization objective O_t, with the area A_{beam} serving as the constraint, which was evaluated in 74 steps from 7 to 80 mm^2, resulting in 74 individual SOO tasks.

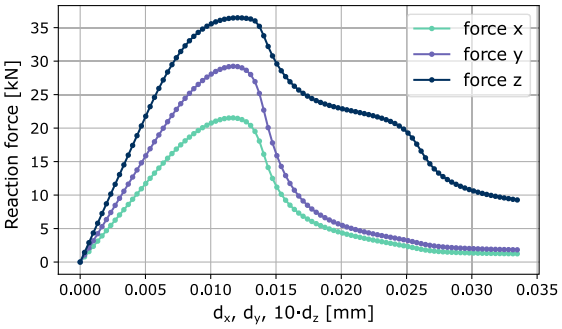
For comparing the evaluation strategies across varying data availability scenarios, additional dataset sizes ranging from 50 to 1000 data points were generated for the use cases using LHS and used for modeling and optimization.

3.2. ML model performance across evaluation strategies

The performance of all models for Sim-PAN with 200 and 50 data points and Sim-TRC with 100 data points is presented in Table 3, showing that Autosklearn performs best for these prediction tasks and evaluation strategies. CatBoost and XGBoost performed similarly, with CatBoost performing slightly better in most settings. In contrast, the FFNN consistently demonstrates the weakest performance. It only surpasses XGBoost in predicting σ_t in Sim-PAN when evaluated with the Pareto-front split strategy. Random splits consistently deliver the lowest errors across all models and prediction tasks, except Sim-PAN with 50 data points. They are followed in most cases by the Euclidean-distance



(a) Overview of the simulated cantilever beam with the 6 design parameters (x_1, x_2, x_3, y_1, y_2, y_3) and the applied displacements (d_z, d_y, d_x).



(b) Force-displacement relation for the simulation with the design parameters $x_1 = x_2 = 6$ mm; $x_3 = y_1 = y_2 = 3$ mm; $y_3 = 1$ mm.

Fig. 2. Illustration of the textile reinforced concrete (TRC) cantilever beam simulation.

split strategy. Conversely, the extrapolation and Pareto-front splitting strategies lead to higher prediction errors, with extrapolation having the highest RMSE in most cases. The standard deviation is only given for the random splitting. This is due to the ability to generate different splits based on the same dataset. In contrast, the other strategies use fixed allocation techniques that produce consistent splits at each iteration. The Pareto-front split, for example, ensures that the test set always contains the Pareto-optimal points, so variations in splits are not possible with this strategy.

To compare the evaluation strategies, a relative score is used to compare the prediction error assessed by the evaluation strategies with the prediction error in the optimized designs. This relative score is based on the RMSE and defined as

$$\mathrm{RMSE}_{\mathrm{rel}} = \mathrm{RMSE}_{\text{evaluation strategy}} / \mathrm{RMSE}_{\text{optimized designs}}. \tag{8}$$

A relative score of 1 indicates that the error estimated by the evaluation strategy matches the error observed in the optimized designs. A value of 2 means the evaluation strategy overestimates the error by a factor of two, while a value of 0.5 indicates that the evaluation strategy underestimates the error by half.

In the Table 3, highlighted in orange are the cases with $\mathrm{RMSE}_{\mathrm{rel}}$ below 0.5 and in blue for $\mathrm{RMSE}_{\mathrm{rel}}$ above 2. The RMSE for the optimized designs of Sim-PAN with 200 and 50 data points are displayed in Fig. 5(b) and for SIM-TRC with 100 data points in Fig. 6(b). For Sim-PAN with 50 data points, all splitting strategies fail to estimate the prediction accuracy of the ML models in the optimization. Excluding the Sim-PAN use case with 50 data points and comparing the frequency with which the splitting strategies overestimate or underestimate the error by more than a factor of 2, the following picture emerges. Random splitting underestimates the prediction error four times and never overestimates it. The Euclidean-distance split underestimates it twice and overestimates it three times. The extrapolation and Pareto-front split never underestimate it and overestimate it five times.

Fig. 3 compares the different evaluation strategies across all used models and datasets ranging from 50 to 1000 data points. For all dataset sizes, the $\mathrm{RMSE}_{\mathrm{rel}}$ shows a clear trend, with random splitting consistently yielding the lowest median $\mathrm{RMSE}_{\mathrm{rel}}$, followed by E-dist, Pareto-front and extrapolation, which yields the highest median. There are significant differences for the smallest data set size of 50 compared to the other data set sizes, with random having the median $\mathrm{RMSE}_{\mathrm{rel}}$ closest to 1. For all other data set sizes, random splitting has a median close to 0.5, which means that the error estimation of the random splitting is twice as low as the observed error in the optimized designs. The E-Dist split provides a significantly better estimate of the expected error than random splitting across all dataset sizes starting at 100. It has the best estimate of all strategies from a data set size of 200. Pareto-front splitting strategies generally overestimate the error. Extrapolation strongly overestimates the error in all cases with median $\mathrm{RMSE}_{\mathrm{rel}}$ between 2 and 3.5.

3.3. Performance of the optimizers

In order to compare the effectiveness of various optimization algorithms in finding optima from the objective function, a normalized performance metric is employed, see Eq. (9). This metric, derived by min–max normalization, scales the achieved optima to a range between 0 and 1. Here, 1 represents the best performance achieved by an optimizer for the specific underlying model at maximum budget, while 0 equates to a score of 0 in the objective function.

$$\text{normalized performance} = \frac{y_{\text{optimizer}}}{\max(y_{\text{all optimizers}})} \tag{9}$$

In Fig. 4, the normalized performance for all 40 optimization objectives of the Sim-PAN use case for the Simulation model, AutoSklearn and CatBoost is summarized. The simulation model shows the complexity of the optimization objective, as no optimal solution is found after 300 000 random runs, with a median normalized performance of 0.953. It is noteworthy that PSO shows weak performance and even underperforms random search. Moreover, with an increasing budget, PSO shows minimal performance improvement. CMA and DE are the most effective methods for these optimization problems, with CMA showing slightly better performance. In particular, CMA converges faster to the optimum and achieves a mean normalized score of 0.998 after 50 000 iterations compared to 0.967 for DE. In addition, there are fewer downward outliers for CMA for all budgets. Similar results can be observed for AutoSklearn, but here, the PSO has a higher median normalized performance than the random search. For CatBoost, both CMA and DE exhibit slower convergence rates, with DE showing no significant improvement after 100 000 iterations.

For the Sim-TRC use case, the investigation of the optimizer shows similar trends to Sim-PAN, as shown in Fig. A.2. Hence, CMA and DE are the best methods for optimizing these problems, with CMA showing slightly better performance. In addition, PSO shows only a slight increase in performance as the budget increases and is generally less efficient than the other methods.

3.4. Optimization performance of the ML models

A normalized performance metric is also used to compare the optimization performance of the ML models. This metric considers the optima already present within the dataset, see Eq. (10). The normalized performance is determined through min–max normalization, ranging from 1 to 0. In this scale, a value of 1 represents the Pareto optimal solution, while 0 denotes the best performance observed within the dataset.

$$\text{normed performance gain} = \frac{y - \max(y_{\text{dataset}})}{y_{\text{Pareto optimum}} - \max(y_{\text{dataset}})} \tag{10}$$

The Pareto-optimal solution is determined based entirely on the simulation results. Results from the simulation are always referred to as

Table 3
Performance of the ML models for the individual prediction tasks for the random (rand), Euclidean-distance (E-dist), extrapolation (extra) and Pareto-front (Pareto) splitting strategies. For the $RMSE_{rel}$ the standard deviation over 5 outer splits is given. Highlighted in orange if the $RMSE_{rel}$ is below 0.5 and in blue if the $RMSE_{rel}$ is above 2.

Use case	Task [unit]	Metric	AutoSklearn	CatBoost	XGBoost	FFNN	AutoSklearn
	Dataset size			200			50
Sim-PAN	WCA [°]	$RMSE_{rand}$	0.284 ± 0.315	**6.857 ± 0.980**	**8.03 ± 1.36**	11.5 ± 1.86	6.42 ± 4.52
		$RMSE_{E-dist}$	**0.310**	12.923	12.7	18.5	16.0
		$RMSE_{extra}$	2.15	19.71	20.14	**26.06**	29.06
		$RMSE_{Pareto}$	2.20	18.0	17.4	23.4	**4.00**
	S [g/g]	$RMSE_{rand}$	**1.22 ± 1.01**	**17.7 ± 1.23**	20.5 ± 1.41	27.1 ± 2.98	**11.88 ± 10.1**
		$RMSE_{E-dist}$	4.027	24.5	**28.5**	33.9	26.0
		$RMSE_{extra}$	35.64	53.08	60.23	**60.67**	55.02
		$RMSE_{Pareto}$	8.88	48.0	49.1	54.3	30.05
	σ_t [MPa]	$RMSE_{rand}$	0.06 ± 0.008	**1.47 ± 0.310**	1.86 ± 0.140	2.28 ± 0.614	**1.34 ± 1.63**
		$RMSE_{E-dist}$	0.285	2.35	**3.15**	**3.36**	1.71
		$RMSE_{extra}$	0.594	3.70	4.27	4.69	0.743
		$RMSE_{Pareto}$	**0.582**	4.44	4.63	4.55	0.762
	Dataset size			100			
Sim-TRC	f_{res} [kN]	$RMSE_{rand}$	3.49 ± 0.89	5.40 ± 0.92	6.57 ± 1.65	**11.73 ± 0.94**	
		$RMSE_{E-dist}$	6.62	**9.08**	10.08	16.22	
		$RMSE_{extra}$	9.14	8.67	8.33	21.13	
		$RMSE_{pareto}$	**6.14**	6.73	**9.06**	18.11	

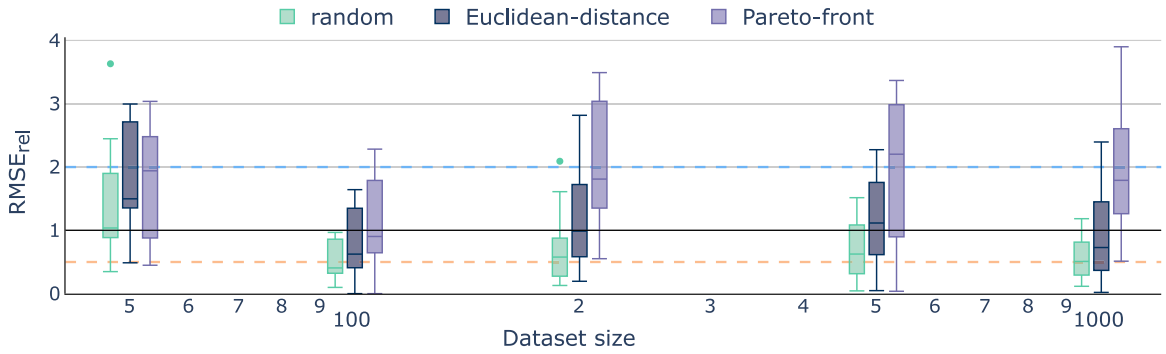


Fig. 3. $RMSE_{rel}$ of the evaluation strategies for all prediction tasks in relation to the dataset size. All evaluation strategies are employed on the same dataset size marked by E-dist, with the variants displayed next to each other. Outliers and the evaluation strategy extrapolation were excluded for better visualization. Marked with the dashed orange line is a $RMSE_{rel}$ of 0.5 and the blue dashed line with $RMSE_{rel}$ of 2.

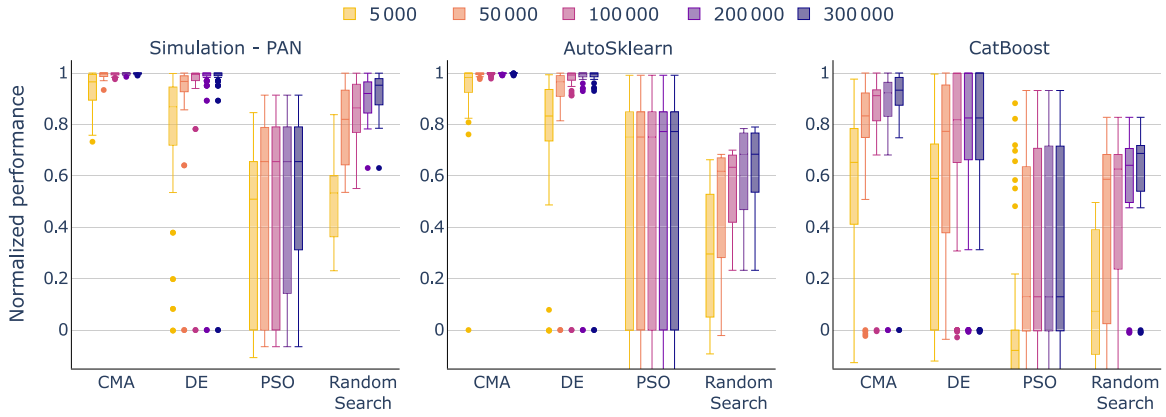
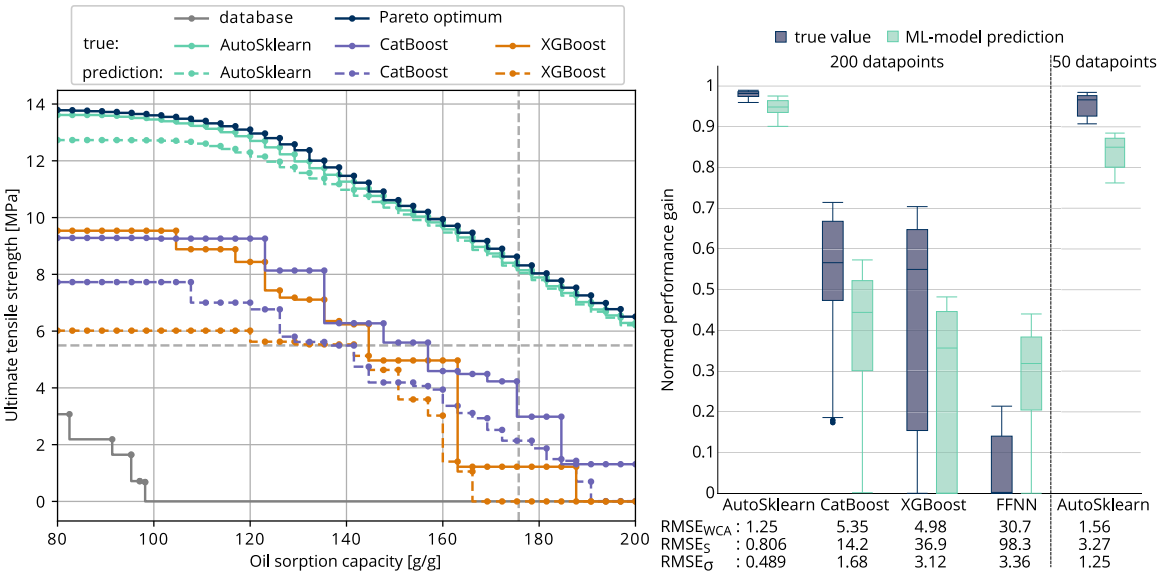


Fig. 4. Comparison of the optimizer on normalized performance for the Sim-PAN use case.

true values in the following. Conversely, the values predicted by the ML models are always referred to as predicted values.

The results for the Sim-PAN use case are presented in Fig. 5. The Pareto optimum shown in Fig. 5(a) was determined by a rigorous optimization process in which the simulation was optimized with CMA at 300 000 iterations for each configured SOO (see Section 2.3). This thorough approach ensures that the Pareto optimum is adequately found. The Pareto front predicted by various machine learning models, the true values of these designs, and the training dataset are also shown. Additionally, Fig. 5(b) shows the normalized performance gain across all SOO steps for both the predicted and true optimized material designs, along with the RMSE between them.

Compared to the existing data points, the tree-based models and AutoSklearn exhibit significantly enhanced designs, as depicted in Fig. 5(a). AutoSklearn, in particular, significantly outperforms the other models and generates new material designs that come close to the Pareto-optimal solution with a budget of only 200 data points. The predicted solutions achieved a normalized performance gain of 0.95, while the true designs scored even higher with 0.98. The smallest error between predicted and true values for new designs underlines



(a) Pareto front of the optimization task, maximizing tensile strength and oil sorption capacity with WCA $\geq 160°$. The grey dashed lines show the maximum S and σ_t in the dataset without WCA constrain.

(b) Normed performance gain of the different ML models.

Fig. 5. Results of Sim-PAN optimization, based on 40 SOO. Pareto optimal solutions were achieved with 300 000 iterations of CMA optimizer.

the superiority of AutoSklearn. All designs achieved the true value for the WCA of at least 160°. Among the tree-based methods, CatBoost and XGBoost perform similarly. On the Pareto front of the true values for the new designs, CatBoost performs better in some areas and XGBoost in others. CatBoost's true values reached a normalized performance of 0.57, compared to 0.55 for XGBoost, with XGBoost showing more downward outliers. The predicted values show a similar behavior. All CatBoost designs achieved a true WCA of at least 158.0°, while XGBoost designs achieved at least 156.5°. FFNN were omitted in Fig. 5(a) as they cannot generate designs that fulfill the predefined conditions of WCA $\geq$ 160°. However, Fig. 5(b) illustrates that both predicted and actual designs failed to achieve satisfactory performance even without the WCA restriction. Overall, AutoSklearn performs by far the best in this use case. CatBoost and XGBoost also perform well, with CatBoost slightly outperforming XGBoost.

The results for the Sim-TRC use case are depicted in Fig. 6. The Pareto optimum shown in Fig. 6(a) was determined by an extensive optimization of the simulation using Bayesian optimization with 300 iterations for each configured SOO (see Section 2.3) to ensure the Pareto optimum is reached. In addition, the figure shows the Pareto front of the true values of the material designs found using the ML models, as well as the training dataset itself. The ML model predictions were not included for visualization purposes and can be found in Fig. A.4. Additionally, Fig. 6(b) shows the normalized performance gain across all SOO steps for both the predicted and true optimized material designs, along with the RMSE between them.

Compared to the existing data points, the tree-based models and AutoSklearn demonstrate significantly improved designs, as shown in Fig. 6(a). Among these, AutoSklearn provides the best-optimized designs with the predicted solutions achieving a median normalized performance gain of 1.0, while the true values of the designs achieve a median of 0.89, as shown in Fig. 6(b). AutoSklearn also exhibited the smallest prediction error between predicted and true values for new designs, with an RMSE of 5.27 for f_{res}. For the tree-based methods, CatBoost and XGBoost performed similarly, achieving a median normalized performance gain of around 1.1 for the predicted values. However, CatBoost outperformed XGBoost in the normed performance gain of the true designs, achieving a median value of 0.80 compared to XGBoost's 0.47. The RMSE for f_{res} between predictions and true values remained close, with 9.61 for CatBoost and 9.18 for XGBoost. As in the Sim-PAN use case, FFNN exhibited the lowest performance, with a median normed performance gain for the true values of −0.35. This reveals that FFNN-optimized designs often performed worse than those already in the dataset. This can also be seen in Fig. 6(a), where the designs optimized with FFNN are mainly below the best values of the existing dataset. Nevertheless, for small areas up to 30 mm^2 and large areas over 70 mm^2, FFNN found better designs than those in the dataset and even reached the Pareto optimum for 6 cases. Despite this, FFNN showed the largest error between predicted and true values for the newly found designs. Overall, the performance ranking of the ML models here remains identical to the Sim-PAN use case, although the advantage of Autosklearn is considerably smaller.

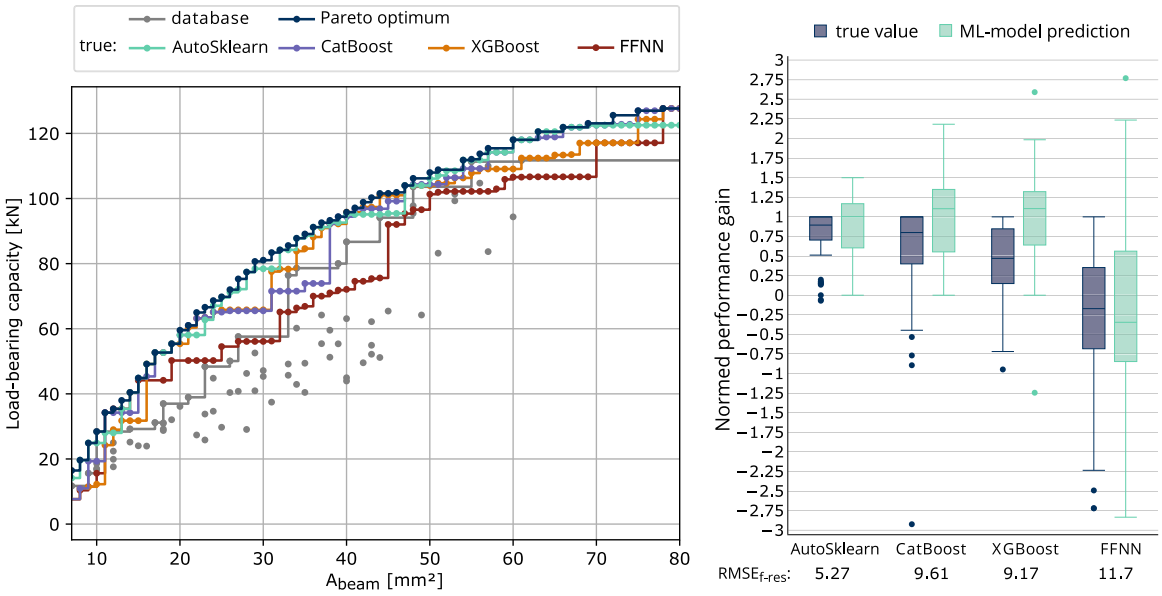
The Sim-PAN and Sim-TRC use cases illustrate the potential of data-driven multi-objective optimization for improving material designs. AutoSklearn was characterized by the fact that it generated designs close to the Pareto front with minimal data. CatBoost and XGBoost performed well, with CatBoost slightly ahead. FFNN performed poorly in comparison.

4. Discussion

4.1. General performance

The workflow demonstrates strong overall performance, yielding good-performing designs even with limited resources. The solutions notably surpass the best designs within the dataset and often approach Pareto-optimal designs, achieved only after numerous optimization iterations in the simulations.

In the Sim-PAN scenario, the optimization challenge becomes apparent with the random optimizer. Even after 300 000 iterations, the Pareto optimum is only found in some optimization tasks. Thus, even an extensive set of 300 000 randomly selected experiments can hardly find Pareto-optimal solutions. In contrast, the presented workflow with AutoSklearn and a database of only 50 experiments surpasses 300 000 random experiments. Comparatively, in a study by Hanaoka et al. [7], only 20 experiments were required to achieve a material design meeting the predefined targets of WCA = 160.0°, S = 100.0 g/g and σ_t = 8.0 MPa. In our approach, with just 50 experiments, our optimization enables exploration in all directions. Thus, a design can be found that achieves WCA = 161°, S = 102 g/g and σ_t = 13.4 MPa, as well as a



(a) Pareto front of the optimization task, maximizing load-bearing capacity f_{res} and minimizing area A_{beam}.

(b) Normed performance gain of the different ML models.

Fig. 6. Results of Sim-PTRC optimization, based on 74 SOO. Pareto optimal solutions were achieved with 300 iterations of Bayesian optimizer.

design that achieves WCA = 160°, S = 173 g/g and σ_t = 8.2 MPa. Moreover, designs can be found for any weighting of the target parameters, compare Fig. 5(a). The presented workflow demonstrates the ability to optimize in all directions without necessitating additional experiments.

In the Sim-TRC use case, the complexity of the problem becomes evident as even 10 000 random iterations fail to reach the optimum in all cases. Our workflow with AutoSklearn and a database of just 100 experiments successfully identifies better designs than those in the original dataset in 72 out of 74 cases. Notably, in 15 of the 74 cases, it finds the Pareto optimal solution, as shown in Fig. 6(a).

This study is limited to simulations, but the methodology holds significant potential for application to experimental datasets. Simulations were crucial for exploring and comparing various ML models, optimization algorithms, and evaluation strategies within a controlled environment while ensuring manageable resource demands. They provide an ideal platform for refining methods in a reproducible and accessible manner. The code for reproducing these simulations is available to support future research and enable cross-study comparisons. Future work could apply this methodology to experimental datasets, enabling more comprehensive validation of the performance in experiment-driven research.

4.2. Comparison of optimizers

In this study, evolutionary algorithms are the most powerful for optimizing ML models, with CMA showing superior performance compared to DE. In Sim-PAN, CMA consistently identifies higher optima, while DE occasionally has difficulties finding designs that meet all constraints. For Sim-TRC, no optimizer can find the optimum even with more than 1000 iterations. Therefore, Bayesian optimization is used for the direct optimization of the Sim-TRC due to the lengthy computing time of the Sim-TRC between 2 and 12 CPU-hours. In contrast to the other methods, Bayesian optimization invests a lot of computing resources to determine the next candidate to arrive at an optimal solution with fewer iterations [46]. Although this trade-off can be less favorable for fast ML models, it becomes advantageous for expensive simulations like Sim-TRC. There is a break-even point where the higher cost per iteration of Bayesian optimization becomes more cost-effective compared to faster optimizers, such as CMA, which require more iterations. This point is unknown beforehand but tends to favor rapid optimizers and more iterations for fast ML models. Fig. A.3 illustrates this by comparing the mean normalized performance of AutoSklearn optimization across all Sim-PAN tasks over computation time, using CMA with 300 000 iterations and Bayesian optimization with 5000 iterations. Despite CMA using 60 times more iterations than Bayesian optimization, both methods have similar total computation times. CMA shows higher overall performance, highlighting its superior effectiveness with fast-computing ML models.

Although swarm-based algorithms have shown exemplary performance in the literature, they do not perform well in this study. Often, they do not find a solution that fulfills the constraints but only comes very close. The swarm-based algorithms get trapped in a local minimum using the optimization function presented here. As a result, they often perform worse than random search in this study.

Notably, all optimization strategies converge slower with tree-based methods than with the other models. The prediction behavior of the tree-based methods possibly hinders optimization. Data points located close to each other can be sorted into the same leaves in all trees, resulting in the same prediction result. As a result, there is no gradient and, therefore, no direction for the optimization algorithm to search for better solutions. This difference is particularly pronounced in the Sim-PAN use case but is also present in Sim-TRC. Since only natural numbers are allowed in Sim-TRC, the minimum distance between 2 possible points is greater, thus decreasing the likelihood that the same value is predicted for two neighboring points with tree-based models.

4.3. Comparison of ML models

AutoSklearn found the models with the smallest prediction error, demonstrating the superiority of AutoML for small tabular data in materials design, which is also found in a previous study [21]. Moreover, AutoSklearn can provide the best material designs for both use cases.

In the Sim-PAN use case, AutoSklearn selected an ensemble of support vector machines and Gaussian process model as the optimal choice for predicting water contact angle and oil sorption capacity and a support vector machine for predicting tensile strength. Meanwhile, in the Sim-TRC use case, AutoSklearn identified an ensemble of Gaussian process model and Automatic Relevance Determination (ARD) Regression as the best-performing model.

The tree-based models used, XGBoost and CatBoost, can solve optimization tasks in the dataset's range, which is underlined by their good performance in the Sim-TRC use case. Optimization far beyond the known space, as with Sim-PAN, is more difficult for these models, where they perform significantly worse than AutoSklearn. Although tree-based models are generally effective, they reach their limits regarding extrapolation, as emphasized and vividly illustrated by Numata and Tanaka [47]. Regression trees are limited to the range of the training data and cannot provide results beyond the highest or lowest data points. Extreme Gradient Boosting in CatBoost and XGBoost allows predictions beyond the training label range by using gradients rather than the actual labels during training. However, this extrapolation is only feasible in high-dimensional spaces and remains limited overall. As soon as the optimal combination of the last leaves in the trees is reached, further increases or decreases in the prediction are no longer possible. This extrapolation limitation directly affects optimization results. For Sim-PAN, XGBoost cannot predict σ_t higher than 6 MPa and S exceeding 166 g/g, as seen in Fig. 5(a) the model predictions are cut off at these values. CatBoost encounters similar issues, with maximum predictions of σ_t at 7.8 MPa and S at 190 g/g. The true values exceed the model's predicted ranges, slightly improving their usability but leading to significant errors between predictions and actual results. One potential strategy to overcome these limitations is adapting tree-based models for better extrapolation capabilities. Quinlan [48] introduced the M5 model tree, which uses linear functions in the leaves of the tree to enable extrapolation. This approach is extended by Numata and Tanaka [47] for tree-based ensembles by a probabilistic determination of the threshold values for the leaf splitting during tree creation. In this way, randomness is introduced into the threshold selection and smooth prediction curves are possible, even in regions with sparse data. Although gradient boosting with classification and regression trees (CART) has become standard in many GBM libraries, improving these techniques to facilitate extrapolation for optimization tasks seems promising.

Autosklearn does not use tree-based methods in the use cases shown here, so it does not have the extrapolation problem of XGBoost and CatBoost. However, since tree-based methods are often effective across various settings, AutoSklearn frequently utilizes them [21]. For better extrapolation in optimization tasks, AutoSklearn's model space can be restricted. For this purpose, it is recommended that tree-based models be excluded from the model space during AutoSklearn training, or even more restrictively, model selection can be limited to SVM, Gaussian Process, and ARD Regression. They have shown excellent optimization performance in this study and can generalize beyond the training data more effectively. SVM also showed the second-best performance in the benchmark of Shmuel et al. [49], but behind an unrestricted AutoML framework. So, excluding tree-based methods may enhance extrapolation capabilities, but it could compromise the model's performance within the data's original range, where tree-based models tend to excel. Therefore, choosing to exclude them should be carefully considered based on the characteristics of the practical use case.

The poor performance of FFNN could be due to the relatively small and tabular datasets in this study, as FFNN generally perform poorly on small datasets and especially struggle with tabular datasets, as observed by Grinsztajn et al. [37].

4.4. Comparison of evaluation strategies

In this study, the order in which the errors of the ML models were observed during training consistently corresponds to the performance of the true values of the proposed designs. The necessity of the nested cross-validation is underlined by the partly high standard deviations in the random split. Thus, the random split is essential for each analysis, as the other splitting strategies do not allow such validation. The other splitting strategies must be applied to a randomly reduced dataset to generate other train test splits. Otherwise, every split would be equal. However, the difference between the size of the training dataset and the application would be more significant, distorting the statements on model accuracy after training for the application. However, the difference between the size of the training dataset during evaluation and retraining with the full data set for the optimization would be more significant, distorting the statements on model accuracy in the evaluation.

All splitting methods face limitations when applied to the smallest datasets with only 50 data points. With such a data set, an 80/20 split of training and test leaves only 10 points for the test, which severely limits an accurate performance assessment, especially without nested cross-validation. This difficulty is also reflected in the high standard deviation for different random splits, which shows the high variability with small test datasets. Therefore, a random split is recommended for such small data sets to allow nested cross-validation.

For larger datasets, starting from 100 data points, the random splitting method has difficulties accurately estimating the expected errors for optimized designs. This is because the test set does not reflect extrapolation into unknown areas. In 80% of this study's prediction tasks with datasets of at least 100 data points, random splitting underestimates the model error, assuming a model accuracy that cannot be achieved in the optimization. This discrepancy between expected and actual model accuracy is occasionally extensive, as shown by the prediction of σ_t in Sim-PAN using AutoSklearn, where an RMSE of 0.06 ± 0.008 contrasts sharply with the actual error of 0.489, which is eight times higher.

The alternative splitting strategies underestimate errors less frequently but occasionally overestimate them. One reason for overestimation is that, when retraining on the entire dataset, models for the optimization process had 20% more training data. In addition, not all objectives may be exhausted to their limits in multi-objective optimization. For example, material designs with a minimum WCA of 160° were searched for in the Sim-PAN use case, while the dataset provides values between 85.5° and 178.1°. Thus, the extrapolation splitting on the WCA label examines areas irrelevant to the optimization in this case.

No splitting strategy can deliver the best results for all cases, so a combination is recommended. The random split should be used to estimate the model's stability, and the Euclidean-distance or Pareto-front split should be used to estimate the prediction accuracy for the optimized designs. The Euclidean-distance split tends to be more accurate but risks underestimating the error. In contrast, the Pareto front split is more restrained, and the upper limit of the error can be estimated. Notably, the Euclidean-distance split performs particularly well for datasets of 200 or more data points, where the Pareto-front split significantly overestimates the expected error.

5. Conclusion and outlook

In summary, two simulation use cases were used to demonstrate the performance of ML model-based materials design optimization. For this purpose, four different ML models and four optimization algorithms were compared, and four evaluation strategies were used to assess the model's performance. The observations show the following three points:

- AutoSklearn is the best-performing model, consistently exhibiting the lowest prediction error and the best optimization capabilities.
- The covariance matrix adaptation evolution strategy proved to be a highly effective approach in tackling the inverse material design problem using ML models.
- Introducing novel evaluation strategies in training ML models provides a more robust evaluation of the expected model accuracy in optimization. Combining random splitting for stability estimation and splits based on Euclidean distance between data points, respectively, the Pareto-front in the dataset provides a balanced approach that avoids overestimating the expected performance.

Overall, these results underline how robust the presented workflow is in identifying new optimized material designs.

Based on these results, three main directions for future research emerge. First, adapting the training process of the ML models by using the proposed data splits in the inner splits of the nested cross-validation seems promising. Doing so could increase the prediction accuracy of the optimized designs and thus bring the performance of the found designs even closer to the Pareto optimum. Furthermore, the workflow should be transferred from the simulated test environments to actual experimental datasets for validation and usability in real-world applications. Simulations allowed for a comprehensive comparison of different strategies. Applying the workflow to real-world data will offer a more robust evaluation of its practicality for materials design and opening pathways for highly optimized and resource-saving material designs. In addition, as sustainability considerations become increasingly crucial in materials design, future research efforts should aim to extend the current workflow to encompass additional sustainability metrics. These could include those defined by life cycle assessment analyses, aiming to achieve optimal solutions from the perspectives of required material performance and environmental friendliness.

CRediT authorship contribution statement

Felix Conrad: Writing – original draft, Visualization, Software, Methodology, Investigation, Conceptualization. **Julien Philipp Stöcker:** Writing – original draft, Software, Methodology. **Cesare Signorini:** Writing – review & editing, Project administration, Conceptualization. **Isabela de Paula Salgado:** Writing – review & editing, Conceptualization. **Hajo Wiemer:** Writing – review & editing, Supervision, Funding acquisition. **Michael Kaliske:** Writing – review & editing, Supervision, Funding acquisition. **Steffen Ihlenfeldt:** Writing – review & editing, Supervision, Funding acquisition.

Declaration of competing interest

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

Data availability

The data and code used in this study are available on GitHub: https://github.com/fc-tud/material-optimization.

Acknowledgments

The financial support of the German Research Foundation (Deutsche Forschungsgemeinschaft, DFG) within the Research Training Group (Graduiertenkolleg, GRK) 2250 “Mineral- bonded composites for enhanced structural impact safety” (grant nr. 287321140) and the transregional collaborative research center (SFB/TRR) 339, Sub-Project B04 (grant nr. 453596084), as well as the financial support of the Federal Ministry of Education and Research (BMBF) within project KIOptiPack grant nr. 033KI129 is gratefully acknowledged.

Appendix A

A.1. Hyperparameter space

Table A.1
Hyperparameter space for the tunning of CatBoost models.

Hyperparameter	Min	Max	Type
Number estimators	250	1500	Integer
Maximum depth	2	10	Integer
Learning rate	0.001	1	Float, log
Bagging temperature	0	100	Float
L2 regularization	0	100	Float

Table A.2
Hyperparameter space for the tunning of XGBoost models.

Hyperparameter	Min	Max	Type
Number estimators	250	1500	Integer
Maximum depth	2	10	Integer
Learning rate	0.001	1	Float, log
L1 regularization	0	100	Float
L2 regularization	0	100	Float

Table A.3
Hyperparameter space for the tunning of the FFNN.

Hyperparameter	Min	Max	Type
Number hidden layers	1	3	Integer
Neurons layer 1	5	35	Integer
Neurons layer 2	5	35	Integer
Neurons layer 3	5	35	Integer
Optimizer	[“Adam”, “RMSprop”, “SGD”]		String
Learning rate	1*e−5	1*e−1	Log
Step size	10	1000	Integer
Gamma	0.1	1	Float
Weigth decay	1*e−7	1*e−2	Log
Dropout	0	0.5	Float
Epochs	50	1000	Integer

A.2. Material parameters in textile-reinforced concrete simulation

Table A.4
Cantilever beam material parameters.

Parameter		TRC	Steel
κ	[MPa]	17 220.0	210 000.0
μ	[MPa]	12 917.0	140 000.0
k_r	[–]	8.182	–
e^{mic}	[MPa]	0.00	–
g^{mic}	[MPa]	0.00	–
A	[–]	[0,0,1]	–
B	[–]	[0,1,0]	–
α	[–]	0.98	–
β	[–]	1000.0	–
γ_0	[–]	2.8E−4	–
c	[–]	1.0	–

A.3. Splitting strategies

(see Fig. A.1.)

A.4. Optimizer comparison

(see Figs. A.2 and A.3.)

A.5. Model comparison

(see Fig. A.4.)

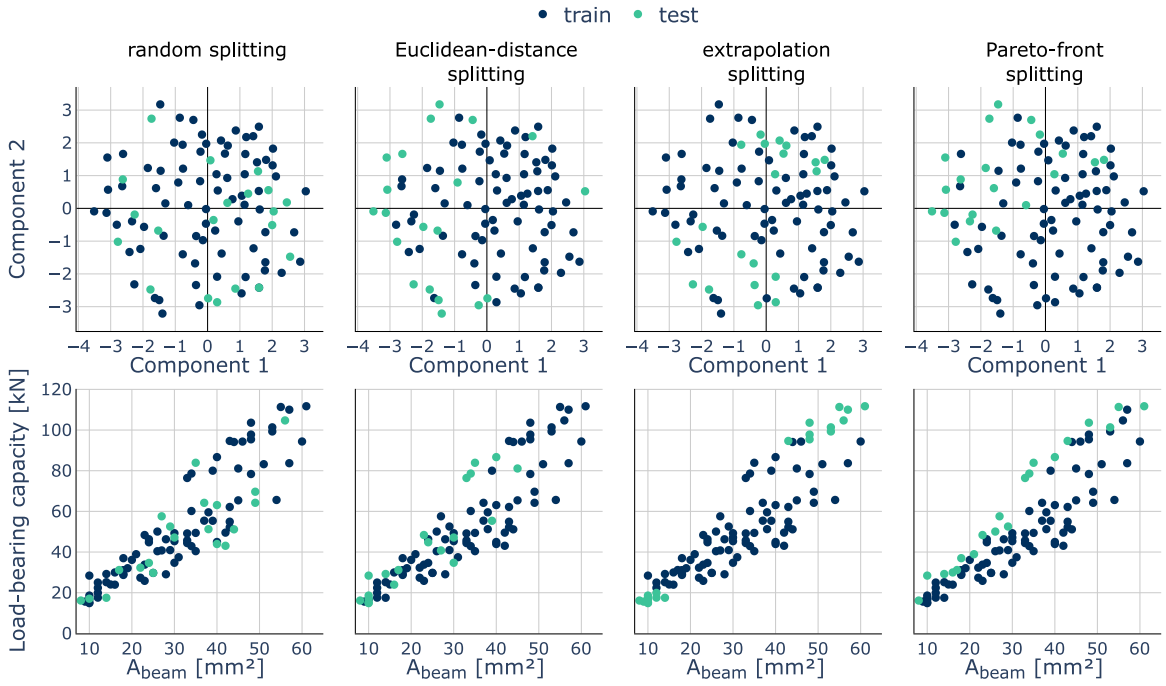


Fig. A.1. Comparison of the 4 splitting strategies for the SIM-TRC data set in two-dimensional PCA space and for the target parameters maximum load-bearing capacity and area.

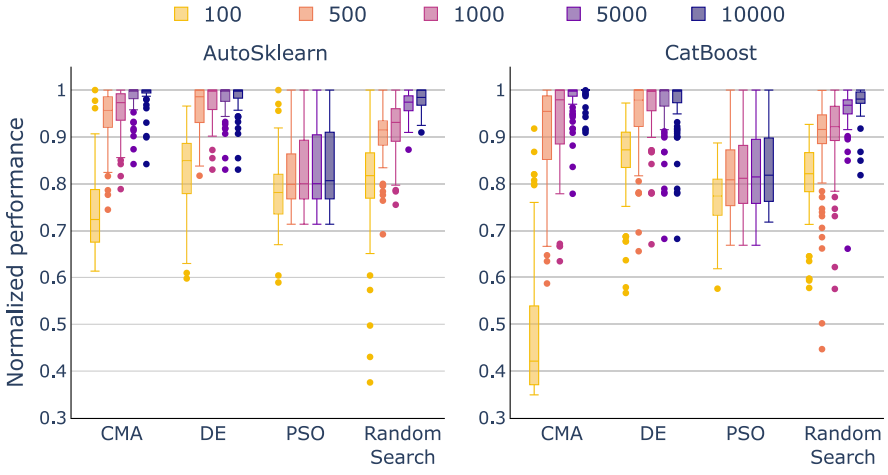


Fig. A.2. Comparison of optimizer for AutoSklearn and CatBoost in the use case Sim-TRC.

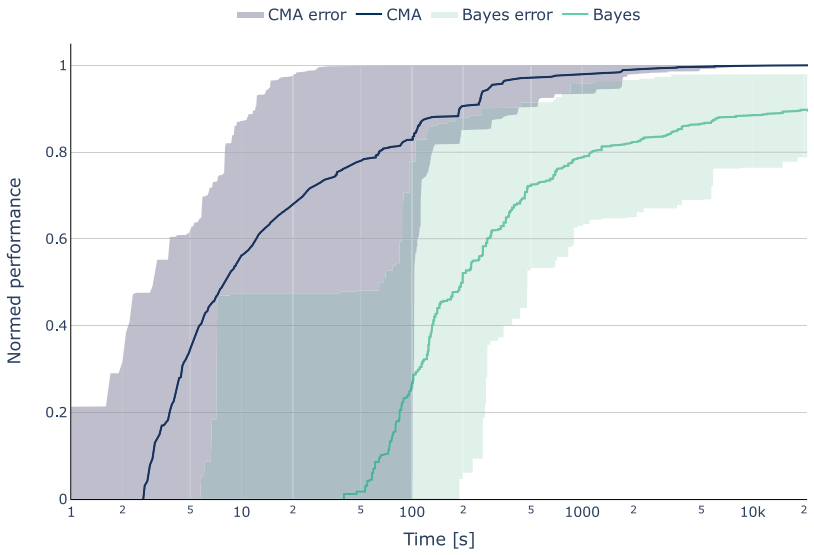


Fig. A.3. Comparison of CMA with 300 000 iterations and Bayesian optimization with 1000 iterations for optimizing AutoSklearn across all tasks within the SIM-PAN use case over time.

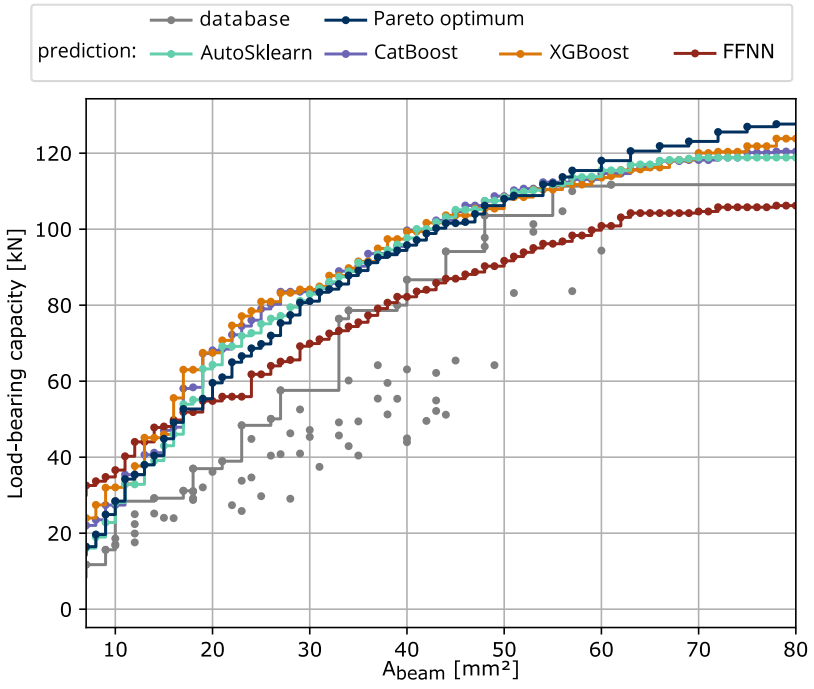


Fig. A.4. Paretofront of the model predictions of the optimization task, achieved through 74 SOO.

References

[1] D. Morgan, R. Jacobs, Opportunities and challenges for machine learning in materials science, Annu. Rev. Mater. Res. 50 (1) (2020) 71–103, http://dx.doi.org/10.1146/annurev-matsci-070218-010015.

[2] K. Guo, Z. Yang, C.-H. Yu, M. J. Buehler, Artificial intelligence and machine learning in design of mechanical materials, Mater. Horiz. 8 (4) (2021) 1153–1172, http://dx.doi.org/10.1039/D0MH01451F.

[3] M. Mohtasham Moein, A. Saradar, K. Rahmati, S.H. Ghasemzadeh Mousavinejad, J. Bristow, V. Aramali, M. Karakouzian, Predictive models for concrete properties using machine learning and deep learning approaches: A review, J. Build. Eng. 63 (2023) 105444, http://dx.doi.org/10.1016/j.jobe.2022.105444.

[4] F. Kazemi, T. Shafighfard, D.-Y. Yoo, Data-driven modeling of mechanical properties of fiber-reinforced concrete: a critical review, Arch. Comput. Methods Eng. (2024) http://dx.doi.org/10.1007/s11831-023-10043-w.

[5] T.B. Martin, D.J. Audus, Emerging trends in machine learning: a polymer perspective, ACS Polym. Au 3 (3) (2023) 239–258, http://dx.doi.org/10.1021/acspolymersau.2c00053.

[6] J. Wang, Y. Wang, Y. Chen, Inverse design of materials by machine learning, Materials 15 (5) (2022) 1811, http://dx.doi.org/10.3390/ma15051811.

[7] K. Hanaoka, Bayesian optimization for goal-oriented multi-objective inverse material design, iScience 24 (7) (2021) 102781, http://dx.doi.org/10.1016/j.isci.2021.102781.

[8] C. Sun, K. Wang, Q. Liu, P. Wang, F. Pan, Machine-learning-based comprehensive properties prediction and mixture design optimization of ultra-high-performance concrete, Sustainability 15 (21) (2023) 15338, http://dx.doi.org/10.3390/su152115338.

[9] B. Wang, J. Cai, C. Liu, J. Yang, X. Ding, Harnessing a novel machine-learning-assisted evolutionary algorithm to co-optimize three characteristics of an electrospun oil sorbent, ACS Appl. Mater. Interfaces 12 (38) (2020) 42842–42849, http://dx.doi.org/10.1021/acsami.0c11667, Publisher: American Chemical Society.

[10] Y. Diao, L. Yan, K. Gao, A strategy assisted machine learning to process multi-objective optimization for improving mechanical properties of carbon steels, J. Mater. Sci. Technol. 109 (2022) 86–93, http://dx.doi.org/10.1016/j.jmst.2021.09.004.

[11] J. Zhang, Y. Huang, Y. Wang, G. Ma, Multi-objective optimization of concrete mixture proportions using machine learning and metaheuristic algorithms, Constr. Build. Mater. 253 (2020) 119208, http://dx.doi.org/10.1016/j.conbuildmat.2020.119208.

[12] J. Zhang, Y. Huang, F. Aslani, G. Ma, B. Nener, A hybrid intelligent system for designing optimal proportions of recycled aggregate concrete, J. Clean. Prod. 273 (2020) 122922, http://dx.doi.org/10.1016/j.jclepro.2020.122922.

[13] J. Zhang, Y. Huang, G. Ma, B. Nener, Mixture optimization for environmental, economical and mechanical objectives in silica fume concrete: A novel framework based on machine learning and a new meta-heuristic algorithm, Resour. Conserv. Recy. 167 (2021) 105395, http://dx.doi.org/10.1016/j.resconrec.2021.105395.

[14] Y. Huang, J. Zhang, F. Tze Ann, G. Ma, Intelligent mixture design of steel fibre reinforced concrete using a support vector regression and firefly algorithm based multi-objective optimization model, Constr. Build. Mater. 260 (2020) 120457, http://dx.doi.org/10.1016/j.conbuildmat.2020.120457.

[15] X. Feng, Z. Wang, L. Jiang, F. Zhao, Z. Zhang, Simultaneous enhancement in mechanical and corrosion properties of Al-Mg-Si alloys using machine learning, J. Mater. Sci. Technol. 167 (2023) 1–13, http://dx.doi.org/10.1016/j.jmst.2023.04.072.

[16] Y. Motoyama, R. Tamura, K. Yoshimi, K. Terayama, T. Ueno, K. Tsuda, Bayesian optimization package: PHYSBO, Comput. Phys. Comm. 278 (2022) 108405, http://dx.doi.org/10.1016/j.cpc.2022.108405.

[17] E.M. Golafshani, A. Behnood, T. Kim, T. Ngo, A. Kashani, A framework for low-carbon mix design of recycled aggregate concrete with supplementary cementitious materials using machine learning and optimization algorithms, Structures 61 (2024) 106143, http://dx.doi.org/10.1016/j.istruc.2024.106143.

[18] E. Asadi Shamsabadi, M. Salehpour, P. Zandifaez, D. Dias-da-Costa, Data-driven multicollinearity-aware multi-objective optimisation of green concrete mixes, J. Clean. Prod. 390 (2023) 136103, http://dx.doi.org/10.1016/j.jclepro.2023.136103.

[19] K. Stergiou, C. Ntakolia, P. Varytis, E. Koumoulos, P. Karlsson, S. Moustakidis, Enhancing property prediction and process optimization in building materials through machine learning: A review, Comput. Mater. Sci. 220 (2023) 112031, http://dx.doi.org/10.1016/j.commatsci.2023.112031.

[20] Y. Zhang, C. Ling, A strategy to apply machine learning to small datasets in materials science, NPJ Comput. Mater. 4 (1) (2018) 1–8, http://dx.doi.org/10.1038/s41524-018-0081-z, Number: 1 Publisher: Nature Publishing Group.

[21] F. Conrad, M. Mälzer, M. Schwarzenberger, H. Wiemer, S. Ihlenfeldt, Benchmarking AutoML for regression tasks on small tabular data in materials design, Sci. Rep. 12 (1) (2022) 19350, http://dx.doi.org/10.1038/s41598-022-23327-1.

[22] S. Mohamed, B. Lakshminarayanan, Learning in implicit generative models, 2017, http://dx.doi.org/10.48550/arXiv.1610.03483.

[23] M. Mirza, S. Osindero, Conditional generative adversarial nets, 2014, http://dx.doi.org/10.48550/arXiv.1411.1784.

[24] D. Bouchacourt, P.K. Mudigonda, S. Nowozin, DISCO nets : Dissimilarity coefficients networks, in: Advances in Neural Information Processing Systems, Vol. 29, Curran Associates, Inc., 2016.

[25] T. Sanchez, B. Caramiaux, P. Thiel, W.E. Mackay, Deep learning uncertainty in machine teaching, in: 27th International Conference on Intelligent User Interfaces, IUI '22, Association for Computing Machinery, 2022, pp. 173–190, http://dx.doi.org/10.1145/3490099.3511117.

[26] C.E. Rasmussen, Gaussian processes in machine learning, in: O. Bousquet, U. von Luxburg, G. Rätsch (Eds.), Advanced Lectures on Machine Learning: ML Summer Schools 2003, Canberra, Australia, February 2 - 14, 2003, Tübingen, Germany, August 4 - 16, 2003, Revised Lectures, in: Lecture Notes in Computer Science, Springer, 2004, pp. 63–71, http://dx.doi.org/10.1007/978-3-540-28650-9_4.

[27] S.P. Vasseur, J.L. Aznarte, Comparing quantile regression methods for probabilistic forecasting of NO2 pollution levels, Sci. Rep. 11 (1) (2021) 11592, http://dx.doi.org/10.1038/s41598-021-90063-3.

[28] D.H. Wolpert, The Lack of a priori distinctions between learning algorithms, Neural Comput. 8 (7) (1996) 1341–1390, http://dx.doi.org/10.1162/neco.1996.8.7.1341.

[29] Z. del Rosario, M. Rupp, Y. Kim, E. Antono, J. Ling, Assessing the frontier: Active learning, model accuracy, and multi-objective candidate discovery and optimization, J. Chem. Phys. 153 (2) (2020) 024112, http://dx.doi.org/10.1063/5.0006124.

[30] P. Zador, Asymptotic quantization error of continuous signals and the quantization dimension, IEEE Trans. Inform. Theory 28 (2) (1982) 139–149, http://dx.doi.org/10.1109/TIT.1982.1056490.

[31] V.R. Joseph, A. Vakayil, SPlit: an optimal method for data splitting, Technometrics (2021) 1–11, http://dx.doi.org/10.1080/00401706.2021.1921037.

[32] F. Conrad, E. Boos, M. Mälzer, H. Wiemer, S. Ihlenfeldt, Impact of data sampling on performance and robustness of machine learning models in production engineering, in: M. Liewald, A. Verl, T. Bauernhansl, H.-C. Möhring (Eds.), Production at the Leading Edge of Technology, in: Lecture Notes in Production Engineering, Springer International Publishing, 2023, pp. 463–472, http://dx.doi.org/10.1007/978-3-031-18318-8_47.

[33] H. Wiemer, D. Schneider, V. Lang, F. Conrad, M. Mälzer, E. Boos, K. Feldhoff, L. Drowatzky, S. Ihlenfeldt, Need for UAI–anatomy of the paradigm of usable artificial intelligence for domain-specific AI applicability, Multimodal Technol. Interact. 7 (3) (2023) 27, http://dx.doi.org/10.3390/mti7030027.

[34] M. Feurer, A. Klein, K. Eggensperger, J. Springenberg, M. Blum, F. Hutter, Efficient and robust automated machine learning, in: C. Cortes, N.D. Lawrence, D.D. Lee, M. Sugiyama, R. Garnett (Eds.), Advances in Neural Information Processing Systems 28, Curran Associates, Inc., 2015, pp. 2962–2970, http://dx.doi.org/10.5555/2969442.2969547.

[35] L. Prokhorenkova, G. Gusev, A. Vorobev, A.V. Dorogush, A. Gulin, CatBoost: Unbiased boosting with categorical features, 2019, http://dx.doi.org/10.48550/arXiv.1706.09516.

[36] T. Chen, C. Guestrin, XGBoost: a scalable tree boosting system, in: Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2016, pp. 785–794, http://dx.doi.org/10.1145/2939672.2939785.

[37] L. Grinsztajn, E. Oyallon, G. Varoquaux, Why do tree-based models still outperform deep learning on typical tabular data? Adv. Neural Inf. Process. Syst. 35 (2022) 507–520, URL https://proceedings.neurips.cc/paper_files/paper/2022/hash/0378c7692da36807bdec87ab043cdadc-Abstract-Datasets_and_Benchmarks.html.

[38] C. Bentéjac, A. Csörgő, G. Martínez-Muñoz, A comparative analysis of gradient boosting algorithms, Artif. Intell. Rev. 54 (3) (2021) 1937–1967, http://dx.doi.org/10.1007/s10462-020-09896-5.

[39] F. Hutter, L. Kotthoff, J. Vanschoren (Eds.), Automated Machine Learning: Methods, Systems, Challenges, in: The Springer Series on Challenges in Machine Learning, Springer International Publishing, Cham, 2019, http://dx.doi.org/10.1007/978-3-030-05318-5.

[40] J. Rapin, O. Teytaud, Nevergrad - a gradient-free optimization platform, 2018, https://GitHub.com/FacebookResearch/Nevergrad.

[41] J. Kennedy, R. Eberhart, Particle swarm optimization, in: Proceedings of ICNN'95 - International Conference on Neural Networks, Vol. 4, IEEE, Perth, WA, Australia, 1995, pp. 1942–1948, http://dx.doi.org/10.1109/ICNN.1995.488968.

[42] R. Storn, K. Price, Differential evolution – a simple and efficient heuristic for global optimization over continuous spaces, J. Global Optim. 11 (4) (1997) 341–359, http://dx.doi.org/10.1023/A:1008202821328.

[43] N. Hansen, S.D. Müller, P. Koumoutsakos, Reducing the time complexity of the derandomized evolution strategy with covariance matrix adaptation (CMA-ES), Evol. Comput. 11 (1) (2003) 1–18, http://dx.doi.org/10.1162/106365603321828970.

[44] J. Platen, I. Zreid, M. Kaliske, A nonlocal microplane approach to model textile reinforced concrete at finite deformations, Int. J. Solids Struct. 267 (2023) 112151, http://dx.doi.org/10.1016/j.ijsolstr.2023.112151.

[45] X. Xu, F. Conrad, X. Xing, O. Loeprech, M. Moeckel, Comparative analysis of small data acquisition strategies in machine learning regression tasks addressing potential uncertainties, Int. J. Adv. Softw. 16 (3 & 4) (2023) 243–253, URL https://www.thinkmind.org/library/Soft/Soft_v16_n34_2023/soft_v16_n34_2023_11.html.

[46] G. Lan, J.M. Tomczak, D.M. Roijers, A.E. Eiben, Time efficiency in optimization with a Bayesian-evolutionary algorithm, Swarm Evol. Comput. 69 (2022) 100970, http://dx.doi.org/10.1016/j.swevo.2021.100970.

[47] K. Numata, K. Tanaka, Stochastic threshold model trees: a tree-based ensemble method for dealing with extrapolation, 2020, http://dx.doi.org/10.48550/arXiv.2009.09171.

[48] J.R. Quinlan, et al., Learning with continuous classes, in: 5th Australian Joint Conference on Artificial Intelligence, Vol. 92, World Scientific, 1992, pp. 343–348.

[49] A. Shmuel, O. Glickman, T. Lazebnik, A comprehensive benchmark of machine and deep learning across diverse tabular datasets, 2024, http://dx.doi.org/10.48550/arXiv.2408.14817.