Bioresource Technology 432 (2025) 132654

Contents lists available at ScienceDirect

Bioresource Technology

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

Meta-tuning and fast optimization of machine learning models for dynamic methane prediction in anaerobic digestion☆

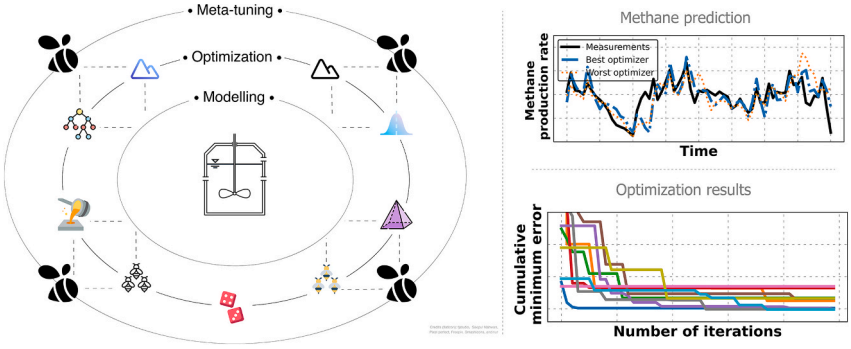
Alberto Meola [a,b], Klara Wolf [a,b], Sören Weinrich [a,c,*]

[a] DBFZ, Deutsches Biomasseforschungszentrum gemeinnützige GmbH, Biochemical Conversion Department, Torgauer Straße 116, Leipzig 04347, Germany
[b] Leipzig University, Faculty of Mathematics and Computer Science, Augustusplatz 10, Leipzig 04109, Germany
[c] Münster University of Applied Sciences, Department of Energy · Building Services · Environmental Engineering, Stegerwaldstraße 39, 48565 Steinfurt, Germany

HIGHLIGHTS

- Bayesian Search is recommended for general biogas prediction routine optimization.
- Simple optimization scenarios can be optimized with a 50-step optimization process.
- Complex scenarios including neural networks require more effective optimization.
- Meta-tuning has a positive influence on prediction results in complex scenarios.

GRAPHICAL ABSTRACT



ARTICLE INFO

Keywords:
Artificial intelligence
Biogas technology
Process modelling
Data processing
Parameter estimation

ABSTRACT

This study evaluates the performance of several optimization algorithms for tuning a data preparation and hyperparameter optimization pipeline applied to machine and deep learning models predicting methane production. Bayesian ridge regression and recurrent neural networks were applied to steady-state and dynamic datasets. Results show that 50 optimization steps are sufficient for optimal performance in simpler cases (62.8 % model accuracy). For complex scenarios, such as recurrent neural networks on dynamic datasets, extended optimization processes improve accuracy. Among the tested algorithms, Bayesian Search performed well without meta-tuning. However, meta-tuned Genetic Algorithm performed better (94.4 % vs 99.2 % baseline). Meta-tuning improves tuning parameter selection and model precision. Differential Evolution and Particle Swarm Optimization with time-varying acceleration also performed well, particularly in steady-state. These findings highlight the need to match optimization to dataset and model complexity, with meta-tuning offering advantages in challenging cases. Improved accuracy can increase revenue in flexible biogas operations.

☆ This article is part of a special issue entitled: 'Innovative Biogasprocess' published in Bioresource Technology.

* Corresponding author at: DBFZ, Deutsches Biomasseforschungszentrum gemeinnützige GmbH, Biochemical Conversion Department, Torgauer Straße 116, Leipzig 04347, Germany.
E-mail address: soeren.weinrich@dbfz.de (S. Weinrich).

https://doi.org/10.1016/j.biortech.2025.132654
Received 30 January 2025; Received in revised form 3 May 2025; Accepted 8 May 2025
Available online 10 May 2025

1. Introduction

Renewable energy production from sources such as wind and solar suffers from high variability due to their dependence on current weather conditions. The application of the Anaerobic Digestion (AD) process for the production of biogas and its subsequent conversion to electrical energy, instead, is suitable for demand-side management, as it can potentially be operated in a flexible manner (Mauky et al., 2017). AD involves the processing of various types of organic waste, including agricultural waste and farm manure, and is performed by bacteria and archaea under strictly anaerobic conditions. To match real-time energy loads, the increasing demand for controllable energy sources can partially be satisfied by dynamically operated biogas plant concepts (Thrän et al., 2023). However, during demand-oriented operation, the microbial AD process needs to be monitored and suitable models are required for the prediction and control of dynamic methane production. Mechanistic models like the ADM1 and several adaptations of it (Batstone et al., 2002; Weinrich and Nelles, 2021) have been developed to describe various process conditions and plant concepts. However, due to the requirement for numerous parameters and expensive offline measurements, these models are usually not suitable for automated application at large-scale industrial biogas plants. Moreover, such models require manual refitting processes and can be sensitive to sensor faults.

As an alternative, machine and deep learning models can predict the methane yield, potentially using a limited number of measurements, being more robust and requiring only automatic refitting processes (Ling et al., 2024; Meola et al., 2023). While Machine Learning (ML) models can perform well when modelling the AD process, a fast and accurate choice for data preprocessing techniques and Hyperparameters (HPs) estimation is crucial for model application on industrial scale. Often, the choice of HPs, such as size and number of layers of a Neural Network (NN), is often set by manual adjustment with little justification (Seo et al., 2021; Lo Sciuto et al., 2016). Sometimes grid search is applied for optimal choice of individual parameters, but without detailed investigation of the applied search space (Capizzi et al., 2020). However, it has been demonstrated that HP optimization is important to achieve optimal results, especially when the search domain has a complex structure (Diaz et al., 2017).

Thus, several authors perform data preprocessing or model HPs optimization (Sun et al., 2023; Yildirim and Ozkaya, 2023), but the combination of both is often absent or left to generic Auto-ML pipelines – such as the Tree-Based Pipeline Optimization Tool (Le et al., 2020) – that might not yield optimal results, since they do not include domain knowledge (Wang et al., 2021, Deng et al., 2024). Furthermore, several authors applied metaheuristic algorithms for the optimization of several parameters of their predictive models (Gogna and Tayal, 2013). Unlike traditional optimizers, which rely on explicit mathematical formulations or derivatives, metaheuristics are inspired by natural or abstract concepts and mimic the behavior of biological or physical systems. Beltramo et al. (2019) used a NN optimized with a Genetic Algorithm (GA) to predict the biogas rate of an agricultural biogas plant. Furthermore, Abu Qdais et al. (2010), Jacob and Banerjee (2016) and Sathish and Vivekanandan (2016) applied a GA for the optimization of NNs to predict biogas production. Also simpler prediction methods – such as the ensemble method combining k-nearest neighbours and random forest regressor developed by Li et al. (2022) – can be optimized by metaheuristic algorithms, such as Particle Swarm Optimization (PSO). Moreover, Beltramo et al. (2016) trained a NN with Ant Colony Optimization (ACO) to predict the biogas production rate of simulated data, mimicking co-digestion of multiple substrates with the ADM1. For combined optimization of data pre-treatment and model HP estimation, a pipeline has been developed (Meola et al., 2023). Such combined optimization is required when datasets, such as biogas production datasets, present potentially a high number of missing values for several features or might have transitory faults in the installed sensors. However, a detailed analysis of suitable optimization algorithms (apart from the applied GA) has yet to be performed, especially considering the high duration of the optimization process (12–56 h).

The current study aims to find the best optimization algorithm for shortening the optimization process without sacrificing model performances. Furthermore, a meta-tuning strategy is introduced to identify the Tuning Parameters (TP) of the pipeline optimizer. To gain knowledge of the functionality and underlying dependencies, the numbers of iterations and optimizer starting points are analysed in detail.

2. Materials and methods

To investigate the complex behaviour of the AD process, two different datasets and two prediction models were tested within the applied optimization pipeline. Prediction models include Bayesian Ridge Regression (BRR) and Recurrent Neural Networks (RNNs) in the form of Long Short-Term Memory (LSTM) and Gated Recurrent Unit (GRU). Individual parameters of the data preparation pipeline (Meola et al., 2023) and the prediction models were optimized with nine different metaheuristic optimizers.

2.1. Process data

For process simulation and validation of implemented optimization procedures, two datasets were evaluated (Meola and Weinrich, 2024). Both datasets were derived from mesophilic full-scale experiments in a single continuous stirred tank reactor with a total volume of 188 m^3 at the Deutsches Biomasseforschungszentrum (DBFZ, German Biomass Research Center).

2.1.1. Dataset A

Dataset A consists of data from steady-state AD of rye whole crop silage (solid substrate) and cattle manure (liquid substrate) with a constant Organic Loading Rate (OLR) of approximately 4 kg Volatile Solid (VS) m^{-3} d^{-1} and a Hydraulic Retention Time (HRT) of 30 d. The reactor was fed on average with 1.4 t of solid substrate and 2.8 m^3 of liquid substrate per day, with an average liquid-to-solid feed ratio of 1:3 (based on mass of VS). The average methane production rate was 7.98 m^3 h^{-1} with a standard deviation of 1.82 m^3 h^{-1}, a maximum of 20.33 m^3 h^{-1} and a minimum of 1.08 m^3 h^{-1}. The dataset consists in a total of 280,561 data points. Five months (149 days) of process data were used for model training (70 % of the available data), while validation and test data consisted of 34 and 17 days, respectively (20 % and 10 %).

2.1.2. Dataset B

In dataset B dynamic AD of corn silage (solid substrate) and cattle manure (liquid substrate) was investigated with variable OLR between 2 and 12 kg VS m^{-3} d^{-1}, with peaks of 16 kg VS m^{-3} d^{-1}, for 165 days. The average HRT per week varied between 22 and 97 d, with a total mean of 40 d. The reactor was fed on average with 1.5 t of solid substrate and 1 m^3 of liquid substrate per day, with a liquid-to-solid feed ratio varying from 1:0 to 1:18 with an average of 1:6 (based on mass of VS). The average methane production rate was 14.66 m^3 h^{-1}, with a standard deviation of 5.11 m^3 h^{-1}, a maximum of 38.45 m^3 h^{-1} and a minimum of 1.35 m^3 h^{-1}. The dataset consists in a total of 239,041 data points. Four months (116 days) of process data were used for model training (70 % of the available data), while validation and test data consisted of 32 and 16 days, respectively (20 % and 10 %).

2.1.3. Prediction specifications

Methane production rate was selected as main output for model prediction, while all remaining features were used for input description. Furthermore, common process variables (such as OLR or HRT) as well as statistical features were added according to Meola et al. (2023). Both datasets were originally recorded at one minute resolution, and were resampled at one hour resolution for dataset A and at six hour resolution

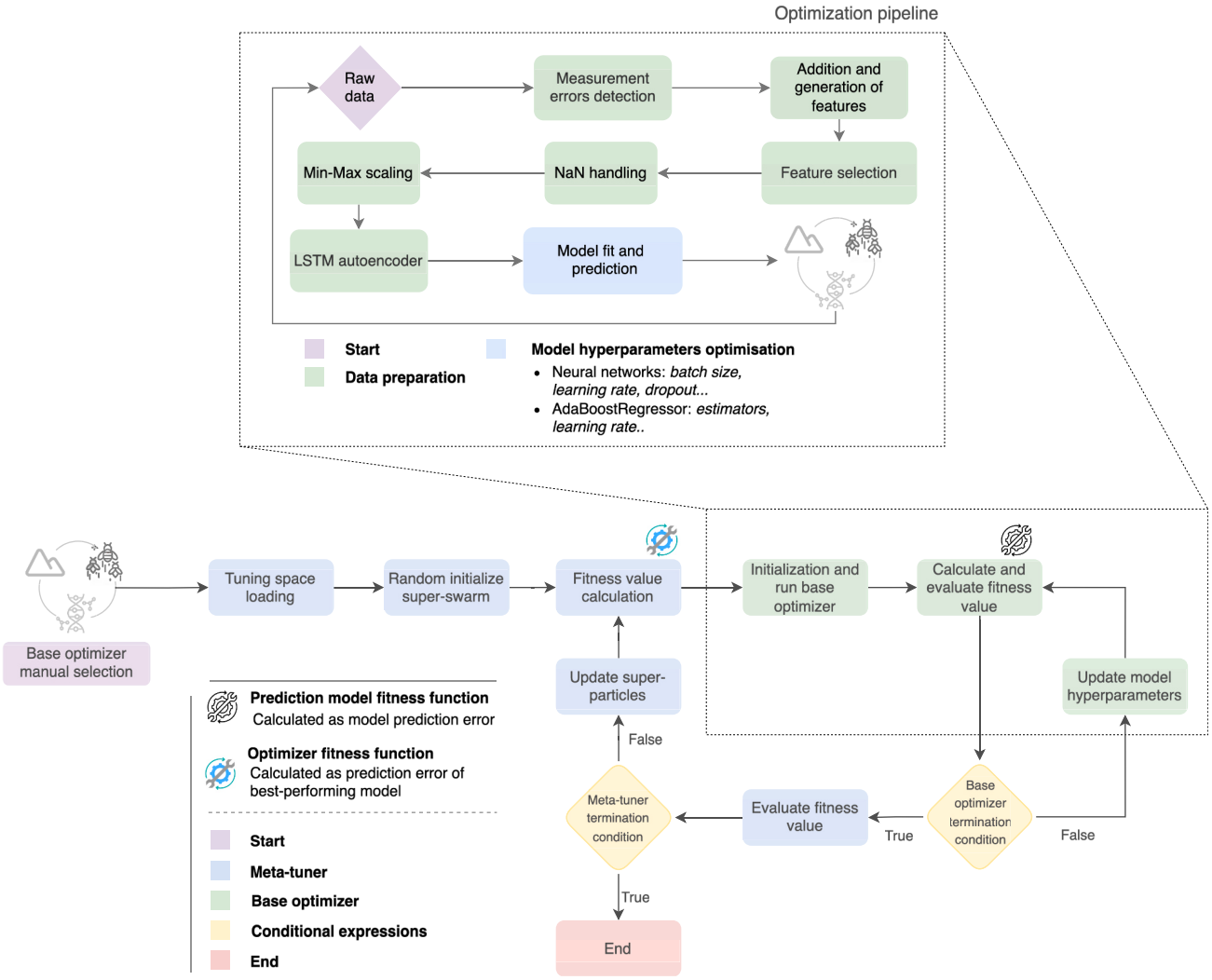


Fig. 1. Flowchart of meta-tuning for optimal results of the applied data preparation and model HP optimization pipeline.

for dataset B. Every prediction is based on historical input and output data. Thus, all available features of previous time steps are considered as input data. Furthermore, the Observation Distance (OD) is defined to characterize the time interval between historical data and the prediction of methane production. For prediction at time t, all measurements of available features t to $t - OD$ are ignored, considering the historical input from $t - OD$ backwards. Only (liquid and solid) feeding quantities are provided to the model until time t at each prediction since they are required as operational variables. Dataset A is simulated with an OD of twelve hours, while Dataset B is simulated with an OD of 24 h. Different OD were chosen for the two separate datasets for differentiating further the two scenarios and increasing prediction difficulty for dataset B.

2.2. Optimization pipeline

For optimization of data preparation procedures (including NaN handling and feature selection) as well as individual model HPs an existing data preparation and model optimization pipeline was applied (Meola et al., 2023). A simplified scheme of the main functional components of the pipeline is illustrated in Fig. 1. The applied optimization pipeline consists of multiple components. It includes a heuristic system for detecting physically impossible values and correcting feeding errors, an isolation forest algorithm for identifying measurement errors, and an autoencoder for generating additional input features. Additionally, a Minimum Redundancy Maximum Relevance (mRMR) algorithm is used for feature selection.

2.3. Prediction models

BRR and RNNs were chosen as prediction models, generally representing less and more complex ML procedures. The number of unknown HPs of each method varies between 17 and 24. The composition of the final HP space is presented in the Supplementary material.

2.3.1. Bayesian ridge regression

BRR is a variant of linear regression that incorporates Bayesian inference (Gelman et al., 2013). For BRR, the prior distribution of the parameters is assumed to be Gaussian with a zero mean and a known variance. The goal is to estimate the posterior distribution of the parameters given by the observed data. This method presents four HPs to be optimized. Specifically, α_1 and α_2 determine the mean and variance of the prior distribution, and λ_1 and λ_2 control the shape of the prior distribution and the noise level of the model, respectively.

2.3.2. Recurrent neural networks

A RNN, formed by LSTM or GRU cells, was also applied for process prediction (Hochreiter and Schmidhuber, 1997; Cho et al., 2014). RNNs are neural networks specially designed for processing sequential data, such as time series. Such NNs use a hidden state that is updated at each time step and that stores information about the previous inputs to the

network. This allows the network to capture the temporal dependencies and patterns in the data. More precisely, LSTM and GRU use specialized gating mechanisms to better control the flow of information through the network and preserve the gradients during backpropagation, mitigating the vanishing gradient problem, typical of baseline RNNs. The applied RNN has eleven HPs, one of whose being the choice of LSTM or GRU cells. The HP space for both BRR and RNN (LSTM/GRU) is summarized in the Supplementary material.

2.4. Optimization procedure

Several optimization algorithms were evaluated for the definition of optimal data preparation and prediction model HPs within the defined HPs' space. Before starting the data preparation pipeline, the chosen optimizer defines a combination of data preparation and model HPs, which are then used as settings for the pipeline. The methane yield prediction error is the evaluation metric. After the evaluation, the optimizer decides on a new parameter combination (or vector), expecting a lower error. The study evaluated a range of optimization algorithms, categorized into three main groups: local search methods (Hill Climbing (HC) and Downhill Simplex (DS)), global search techniques (Simulated Annealing (SA), Hill Climbing with Random Restart (RHC), and Bayesian Search (BS)), and population-based approaches (Particle Swarm Optimization (PSO), Hybrid PSO-TVAC (HPSO-TVAC), Differential Evolution (DE), and Genetic Algorithm (GA)). Other optimization techniques, such as gradient-based optimizers (Bengio, 2000) and Hyperband (Li et al., 2018) were not taken into consideration in this study, either for the requirement of the function to be optimized to be convex, or for too high computational requirements. On top of these nine base optimizers, a meta-tuning strategy was implemented. To differentiate between the prediction model hyperparameters and the optimizer parameters, the terms HPs (hyperparameters) and TPs (tuning parameters) were previously defined. HPs are parameters provided by the framework, representing data preparation pipeline parameters and the prediction models' parameters. The HPs are optimized by the base optimizers. This operation is named optimization. TPs are parameters related to the base optimizers themselves, such as the number of neighbours in HC or the social weight for the PSO. These parameters are optimized by the meta-tuning algorithm. This operation is defined as tuning. The meta-tuning strategy is based on a PSO, instantiating a base optimizer with the given set of tuning parameters (see Section 2.7).

2.5. Benchmark optimizers

For effective comparison of obtained results, Random Search (RS) was chosen to compare the optimized methane predictions with choice randomness. RS takes random samples from the search space in each iteration. Each sample represents a possible solution for the optimization problem. It does not rely on the shape of the search space or the distribution of the optimal solution.

2.6. Base optimizers

The selected base optimizers are applied for the optimization of the data preparation and model HPs within the pipeline. All the base optimizers are programmed to find the best parameter combination depending on the resulting error in the prediction of biomethane production. Those optimizers are either used with standard TPs or are tuned through the meta-tuner presented in Section 2.7.

2.6.1. Hill climbing

The HC method explores the search space locally, by moving to positions within its neighbourhood with a better solution. These neighbour points are sampled at random with a given distribution. In order to instantiate this algorithm, there are three TPs to specify: ε, representing the tolerance for improvement between steps, the number of neighbors representing how many potential solutions the algorithm evaluates at each step before choosing the best one, and distribution indicating the probability distribution used to generate neighbors (perturbations) around the current solution (Selman and Gomes, 2006).

2.6.2. Random restart hill climbing

The RHC method works like the above-introduced HC algorithm, but it performs a random restart at a new, random position in the space after a previously set count of iterations. This process should allow the avoidance of local minima that the traditional HC algorithm can suffer from (Russell and Norvig, 2016). The tuned HP within this model are the same as in the HC optimizer, plus the number of iterations before a new restart.

2.6.3. Random simulated annealing

The SA algorithm is based on HC and derived from the regular RS algorithm. Unlike HC, it may also move to points with a worse solution, with a probability that decreases over time, called temperature. This probabilistic movement allows the algorithm to search the space globally and provides a technique to escape local minima (Kirkpatrick et al., 1983). The four TPs of SA are ε, the number of neighbors, the neighbors distribution, the annealing rate, representing the rate at which the temperature decreases during the cooling process, and the start temperature of the system, which determines the likelihood of accepting worse solutions at the beginning of the optimization process.

2.6.4. Downhill simplex

The DS works by constructing a simplex, a geometrical shape with N + 1 vertices in an *N*-dimensional search space (Powell, 1973). At each iteration, the simplex is transformed by reflecting, expanding, contracting or shrinking one of its vertices, depending on the function values. The new vertices are again evaluated to eventually find the local minimum. TPs belonging to the DS algorithm are α, the size of the reflection step when exploring a potentially better solution, β, representing how much the simplex contracts towards the centroid when no improvement is observed, γ, how far the simplex expands beyond the reflected point and σ, representing how much the simplex shrinks towards its best vertex when the algorithm detects stagnation.

2.6.5. Bayesian search

BS is a probabilistic technique that selects new positions by calculating the expected improvement of every position in the search space based on a Gaussian process that trains on already evaluated positions (Iida, 1992). The core idea is to maintain the probabilistic model of the objective function, which captures the uncertainty and tradeoffs between different HPs. Two TPs are optimized: ξ, the parameter controlling the balance between exploration and exploitation, and the number of iterations before restart.

2.6.6. Particle swarm optimization

The PSO is a population-based optimization algorithm inspired by the social behavior of bird flocks or fish schools (Kennedy and Eberhart, 1995; Shi and Eberhart, 1998). In the PSO, a group of particles, forming a swarm, move through the space to find the optimal solution. Each particle represents a potential solution to the problem and has a position and velocity. At each iteration, the algorithm updates the position and velocity of each particle. The update considers the particle's current position and velocity, the best position the particle itself has found (personal best), and the best position found by the entire swarm (global best). Four TP need to instantiate the implemented PSO optimizer: the inertia weight parameter, *W*, set as 0.5 to keep the optimization stable between local and global search, balancing exploration and exploitation. The cognitive and social weight parameters, c_1 and c_2, respectively, determine the influence of the personal best and the global best solution, and were set as two as proposed by Kennedy and Eberhart (1995). The number of particles parameter sets the size of the swarm. It was decided

for a small swarm size (here two) to make it comparable with the other base optimizers in terms of complexity.

2.6.7. Hierarchical particle swarm optimization with time varying acceleration

For effective control of the global search convergence and global best solution, Ratnaweera et al. (2004) proposed a PSO algorithm with linearly Time-Varying Acceleration Coefficients (TVAC), where a larger c_1 and a smaller c_2 were set at the beginning and were gradually reversed during the search. As proposed in the paper, c_{min} and c_{max} were respectively set to 0.5 and 2.5. Moreover, the authors combined the PSO-TVAC with a self-organised Hierarchical PSO (HPSO), where the momentum for the particles to roam through the search space is maintained by reinitializing particles with random velocities whenever they stagnate in the search space. This approach was implemented as HPSO-TVAC.

2.6.8. Differential evolution

In the DE algorithm (Storn and Price, 1997), a population of possible solution candidates – with every candidate consisting of a TPs vector – is initialized, and further candidates are generated by combining the existing solution vectors through a mutation and crossover process. The mutation operation consists of calculating the difference between two vectors, multiplying it by a scaling factor F and adding this to a third vector. The binomial crossover operation involves combining the new candidate solution generated by the mutation with an existing candidate solution in the population, with the probability of choosing one of them over the other c_r. F and c_r were set as 0.5 and 0.7, respectively (Virtanen et al., 2020). For direct comparison, same population size value was chosen as in other base optimizers.

2.6.9. Genetic algorithm

The GA is an evolutionary optimization technique inspired by natural selection, which iteratively refines solutions through mechanisms such as selection, crossover, and mutation (Katoch et al., 2021). A population is defined – with each individual containing a combination of HPs – and parents that are chosen for mating originate new offspring, crossing genetic material and originating mutations. In the applied version of the GA, an adaptive mutation is used, consisting of having different mutation probabilities for good and bad performing genes. Six TPs are taken into consideration for the GA. The percentage of mating parents determines the proportion of the parent population selected for reproduction. The kept parents define the fraction of the parent population preserved unchanged in the next generation. The parents' selection algorithm governs how parents are chosen based on their fitness, influencing the algorithm's ability to prioritize high-performing individuals while maintaining diversity. Crossover type (like single-point, multi-point, or uniform crossover) dictates how genetic material is exchanged between parents to produce offspring. The lower mutation applies to low-performing genes, while the higher mutation applies to well-performing genes.

2.7. Meta-tuning

For optimal optimization results, individual TPs of the base optimizers need to be optimized. While this additional optimization (tuning) can be performed manually or with trivial and time-consuming operations (such as grid search), more complex methods might be required to yield optimal results within complex search spaces. Thus, a PSO on top of the underlying base optimizers was applied, here defined as super-PSO. The role of the super-PSO is the tuning of the base optimizers' TPs. The swarm belonging to the super-PSO is here called super-swarm, and each particle belonging to the super-swarm is defined as super-particle. A flowchart of the developed meta-tuning method is shown in Fig. 1. This tuning is applied to all tested base optimizers. Once the base optimizer is manually selected, its tuning parameter space is loaded within the super-PSO, and its super-swarm is randomly initialized with a fixed number of super-particles. Each of the super-particles evaluates its fitness according to its current random position in the parameter space. Thus, an instance of the base optimizer is created with the corresponding TPs and performs its optimization on the HPs space to find an optimal solution for the predicted methane yield. The applied base optimizer performs a search until the prediction error does not improve absolutely by 0.1 or relatively by 0.1 % within the last three iterations. The output of the optimizer, being the best-found prediction error, performs as the fitness value of the current particle of the super-swarm. After each super-particle is evaluated, the super-PSO operates as the original PSO, adjusting the position and velocity of each particle (as described in Section 2.6.7). The super-PSO runs until the error metric does not improve absolutely by 0.05 or relatively by 0.1 % within the last five generations, and it returns the best tuning parameters of the underlying base optimizer. These termination conditions were determined based on previous experiments. The TP of the super-PSO were set according to Kennedy and Eberhart (1995), with $w = 0.5$ and $c_1 = c_2 = 2$. The size of the swarm was fixed to $n = 3$ due to the smaller size of the TP space to be searched in comparison to the HP space. An example of a meta-tuning scenario consists of a meta-PSO adjusting the TPs of a genetic algorithm optimizing the data preparation and HP of an RNN for the prediction of biomethane production of Dataset B 24 h in advance.

2.8. Error metrics

Two separate error metrics were chosen for optimization of LSTM neural weights, BRR, and model optimization parameters (such as sequence length, autoencoder switch or weight initialization).

2.8.1. Root mean square error

For optimization of neuronal weights, Root Mean Square Error (RMSE) was chosen, since it penalizes large differences between prediction results and measurements more than other methods, such as the Mean Absolute Error (MAE). Penalization of large differences is required to enable accurate depiction of methane production peaks.

2.8.2. Root mean square scaled error

For optimization of characteristic model parameters, Root Mean Square Scaled Error (RMSSE) was applied, according to Eq. (4).

$$RMSSE = \frac{\sqrt{\frac{1}{T}\sum_{i=1}^{T}(Y_t - P_t)^2}}{\sqrt{\frac{1}{T}\sum_{i=1}^{T}(Y_t - Y_{t-1})^2}} \tag{4}$$

with T being the total number of time steps, Y_t the measurement and P_t the prediction at the actual time step t, respectively. The RMSSE is a relative (scaleless) error metric. Thus, it allows to compare errors among different datasets and to evaluate model predictions against a naïve forecast (Hyndman and Koehler, 2006).

2.9. Implementation

All the simulations were executed on a high-performance cluster, with 64 Intel Xeon E5-2698 V3 CPUs, a maximum clock of 3.6 GHz and 110 GB of RAM. The optimization pipeline was programmed in Python 3.7.1, making use of several packages. Scikit-learn (Pedregosa et al., 2011), Pandas (McKinney, 2010) and Numpy (Harris et al., 2020) were used for the data preparation process and the development of PSO, HPSO and DE, Tensorflow (Abadi et al., 2016) was applied for the autoencoder and for the prediction model, while the library Gradient Free Optimizers (Simon Blanke, 2020) was used for the development of the HC, Random SA, DS, and Bayes' optimization.

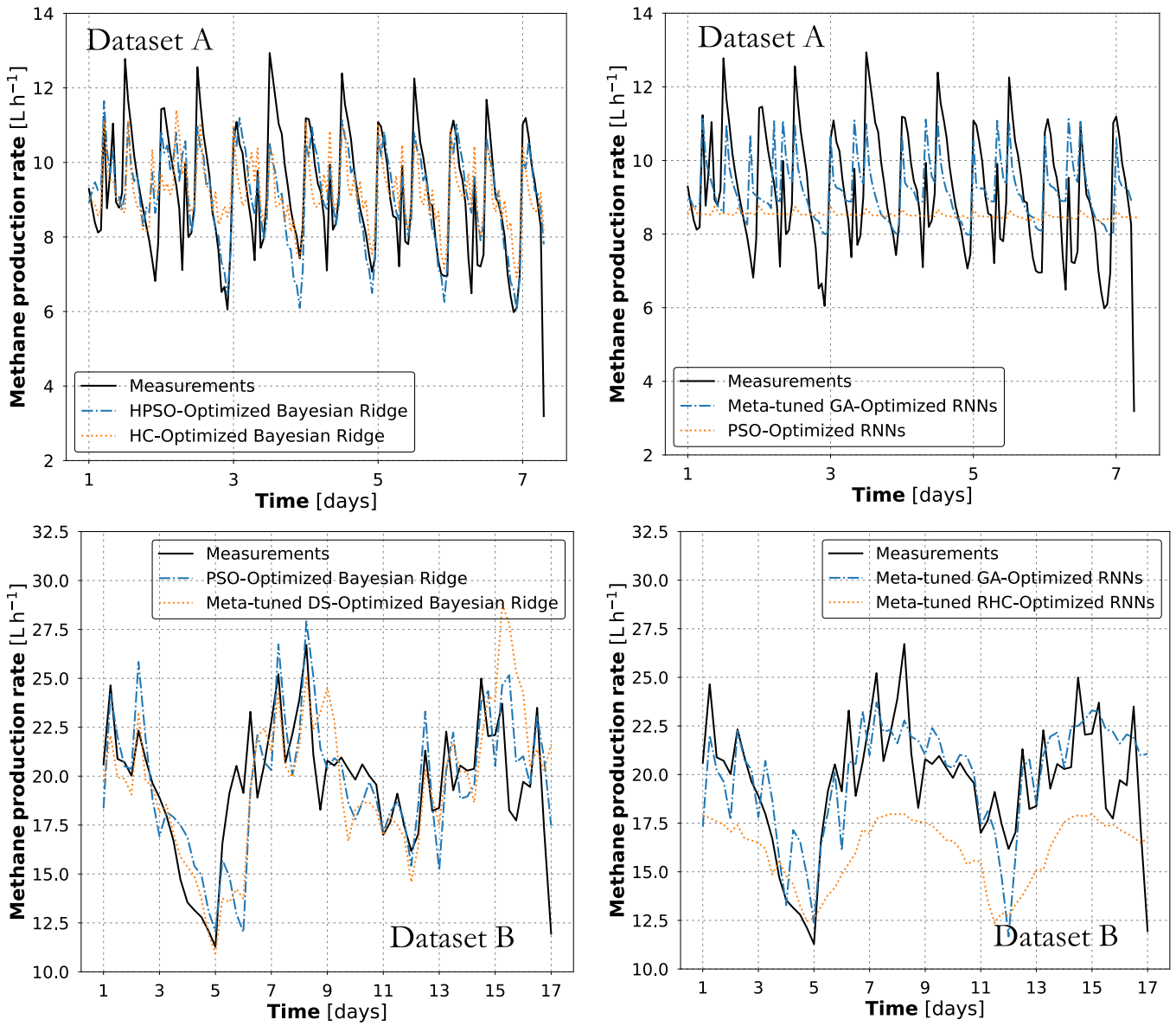


Fig. 2. Results of best and worst optimization results of datasets A and B.

3. Results and discussion

For detailed evaluation of the error metrics, the best and worst optimizer results for each dataset and prediction model are shown in Fig. 2. The visual difference between RMSSE quantities of 167.6 and 94.4 can be observed in Fig. 2 in RNNs prediction of Dataset B. The individual error of each prediction is summarized in Table 1. Generally, more complex models (such as RNNs) tend to perform worse than simpler models (such as BRR) at this number of iterations. This can be explained by the complex structure and optimization process of the RNNs, which requires a higher number of HPs to be optimized, many of them with high sensitivity (Gorgolis et al., 2019; Meola and Weinrich, 2024).

3.1. Evaluation of base optimizers

For analysis of optimization performances, the applied base optimizers and RS were initially run for 5, 10, 20 and 50 iterations. Due to the stochastic nature of metaheuristic algorithms, each optimizer performance is run three times, and the results are then averaged. Results show that while the results at 5, 10 and 20 iterations show high standard deviations, at 50 iterations the standard deviation is, in general, reduced. This fact indicates that at 50 iterations the algorithms behave more similarly regardless from the starting point of each optimization run (see Supplementary material). Thus, the evaluation of optimizer performances is carried out for 50 iterations only. To analyze the uncertainty caused by different starting points, individual optimization scenarios were repeated five times (Section 3.1.3). For comparison with longer simulation times, selected optimization scenarios were run for 2,000 steps (Section 3.1.4). This limit was chosen because previous studies have shown that models do not significantly improve beyond this number of optimization steps (Meola et al., 2023; Putra et al., 2025).

3.1.1. Dataset A

Table 1 shows the individual performance of different prediction models for optimal pipeline settings of each optimization algorithm (base and benchmark optimization). For each optimizer, the average time per evaluation is included. For the population-based optimizers, one iteration refers to one generation with five individuals. The evaluation times for both BRR and RNNs show high differences among the

Table 1
Performance evaluation of individual benchmark and base optimizers.

		Bayesian Ridge Regression				Recurrent Neural Networks			
	Applied optimizer	Average time per evaluation[a] [min]	Best test RMSSE[b] [%]	Average iterations before best result	Average time before best result [min]	Average time per evaluation[a] [min]	Best test RMSSE[b] [%]	Iterations before reaching best test result	Average time before best result [min]
Dataset A	Random	3.1	66.5	17.5 ± 15.7	54.3	6.0	82.1	30.2 ± 9.1	181.2
	Hill Climbing	3.1	71.8	18 ± 7.8	55.8	3.9	94.2	6.0 ± 7.5	23.4
	Random Hill Climbing	1.7	67.6	31.2 ± 15.3	53.0	2.0	84.6	13.0 ± 14.0	26.0
	Bayesian	1.7	67.0	31.8 ± 8.9	54.1	3.2	77.8	14.6 ± 8.1	46.7
	Downhill Simplex	2.4	65.9	21.3 ± 13.3	51.1	2.1	90.8	20.5 ± 3.6	43.1
	Simulated Annealing	3.2	66.1	24.7 ± 3.77	79.0	1.8	94.1	12.0 ± 16.8	21.6
	Particle Swarm Optimization	1.6	66.7	15.3 ± 2.0	24.5	1.9	111.0	42.0 ± 6.0	79.8
	Hierarchical Particle Swarm Optimization	1.5	**62.8**	29.33 ± 11.9	44.0	2.0	83.5	33.5 ± 10.3	67.0
	Differential evolution	1.8	65.1	13.67 ± 12.47	24.6	3.0	94.2	20.3 ± 18.9	60.9
	Genetic Algorithm	**0.6**	69.4	20.33 ± 4.0	12.2	2.3	77.8	34.7 ± 12.0	79.81
Dataset B	Random	0.8	98.5	21.3 ± 9.7	17.0	1.0	116.4	23.3 ± 14.5	23.3
	Hill Climbing	0.7	97.4	3.7 ± 3.6	2.59	1.2	132.3	9.2 ± 11.3	11.0
	Random Hill Climbing	0.6	99.9	21.0 ± 14.0	12.6	0.8	115.9	19.5 ± 14.4	15.6
	Bayesian	**0.4**	97.7	14.2 ± 12.4	5.7	1.0	99.5	32.5 ± 10.0	32.5
	Downhill Simplex	0.5	99.0	38.2 ± 4.4	19.1	1.0	112.6	22.4 ± 9.1	22.4
	Simulated Annealing	0.7	104.8	24.5 ± 22.5	17.1	1.2	135.0	1.33 ± 1.25	1.6
	Particle Swarm Optimization	0.7	**97.6**	2.0 ± 0	1.4	1.0	99.6	35.0 ± 11.0	35.0
	Hierarchical Particle Swarm Optimization	0.6	112.6	5.8 ± 3.5	3.5	1.0	117.0	24.8 ± 20.0	24.8
	Differential evolution	0.6	102.3	13.0 ± 11.7	7.8	1.0	101.3	14.5 ± 1.5	14.5
	Genetic Algorithm	**0.4**	110.1	25.67 ± 6.8	10.3	0.8	99.2	32.0 ± 15.5	25.6

[a] Bold and underlined evaluations times indicate best times for the relative dataset. Underlined evaluation times indicate best times for relative dataset and applied prediction method.
[b] Bold and underlined errors indicate best error values for the relative dataset. Underlined errors indicate best error values for relative dataset and applied prediction method.

different algorithms, with BRR average iteration speed of the GA being one-fifth of the speed of RS, HC and SA. RNN iterations speed differ slightly less, probably due to the high influence of the NN training time. Results evince that algorithms with the fastest and slowest convergence (e. g., GA and SA for BRR and SA and RS for RNN, respectively) are never in the top three best performing algorithms. This indicates that fast convergence leads towards too simple pipeline architectures. Long convergence results in too complex structures. The only exception is represented by the RS, whose results are not controlled by any optimization mechanism and can at times represent complex architectures, and at other times very simple ones. Best performing algorithms for BRR are HPSO (RMSSE = 62.8 %), DE (RMSSE = 65.1 %) and DS (RMSSE = 65.9 %). However, most optimizers (except HC and GA) reach similar performance levels between 66.1 and 67.6 % RMSSE. Individual results from RNNs optimization differ more, with a variation between the best (GA) and the worst (PSO) performer between 77.8 and 111.0 %. A group represented by GA, BS, RS, HPSO and RHC offer performances below 85 % RMSSE, while all the other optimizers do not fall below this threshold. Regarding the steady-state operating conditions within dataset A, the only optimizer that does not seem fit for the purpose is HC. Depending on optimization complexity, several algorithms might be used, with HPSO working well for BRR and RNNs. In general, ML algorithms (such as BRR) are easier to optimize than RNN algorithms within the applied pipeline, as expected, with the best RNN result being worse than the worst ML result. This suggests already that a low number of optimization iterations might not be suitable for more complex models but suffices for simpler models. Optimization trajectories of BRR and RNNs are illustrated in Fig. 3a and b. While most of the algorithms applied to BRR have already found an optimal solution between step 20 and step 30, they require on average more time when optimizing RNNs. Thus, several optimizers still keep experiencing error improvements between step 30 and step 40, while only the worst-performing algorithm (PSO) improves past 40 optimization steps. Nevertheless, half of the optimizers applied to RNNs are able to find a solution below 100 % RMSSE, while when applied to BRR models, optimizers find solutions under 100 % RMSSE after ten iterations. Moreover, the optimizers stagnate at different levels, indicating that the optimizer. Fig. 3c and d summarize the mean errors for all test runs performed on dataset A for BRR and RNN models, respectively. The error bars indicate the standard deviation across the runs. Interestingly, the mean error rankings don't quite match the rankings based on the absolute best results from Table 1, even though it is closely resembled. When optimizing BRR, optimization algorithms

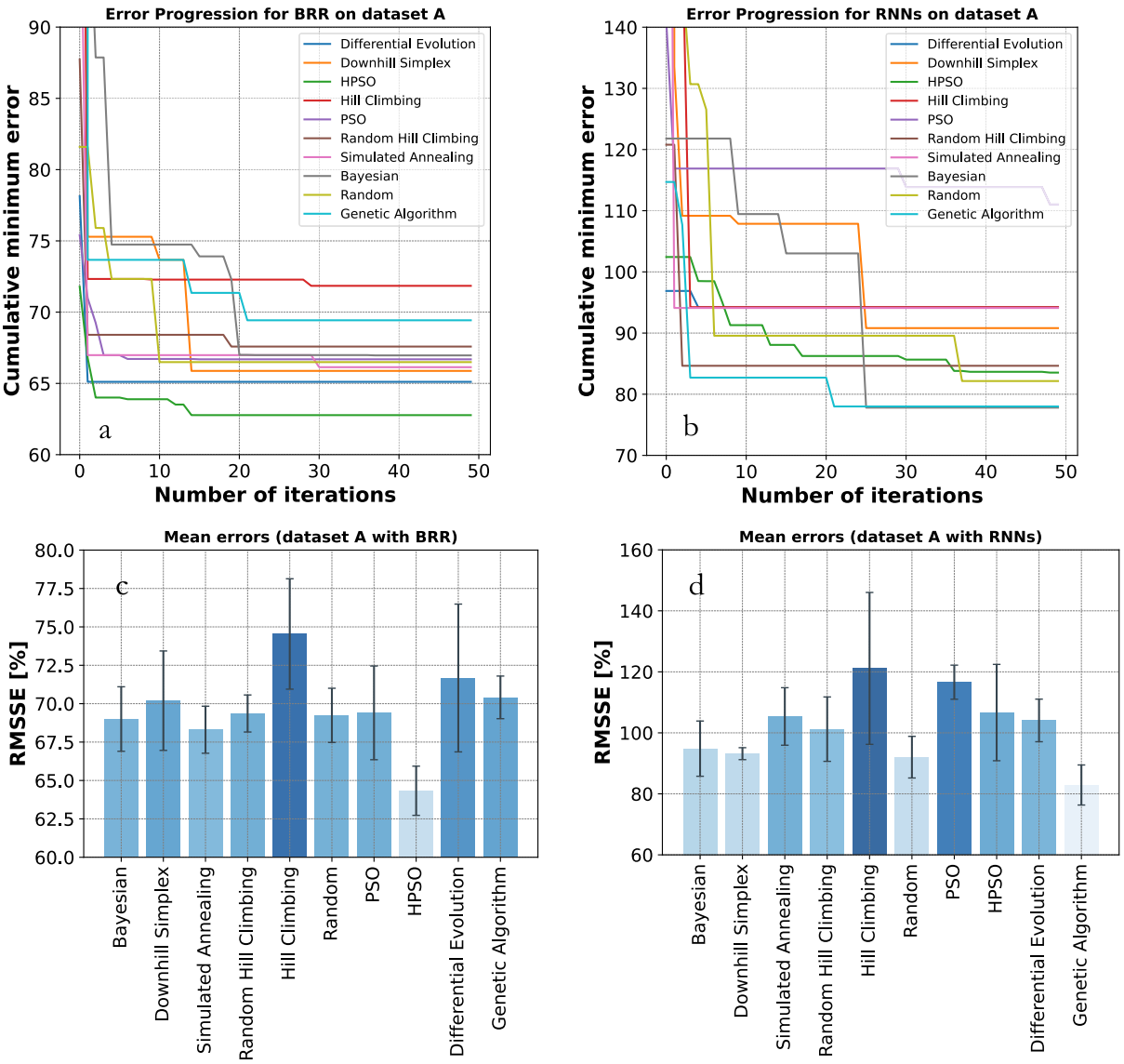


Fig. 3. Comparison of mean errors, standard deviations and number of iterations for dataset A (BRR and RNNs).

can be divided into two groups regarding to their standard deviation, with the group with a higher standard deviation being represented by DS, DE and HC (3.23, 4.81 and 3.59 %, respectively), and the rest having considerably lower standard deviation (between 1.21 and 3.06 %). The distribution of the standard deviation within the errors on RNNs is broader, with DS and PSO having a low standard deviation (1.96 and 5.59 %, respectively), HC and HPSO having very high standard deviations (24.90 and 15.80 %, respectively). Considering all the observations regarding the best performances and the standard deviation of the different runs, BS, RS and RHC are best suitable for prediction of steady-state methane production at full-scale using BRR and RNNs.

3.1.2. *Dataset B*

The performance of the applied benchmark and base optimizers for accurate prediction of methane production rate in dataset B by BRR and RNNs is also included in Table 1. While optimizing BRR, the models having RMSSE lower than 100 % are RS, HC, RHC, BS, DS and PSO, with PSO being the best performing optimizer. When optimizing the RNNs, only three optimizers (BS, PSO and GA) are able to score lower than 100 % RMSSE. In general, results and execution times are less widely spread in comparison to dataset A, with execution time varying between 0.4 and 1.2 % among the optimization of both BRR and RNNs. Furthermore, the lowest error reached by the optimizer on RNNs is close to the optimal error reached by the best-performing BRR model. This behaviour might be explained by the higher variation in methane production rate and substrate amount within dataset B compared to dataset A, which makes the prediction in general more difficult. Thus, the complexity of RNNs helps reaching optimal results, even though they do not match the performances of BRR (probably because of the low number of iterations). In Fig. 4a and b the error progression of the optimizers for both BRR and RNNs is illustrated. In general, it can be noticed that models tend to reach optimal results later than within dataset A, demonstrating again a difficulty in finding optimal HPs for simulating dataset B. Nevertheless, the optimization of BRR applied to dataset B seems to stagnate around 100 % RMSSE, suggesting the presence of a global minimum. While for dataset A the optimized RNNs converged between 75 % and 95 % RMSSE, optimized RNNs applied on dataset B converged in a larger range between 95 % and 135 % RMSSE. This fact indicates that the importance of HPs choice when optimizing complex models on highly dynamic datasets. Fig. 4c and d show the average results among the three runs of the applied optimizers on dataset B. HC, HPSO and DE show a high standard deviation (69.03, 46.30 and 32.78 %, respectively) compared to the other models (from 0 to 5.01 %) when optimizing BRR, with HC having a high standard deviation also when optimizing RNNs

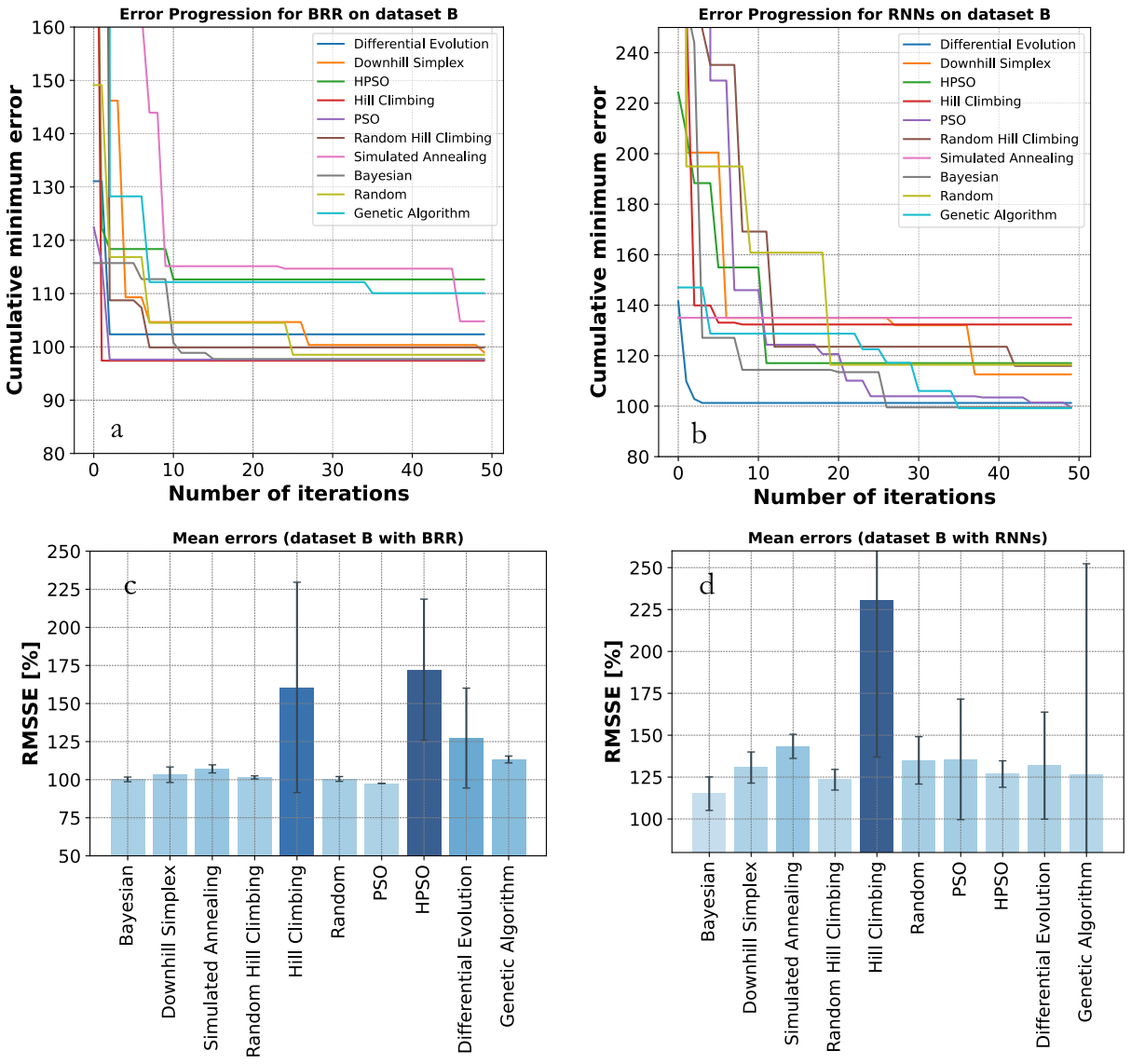


Fig. 4. Comparison of mean errors, standard deviations and number of iterations for dataset B (BRR and RNNs).

(93.46 %). GA, PSO and DE also show high standard deviations (126.15, 35.97 and 31.94, respectively), while the other models show lower levels of standard deviation, with RHC showing very low levels of uncertainty (6.17 %). Nevertheless, the average among the three runs of all the applied optimizers lies above 100 % RMSSE, showing that 50 iterations do not suffice for complex models and complex datasets. Among the top three performing optimizers when applied to RNNs, the one showing the lowest standard deviation is BS (9.98 %). In conclusion, BS seems to be the most robust optimizer at 50 iterations, consistently performing well and with low standard deviation among the four analysed scenarios (based on two datasets and prediction algorithms, respectively).

3.1.3. Uncertainty analysis

As described in Section 3.1., a high difference in optimizer performances can be expected, since the direction of the optimization strongly depends on the starting point and individual tuning parameters required by the applied base optimizer. During the optimization of models for real-time prediction of methane production, it is imperative to obtain optimal results in the shortest time possible to predict process disturbances or interruptions, without the need for repeating long optimization processes. Furthermore, continuous re-training with newly obtained data be needed to gain high prediction accuracies. Therefore, selected optimization algorithms (DE, DS and HPSO) were applied on two different optimization scenarios to evaluate the uncertainty in results when started from different initial points and set with different random seeds. The optimization scenarios were applied five times each. The tested scenarios included the optimization of BRR applied on dataset A for simulating a simpler scenario, and the application of RNNs to dataset B for simulating a more complex scenario. The first scenario in Fig. 5a returns similar uncertainties for all the tested optimization procedures. While DE shows a higher variation in the first 10 iterations, DS and HPSO return comparable uncertainty results from iteration 10. Thus, other algorithms are expected to perform similarly when optimizing methane prediction by BRR for steady-state process conditions. The second scenario in Fig. 5b pictures higher differences between the applied optimizers. The DE has a high variation during the entire optimization process, probably due to its sensitivity to initial values. HPSO shows initially lower variance, even if at a higher cumulative error compared to the other two optimizers. But from step 20, its variance increases to values similar to the DE variance. Instead, DS shows a low variance throughout the entire optimization run, also achieving the best average result by the end of the iteration process.

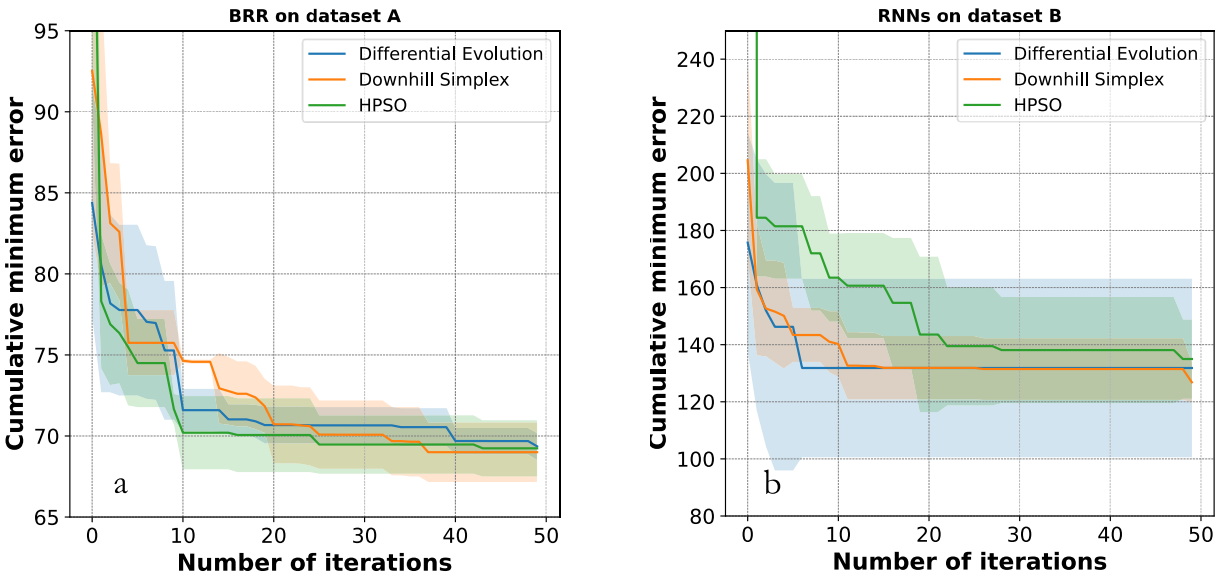


Fig. 5. Error progression with uncertainty for different optimization scenarios and methane prediction algorithms.

Table 2
Comparison of selected optimization scenarios for 50 and 2,000 optimizaion steps.

Optimization Model	Dataset A and Bayesian Ridge Regression			Dataset B and Recurrent Neuronal Networks		
	50-steps	2,000-steps[a]	Relative improvement	50-steps	2,000-steps[a]	Relative improvement
Downhill Simplex	65.9	63.6	3.5 %	112.6	102.3	9.1 %
Random Hill Climbing	67.6	64.7	4.3 %	115.9	112.8	2.6 %
Genetic Algorithm	69.4	**61.2**	11.8 %	99.2	**99.2**	0.1 %
Random search	66.5	65.1	2.1 %	116.4	101.0	13.2 %

[a] Bold and underlined errors indicate best error values for the relative optimization instance.

Table 3
Performance evaluation of meta-tuning of individual base optimizers.

	Applied optimizer	Bayesian Ridge Regression			Recurrent Neural Networks		
		Average time per evaluation[a] [h]	Best test RMSSE[b] [%]	Super-generations before reaching best test result	Average time per evaluation[a] [h]	Best test RMSSE[b] [%]	Iterations before reaching best test result
Dataset A	Hill Climbing	2.7	**63.6**	5	2.9	90.7	1
	Random Hill Climbing	2.8	**63.6**	8	3.3	88.2	2
	Bayesian	5.5	68.9	5	7.8	86.3	6
	Downhill Simplex	2.5	70.3	1	4.2	83.8	1
	Simulated Annealing	**2.3**	**63.6**	5	3.5	93.2	2
	PSO	24.8	63.9	3	35.7	81.8	1
	HPSO	12.5	64.0	2	37.4	89.1	3
	Differential evolution	32.5	67.0	1	36.6	81.7	2
	Genetic Algorithm	8.7	64.9	2	20.6	75.0	2
Dataset B	Hill Climbing	1.5	109.6	1	2.3	122.9	3
	Random Hill Climbing	1.4	115.2	1	2.3	167.6	1
	Bayesian Search	1.6	102.6	1	2.0	124.7	1
	Downhill Simplex	1.5	115.7	1	2.6	146.1	1
	Simulated Annealing	**1.3**	108.1	1	2.5	136.8	3
	Particle swarm optimization	9.4	106.9	1	15.2	121.4	1
	Hierarchical particle swarm optimization	5.4	98.4	3	37.4	99.7	1
	Differential evolution	7.2	99.2	1	16.6	98.4	1
	Genetic Algorithm	5.8	98.3	3	10.9	**94.4**	2

[a] Bold and underlined evaluations times indicate best times for the relative dataset. Underlined evaluation times indicate best times for relative dataset and applied prediction method.

[b] Bold and underlined errors indicate best error values for the relative dataset. Underlined errors indicate best error values for relative dataset and applied prediction method.

Table 4
Tuned parameters for applied optimization algorithms.

Applied Optimizer		Default value	Tuned[a]			
			Dataset A		Dataset B	
			BRR[b]	RNN[c]	BRR[b]	RNN[c]
Hill climbing	ε	0.03	0.1	0.3	–	–
	Number neighbors	3	14	15	–	–
	Distribution	Normal	Gumble	Logistic	–	–
Random Hill Climbing	ε	0.03	0.0	–	–	–
	Number neighbors	3	8	–	–	–
	Distribution	normal	Gumble	–	–	–
	Number restart iter.	10	20	–	–	–
Simulated Annealing	ε	0.03	0.0	0	–	–
	Number neighbors	3	4	9	–	–
	Distribution	Normal	Normal	Gumbel	–	–
	Annealing rate	0.97	0.95	0.9	–	–
	Starting temperature	1	1.2	2.4	–	–
Downhill Simplex	α	1	–	1.7	–	–
	β	0.5	–	0.5	–	–
	γ	2	–	2.5	–	–
	σ	0.5	–	0.8	–	–
Bayesian Search	ξ	0.3	–	–	–	–
	Number restart iter.	0	–	–	–	–
Particle Swarm Optimization	Number particles	5	6	5	–	–
	W	0.5	0.8	0.1	–	–
	c_1	2	0	2.2	–	–
	c_2	2	0.3	1.5	–	–
Hybrid Particle Swarm Optimization	Number particles	5	–	–	5	10
	W	0.5	–	–	0.9	0.2
Differential Evolution	Population size	5	–	12	12	19
	F	0.5	–	2.2	1.5	2.9
	c_r	0.7	–	0.4	0.9	0.4
Genetic Algorithm	Mating parents %	50	47	41	86	33
	Kept parents %	60	64	38	26	97
	Parents selection	Steady state	Steady state	Rank	Random	Random
	Crossover	Single	Uniform	Single	Uniform	Single
	Lower mutation	0.08	0.1	0.01	0.01	0.01
	Higher mutation	0.2	0.11	0.35	0.11	0.2

[a] Several optimization instances do not show improvement with tuned parameters. Thus, default values were used for optimal results. These cases are marked by the sign "–".
[b] Bayesian Ridge Regression.
[c] Recurrent Neural Network.

3.1.4. Number of iterations

For analysis of possible optimization improvements, best performing base optimizers (DS, RHC and GA) and the benchmark optimizer (RS) were evaluated and compared for 2,000 iterations steps (Table 2). While more significant improvements can be noticed in the second scenario (dataset B and RNNs), the difference between running the algorithms for 50 versus 2,000 optimization steps generally does not show absolute improvements higher than 5 %. Exceptions include the GA in the first scenario and RS and DS in the second, which show relative improvements close to 10 %. These findings suggest that increasing the number of iterations may be beneficial in such cases. Moreover, the GA demonstrates to be still the best performing among the considered algorithms, and the RS also proves effective in finding HPs combinations leading to optimal results.

3.2. Evaluation of meta-tuning

The meta-tuning includes applying a PSO (the meta-PSO) for the optimization of the optimizers' TPs. The meta-PSO is programmed to find the best TPs combination depending on the best error found by the tuned optimizer.

3.2.1. Dataset A

Individual results of the identification of TP of all the applied optimizers and datasets are summarized in Table 4. In comparison to non-tuned results of DS, only five out of nine optimizers (HC, RHC, SA, PSO and GA) demonstrated improvements for TP optimization of BRR. However, all optimizers are able to reach a RMSSE lower than 70 %. In general, the models that perform best also show a higher number of super-generations required for reaching optimal results. Slight improvements can also be observed for meta-tuning of RNNs on dataset A, while the best-performing RNN is still outperformed by the worst performing BRR model. Furthermore, results indicate that population-based optimizers require substantially longer computation times for parameter tuning compared to the other methods.

3.2.2. Dataset B

While most of the models did not benefit from the meta-tuning when applied to dataset B, the population-based optimizers all improved (except for PSO). In particular, the GA is the best performing optimizer on both the optimization of BRR and RNNs, considerably improving its previous performances. In general, the best-performing model applied on dataset B is now the meta-tuned GA, demonstrating that complex optimizers tuning complex models in the prediction of non-steady-state datasets require meta-tuning for optimal performances.

3.2.3. Evaluation of tuning parameters

The best performing TP for each optimization scenario are shown in Table 4. The optimizers that showed higher advantages when tuned with the meta-PSO are the GA and the DE. Moreover, other four optimizers (HC, SA, PSO and GA) benefitted two times out of four of the meta-tuning, and the rest of the optimizers benefitted only one-fourth of the times. No optimizer was able to perform in all four optimization scenarios better with the standard parameters rather than with the tuned

parameters. Individual patterns can be concluded from the optimized TP. DE shows in general improved performances with a higher F value. This indicates the importance of a wide exploration of the search space, favoring a larger step size to jump over local minima instead of making small incremental changes to the HPs (Price et al., 2005). The HC shows a tendency to perform better on Dataset A with a higher amount of neighbours, indicating the presence of several local optima that the algorithm is able to better evaluate going towards different directions (Aarts and Lenstra, 2003). The other optimizers do not show generalizable patterns but exhibit several performance improvements for each case.

3.2.4. Discussion

Results show that, in general, the prediction of steady-state methane production through BRR and RNN models does not require extensive HP optimizations within the applied pipeline. Repeating the optimization process of 50 iterations three times (resulting in a total of 150 iterations) is generally sufficient to reach optimal results depending on the chosen algorithm. Nevertheless, the application of several optimizers for the tuning of BRR on dataset A for 2,000 steps demonstrated that longer optimization times could improve the prediction further, overperforming any meta-tuned and non-meta-tuned algorithm at 50 iterations (see Table 2 and Table 3). The non-steady-state dataset B instead benefitted from meta-tuning, probably due to the higher complexity of the dataset, which requires specific HP combinations to be properly modelled. While the BRR reached already high performances without meta-tuning, meta-tuning of RNNs resulted in an absolute performance increase of 4.8 % RMSSE, with the meta-tuned GA outperforming any other non-meta-tuned optimizer. Moreover, the meta-tuned GA overperformed even the 2,000 iterations tested models, demonstrating that complex prediction algorithms applied to complex datasets benefit from additional meta-tuning. In general, BS performs well in any optimization instance, even without meta-tuning, and is therefore the suggested optimizer for general purposes when the complexity of the model or of the dataset is unknown. For optimal optimization performances in any context, a 50 iteration meta-tuning might be applied and subsequently use the tuned parameters for a 2,000 iteration optimization process. Since the computational effort of meta-tuning followed by a 2,000-step optimization can be high depending on the application, operators may need to weigh the trade-offs. In cases where slightly lower prediction accuracy is acceptable, simpler approaches– such as only performing meta-tuning or only running a 2,000-step optimization – may be sufficient. Further improvement of the results might be obtained with specific algorithm initialization strategies (Cheng et al., 2018).

4. Conclusions

This study demonstrates that the choice of optimization strategies significantly impacts performances of ML and DL models predicting methane production from AD. For steady-state datasets, 50 iterations repeated three times typically suffice for optimal results, with HPSO being the best performer (62.8 % RMSSE). Moreover, a GA optimized for 2000 iterations overperforms it (61.2 % RMSSE). However, dynamic datasets and complex models, such as RNNs, benefit from extended optimization or meta-tuning, with DE and GA showing improved results in such cases (94.4 % and 98.4 % RMSSE, respectively). BS was proven to be a robust general-purpose optimizer, performing consistently well without meta-tuning. The tested strategies can be used for tuning ML models when applied to stochastic MPC for industrial plant control. The utilization of training datasets from several reactors might be further tested for improved prediction accuracy and model robustness.

CRediT authorship contribution statement

Alberto Meola: Writing – original draft, Software, Methodology, Investigation, Data curation, Conceptualization. **Klara Wolf:** Visualization, Software, Investigation, Data curation. **Sören Weinrich:** Writing – review & editing, Supervision, Resources, Project administration, Funding acquisition, Conceptualization.

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.

Acknowledgments

The authors are thankful for funding from the European Regional Development Fund (ERDF) of the research project Integrated control of biogas plants for flexibilization and energetic optimisation (grant 100267056) and from the German Federal Ministry of Food and Agriculture of the junior research group on Simulation, monitoring and control of anaerobic digestion plants (grant 2219NR333). Martin Bogdan is acknowledged for supporting the writing of the manuscript.

Appendix A. Supplementary data

Supplementary data to this article can be found online at https://doi.org/10.1016/j.biortech.2025.132654.

Data availability

Data will be made available on request.

References

Aarts, E.H.L., Lenstra, J.K., 2003. Local Search in Combinatorial Optimization. Princeton University Press.

Abadi, M., Barham, P., Chen, J., Chen, Z., Davis, A., Dean, J., Devin, M., Ghemawat, S., Irving, G., Isard, M., et al., 2016. {TensorFlow}: a system for {Large-Scale} machine learning. In: 12th USENIX Symposium on Operating Systems Design and Implementation (OSDI 16). pp. 265–283.

Abu Qdais, H., Bani Hani, K., Shatnawi, N., 2010. Modeling and optimization of biogas production from a waste digester using artificial neural network and genetic algorithm. Resour. Conserv. Recycl. 54, 359–363.

Batstone, D.J., Keller, J., Angelidaki, I., Kalyuzhnyi, S.V., Pavlostathis, S.G., Rozzi, A., Sanders, W.T.M., Siegrist, H., Vavilin, V.A., 2002. The IWA anaerobic digestion model No 1 (ADM1). Water Sci. Technol. 45, 65–73. https://doi.org/10.2166/wst.2002.0292.

Beltramo, T., Klocke, M., Hitzmann, B., 2019. Prediction of the biogas production using GA and ACO input features selection method for ANN model. Inf. Process. Agric. 6, 349–356.

Beltramo, T., Ranzan, C., Hinrichs, J., Hitzmann, B., 2016. Artificial neural network prediction of the biogas flow rate optimised with an ant colony algorithm. Biosyst. Eng. 143, 68–78.

Bengio, Y., 2000. Gradient-based optimization of hyperparameters. Neural Comput. 12, 1889–1900.

Capizzi, G., Lo Sciuto, G., Napoli, C., Woźniak, M., Susi, G., 2020. A spiking neural network-based long-term prediction system for biogas production. Neural Netw. 129, 271–279.

Cheng, M.-Y., Huang, K.-Y., Hutomo, M., 2018. Multiobjective dynamic-guiding PSO for optimizing work shift schedules. J. Constr. Eng. Manag. 144, 04018089.

Cho, K., van Merriënboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H., Bengio, Y., 2014. Learning phrase representations using RNN encoder–decoder for statistical machine translation. In: Moschitti, A., Pang, B., Daelemans, W. (Eds.), Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP). Presented at the EMNLP 2014. Association for Computational Linguistics, Doha, Qatar, pp. 1724–1734. https://doi.org/10.3115/v1/D14-1179.

Deng, Y., Zhang, Y., Zhao, Z., 2024. A data-driven approach for revealing the linkages between differences in electrochemical properties of biochar during anaerobic digestion using automated machine learning. Sci. Total Environ. 927, 172291.

Diaz, G.I., Fokoue-Nkoutche, A., Nannicini, G., Samulowitz, H., 2017. An effective algorithm for hyperparameter optimization of neural networks. IBM J. Res. Dev. 61, 9:1–9:11.

Gelman, A., Carlin, J.B., Stern, H.S., Dunson, D.B., Vehtari, A., Rubin, D.B., 2013. Bayesian Data Analysis, third ed. Chapman and Hall/CRC, New York.

Gogna, A., Tayal, A., 2013. Metaheuristics: review and application. J. Exp. Theor. Artif. Intell. 25, 503–526.

Gorgolis, N., Hatzilygeroudis, I., Istenes, Z., Gyenne, L.- G., 2019. Hyperparameter optimization of LSTM network models through genetic algorithm. In: 2019 10th International Conference on Information, Intelligence, Systems and Applications

(IISA). Presented at the 2019 10th International Conference on Information, Intelligence, Systems and Applications (IISA). pp. 1–4.
Harris, C.R., Millman, K.J., van der Walt, S.J., Gommers, R., Virtanen, P., Cournapeau, D., Wieser, E., Taylor, J., Berg, S., Smith, N.J., Kern, R., Picus, M., Hoyer, S., van Kerkwijk, M.H., Brett, M., Haldane, A., del Río, J.F., Wiebe, M., Peterson, P., Gérard-Marchant, P., Sheppard, K., Reddy, T., Weckesser, W., Abbasi, H., Gohlke, C., Oliphant, T.E., 2020. Array programming with NumPy. Nature 585, 357–362.
Hochreiter, S., Schmidhuber, J., 1997. Long short-term memory. Neural Comput. 9, 1735–1780.
Hyndman, R.J., Koehler, A.B., 2006. Another look at measures of forecast accuracy. Int. J. Forecast. 22, 679–688. https://doi.org/10.1016/j.ijforecast.2006.03.001.
Iida, K., 1992. Studies on the Optimal Search Plan, Lecture Notes in Statistics. Springer, New York, NY.
Jacob, S., Banerjee, R., 2016. Modeling and optimization of anaerobic codigestion of potato waste and aquatic weed by response surface methodology and artificial neural network coupled genetic algorithm. Bioresour. Technol. 214, 386–395.
Katoch, S., Chauhan, S.S., Kumar, V., 2021. A review on genetic algorithm: past, present, and future. Multimed. Tools Appl. 80, 8091–8126. https://doi.org/10.1007/s11042-020-10139-6.
Kennedy, J., Eberhart, R., 1995. Particle swarm optimization. In: Proceedings of ICNN'95 – International Conference on Neural Networks. Presented at the Proceedings of ICNN'95 – International Conference on Neural Networks. pp. 1942–1948 vol.4.
Kirkpatrick, S., Gelatt, C.D., Vecchi, M.P., 1983. Optimization by simulated annealing. Science 220, 671–680. https://doi.org/10.1126/science.220.4598.671.
Le, T.T., Fu, W., Moore, J.H., 2020. Scaling tree-based automated machine learning to biomedical big data with a feature set selector. Bioinformatics 36, 250–256.
Li, J., Zhang, L., Li, C., Tian, H., Ning, J., Zhang, J., Tong, Y.W., Wang, X., 2022. Data-driven based in-depth interpretation and inverse design of anaerobic digestion for CH4-rich biogas production. ACS EST Eng. 2, 642–652.
Li, L., Jamieson, K., DeSalvo, G., Rostamizadeh, A., Talwalkar, A., 2018. Hyperband: a novel bandit-based approach to hyperparameter optimization. J. Mach. Learn. Res. 18, 1–52.
Ling, J.Y.X., Chan, Y.J., Chen, J.W., Chong, D.J.S., Tan, A.L.L., Arumugasamy, S.K., Lau, P.L., 2024. Machine learning methods for the modelling and optimisation of biogas production from anaerobic digestion: a review. Environ. Sci. Pollut. Res. 31, 19085–19104.
Lo Sciuto, G., Susi, G., Cammarata, G., Capizzi, G., 2016. A spiking neural network-based model for anaerobic digestion process. In: 2016 International Symposium on Power Electronics, Electrical Drives, Automation and Motion (SPEEDAM). Presented at the 2016 International Symposium on Power Electronics, Electrical Drives, Automation and Motion (SPEEDAM). pp. 996–1003.
Mauky, E., Weinrich, S., Jacobi, H.-F., Nägele, H.-J., Liebetrau, J., Nelles, M., 2017. Demand-driven biogas production by flexible feeding in full-scale – process stability and flexibility potentials. Anaerobe Biogas Sci. 2016 (46), 86–95.
McKinney, W., 2010. Data structures for statistical computing in python. In: van der Walt, S., Millman, J. (Eds.), Proceedings of the 9th Python in Science Conference, pp. 56–61.
Meola, A., Weinrich, S., 2024. Full-scale dynamic anaerobic digestion process simulation with machine and deep learning algorithms at intra-day resolution. Review at Applied Energy.
Meola, A., Winkler, M., Weinrich, S., 2023. Metaheuristic optimization of data preparation and machine learning hyperparameters for prediction of dynamic methane production. Bioresour. Technol. 372, 128604. https://doi.org/10.1016/j.biortech.2023.128604.
Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., Duchesnay, É., 2011. Scikit-learn: machine learning in python. J. Mach. Learn. Res. 12, 2825–2830.
Powell, M.J.D., 1973. On search directions for minimization algorithms. Math. Program. 4, 193–201. https://doi.org/10.1007/BF01584660.
Price, K., Storn, R., Lampinen, J., 2005. Differential evolution-a practical approach to global optimization. Natural Computing – NC. https://doi.org/10.1007/3-540-31306-0.
Putra, L.A., Köstler, M., Grundwürmer, M., Li, L., Huber, B., Gaderer, M., 2025. State estimation of a biogas plant based on spectral analysis using a combination of machine learning and metaheuristic algorithms. Appl. Energy 377, 124447.
Ratnaweera, A., Halgamuge, S.K., Watson, H.C., 2004. Self-organizing hierarchical particle swarm optimizer with time-varying acceleration coefficients. IEEE Trans. Evol. Comput. 8, 240–255. https://doi.org/10.1109/TEVC.2004.826071.
Russell, S.J., Norvig, P., 2016. Artificial Intelligence: A Modern Approach. Pearson.
Sathish, S., Vivekanandan, S., 2016. Parametric optimization for floating drum anaerobic bio-digester using Response Surface Methodology and Artificial Neural Network. Alex. Eng. J. 55, 3297–3307. https://doi.org/10.1016/j.aej.2016.08.010.
Selman, B., Gomes, C.P., 2006. Hill-climbing search. In: Encyclopedia of Cognitive Science. John Wiley & Sons, Ltd. doi: 10.1002/0470018860.s00015.
Seo, K.W., Seo, J., Kim, K., Ji Lim, S., Chung, J., 2021. Prediction of biogas production rate from dry anaerobic digestion of food waste: process-based approach vs. recurrent neural network black-box model. Bioresour. Technol. 341, 125829.
Shi, Y., Eberhart, R., 1998. A modified particle swarm optimizer. In: 1998 IEEE International Conference on Evolutionary Computation Proceedings. IEEE World Congress on Computational Intelligence (Cat. No.98TH8360). Presented at the 1998 IEEE International Conference on Evolutionary Computation Proceedings. IEEE World Congress on Computational Intelligence, pp. 69–73.
Simon Blanke, 2020. Gradient-Free-Optimizers: Simple and reliable optimization with local, global, population-based and sequential techniques in numerical search spaces.
Storn, R., Price, K., 1997. Differential evolution – a simple and efficient heuristic for global optimization over continuous spaces. J. Glob. Optim. 11, 341–359.
Sun, J., Xu, Y., Nairat, S., Zhou, J., He, Z., 2023. Prediction of biogas production in anaerobic digestion of a full-scale wastewater treatment plant using ensembled machine learning models. Water Environ. Res. 95, e10893.
Thrän, D., Deprie, K., Dotzauer, M., Kornatz, P., Nelles, M., Radtke, K.S., Schindler, H., 2023. The potential contribution of biogas to the security of gas supply in Germany. Energy Sustain. Soc. 13, 12.
Virtanen, P., Gommers, R., Oliphant, T.E., Haberland, M., Reddy, T., Cournapeau, D., Burovski, E., Peterson, P., Weckesser, W., Bright, J., van der Walt, S.J., Brett, M., Wilson, J., Millman, K.J., Mayorov, N., Nelson, A.R.J., Jones, E., Kern, R., Larson, E., Carey, C.J., Polat, İ., Feng, Y., Moore, E.W., VanderPlas, J., Laxalde, D., Perktold, J., Cimrman, R., Henriksen, I., Quintero, E.A., Harris, C.R., Archibald, A.M., Ribeiro, A. H., Pedregosa, F., van Mulbregt, P., 2020. SciPy 1.0: fundamental algorithms for scientific computing in Python. Nat. Methods 17, 261–272.
Wang, Y., Huntington, T., Scown, C.D., 2021. Tree-based automated machine learning to predict biogas production for anaerobic co-digestion of organic waste. ACS Sustain. Chem. Eng. 9, 12990–13000.
Weinrich, S., Nelles, M., 2021. Systematic simplification of the Anaerobic Digestion Model No. 1 (ADM1) – Model development and stoichiometric analysis. Bioresour. Technol. 333, 125124.
Yildirim, O., Ozkaya, B., 2023. Prediction of biogas production of industrial scale anaerobic digestion plant by machine learning algorithms. Chemosphere 335, 138976.