A survey on multi-objective hyperparameter optimization algorithms for Machine Learning

A survey on multi-objective hyperparameter optimization algorithms for Machine Learning

Alejandro Morales-Hernández[1,2*], Inneke Van Nieuwenhuyse[1,2] and Sebastian Rojas Gonzalez[1,2,3]

[1*]Research Group Logistics, Hasselt University, Belgium.
[2]Data Science Institute, Hasselt University, Belgium.
[3]Surrogate Modeling Lab, Ghent University, Belgium.

*Corresponding author(s). E-mail(s): alejandro.moraleshernandez@uhasselt.be;
Contributing authors: inneke.vannieuwenhuyse@uhasselt.be; sebastian.rojasgonzalez@uhasselt.be;

Abstract

Hyperparameter optimization (HPO) is a necessary step to ensure the best possible performance of Machine Learning (ML) algorithms. Several methods have been developed to perform HPO; most of these are focused on optimizing one performance measure (usually an error-based measure), and the literature on such single-objective HPO problems is vast. Recently, though, algorithms have appeared which focus on optimizing multiple conflicting objectives simultaneously. This article presents a systematic survey of the literature published between 2014 and 2020 on multi-objective HPO algorithms, distinguishing between metaheuristic-based algorithms, metamodel-based algorithms and approaches using a mixture of both. We also discuss the quality metrics used to compare multi-objective HPO procedures, and present future research directions.

Keywords: hyperparameter optimization, multi-objective optimization, metamodel, meta-heuristic, machine learning

1 Introduction

Nowadays, Artificial Intelligence (AI) is omnipresent in everyday life. Current technological advances allow us to analyze huge amounts of data to generate knowledge that is used in many different ways, e.g. for automatic user recommendations (Cai, Hu, Zhao, Zhang, & Chen, 2020), image recognition (Andreopoulos & Tsotsos, 2013; Phillips et al., 2005), and supporting healthcare related tasks (F. Jiang et al., 2017). In general, AI can been seen as a computer technology capable of carrying out functions that traditionally required human intelligence (Ertel, 2018). Although learning is a key element in many areas of artificial intelligence, the very concept of learning is mainly studied in the Machine Learning (ML) subfield. According to T.M. Mitchell et al. (1997), "a computer program is said to learn from experience **E** with respect to some class of tasks **T** and performance measure **P** if its performance at tasks in **T**, as measured by **P**, improves with experience **E**". ML algorithms and their parameters must be intelligently configured to make the most of the data. Those parameters that need to be specified *before* training the algorithm are usually referred to as *hyperparameters*: they influence the learning process, but they are not optimized as part of the training algorithm.

The impact of these hyperparameters on algorithm performance should not be underestimated (Cooney, Korik, Folli, & Coyle, 2020; Kim, Reddy, Yun, & Seo, 2017; Kong et al., 2017; Singh, Kumar, & Kaur, 2020); yet, their optimization (hereafter referred to as *hyperparameter optimization* or HPO) is a challenging task, as traditional optimization methods are often not applicable (Luo, 2016). Indeed, classic convex optimization methods such as gradient descent tend to be ill-suited for HPO, as the measure to optimize is usually a non-convex and non-differentiable function (Parsa, Ankit, Ziabari, & Roy, 2019; Stamoulis, Chin, et al., 2018). Furthermore, the hyperparameters to optimize may be discrete, categorical and/or continuous (typical hyperparameters for an Artificial Neural Network (ANN), for instance, are the number of layers, the number of neurons per layer, the type of optimizer, and the learning rate). The search space can also contain *conditional hyperparameters*; e.g., the hyperparameters in a support vector machine algorithm depend on the type of kernel used. Finally, the time needed to train a machine learning model with a given hyperparameter configuration on a *given* dataset may already be substantial, particularly for moderate to large datasets; as a common HPO algorithm requires multiple such training cycles, the algorithm itself needs to be computationally efficient to be useful in practice.

HPO should not be confused with the more general topic of *automatic algorithm configuration*, which is much broader in scope (see Hutter, Hoos, Leyton-Brown, and Stützle (2009); López-Ibáñez, Dubois-Lacoste, Cáceres, Birattari, and Stützle (2016) for examples on this topic). Analogous to HPO, automatic algorithm configuration tries to find parameter values for which an algorithm achieves the best possible performance on the input data; yet, the target algorithm does not necessarily carry out a learning process of a certain task (classification, image recognition, or other).

HPO has gained increasing attention in recent years, probably spurred by the popularity of deep learning algorithms, which have demanding characteristics (e.g., the need for large amounts of data and time to train the models, high model complexity, and a diverse mix of hyperparameter types). Previously, analysts tended to use simple methods to look for the "best" hyperparameter settings. The most basic of these is grid search (Montgomery, 2017): the user creates a set of possible values for each hyperparameter, and the search evaluates the Cartesian product of these sets. Although this strategy is easy to implement and easy to understand, its performance is influenced by the number of hyperparameters to optimize, and the (number of) values chosen on the grid. Random search (Bergstra & Bengio, 2012) provides an alternative to grid search, and tends to be popular when some of the hyperparameters are more important than others; e.g, learning rate and momentum are critical to guarantee a faster convergence of neural networks (C. Guo, Li, Hu, & Yan, 2020). More advanced optimization methods have also been put forward, such as population-based algorithms (Keshtkaran & Pandarinath, 2019), meta-learning methods (Bui & Yi, 2020), neural architecture search (NAS) methods (Jing, Lin, & Wang, 2020), and surrogate-based or model-based optimization approaches (Kim & Chung, 2019).

Our work aims to provide an overview of the state-of-the-art in the field of *multi-objective hyperparameter optimization* for machine learning algorithms, highlighting the approaches currently used in the literature, the typical performance measures used as objectives, and discussing remaining challenges in the field. Multi-objective optimization is particularly relevant for HPO, as different conflicting objectives may be important for the analyst (e.g., error-based performance of the target ML algorithm, inference time, model size, energy consumption, etc.). To the best of our knowledge, our work presents the first comprehensive review of these multi-objective HPO approaches. Previous reviews (Feurer & Hutter, 2019; Hutter, Lücke, & Schmidt-Thieme, 2015; Luo, 2016; Talbi, 2021; Yang & Shami, 2020) mainly discuss single-objective HPO approaches, often focusing on particular contexts (such as biomedical data analysis), specific target algorithms (such as Deep Neural Networks) or specific approaches (Sequential Model-based Bayesian Optimization, multi-fidelity approaches). While two of the most recent surveys (Feurer & Hutter, 2019; Talbi, 2021) mention multi-objective HPO on the sidelines, they only list some examples or common strategies relevant to this topic, without discussing the actual approaches.

The remainder of this article is organized as follows. Section 2 discusses the methodology used in the literature search. Section 3 formalizes the concepts of single- and multi-objective hyperparameter optimization, and discusses the most commonly used performance measures in HPO algorithms. Section 4 discusses the existing methods for multi-objective hyperparameter optimization. Finally, section 5 summarizes the findings, highlighting open challenges and potential solutions.

2 Methodology

Given the remarkable surge in publications on HPO since 2014, we focused on research published between 2014 and 2020. Figure 1 shows an overview of the search and selection process.

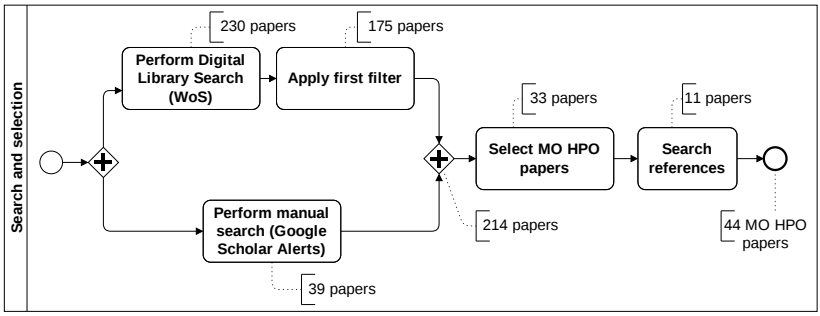


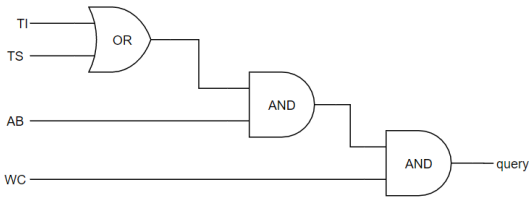
Figure 1: Overview of search and selection process

We first performed a WoS (Web of Science) search, using the search terms shown in Table 1. Although the main focus is on multi-objective HPO, we also consider the occurrence of the phrase "single objective" in the abstract (AB), as it is common to transform multiple objectives into a single objective by means of a scalarization function. As the use of surrogates is common in single-objective HPO for deep learning networks (e.g., Sjöberg, Önnheim, Gustavsson, and Jirstrand (2019); Victoria and Maragatham (2021); Wistuba, Schilling, and Schmidt-Thieme (2018)), we also searched for articles mentioning the terms "surrogate", "deep learning", "neural networks", "Gaussian process", and "kriging" in the abstract. The choice of hyperparameters is also related to overfitting (Feurer & Hutter, 2019). Finally, we also include the term "constraint", as the required performance targets (e.g., maximum memory consumption, training time (Hu, Jin, Liu, & Zhang, 2019; Stamoulis, Cai, Juan, & Marculescu, 2018)) may be presented as constraints in (multi-objective) HPO. We limited our search to publications (including conference proceedings, articles, book chapters and meeting abstracts) in computer science related categories (WC).

We subsequently completed the set of papers through (1) scanning suggestions of papers on Google Scholar alerts, and (2) a reference search. We limited the latter to electronic collections only, and solely considered journals/conference proceedings/workshop proceedings that were indexed on WoS (for the WoS journals, we included accepted preprints of forthcoming articles).

The papers obtained through the WoS and manual search were manually filtered based on the title and abstract, to ensure they were related to the topic of discussion. We filtered out irrelevant papers, such as those that

Table 1: Search term details



TI	hyperparameter optimization, hyperparameter tuning, parameter optimization, hyperparameter, parameter tuning
TS	hyperparameter optimization, hyperparameter tuning
AB	neural networks, deep learning, constraint, overfitting, multiobjective, multi objective, multi-objective, many-objective, many objective, single objective, surrogate, gaussian process, kriging
WC	Computer Science, Artificial Intelligence Computer Science, Information Systems Computer Science, Theory & Methods Computer Science, Interdisciplinary Applications

TI: Title, TS: Topic, AB: Abstract, WC: Web of Science Categories

focus on optimization of industrial processes (Chen, Jiang, Chang, & Chen, 2014), meta-learning (Vanschoren, 2019), optimization of internal parameters (Wawrzyński, 2017), and papers related to AutoML systems that are not focused on hyperparameter optimization (such as model selection algorithms (Silva et al., 2016; van Rijn, Abdulrahman, Brazdil, & Vanschoren, 2015) or pure feature selection methods (Hegde & Mundada, 2020)). Neural Architecture Search (NAS) is usually considered as a distinct category with its own methods and techniques for optimizing the structure of a neural network; hence, articles on NAS were only considered when the problem was addressed as an HPO problem. Articles focusing on more specific aspects of NAS (such as Negrinho et al. (2019)) are beyond the scope of this research.

A full read of the articles, combined with a reference search, resulted in a final selection of 44 relevant articles. Most of these articles (about 60%) were published in conferences or workshops, though there has been a notable increase in scientific journal articles in 2020 (see Figure 2); these were mainly published in Q1/Q2 journals belonging to the Computer Science field.

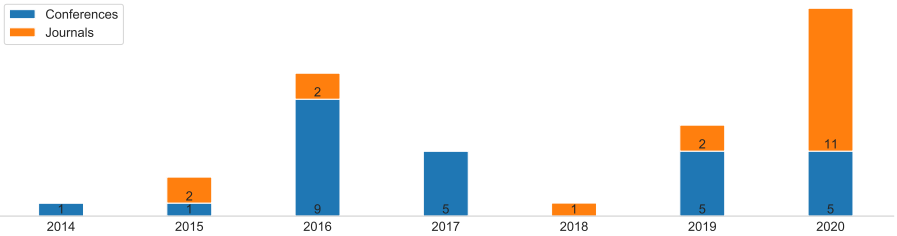


Figure 2: Number of articles that address multi-objective HPO, according to the publication source (2014-2020)

3 HPO: Concepts and performance measures

Section 3.1 provides an overview of the basic concepts related to HPO, while Section 3.2 discusses the main performance measures (objectives) used in such optimization. Finally, Section 3.3 discusses the quality metrics used for comparing the performance of multi-objective HPO algorithms.

3.1 HPO: Concepts and terminology

In mathematics and computer science, an algorithm is a finite sequence of well-defined instructions that, when fed with a set of initial inputs, eventually produces an output. Figure 4 shows that in HPO, the optimization algorithm forms an "outer" shell of optimization instructions; the "inner" optimization refers to the training and cross-validation of the target ML algorithm (e.g., ANN, SVM, etc.). This inner optimization trains the target algorithm to perform the task it should perform (e.g., predicting house prices from a data set, using a set of features). In turn, the HPO algorithm takes the hyperparameters of the target ML algorithm as input, and produces a number of performance measures as output (e.g., RMSE, energy consumption, etc.). The aim of the HPO algorithm is to optimize the set of hyperparameters, in view of obtaining the best possible outcomes for the performance measures considered.

More formally, the single-objective HPO problem can then be formalized as follows. Consider a target ML algorithm $\mathcal{A}$ with N hyperparameters, such that the n-th hyperparameter has a domain denoted by Λ_n. The overall *hyperparameter configuration space* is denoted as $\Lambda = \Lambda_1 \times \Lambda_2 \times ... \times \Lambda_N$. A vector of hyperparameters is denoted by $\boldsymbol{\lambda} \in \Lambda$, and an algorithm $\mathcal{A}$ with its hyperparameters set to $\boldsymbol{\lambda}$ is denoted by $\mathcal{A}_{\boldsymbol{\lambda}}$. In the case of HPO, the available data are split into a training set, a validation set and a test set. The *learning* process of the algorithm take place on the training set ($\mathcal{D}_{train}$), and is validated on the validation set ($\mathcal{D}_{valid}$). We can then formalize the *single-objective* HPO problem as (Feurer & Hutter, 2019):

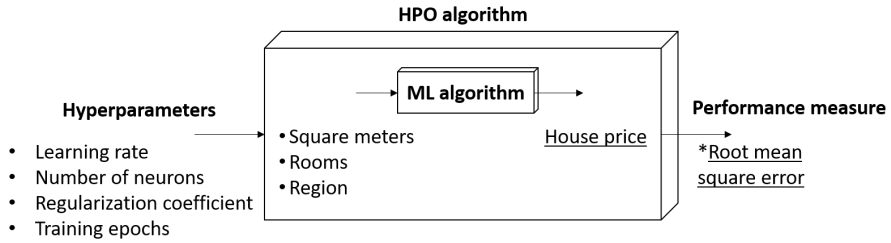


Figure 3: Input/Output of an HPO algorithm

Figure 4: Example of the interplay between a target ML algorithm (ANN, in this case) and the HPO algorithm.

$$\boldsymbol{\lambda}^* = \arg\min_{\boldsymbol{\lambda} \in \Lambda} V(\mathcal{L} \mid \mathcal{A}_{\boldsymbol{\lambda}}, \mathcal{D}_{train}, \mathcal{D}_{valid})$$

where $V(\mathcal{L} \mid \mathcal{A}_{\boldsymbol{\lambda}}, \mathcal{D}_{train}, \mathcal{D}_{valid})$ is a validation protocol that uses a loss function $\mathcal{L}$ to estimate the performance of a model $\mathcal{A}_{\boldsymbol{\lambda}}$ trained on $\mathcal{D}_{train}$ and validated on $\mathcal{D}_{valid}$. Popular choices for the validation protocol $V(\cdot)$ are the holdout and cross-validation process (see Bischl, Mersmann, Trautmann, and Weihs (2012) for an overview of validation protocols). Without loss of generality, we assume in the remainder of this article that the loss function should be minimized.

The previous definition can be readily extended to multi-objective optimization (see M. Li and Yao (2019)). Consider a multi-objective hyperparameter optimization problem with N hyperparameters and m performance measures (objectives). Each hyperparameter configuration $\boldsymbol{\lambda} \in \Lambda^N$ in the search space is then mapped to an objective vector $f \mid \mathcal{A}_{\boldsymbol{\lambda}} = (f_1 \mid \mathcal{A}_{\boldsymbol{\lambda}}, ..., f_m \mid \mathcal{A}_{\boldsymbol{\lambda}}) \subset \mathbb{R}^m$ in the objective space. The multi-objective HPO problem can then be formalized as follows (assuming that all performance measures should be minimized):

$$\boldsymbol{\lambda}^* = \arg\min_{\boldsymbol{\lambda} \in \Lambda} V(f \mid \mathcal{A}_{\boldsymbol{\lambda}}, \mathcal{D}_{train}, \mathcal{D}_{valid})$$

Usually, there is a trade-off among the different objectives: for instance, between the performance of a model and training time (increasing the accuracy of a model often requires larger amounts of data and, hence, a higher training time; see e.g., Rajagopal et al. (2020)), or between different error-based measures (e.g., between confusion matrix-based measures (Tharwat, 2020) of a binary classification problem; see Horn and Bischl (2016)). Considering these trade-offs is often crucial: e.g., in medical diagnostics (de Toro, Ros, Mota, & Ortega, 2002), the simultaneous consideration of objectives such as sensitivity and specificity is essential to determine if the machine learning model can

be used in practice. The goal in multi-objective HPO is to obtain the *Pareto-optimal* solutions, i.e., those solutions for which none of the objectives can be improved without negatively affecting any other objective. In the decision space, the set of optimal solutions is referred to as the *Pareto set*; in objective space, it yields the *Pareto front* (or Pareto frontier). The Pareto-optimal solutions are also referred to as the *non-dominated* solutions (Emmerich & Deutz, 2018).

It is crucial that HPO approaches (both single-objective and multi-objective) balance *exploration* and *exploitation* in their search process. Exploitation refers to the selection of hyperparameter configurations that are located in regions of the search space that have already shown to be promising (i.e., that are known to yield configurations with good performance), while exploration refers to the selection of configurations in regions where little is known yet about the related performance. In (general) multi-objective optimization problems, the multiple objectives are often *scalarized* into one single function, such that the problem can be solved as a single-objective problem. Care should be taken, though, when selecting the scalarization approach: e.g., not all approaches allow to detect non-convex parts of the front (see Miettinen and Mäkelä (2002) for further details about scalarization functions). Scalarization methods have also been applied in multi-objective HPO; see Section 4 for further details.

3.2 Multi-objective HPO: Typical objectives

Table 2 shows an overview and concise description of the most common error-based performance measures in multi-objective HPO. As evident from the table, for regression problems, these are commonly based on the squared errors. For classification problems, they are commonly related to the elements of the confusion matrix. By definition, a confusion matrix C has elements $C_{i,j}$ which show the number of observations known to be in class i and predicted to be in class j (see Figure 5 for the confusion matrix of a binary classification problem).

Table 2: Error-based measures used in multi-objective HPO algorithms. The description given for the classification metrics assumes a two-class problem; in multi-class settings, the metrics are computed for each individual class.

Type of problem	Performance metric	Description	References
Classification	Classification error	$\frac{FN+FP}{P+N}$	Abdolsh, Shilton, Rana, Gupta, and Venkatesh (2019); Albelwi and Mah. (2016); Binder, Moosbauer, Thomas, and Bischl (2020); Bouraoui, Jamoussi, and BenAyed (2018); Chin, Morcos, and Marculescu (2020); Elsken, Metzen, and Hutter (2019); Faris, Habib, Faris, Alomari, and Alomari (2020); Garrido and Hernández (2019); Hernández, Hernandez-Lobato, Shah, and Adams (2016); J. Jiang et al. (2020); Kim et al. (2017); Laskaridis, Venieris, Kim, and Lane (2020); Liang et al. (2019); Loni, Zoljodi, Sinaei, Daneshtalab, and Sjödin (2019); Lu, Deb, Goodman, Banzhaf, and Boddeti (2020); Mostafa, Mendonça, Ravelo-Garcia, Juliá-Serdá, and Morgado-Dias (2020); Parsa et al. (2019,?); H. Qin, Shinozaki, and Duh (2017); Rajagopal et al. (2020); Salt, Howard, Indiveri, and Sandamirskaya (2019); Shah and Ghahramani (2016); Smithson, Yang, Gross, and Meyer (2016); Sopov and Ivanov (2015); Wang, Sun, Xue, and Zhang (2019); Wang, Xue, and Zhang (2020); C. Zhang, Lim, Qin, and Tan (2016)

Continued on next page

Type of problem	Performance metric	Description	References
	Recall / Sensitivity	$\frac{TP}{TP+FN}$	Chatelain, Adam, Lecourtier, Heutte, and Paquet (2007); Ekbal and Saha (2015, 2016); Mostafa et al. (2020); Pathak, Shukla, and Arya (2020); Singh et al. (2020)
	Precision	$\frac{TP}{TP+FP}$	Ekbal and Saha (2015, 2016)
	Specificity	$\frac{TN}{TN+FP}$	(Mostafa et al., 2020; Pathak et al., 2020; Singh et al., 2020)
	False positive rate	$\frac{FP}{FP+TN}$	Chatelain et al. (2007); Horn and Bischl (2016); Horn, Dagge, Sun, and Bischl (2017)
	False negative rate	$\frac{FN}{FN+TP}$	Horn and Bischl (2016); Horn et al. (2017)
Regression	Root mean square error	$\sqrt{\frac{\sum (Real-Prediction)^2}{Total\ of\ observations}}$	Juang and Hsu (2014); Magda, Martinez-Alvarez, and Cuenca-Asensi (2017); Martinez-de Pison, Gonzalez-Sendino, Aldama, Ferreiro, and Fraile (2017)
	Mean square error	$\frac{\sum (Real-Prediction)^2}{Total\ of\ observations}$	Smith and Jin (2014)
	Sum square error	$\sum (Real - Prediction)^2$	Salt et al. (2019)

Continued on next page

Type of problem	Performance metric	Description	References
Speech recognition	Word error rate	Measures how different the recognised word is from reference word	Chandra and Lane (2016); Shinozaki, Watanabe, and Duh (2020); Tanaka et al. (2016)
Image recognition	Segmentation accuracy	Computed as two times the area of overlap of two images divided by the total number of pixels in both images	Baldeon and Lai-Yuen (2020); Calisto and Lai-Yuen (2020)
	Mutual Information	Measures the dependencies between two images, or the amount of information that one image contains about the other	Albelwi and Mah. (2016)

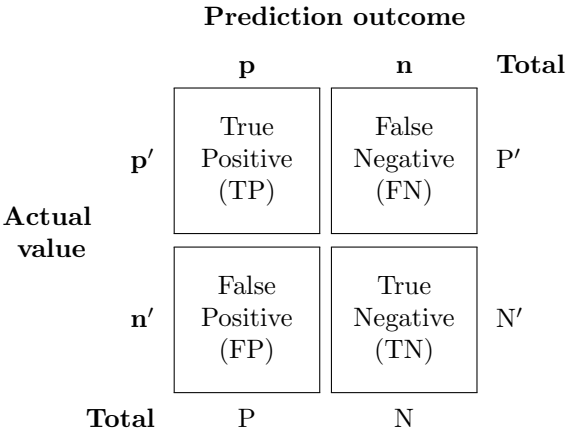


Figure 5: Confusion matrix of a binary classification problem

In single-objective HPO, the objective considered is commonly an error-based measure. The references in Table 2 show that error-based measures are also heavily used in multi-objective HPO, as they ensure a response from the model that is close to reality. Additionally, model complexity objectives are often included (following Occam's razor principle; Blumer, Ehrenfeucht, Haussler, and Warmuth (1987)), along with time-based metrics (e.g., training time on embedded devices) and/or (computational) cost objectives. The complexity of a neural network, for instance, is often estimated using the number of parameters (weights of the connections between neurons); e.g., (Baldeon & Lai-Yuen, 2020; Calisto & Lai-Yuen, 2020; Elsken et al., 2019; J. Jiang et al., 2020; Liang et al., 2019; Lu et al., 2020; Smith & Jin, 2014) use this measure as a second objective (along with an error-based measure). The number of features can also be used as a complexity measure: see Binder et al. (2020); Bouraoui et al. (2018); Faris et al. (2020); Martinez-de Pison et al. (2017); Sopov and Ivanov (2015). The more features the training algorithm has to consider, the more expensive it will be. Likewise, considering fewer features may negatively affect the performance of the algorithm.

A complexity measure can also be defined in terms of the "model size"; therefore it depends of the target ML algorithm to be optimized (e.g., the number of neurons in one layer (Juang & Hsu, 2014), the number of support vectors in a SVM (Bouraoui et al., 2018), the DNN file size (Shinozaki et al., 2020) or the ensemble size (Garrido & Hernández, 2019) for ensemble algorithms). Alternatively, the number of floating point operations (FLOPS) can be used (Chin et al., 2020; Elsken et al., 2019; Lu et al., 2020; Wang et al., 2019, 2020). This metric is also used to reflect the energy consumption (Han, Pool, Tran, & Dally, 2015), and used along the number of parameters in the network (Smithson et al., 2016). Both FLOPS and the number of parameters are sometimes used as a memory consumption measures (Laskaridis et al.,

2020), and can be combined with a time-based measure (Shah & Ghahramani, 2016). Time-based measures can be related to the training phase (Laskaridis et al., 2020; Lu et al., 2020; Rajagopal et al., 2020; Tanaka et al., 2016), the prediction phase (Abdolsh et al., 2019; Garrido & Hernández, 2019; Hernández et al., 2016), the inference process on forward passes in ANNs (Kim et al., 2017), or the whole optimization process (Richter et al., 2016).

The increasing computational cost of Deep Learning models generally translates into higher hardware costs. As a result, an optimization using both algorithm performance and hardware cost should be considered, especially for edge devices. Hardware-related costs can be measured in different ways; e.g., through energy consumption or memory utilization. In many cases, these measures are estimated as a function of the hyperparameters. For instance, Parsa et al. (2019) present an abstract energy consumption model that depends on the neural network architecture (number of layers, number of outputs of each layer, kernel size, etc).

3.3 Quality metrics for comparing multi-objective HPO algorithms

The surveyed literature presents different metrics to judge and/or compare the strengths and weaknesses of multi-objective HPO algorithms. A first set of quality metrics is related to the resulting Pareto front. Here, hypervolume is the most widely used; see Garrido and Hernández (2019); Hernández et al. (2016); Horn and Bischl (2016); Horn et al. (2017); Lu et al. (2020); Shah and Ghahramani (2016). It computes the volume of the area enclosed by the Pareto front and a reference point, specified by the user. Binder et al. (2020) compute the *generalization* dominated hypervolume, which is obtained by evaluating the non-dominated solutions of the validation set on the test set data. Other quality metrics based on the Pareto front are the difference in performance between each solution on the front and the single-objective version of the algorithm (holding the other objectives steady) (Chatelain et al., 2007), the average distance (or Generational Distance) of the front to a reference set (such as the approximated true Pareto front obtained by exhaustive search, see Smithson et al. (2016); or an aggregated front, see Gülcü and Kuş (2021)), a coverage measure computed as the percentage of the solutions of an algorithm A dominated by the solutions of another algorithm B (Juang & Hsu, 2014; H. Li, Zhang, Tsang, & Ford, 2004), or metrics based on the shape of the Pareto front (Abdolsh et al., 2019) or its diversity (Juang & Hsu, 2014; H. Li et al., 2004). The latter can be computed using the spacing and the spread of the solutions: spacing evaluates the diversity of the Pareto points along a given front (Gülcü & Kuş, 2021), whereas spread evaluates the range of the objective function values (see Zitzler, Deb, and Thiele (2000)).

Some authors use performance measures that do not relate to the quality of the front obtained; e.g., execution time (Horn et al., 2017; Parsa et al., 2019; Richter et al., 2016), number of performance evaluations (Parsa et al., 2019), CPU utilization in parallel computer architectures (Richter et al., 2016),

measures that were not considered as an objective and that are evaluated in the Pareto solutions (usually, confusion matrix-based measures for classification problems; see Salt et al. (2019)), or measures that are specific for the HPO algorithm used (e.g., the number of new points suggested per batch is used by Gupta, Shilton, Rana, and Venkatesh (2018) to evaluate the performance of the search executed during batch Bayesian optimization).

4 Multi-objective HPO: categorization

This section categorizes the selected articles, distinguishing between metaheuristic-based algorithms (Section 4.1) and metamodel-based algorithms (Section 4.2). Finally, Section 4.3 discusses the (scarce) research on hybrid HPO algorithms.

4.1 Metaheuristic-based HPO algorithms

Heuristic search attempts to optimize a problem by improving the solution based on a given heuristic function or a cost measure (Russell & Norvig, 2010). A heuristic search method does not always guarantee to find the optimal solution, but aims to find a good or acceptable solution within a reasonable amount of time and memory space. Metaheuristics are algorithms that combine heuristics (which are often problem-specific) in a more general framework (Bianchi, Dorigo, Gambardella, & Gutjahr, 2009). Figure 6 summarizes the general procedure of a metaheuristic-based algorithm. The algorithm generates new solution(s) (exploration and/or exploitation of the search space) starting from one or more initial solution(s). Depending on the algorithm, the information available at the moment of the search can be updated before the next iteration and/or bad solutions can be discarded. The process is repeated until a stop criterion is met.

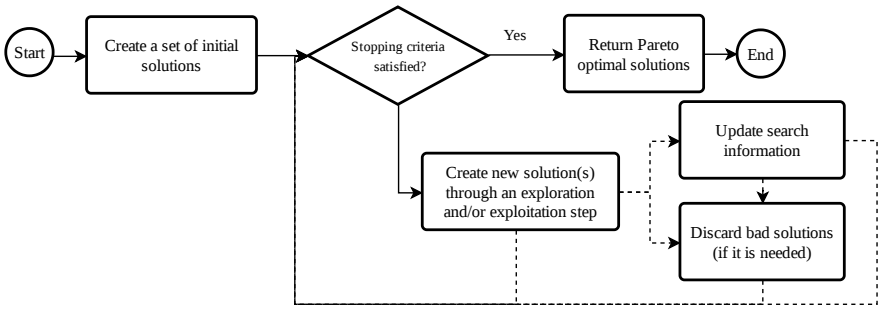


Figure 6: General procedure in metaheuristic-based algorithms.

While some metaheuristics start from a single initial solution (e.g., Tabu Search (Glover, 1986)), others (referred to as population-based algorithms) start from a set of solutions (e.g., Ant Colony Optimization (Dorigo & Blum, 2005) and Genetic Algorithms (M. Mitchell, 1998)). In general, single-solution based metaheuristics are more exploitation oriented, whereas basic population-based metaheuristics are more exploration oriented (Boussaïd, Lepagnot, & Siarry, 2013). Table 3 gives an overview of the metaheuristics-based algorithms currently used in multi-objective HPO.

Clearly, the most popular metaheuristic-based algorithm for multi-objective HPO is the Non-dominated Sorting Genetic Algorithm II (NSGA-II; Deb, Pratap, Agarwal, and Meyarivan (2002)). This is not surprising, as genetic algorithms have shown to perform quite well in single-objective HPO settings: see, e.g., Deighan, Field, Capano, and Khanna (2021), who showed that they cannot only obtain CNN configurations from scratch, but can also refine state-of-the-art CNNs. NSGA-II builds on the original NSGA algorithm (Srinivas & Deb, 1994); yet, it is computationally less expensive (a temporal complexity of $O(MN^2)$ versus $O(MN^3)$ for the original algorithm, where M is the number of objectives and N is the population size). Another important difference is the preservation of the best solutions, through an elitist selection according to the fitness and spread of solutions. Ekbal and Saha (2015) applied NSGA-II to jointly optimize hyperparameters and features, and demonstrated the superiority of the resulting models over others constructed with all the features and default hyperparameters. Binder et al. (2020) observed analogous results optimizing a SVM, kkNN, and XGBoost. Yet, according to the generalization dominated hypervolume, NSGA-II performed slightly worse than ParEGO, a Bayesian optimization-based approach (see Knowles (2006) for further details). Binder et al. (2020) thus suggest to prefer NSGA-II over ParEGO only when model evaluations are cheap, and marginal degradation of performance is acceptable.

Contrary to NSGA-II, the Multi-Objective Evolutionary Algorithm based on Decomposition (MOEA/D) (Q. Zhang & Li, 2007) uses *scalarization* to solve the multi-objective HPO problem. Both MOEA/D and NSGA-II have shown to improve the accuracy of the resulting model compared with manual hyperparameter selection (Calisto & Lai-Yuen, 2020; Magda et al., 2017). In Baldeon and Lai-Yuen (2020), MOEA/D is compared with a Bayesian Optimization approach (using Gaussian Process Regression with Expected Improvement as acquisition function), for tuning an adaptive convolutional neural network (AdaResU-Net) used for medical image segmentation. The use of MOEA/D results in a reduction in the number of parameters to train; the comparison is not really reliable, though, as the Bayesian approach was used in a single-objective optimizer, focusing only on segmentation accuracy and not on model size.

Table 3: Overview of Metaheuristics-based HPO algorithms (N, D, C refer to the number of numeric, discrete and categorical hyperparameters respectively). Some algorithms require scalarization; if so, this is indicated in the first column.

HPO Algorithm	Ref.	HP	Target ML algorithm	Performance measure	Application field	Compared against
NSGA-II	Ekbal and Saha (2015)	**N:** 1, **D:** 1, **C:** -	CRF	- Recall - Precision	Mention recognition in texts	-
		N: 2, **D:** 2, **C:** 1	SVM			
	Magda et al. (2017)	**N:** -, **D:** 2, **C:** -	Komi threshold algorithm	- Mean error - Number of outliers	Muscle onset detection	Manual selection
	Mostafa et al. (2020)	**N:** 3, **D:** 2, **C:** -	CNN	- Accuracy - Recall - Specificity	Apnea detection	-
	Sopov and Ivanov (2015)	**N:** -, **D:** 2, **C:** -	MLP	- Classification rate - Number of neurons	Emotion recognition	SPEA (Zitzler & Thiele, 1999), VEGA (Schaffer, 1985), SelfCOMO-GA (Sopov & Ivanov, 2015)

Continued on next page

HPO Algorithm	Ref.	HP	Target ML algorithm	Performance measure	Application field	Compared against
	Binder et al. (2020)	**N:** -, **D:** 2, **C:** -	SVM	- Fraction of selected features - Generalization error	OpenML benchmark	ParEGO (Knowles, 2006)
		N: 5, **D:** 4, **C:** -	XGBoost			
		N: -, **D:** 2, **C:** 1	kkNN			
	Shinozaki et al. (2020)	**N:** 7, **D:** 3, **C:** 1	DNN-based Spoken Language Systems	- Word error rate - Model size	Speech recognition	CMA-ES (Hansen, Müller, & Koumoutsakos, 2003)
	Kim et al. (2017)	**N:** 2, **D:** 1, **C:** -	LeNet	- Accuracy - Running time	Image recognition	-
	Bouraoui et al. (2018)	**N:** 4, **D:** 1, **C:** -	SVM	- Accuracy - Number of SV - Number of features	UCI datasets	Grid search

Continued on next page

HPO Algorithm	Ref.	HP	Target ML algorithm	Performance measure	Application field	Compared against
	Loni, Sinaei, Zoljodi, Daneshtalab, and Sjödin (2020)	**N:** -, **D:** 3, **C:** 3	CNN	- Accuracy - Network size	Image recognition	-
GA (scalarized objectives)	Deighan et al. (2021)	**N:** 4, **D:** 6, **C:** -	CNN	- Accuracy - Network size	Gravitational wave classification	GA variants
MOEA/D (scalarized objectives)	Calisto and Lai-Yuen (2020)	**N:** 1, **D:** 3, **C:** 3	AdaEn-net	- Segmentation accuracy - Model size	Image segmentation	Manual selection
	Baldeon and Lai-Yuen (2020)	**N:** 2, **D:** 1, **C:** 2	AdaResU-net	- Segmentation accuracy - Model size	Image segmentation	GP-EI (Snoek, Larochelle, & Adams, 2012)
	C. Zhang et al. (2016)	**N:** 2, **D:** 1, **C:** -	DBN ensembles	- Accuracy - Diversity	Remaining useful life prediction	-
CMA-ES	Tanaka et al. (2016)	**N:**19, **D:** 6, **C:** 2	NNLM	- Word error rate - Computational time	Speech recognition	-

Continued on next page

HPO Algorithm	Ref.	HP	Target ML algorithm	Performance measure	Application field	Compared against
	Shinozaki et al. (2020)	**N:** 7, **D:** 3, **C:** 1	Spoken Language Systems	- Word error rate - Model size	Speech recognition	NSGA-II (Deb et al., 2002)
	H. Qin et al. (2017)	**N:** 6, **D:** 4, **C:** -	NMT System	- BLEU score - Validation time	Machine translation	-
	Ekbal and Saha (2016)	**N:** -, **D:** 2, **C:** -	CRF	- Recall - Precision	Named entity recognition	-
		N: -, **D:** 1, **C:** -	SVM			
		N: -, **D:** 1, **C:** -	MBL			
OMOPSO	Wang et al. (2019, 2020)	**N:** -, **D:** 5, **C:** -	Densenet-121	- Accuracy - FLOPs	Image classification	-

Continued on next page

HPO Algorithm	Ref.	HP	Target ML algorithm	Performance measure	Application field	Compared against
	Rajagopal et al. (2020)	**N:** 3, **D:** 3, **C:** -	CNN	- Accuracy - FLOPs	Scene classification	-
PSO (scalarized objectives)	Faris et al. (2020)	**N:** 1, **D:** 2, **C:** -	SVM	- Error - Feature selection rate	Medical specialties classification	-
CoDeepNeat	Liang et al. (2019)	**N:** 2, **D:** 2, **C:** 3	CNN	- Error - Number of parameters	Medical image classification	-
SPEA II (scalarized objectives)	Loni et al. (2019)	**N:** -, **D:** 2, **C:** 4	CNN	- Accuracy - Number of parameters	Image classification	-
MADE	Pathak et al. (2020)	**N:** 1, **D:** 5, **C:** 4	Bidirectional LSTM	- Recall - Specificity	Classification	-
MODE (scalarized objectives)	Singh et al. (2020)	**N:** 2, **D:** 5, **C:** 3	CNN	- Recall - Specificity	Classification	-
MO-RACACO	Juang and Hsu (2014)	**N:** -, **D:** 1, **C:** -	Fuzzy Neural Networks	- RMSE - Number of rules nodes	Regression	MO-EA (Juang, 2002), MO-ACOr (Socha & Dorigo, 2008)

Continued on next page

HPO Algorithm	Ref.	HP	Target ML algorithm	Performance measure	Application field	Compared against
MOSA	Gülcü and Kuş (2021)	**N:** -, **D:** 10, **C:** 4	CNN	- Accuracy - FLOPs	Image classification	-
Nelder-Mead (scalarized objectives)	Albelwi and Mah. (2016)	**N:** -, **D:** 7, **C:** -	CNN	- Accuracy - Mutual information	Image classification	-

The Covariance matrix adaptation-evolutionary strategy (CMA-ES) (Hansen et al., 2003) is a population-based metaheuristic that differs from Genetic Algorithms in the use of a fixed length real valued vector as a gene (instead of the typical vector of binary components), and a multivariate Gaussian distribution to generate new solutions. Multi-objective CMA-ES can be formulated considering the dominance of solutions on the Pareto Frontier (H. Qin et al., 2017; Shinozaki et al., 2020; Tanaka et al., 2016). Shinozaki et al. (2020) optimize DNN-based Spoken Language Systems using this approach; the resulting networks had lower word error rates and were smaller than the networks designed by NSGA-II. Additionally, multi-objective CMA-ES generated smaller networks than the one obtained with single-objective CMA-ES (using the error-based measure as objective to optimized). In our opinion, though, this last comparison does not make much sense, since network size did not appear as objective in the single-objective setting.

Analogous to Genetic Algorithms, Particle Swarm Optimization (PSO) (Eberhart & Kennedy, 1995) works with a population of candidate solutions, known as *particles*. Each particle is characterized by a velocity and a position. The particles search for the optimal solutions by continuously updating their position and velocity. Their movement is influenced not only by their own local best known position, but is also guided towards the best known position found by other particles in the search space. A multi-objective PSO algorithm (OMOPSO) was developed by Sierra and Coello (2005), using Pareto dominance and crowding distance to filter out the best particles. It employs different mutation operators which act on subsets of the swarm, and applies the ϵ-dominance concept (see Laumanns, Thiele, Deb, and Zitzler (2002) for more details) to fix the size of the set of final solutions produced by the algorithm.

Strength Pareto Evolutionary Algorithm II (SPEA-II) (Zitzler, Laumanns, & Thiele, 2001) adds several improvements to the original SPEA algorithm presented by Zitzler and Thiele (1999). Loni et al. (2019) used the algorithm to optimize six hyperparameters of a CNN, yielding more accurate and less complex networks than could be obtained with hand-crafted networks, or with NAS algorithms.

Differential Evolution (DE) (Storn & Price, 1997) is similar to Genetic Algorithms, but differs in the way in which the solutions are coded (using real vectors instead of binary-coded ones) and, consequently, in the way in which the evolutionary operators are applied. Multi-Objective Differential Evolution (MODE) (Babu & Gujarathi, 2007) selects the non-dominated solutions to generate new solutions on each iteration. To reduce the computational effort while maintaining accuracy, a memetic adaptive DE method (MADE) was developed by S. Li et al. (2019). DE depends significantly on its control parameter settings. Therefore, MADE uses a historical memory of successful control parameter settings to guide the selection of future control parameter values (Tanabe & Fukunaga, 2013). Additionally, a local search method (e.g., the Nelder-Mead simplex method (NMM) (S. Li et al., 2019), or chaotic local search (Pathak et al., 2020)) is employed to refine the solutions,

and a ranking-based elimination strategy (using non-dominated and crowding distance sorting) is proposed to maintain the most promising solutions.

Ant Colony Optimization (ACO) (Dorigo, Maniezzo, & Colorni, 1996) is inspired by the behavior of real ants; the basic idea is to model the HPO problem as the search for a minimum cost path in a graph. ACO algorithms can be applied to solve multi-objective problems, and may differ in three respects (Alaya, Solnon, & Ghedira, 2007): (1) the way solutions are built, using only one *pheromone structure* for an aggregation of several objectives, or associating a different pheromone structure with each objective (Gravel, Price, & Gagné, 2002; Iredi, Merkle, & Middendorf, 2001); (2) the way in which solutions are updated (Barán & Schaerer, 2003; Iredi et al., 2001) and (3) the incorporation of existing problem-specific knowledge into the transition rule that defines how to create new solutions from existing ones (Doerner, Gutjahr, Hartl, Strauss, & Stummer, 2004; Gravel et al., 2002). The latter is included in a multi-objective version of ACO (MO-RACACO, Hsu and Juang (2013)) for Fuzzy Neural Network (FNN) optimization (Juang & Hsu, 2014). The results showed that MO-RACACO outperformed other population-based MO algorithms (MO-EA, Juang (2002); and MO-ACOr, Socha and Dorigo (2008)) in terms of the coverage measure obtained, yet it did not always obtain the best diversity values.

Simulated annealing (SA) is a probabilistic technique for finding the global optimum of a single-objective problem (Kirkpatrick, Gelatt, & Vecchi, 1983). Gülcü and Kuş (2021) applied a multi-objective approach (MOSA) to optimize 14 hyperparameters of a CNN. Their MOSA algorithm selects new solutions based on their relative merit (measured by the dominance relationship) w.r.t. the current solutions.

Lastly, the Nelder-Mead simplex method (NMM) (Olsson & Nelson, 1975) has been applied by Albelwi and Mah. (2016) to optimize seven hyperparameters for a CNN. As NMM is a single-objective optimization procedure, the objectives need to be scalarized (the authors used a weighted sum approach). NMM is a local optimization procedure, so it may get stuck in a local minimum. This may be avoided by running the algorithm from different starting points, which increases the probability of reaching the global minimum. Alternatively, modifications to the algorithm have been proposed (as in McKinnon (1998)) that allow the algorithm to escape from local minima, yet at the cost of a large number of iterations.

4.2 Metamodel-based HPO algorithms

Training a machine learning algorithm can be computationally expensive, e.g. due to the target algorithm's own structure (e.g., Deep Learning models), the amount and complexity of the data to process, resource limitations (execution time, memory and energy consumption, etc), and/or the type of training algorithm used. Therefore, different HPO approaches have been developed that employ less expensive models (referred to as metamodels or surrogate models) to emulate the computation of the real performance functions. A

metamodel can be used in combination with a metaheuristic (yielding a so-called *hybrid* method, see Section 4.3) to reduce the computational complexity of the optimization (Pech, Kandler, Lukacevic, & Füssl, 2019; Regis, 2014; Sun, Gong, & Ma, 2009), or can be used independently to guide the search. The latter requires an *acquisition function* or *infill criterion* to guide the exploration/exploitation of the search space.

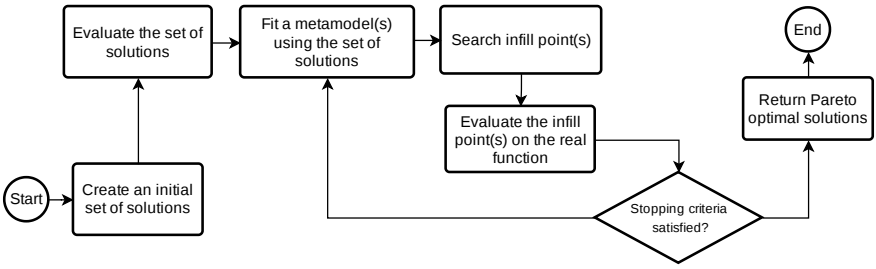


Figure 7: Generic optimization procedure in metamodel-based algorithms.

Figure 7 summarizes the main steps in such a metamodel-based optimization algorithm. The optimization starts with a set of initial points (input/output observations) to train the metamodel. Then, the acquisition function is used to select one or more new points (infill points) to be evaluated. The metamodel is updated with this new information (adding the new I/O observations to the initial set), and the procedure continues until a stopping criterion is met.

Table 4 gives an overview of the metamodel-based algorithms currently used for multi-objective HPO. Most multi-objective HPO articles use a Gaussian Process (GP) as metamodel. GPs use a covariance function, or kernel, to compute the *spatial correlation* among several output observations for a given performance measure (i.e., a given objective of the HPO algorithm; see Figure 4). In this approach, it is assumed that HPO input configurations that differ only slightly from one another (i.e., they are *close* to each other in the search space) are strongly positively correlated w.r.t. their outputs; as the configurations are further apart in the search space, the correlation dies out.

Table 4: Overview of Metamodel-based HPO algorithms. N, D, C refer to numeric, discrete and categorical hyperparameters respectively. Some algorithms require scalarization; if so, this is indicated below the acquisition function.

Metamodel	Infill criteria	Ref.	HP	Target ML algorithm	Performance measure	Application field	Compared against
Gaussian process	EI	Salt et al. (2019)	**N:**18, **D:** -, **C:** -	SNN	- Accuracy - MVS error - Reward measure	Classification	Random search, DE (Storn & Price, 1997), SADE (A.K. Qin, Huang, & Suganthan, 2008)
	Dominance*	Parsa et al. (2019)	**N:** 3, **D:** 6, **C:** -	AlexNET	- Error - Energy requirement	Image classification	Grid search, Random search, NSGA-II (Deb et al., 2002)
			N: -, **D:**11, **C:** -	VGG19			
	PEHI	Abdolsh et al. (2019)	**N:** 4, **D:** 2, **C:** -	NN	- Prediction error - Prediction time	Image classification	PESMO (Hernández et al., 2016), SMS-EGO (Ponweiser, Wagner, Biermann, & Vincze, 2008), SUR (Picheny, 2014), ParEGO (Knowles, 2006)
	CEIPV	Shah and Ghahramani (2016)	**N:** 1, **D:** 2, **C:** -	NN	- Accuracy - Memory consumption	Classification	ParEGO (Knowles, 2006), Random search, GP-EHV

Continued on next page

Metamodel	Infill criteria	Ref.	HP	Target ML algorithm	Performance measure	Application field	Compared against
	LCB	Richter et al. (2016)	**N:** -, **D:** 2, **C:** -	SVM	- Classification error - Running time	Classification	Random search
	UCB (scalarized objectives)	Chin et al. (2020)	**N:** 6, **D:** -, **C:** -	Slimmable NN	- Cross entropy loss - FLOPs	Classification	-
	PES	Hernández et al. (2016)	**N:** 4, **D:** 2, **C:** -	NN	- Prediction error - Prediction time	Image classification	ParEGO (Knowles, 2006), SMS-EGO (Ponweiser et al., 2008), SUR (Picheny, 2014)
		Garrido and Hernández (2019)	**N:** 2, **D:** 3, **C:** -	Ensemble of Decision Trees	- Prediction error - Ensemble size	Classification	Random search, BMOO (Feliot, Bect, & Vazquez, 2017)
		Hernández-Lobato et al. (2016)	**N:** 4, **D:** 4, **C:** -	NN	- Prediction error - Energy	Classification	Random search, NSGA-II (Deb et al., 2002)

Continued on next page

Metamodel	Infill criteria	Ref.	HP	Target ML algorithm	Performance measure	Application field	Compared against
Random Forest	LCB	Binder et al. (2020)	**N:** -, **D:** 2, **C:** -	SVM	- Fraction of selected features - Generalization error	OpenML benchmark	-
			N: 5, **D:** 4, **C:** -	XGBoost			
			N: -, **D:** 2, **C:** 1	kkNN			
		Horn and Bischl (2016)	**N:** -, **D:** 3, **C:** -	SVM	- False Negative rate - False Positive rate	OpenML Benchmark	SMS-EGO (Ponweiser et al., 2008), ParEGO (Knowles, 2006), Random sampling, NSGA-II (Deb et al., 2002)
			N: -, **D:** 2, **C:** -	Random Forest			
			N: -, **D:** 2, **C:** -	Logistic regression			

Continued on next page

Metamodel	Infill criteria	Ref.	HP	Target ML algorithm	Performance measure	Application field	Compared against
Tree Parzen Estimators	Modified EHV (scalarized objectives)	Chandra and Lane (2016)	**N:** 3, **D:** 3, **C:** -	NN-decoder	- Word error rate - Real time factor - Memory usage	Speech recognition	Random sampling, GP, Genetic Algorithm (Zames et al., 1981)
Noisy SMS-EGO	EHI	Horn et al. (2017)	**N:** -, **D:** 3, **C:** -	SVM	- False Negative Rate - False Positive Rate	OpenML benchmark	RTEA (Fieldsend & Everson, 2014) , Random search
SExI-EGO	EHI	Koch, Wagner, Emmerich, Bäck, and Konen (2015)	**N:** 2(5), **D:** -, **C:** -	SVM	- Accuracy (Mean accuracy) - Training time	UCI benchmark	SMS-EGO (Ponweiser et al., 2008) , Latin Hypercube sampling

The choice of the kernel in a GP is important, as it determines the shape of the assumed correlation function. In general, the most common kernels used in GP-based metamodels are the Gaussian kernel and the Matérn kernel (Ounpraseuth, 2008). Using the kernel, the analyst can not only *predict* the estimated outputs (i.e., in our case, the performance measures) at non-observed input locations (i.e., hyperparameter configurations), but can also estimate the *uncertainty* on these output predictions. Both the predictions and their uncertainty are reflected in the acquisition function, to balance exploration and exploitation in the search for new hyperparameter settings to be evaluated. We refer the reader to Rojas-Gonzalez and Van Nieuwenhuyse (2020) for a detailed review of acquisition functions, for (general, non HPO related) single and multi-objective optimization problems.

Table 4 also shows the acquisition functions that have been used so far in multi-objective HP. Clearly, the most popular one is Expected Improvement (EI, which was originally proposed by Jones, Schonlau, and Welch (1998)). The EI represents the expected improvement over the best outputs found so far, at an (arbitrary) non-observed input configuration. As EI was originally developed for single-objective problems, it is usually applied in multi-objective problems where the objectives are scalarized. Salt et al. (2019), for instance, optimize a Spiking Neural Network (SNN) using a weighted function of three individual objectives (the accuracy, the sum square error of the membrane voltage signal, and the reward of the spiking trace). Three types of acquisition function were studied; EI, Probability of Improvement (POI) and Upper Confidence Bound (UCB). The performance obtained with POI was significantly better than that obtained with EI and UCB, and overall, the BO-based approach required significantly fewer evaluations than evolutionary strategies such as SADE.

Another way to use BO in multi-objective HPO is to fit a metamodel to each objective independently. Parsa et al. (2019) use such an approach in their Pseudo Agent-Based multi-objective Bayesian hyperparameter Optimization (PABO) algorithm; they use the dominance rank (based on the predictor values of each objective) as infill criterion. This evidently yields different infill points for the respective objectives (in their case, an error-based objective and an energy-related objective); the infill point suggested for one objective function is then also evaluated for the other objective function, provided that it is not dominated by any previous HPO configuration analyzed. In this way, the algorithm speeds up the search for Pareto-optimal solutions. The experiments indeed demonstrated that PABO outperforms NSGA-II in terms of speed.

Other authors have studied HPO problems when the performance measures are correlated (Shah & Ghahramani, 2016), or when one of the measures is clearly more important than the others (Abdolsh et al., 2019). The algorithm proposed by Shah and Ghahramani (2016) models the correlations between accuracy, memory consumption and training time of an ANN using a multi-output Gaussian process or Co-Kriging (H. Liu, Cai, & Ong, 2018). The authors propose a modification to the expected hypervolume (EHV) that

reflects these correlations; this modified EHV is then used as acquisition function, preferring the infill point that increases the expected hypervolume of the Pareto front the most. The algorithm is compared to ParEGO, (Knowles, 2006), random search, and a GP using the original EHV metric. The results suggest that the modified EHV criterion increases the speed of the optimization, requiring fewer iterations to converge to the Pareto optimal solutions.

The PEHI algorithm proposed by Abdolsh et al. (2019) accounts for user preferences. While the other algorithms that are used as comparison in the paper (PESMO, Hernández et al. (2016); SMS-EGO, Ponweiser et al. (2008); Stepwise Uncertainty Reduction, Picheny (2014); and ParEGo, Knowles (2006)) try to find solutions across the entire Pareto front, the proposed algorithm manages to focus its budget on the Pareto solutions that are most interesting for the analyst.

Other acquisition functions used in metamodel-based algorithms are the Lower Confidence Bound (LCB) or Upper Confidence Bound (UCB). These use a (user-defined) confidence bound to focus the search on local areas, or explore the search space more globally. Richter et al. (2016) use a multipoint LCB which simultaneously generates q hyperparameter configurations. A GP is used to model the misclassification error and the logarithmic runtime. The results demonstrated an improvement in CPU utilization (and, thus, an increase in the number of hyperparameter evaluations) within the same time budget. Confidence bounds are also used by Chin et al. (2020) to optimize the hyperparameters of Slimmable Neural Networks. The algorithm fits a GP to each individual performance measure, hence obtaining information to compute individual UCBs. These UCBs are then scalarized, and the resulting single objective function is minimized to obtain the next infill point. The proposed algorithm succeeds in reducing the complexity of the NNs studied; yet, the authors did not compare its performance with any other multi-objective HPO algorithms.

The Predictive Entropy Search (PES) criterion is used by multiple authors, as infill criterion for different algorithms. Hernández et al. (2016) use PESMO (multi-objective PES) to optimize a NN with six hyperparameters, in view of minimizing the prediction error and the training time. PESMO seeks to minimize the uncertainty in the *location of the Pareto set*. The algorithm is compared with ParEGO, SMS-EGO and SUR, showing that PESMO gives the best overall results in terms of hypervolume and number of expensive evaluations required for training/testing the neural network. Garrido and Hernández (2019) use PESMOC (a modified version of PESMO which takes into account constraints) to optimize an ensemble of Decision Trees. The experiments show that PESMOC is able to obtain better results than a state-of-the-art method for constrained multi-objective Bayesian optimization (Feliot et al., 2017), in terms of the hypervolume obtained and the number of evaluations required. Finally, Hernández-Lobato et al. (2016) used PES to design a neural network with three layers. While most of the HPO methods collect data in a *coupled*

way by always evaluating all performance measures jointly at a given input, these authors consider a *decoupled* approach in which, at each iteration, the next infill configuration is selected according to the maximum value of the acquisition functions across all objectives. The results showed that this approach obtains better solutions (compared to NSGA-II and random search) when computational resources are limited; yet, the trade-offs found among the performance measures may be affected and one of the objectives can turn out to be prioritized over the others.

Random forests (RFs) (Ho, 1995) are an ensemble learning method that trains a set of decision trees having low computational complexity. Each tree is trained with different samples, taken from the initial set of observations. For classification outputs, the RF uses a voting procedure to determine the decision class; for regression output, it returns the average value over the different trees. As for GP, RFs allow the analyst to obtain an uncertainty estimator for the prediction values. Some examples are the quantile regression forests method (Meinshausen & Ridgeway, 2006), which estimates the prediction intervals, and the U-statistics approach (Mentch & Hooker, 2016). Horn and Bischl (2016) use RFs as metamodel to optimize the hyperparameters of three ML algorithms: SVM, Random Forest, and Logistic regression. Using LCB as acquisition function, the authors show that SMS-EGO and ParEGO outperform random sampling and NSGA-II.

Whereas GP-based approaches model the density function of the resulting outcomes (performance measures) given a candidate input configuration, Tree-structured Parzen Estimators (TPE) (Bergstra, Bardenet, Bengio, & Kégl, 2011) model the probability of obtaining an input configuration, given a condition on the outcomes. TPEs naturally handle not only continuous but also discrete and categorical inputs, which are difficult to handle with a GP. Moreover, TPE also works well for conditional search spaces (where the value of a given hyperparameter may depend on the value of another hyperparameter), and has demonstrated good performance on HPO problems for single-objective optimization (Bergstra, Yamins, & Cox, 2013; Falkner, Klein, & Hutter, 2018; Thornton, Hutter, Hoos, & Leyton-Brown, 2013). While it can, in theory, also be applied to multi-objective settings by scalarizing the performance measures, Chandra and Lane (2016) obtained disappointing results when comparing this approach with random sampling, GP and Genetic Algorithms for optimizing an Augmented Tchebycheff scalarized function (Miettinen, 2012) (using fixed weights) of three performance measures for ANNs: GP performed best, while TPE performed worst. Unfortunately, the authors reported the performance based solely on the scalarized value of the three performance measures; they did not report on any other quality metrics, such as hypervolume. They also didn't discuss the reason of the poor TPE performance, such that it remains unclear whether this is due to the scalarization function, or to the characteristics of the search space. A (non-scalarized) multi-objective version of TPE has been proposed by Ozaki, Tanigaki, Watanabe, and Onishi (2020) and is included in the software Optuna (Akiba, Sano, Yanase, Ohta, & Koyama, 2019).

Strikingly, the majority of current HPO algorithms routinely ignore the fact that the obtained performance measures are *noisy*. The noise can be due to either the target ML algorithm itself (when it contains randomness in its procedure, such as a NN that randomly initializes the weights), but even if there is no randomness involved, there will be noise on the outcomes due to the use of k-fold cross validation during the training of the algorithm. This type of cross-validation is common in HPO: it involves the creation of different *splits* of the data into a training and validation set. This process is repeated k times; the performance measures of a given hyperparameter combination will thus differ for each split. Current HPO algorithms focus simply on the *average* performance measures *over the different splits* during the search for the Pareto-optimal points; the inherent uncertainty on these performance measures is ignored. Horn et al. (2017) are one of the few authors to highlight the presence of noise. The paper assumes, though, that noise is homogenous (i.e., it doesn't differ over the search space), and only focuses on different strategies for handling this noise. These strategies are used in combination with the SMS-EGO algorithm (Ponweiser et al., 2008), and compared with the rolling tide evolutionary algorithm (RTEA) (Fieldsend & Everson, 2014) and random search. The results show that simply ignoring the noise (by evaluating a given HPO combination only once, and considering the resulting performance measures as deterministic) performs poorly, even worse than a repeated random search. The best strategy is to reevaluate the (most promising) HP settings. According to the authors, this can likely be explained by the fact that the *true* noise on the performance measures in HPO settings is heterogeneous (i.e., its magnitude differs over the search space). Reevaluation of already observed HP settings is then required to improve the reliability of the observed performance measures. The interested reader is referred to Jalali, Van Nieuwenhuyse, and Picheny (2017) for a discussion of the impact of noise magnitude and noise structure on the performance of (general) optimization algorithms.

Koch et al. (2015) adapt SMS-EGO (Ponweiser et al., 2008) and SExI-EGO (Emmerich, Deutz, & Klinkenberg, 2011) for noisy evaluations, to optimize the hyperparameters of a SVM. The authors again assume that the noise is homogenous, and compare the performance of both algorithms with different noise handling strategies (the reinterpolation method proposed by Forrester, Keane, and Bressloff (2006), and static resampling). Both algorithms use the expected hypervolume improvement (EHI) as infill criterion, though the actual calculation of the criterion is different (causing Sexi-EGO to require larger runtimes). The results show that both SMS-EGO and SExI-EGO work well with the reinterpolation method, yielding comparable results in terms of the hypervolume.

4.3 Hybrid HPO algorithms

A limited number of papers have combined aspects of metamodel-based and population-based HPO approaches: these are referred to in Table 5, summarizing their main characteristics.

Smithson et al. (2016) use an ANN as metamodel to estimate the performance of the target ML algorithm. The neural network is embedded into a Design Space Exploration (DSE) metaheuristic, and is used to intelligently select new solutions that are likely to be Pareto optimal. The algorithm starts with a random solution, and iteratively generates new solutions that are evaluated with the ANN. DSE decides if the solution should be used to update the ANN knowledge, or should be discarded. Compared with manually designed networks from the literature, the proposed algorithm yields results with nearly identical performance, while reducing the associated costs (in terms of energy consumption).

The algorithm proposed by Martinez-de Pison et al. (2017) combines HPO with feature selection. Unlike other algorithms (Ekbal & Saha, 2015; J. Guo et al., 2019; León, Ortega, & Ortiz, 2019), the proposed method considers feature selection combined with HPO. A GP (with UCB as acquisition function) with all features is used to obtain the best setting of hyperparameters (according to the RMSE). Next, a variant of GA (GA-PARSIMONY, Sanz-García, Fernández-Ceniceros, Antonanzas-Torres, Pernia-Espinoza, and Martinez-De-Pison (2015)) is used to select the best features of the problem with the fixed hyperparameters obtained in the first step. In this way, the final model has high accuracy and less complexity (i.e., fewer features), and optimization time is significantly reduced. In our opinion, however, this approach is still suboptimal, as the two optimization problems (HPO and feature selection) are solved sequentially, instead of jointly. Calisto and Lai-Yuen (2021) use an evolutionary strategy combined with a Random Forest metamodel, to optimize 10 hyperparameters of a CNN. In the beginning of the optimization, the algorithm updates the population of solutions using the evolutionary strategy; only after some iterations, the selection of the new candidates is guided by the RF, which is updated each time with all new Pareto front solutions. The final networks found by the algorithm perform better than (or equivalent to) state-of-the-art architectures, while the size of the architectures and the search time is significantly reduced.

Although most NAS algorithms are out of scope for this survey, we include the work by Lu et al. (2020), as it can be considered as an HPO algorithm. The algorithm (NSGANetV2) simultaneously optimizes the architectural hyperparameters and the model weights of a CNN, using a bi-level approach consisting of NSGA-II combined with a metamodel. The metamodel is used to estimate performance measures, which are then optimized by an evolutionary algorithm (such approaches have also been applied successfully non-HPO settings, see e.g., Dutta and Gandomi (2020); Jin (2011)). In the upper level of the optimization, a metamodel is built using an initial set of candidate solutions. In each iteration of the upper level, NSGA-II is executed on the metamodel to detect the Pareto-optimal HP settings (configuration of layers, channels, kernel size and input resolution of the CNN). At the lower level, the weights of the CNN are trained on a subset of the Pareto-optimal solutions. The metamodel is then updated with the results of the actual performance evaluations. Four

different metamodels were studied; Multilayer Perceptron (MLP), Classification and Regression Trees (CART), Radial Basis Functions (RBF) and GP. Given that none of them was consistently better than the others, the authors propose to select the best metamodel on every iteration. On standard datasets (CIFAR-10, CIFAR-100 and ImageNet), the resulting algorithm matches the performance of state-of-the-art NAS algorithms (Lu et al., 2019; Mei et al., 2020), but at a reduced search cost.

Table 5: Overview of hybrid HPO algorithms. (N, D, C refer to numeric, discrete and categorical hyperparameters respectively)

HPO algorithm	Ref.	HP	Target ML algorithm	Performance measure	Application field	Compared against
ANN + DSE	Smithson et al. (2016)	**N:** 1, **D:** 2, **C:** 1	MLP	- Accuracy - Computational complexity	Image classification	Exhaustive search
		N: 1, **D:** 4, **C:** 2	CNN			
GP + GA Parsimony	Martinez-de Pison et al. (2017)	**N:** 5, **D:** 3, **C:** -	XGBoost	- RMSE - Complexity	Image classification	-
Metamodel + NSGA-II	Lu et al. (2020)	**N:** -, **D:** 4, **C:** -	CNN	- Accuracy - MAdds	Image classification	NAS algorithms
Random Forest + ES	Calisto and Lai-Yuen (2021)	**N:** -, **D:** 6, **C:** 4	CNN	- Segmentation error - Number of parameters	Image segmentation	-

5 Conclusions and research opportunities

This paper has reviewed the literature on multi-objective HPO algorithms, categorizing relevant papers into metaheuristics-based, metamodel-based and hybrid approaches. Taking a multi-objective perspective on HPO does not only allow the analyst to optimize trade-offs between different performance measures, it may even yield *better* solutions than the corresponding single-objective HPO problem. For instance, it has been shown that including complexity as an objective in multi-objective HPO does not necessarily compromise the loss-based performance of the ML algorithm w.r.t. the task for which it is trained: particularly, the minimization of the number of features used for training can *improve* the performance of the ML algorithm (Binder et al., 2020; Bouraoui et al., 2018; Faris et al., 2020; Sopov & Ivanov, 2015).

As shown, metaheuristics-based approaches are the most popular. This is quite striking, as such approaches require the evaluation of a large amount of HP configurations, and training/testing the target algorithm for any given HP configuration is usually the most expensive step in the HPO algorithm (due to, e.g., the k-fold cross validation, the optimization steps required for the algorithm's internal parameters, the evaluation of potentially expensive performance measures such as energy consumption or inference time, etc.). Table 6 compares the number of expensive HP evaluations required for multiple HPO algorithms included in this survey, according to two parameters: N (which refers to the initial population size for metaheuristic-based algorithms, and to the size of the initial design for metamodel-based algorithms) and I (the number of iterations performed). The order of the rows in the table reflects an increase in the number of HP evaluations required (for this comparison, parameters that are specific to given algorithms, such as A, Ns, T, n, and N'', are assumed to take specific values which are mentioned in the table). Clearly, the number of costly function evaluations in a typical metamodel-based optimization is much lower than in a metaheuristics-based algorithm, as usually only a single new solution is evaluated in each iteration. Nevertheless, the metamodel-based algorithms by Chin et al. (2020) (row 7 in Table 6) require surprisingly many evaluations since the authors dedicate additional evaluations to further improve the solutions found. The metaheuristic-based algorithm by Pathak et al. (2020) (row 9 in Table 6) can be particularly expensive, as it performs a chaotic local search to generate N'' additional solutions for each solution present in the population of a given iteration.

Table 6: Number of expensive HP evaluations required for different algorithms. An initial set of N solutions is used to start the optimization, and stops after I iterations have been performed. The rows have been ordered according to the number of HP evaluations performed when N, I are incremented. Variables A, Ns, T, n, and N'' are specific to the algorithms and are assumed to take specific values (as mentioned).

ID	HPO type	HP evaluations	Ref.
1	Metamodel-based / Hybrid	$N + I$	(Abdolsh et al., 2019; Chandra & Lane, 2016; Garrido & Hernández, 2019; Hernández et al., 2016; Horn & Bischl, 2016; Horn et al., 2017; Koch et al., 2015; Parsa et al., 2019; Richter et al., 2016; Shah & Ghahramani, 2016; Smithson et al., 2016)
2	Hybrid	$(N+I)+(N+I*2)$	(Martinez-de Pison et al., 2017)
3	Hybrid	$N + I * A, A < N$ (assuming $A = N/2$)	(Lu et al., 2020)
4	Metaheuristic	$N * I$	(Gülcü & Kuş, 2021; H. Qin et al., 2017; Shinozaki et al., 2020; Tanaka et al., 2016)
5	Metaheuristic	$N+I*N'$ (assuming $N = N'$)	(Baldeon & Lai-Yuen, 2020; Binder et al., 2020; Bouraoui et al., 2018; Calisto & Lai-Yuen, 2020; Deighan et al., 2021; Ekbal & Saha, 2016; Faris et al., 2020; Juang & Hsu, 2014; Kim et al., 2017; Liang et al., 2019; Loni et al., 2019; Magda et al., 2017; Mostafa et al., 2020; Rajagopal et al., 2020; Salt et al., 2019; Singh et al., 2020; Sopov & Ivanov, 2015; Wang et al., 2019, 2020; C. Zhang et al., 2016)
6	Metaheuristic	$T + N + I * (N' + 1)$ (assuming $N = N', T = 20$)	(Albelwi & Mah., 2016)
7	Metamodel-based	$I * (N + n)$ (assuming $N = n$)	(Chin et al., 2020)

Continued on next page

ID	HPO algorithm	HP evaluations	Ref.
8	Metaheuristic	$(N+I*N')+(Ns+I*Ns')$ (assuming $N = N' = Ns = Ns'$)	(Ekbal & Saha, 2015)
9	Metaheuristic	$N+I*(N'*N'')$ (assuming $N = N' = N''$)	(Pathak et al., 2020)

Our research leads us to different recommendations for future work. Firstly, current results have demonstrated that using *ensembles* of optimal HP configurations can yield improvements (Ekbal & Saha, 2015, 2016; Sopov & Ivanov, 2015; C. Zhang et al., 2016). Yet, this evidently increases the number of HP evaluations required. In future research, it may be promising to look at ensembles of multiple metamodels (Cho et al., 2020; Wistuba et al., 2018), multiple acquisition functions (Cowen-Rivers et al., 2020), or even multiple optimization procedures (J. Liu, Tunguz, & Titericz, 2020).

Secondly, it is quite surprising that hybrid HPO algorithms remain scarce; the use of metamodels helps to drastically reduce the computational costs related to the metaheuristic approaches (row 3 in Table 6), such that large numbers of inexpensive evaluations can be performed. One would expect that such a hybrid algorithm combines the best of two worlds, providing low computational cost combined with a heuristic search that avoids the (often nasty) optimization of an infill criterion. Unfortunately, the few hybrid approaches present in this survey did not extensively compare their results with other state-of-the-art approaches. This may provide a useful avenue for future research. Thirdly, it is recommended to use individual performance measures as objectives in HPO settings, rather than an aggregate measure such as the *F-measure* (combining recall and precision for classification problems (Ekbal & Saha, 2015, 2016)) or the *Area Under the Curve* measure (AUC), which combines the False Positive rate and the True Positive rate. Such aggregated measures reflect a fixed relationship between the individual measures, which may result in solutions that perform really well on the aggregated measure (for instance, the F-measure), but are suboptimal for the individual measures (recall and precision). Moreover, the aggregation of multiple performance measures into a single objective by means of scalarization should be done carefully, as not all scalarization methods (e.g., weighted sum) allow to detect all parts of the Pareto front. The Augmented Tchebycheff function (Miettinen, 2012), for instance, is recommended when the front contains non-convex areas. The nonlinear term in the scalarization function ensures that these areas can be detected, while the linear term ensures that weak Pareto optimal solutions are less rewarded (see Miettinen and Mäkelä (2002) for a further discussion on scalarization functions).

It is equally surprising that, apart from the work of Koch et al. (2015) and Horn et al. (2017), the uncertainty in the performance measures is commonly ignored in HPO optimization. These two algorithms have mainly explored the impact of different noise handling strategies on the results of *existing* algorithms, while it may be more beneficial to account for the noise by adjusting the metamodels used, and/or the algorithmic approach. Furthermore, they assume homogenous noise, which is likely not the case in practice. Stochastic algorithms (such as (Binois, Huang, Gramacy, & Ludkovski, 2019; Gonzalez, Jalali, & Van Nieuwenhuyse, 2020)) can potentially be useful to determine the number of (extra)replications dynamically during HPO optimization, thus ensuring that computational budget is spent in (re-)evaluating the configuration that yields most information. To the best of our knowledge, such approaches have not yet been studied in the context of HPO optimization.

Finally, recent research has shown potential benefits in studying cheaply available (yet lower fidelity) information, obtained for instance by evaluating only a fraction of the training data or a small number of iterations. Low fidelity methods such as bandit-based approaches (L. Li, Jamieson, DeSalvo, Rostamizadeh, & Talwalkar, 2017) have been mainly applied in single-objective HPO; to the best of our knowledge, there are no studies in multi-objective HPO. Also, early stopping criteria (Dai, Yu, Low, & Jaillet, 2019) could be considered to ensure a more intelligent use of the available computational budget. Previously, this has been applied in single-objective optimization (Kohavi & John, 1995; Provost, Jensen, & Oates, 1999) by considering the algorithm's learning curve. The training procedure for a given hyperparameter configuration is then stopped when adding further resources (training instance, iterations, training time, etc) is predicted to be futile. Likewise, early stopping criteria have been used to reduce the overfitting level of the ML algorithm (Makarova et al., 2021). To the best of our knowledge, none of these strategies has been applied in multi-objective HPO algorithms.

Declarations of interests. 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. This work was supported by the Flanders Artificial Intelligence Research Program (FLAIR), and by the Research Foundation Flanders (FWO Grant 1216021N). The authors would like to thank Gonzalo Nápoles from Tilburg University for his comments on a previous version of this paper.

References

Abdolsh, M., Shilton, A., Rana, S., Gupta, S., Venkatesh, S. (2019). Multi-objective bayesian optimisation with preferences over objectives. *Advances in neural information processing systems* (pp. 12235–12245).

Akiba, T., Sano, S., Yanase, T., Ohta, T., Koyama, M. (2019). Optuna: A next-generation hyperparameter optimization framework. *Proceedings of the 25th acm sigkdd international conference on knowledge discovery & data mining* (pp. 2623–2631).

Alaya, I., Solnon, C., Ghedira, K. (2007). Ant colony optimization for multi-objective optimization problems. *19th ieee international conference on tools with artificial intelligence (ictai 2007)* (Vol. 1, pp. 450–457). (https://doi.org/10.1109/ICTAI.2007.108)

Albelwi, S., & Mah., A. (2016). Automated optimal architecture of deep convolutional neural networks for image recognition. *2016 15th ieee international conference on machine learning and applications (icmla)* (pp. 53–60). (https://doi.org/10.1109/ICMLA.2016.0018)

Andreopoulos, A., & Tsotsos, J.K. (2013). 50 years of object recognition: Directions forward. *Computer vision and image understanding*, *117*(8), 827–891. (https://doi.org/10.1016/j.cviu.2013.04.005)

Babu, B., & Gujarathi, A.M. (2007). Multi-objective differential evolution (mode) algorithm for multi-objective optimization: parametric study on benchmark test problems. *Journal on Future Engineering and Technology*, *3*(1), 47–59. (https://doi.org/10.26634/jfet.3.1.697)

Baldeon, M., & Lai-Yuen, S.K. (2020). Adaresu-net: Multiobjective adaptive convolutional neural network for medical image segmentation. *Neurocomputing*, *392*, 325–340. (https://doi.org/10.1016/j.neucom.2019.01.110)

Barán, B., & Schaerer, M. (2003). A multiobjective ant colony system for vehicle routing problem with time windows. *Applied informatics* (pp. 97–102).

Bergstra, J., Bardenet, R., Bengio, Y., Kégl, B. (2011). Algorithms for hyper-parameter optimization. *25th annual conference on neural information processing systems (nips 2011)* (Vol. 24).

Bergstra, J., & Bengio, Y. (2012). Random search for hyper-parameter optimization. *The Journal of Machine Learning Research*, *13*(1), 281–305. Retrieved from http://jmlr.org/papers/v13/bergstra12a.html

Bergstra, J., Yamins, D., Cox, D. (2013). Making a science of model search: Hyperparameter optimization in hundreds of dimensions for vision architectures. *International conference on machine learning* (pp. 115–123). Retrieved from http://proceedings.mlr.press/v28/bergstra13.html

Bianchi, L., Dorigo, M., Gambardella, L.M., Gutjahr, W.J. (2009). A survey on metaheuristics for stochastic combinatorial optimization. *Natural Computing*, *8*(2), 239–287. (https://doi.org/10.1007/s11047-008-9098-4)

Binder, M., Moosbauer, J., Thomas, J., Bischl, B. (2020). Multi-objective hyperparameter tuning and feature selection using filter ensembles. *stat*, *1050*, 13. (https://doi.org/10.1145/3377930.3389815)

Binois, M., Huang, J., Gramacy, R.B., Ludkovski, M. (2019). Replication or exploration? sequential design for stochastic simulation experiments. *Technometrics*, *61*(1), 7–23. (https://doi.org/10.1080/00401706.2018.1469433)

Bischl, B., Mersmann, O., Trautmann, H., Weihs, C. (2012). Resampling methods for meta-model validation with recommendations for evolutionary computation. *Evolutionary Computation*, *20*(2), 249–275. (https://doi.org/10.1162/EVCO_a_00069)

Blumer, A., Ehrenfeucht, A., Haussler, D., Warmuth, M.K. (1987). Occam's razor. *Information processing letters*, *24*(6), 377–380. (https://doi.org/10.1016/0020-0190(87)90114-1)

Bouraoui, A., Jamoussi, S., BenAyed, Y. (2018). A multi-objective genetic algorithm for simultaneous model and feature selection for support vector machines. *Artificial Intelligence Review*, *50*(2), 261–281. (https://doi.org/10.1007/s10462-017-9543-9)

Boussaïd, I., Lepagnot, J., Siarry, P. (2013). A survey on optimization metaheuristics. *Information sciences*, *237*, 82–117.

Bui, K.-H.N., & Yi, H. (2020). Optimal hyperparameter tuning using meta-learning for big traffic datasets. W. Lee et al. (Eds.), *2020 ieee international conference on big data and smart computing (bigcomp 2020)* (pp. 48–54). IEEE. (https://doi.org/10.1109/BigComp48618.2020.0

-100)

Cai, X., Hu, Z., Zhao, P., Zhang, W., Chen, J. (2020). A hybrid recommendation system with many-objective evolutionary algorithm. *Expert Systems with Applications*, *159*, 113648. (https://doi.org/10.1016/j.eswa.2020.113648)

Calisto, M.B., & Lai-Yuen, S.K. (2020). Adaen-net: An ensemble of adaptive 2d-3d fully convolutional networks for medical image segmentation. *Neural Networks*. (https://doi.org/10.1016/j.neunet.2020.03.007)

Calisto, M.B., & Lai-Yuen, S.K. (2021). Emonas-net: Efficient multiobjective neural architecture search using surrogate-assisted evolutionary algorithm for 3d medical image segmentation. *Artificial Intelligence in Medicine*, *119*, 102154. (https://doi.org/10.1016/j.artmed.2021.102154)

Chandra, A., & Lane, I. (2016). Automated optimization of decoder hyper-parameters for online lvcsr. *2016 ieee spoken language technology workshop (slt)* (pp. 454–460). (https://doi.org/10.1109/SLT.2016.7846303)

Chatelain, C., Adam, S., Lecourtier, Y., Heutte, L., Paquet, T. (2007). Multi-objective optimization for svm model selection. *Ninth international conference on document analysis and recognition (icdar 2007)* (Vol. 1, pp. 427–431). (https://doi.org/10.1109/ICDAR.2007.4378745)

Chen, W.-C., Jiang, X.-Y., Chang, H.-P., Chen, H.-P. (2014). An effective system for parameter optimization in photolithography process of a lgp stamper. *Neural Computing and Applications*, *24*(6), 1391–1401. (https://doi.org/10.1007/s00521-013-1353-7)

Chin, T.-W., Morcos, A.S., Marculescu, D. (2020). *Pareco: Pareto-aware channel optimization for slimmable neural networks.* 2nd Workshop on Adversarial Learning Methods for Machine Learning and Data Mining, KDD'2020. Retrieved from https://openreview.net/forum?id=SPyxaz%5Fh9Nd

Cho, H., Kim, Y., Lee, E., Choi, D., Lee, Y., Rhee, W. (2020). Basic enhancement strategies when using bayesian optimization for hyperparameter tuning of deep neural networks. *IEEE Access*, *8*, 52588–52608. (https://doi.org/10.1109/ACCESS.2020.2981072)

Cooney, C., Korik, A., Folli, R., Coyle, D. (2020). Evaluation of hyperparameter optimization in machine and deep learning methods for decoding imagined speech eeg. *Sensors*, *20*(16), 4629. (https://doi.org/10.3390/s20164629)

Cowen-Rivers, A.I., Lyu, W., Wang, Z., Tutunov, R., Jianye, H., Wang, J., Ammar, H.B. (2020). *Hebo: Heteroscedastic evolutionary bayesian optimisation.* Workshop at NeurIPS 2020 Competition Track on Black-Box Optimization Challenge.

Dai, Z., Yu, H., Low, B.K.H., Jaillet, P. (2019). Bayesian optimization meets bayesian optimal stopping. *International conference on machine learning* (pp. 1496–1506). Retrieved from http://proceedings.mlr.press/v97/dai19a.html

Deb, K., Pratap, A., Agarwal, S., Meyarivan, T. (2002). A fast and elitist multiobjective genetic algorithm: Nsga-ii. *IEEE transactions on evolutionary computation*, *6*(2), 182–197. (https://doi.org/10.1109/4235.996017)

Deighan, D.S., Field, S.E., Capano, C.D., Khanna, G. (2021). Genetic-algorithm-optimized neural networks for gravitational wave classification. *Neural Computing and Applications*, 1–25. (https://doi.org/10.1007/s00521-021-06024-4)

de Toro, F., Ros, E., Mota, S., Ortega, J. (2002). Multi-objective optimization evolutionary algorithms applied to paroxysmal atrial fibrillation diagnosis based on the k-nearest neighbours classifier. *Ibero-american conference on artificial intelligence* (pp. 313–318). (https://doi.org/10.1007/3-540-36131-6_32)

Doerner, K., Gutjahr, W.J., Hartl, R.F., Strauss, C., Stummer, C. (2004). Pareto ant colony optimization: A metaheuristic approach to multiobjective portfolio selection. *Annals of operations research*, *131*(1), 79–99. (https://doi.org/10.1023/B:ANOR.0000039513.99038.c6)

Dorigo, M., & Blum, C. (2005). Ant colony optimization theory: A survey. *Theoretical computer science*, *344*(2-3), 243–278. (https://doi.org/10.1016/j.tcs.2005.05.020)

Dorigo, M., Maniezzo, V., Colorni, A. (1996). Ant system: optimization by a colony of cooperating agents. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, *26*(1), 29–41. (https://doi.org/

10.1109/3477.484436)

Dutta, S., & Gandomi, A.H. (2020). Surrogate model-driven evolutionary algorithms: Theory and applications. *Evolution in action: Past, present and future* (pp. 435–451). Springer. (https://doi.org/10.1007/978-3-030-39831-6_29)

Eberhart, R., & Kennedy, J. (1995). A new optimizer using particle swarm theory. *Mhs'95. proceedings of the sixth international symposium on micro machine and human science* (pp. 39–43). (https://doi.org/10.1109/MHS.1995.494215)

Ekbal, A., & Saha, S. (2015). Joint model for feature selection and parameter optimization coupled with classifier ensemble in chemical mention recognition. *Knowledge-Based Systems*, *85*, 37–51. (https://doi.org/10.1016/j.knosys.2015.04.015)

Ekbal, A., & Saha, S. (2016). Simultaneous feature and parameter selection using multiobjective optimization: application to named entity recognition. *International Journal of Machine Learning and Cybernetics*, *7*(4), 597–611. (https://doi.org/10.1007/s13042-014-0268-7)

Elsken, T., Metzen, J.H., Hutter, F. (2019). *Efficient multi-objective neural architecture search via lamarckian evolution.* International Conference on Learning Representations.

Emmerich, M.T., & Deutz, A.H. (2018). A tutorial on multiobjective optimization: fundamentals and evolutionary methods. *Natural computing*, *17*(3), 585–609.

Emmerich, M.T., Deutz, A.H., Klinkenberg, J.W. (2011). Hypervolume-based expected improvement: Monotonicity properties and exact computation. *2011 ieee congress of evolutionary computation (cec)* (pp. 2147–2154). (https://doi.org/10.1109/CEC.2011.5949880)

Ertel, W. (2018). *Introduction to artificial intelligence.* Springer.

Falkner, S., Klein, A., Hutter, F. (2018). Bohb: Robust and efficient hyperparameter optimization at scale. *International conference on machine learning* (pp. 1437–1446). Retrieved from http://proceedings.mlr.press/v80/falkner18a.html

Faris, H., Habib, M., Faris, M., Alomari, M., Alomari, A. (2020). Medical speciality classification system based on binary particle swarms and ensemble of one vs. rest support vector machines. *Journal of biomedical informatics*, *109*, 103525. (https://doi.org/10.1016/j.jbi.2020.103525)

Feliot, P., Bect, J., Vazquez, E. (2017). A bayesian approach to constrained single-and multi-objective optimization. *Journal of Global Optimization*, *67*(1-2), 97–133. (https://doi.org/10.1007/s10898-016-0427-3)

Feurer, M., & Hutter, F. (2019). Hyperparameter optimization. *Automated machine learning: methods, systems, challenges* (pp. 3–33). Springer, Cham.

Fieldsend, J.E., & Everson, R.M. (2014). The rolling tide evolutionary algorithm: A multiobjective optimizer for noisy optimization problems. *IEEE Transactions on Evolutionary Computation*, *19*(1), 103–117. (https://doi.org/10.1109/TEVC.2014.2304415)

Forrester, A.I., Keane, A.J., Bressloff, N.W. (2006). Design and analysis of" noisy" computer experiments. *AIAA journal*, *44*(10), 2331–2339. (https://doi.org/10.2514/1.20068)

Garrido, E.C., & Hernández, D. (2019). Predictive entropy search for multi-objective bayesian optimization with constraints. *Neurocomputing*, *361*, 50–68. (https://doi.org/10.1016/j.neucom.2019.06.025)

Glover, F. (1986). Future paths for integer programming and links to artificial intelligence. *Computers & operations research*, *13*(5), 533–549. (https://doi.org/10.1016/0305-0548(86)90048-1)

Gonzalez, S.R., Jalali, H., Van Nieuwenhuyse, I. (2020). A multiobjective stochastic simulation optimization algorithm. *European Journal of Operational Research*, *284*(1), 212–226. (https://doi.org/10.1016/j.ejor.2019.12.014)

Gravel, M., Price, W.L., Gagné, C. (2002). Scheduling continuous casting of aluminum using a multiple objective ant colony optimization metaheuristic. *European Journal of Operational Research*, *143*(1), 218–229. (https://doi.org/10.1016/S0377-2217(01)00329-0)

Gülcü, A., & Kuş, Z. (2021). Multi-objective simulated annealing for hyper-parameter optimization in convolutional neural networks. *PeerJ Computer Science*, *7*, e338. (https://doi.org/10.7717/peerj-cs.338)

Guo, C., Li, L., Hu, Y., Yan, J. (2020). A deep learning based fault diagnosis method with hyperparameter optimization by using parallel computing. *IEEE Access*, *8*, 131248–131256. (https://doi.org/10.1109/ACCESS.2020.3009644)

Guo, J., Yang, L., Bie, R., Yu, J., Gao, Y., Shen, Y., Kos, A. (2019). An xgboost-based physical fitness evaluation model using advanced feature selection and bayesian hyper-parameter optimization for wearable running monitoring. *Computer Networks*, *151*, 166–180. (https://doi.org/10.1016/j.comnet.2019.01.026)

Gupta, S., Shilton, A., Rana, S., Venkatesh, S. (2018). Exploiting strategy-space diversity for batch bayesian optimization. *International conference on artificial intelligence and statistics* (pp. 538–547). Retrieved from http://proceedings.mlr.press/v84/gupta18a.html

Han, S., Pool, J., Tran, J., Dally, W. (2015). Learning both weights and connections for efficient neural network. C. Cortes, N. Lawrence, D. Lee, M. Sugiyama, & R. Garnett (Eds.), *Advances in neural information processing systems* (Vol. 28, pp. 1135–1143). Curran Associates, Inc. Retrieved from https://dl.acm.org/doi/10.5555/2969239.2969366

Hansen, N., Müller, S.D., Koumoutsakos, P. (2003). Reducing the time complexity of the derandomized evolution strategy with covariance matrix adaptation (cma-es). *Evolutionary computation*, *11*(1), 1–18. (https://doi.org/10.1162/106365603321828970)

Hegde, S., & Mundada, M.R. (2020). Early prediction of chronic disease using an efficient machine learning algorithm through adaptive probabilistic divergence based feature selection approach. *International Journal of Pervasive Computing and Communications*. (https://doi.org/10.1108/IJPCC-04-2020-0018)

Hernández, D., Hernandez-Lobato, J., Shah, A., Adams, R. (2016). Predictive entropy search for multi-objective bayesian optimization. *International conference on machine learning* (pp. 1492–1501). Retrieved from http://proceedings.mlr.press/v48/hernandez-lobatoa16.html

Hernández-Lobato, J.M., Gelbart, M.A., Reagen, B., Adolf, R., Hernández-Lobato, D., Whatmough, P.N., ... Adams, R.P. (2016). Designing neural network hardware accelerators with decoupled objective evaluations. *Nips workshop on bayesian optimization* (p. 10).

Ho, T.K. (1995). Random decision forests. *Proceedings of 3rd international conference on document analysis and recognition* (Vol. 1, pp. 278–282). (https://doi.org/10.1109/ICDAR.1995.598994)

Horn, D., & Bischl, B. (2016). Multi-objective parameter configuration of machine learning algorithms using model-based optimization. *2016 ieee symposium series on computational intelligence (ssci)* (pp. 1–8). (https://doi.org/10.1109/SSCI.2016.7850221)

Horn, D., Dagge, M., Sun, X., Bischl, B. (2017). First investigations on noisy model-based multi-objective optimization. *International conference on evolutionary multi-criterion optimization* (pp. 298–313). (https://doi.org/10.1007/978-3-319-54157-0_21)

Hsu, C.-H., & Juang, C.-F. (2013). Multi-objective continuous-ant-colony-optimized fc for robot wall-following control. *IEEE Computational Intelligence Magazine*, *8*(3), 28–40. (https://doi.org/10.1109/MCI.2013.2264233)

Hu, W., Jin, J., Liu, T.-Y., Zhang, C. (2019). Automatically design convolutional neural networks by optimization with submodularity and supermodularity. *IEEE Transactions on Neural Networks and Learning Systems*. (https://doi.org/10.1109/TNNLS.2019.2939157)

Hutter, F., Hoos, H.H., Leyton-Brown, K., Stützle, T. (2009). Paramils: an automatic algorithm configuration framework. *Journal of Artificial Intelligence Research*, *36*, 267–306. (https://doi.org/10.1613/jair.2861)

Hutter, F., Lücke, J., Schmidt-Thieme, L. (2015). Beyond manual tuning of hyperparameters. *KI-Künstliche Intelligenz*, *29*(4), 329–337. (https://doi.org/10.1007/s13218-015-0381-0)

Iredi, S., Merkle, D., Middendorf, M. (2001). Bi-criterion optimization with multi colony ant algorithms. *International conference on evolutionary multi-criterion optimization* (pp. 359–372). (https://doi.org/10.1007/3-540-44719-9_25)

Jalali, H., Van Nieuwenhuyse, I., Picheny, V. (2017). Comparison of kriging-based algorithms for simulation optimization with heterogeneous noise. *European Journal of Operational Research*, *261*(1), 279–301. (https://doi.org/10.1016/j.ejor.2017.01.035)

Jiang, F., Jiang, Y., Zhi, H., Dong, Y., Li, H., Ma, S., ... Wang, Y. (2017). Artificial intelligence in healthcare: past, present and future. *Stroke and vascular neurology*, *2*(4), 230–243. (https://doi.org/10.1136/svn-2017-000101)

Jiang, J., Han, F., Ling, Q., Wang, J., Li, T., Han, H. (2020). Efficient network architecture search via multiobjective particle swarm optimization based on decomposition. *Neural Networks*, *123*, 305–316. (https://doi.org/10.1016/j.neunet.2019.12.005)

Jin, Y. (2011). Surrogate-assisted evolutionary computation: Recent advances and future challenges. *Swarm and Evolutionary Computation*, *1*(2), 61–70. (https://doi.org/10.1016/j.swevo.2011.05.001)

Jing, W., Lin, J., Wang, H. (2020). Building nas: Automatic designation of efficient neural architectures for building extraction in high-resolution aerial images. *IMAGE AND VISION COMPUTING*, *103*. (https://doi.org/10.1016/j.imavis.2020.104025)

Jones, D.R., Schonlau, M., Welch, W.J. (1998). Efficient global optimization of expensive black-box functions. *Journal of Global optimization*, *13*(4), 455–492. (https://doi.org/10.1023/A:1008306431147)

Juang, C.-F. (2002). A tsk-type recurrent fuzzy network for dynamic systems processing by neural network and genetic algorithms. *IEEE Transactions on Fuzzy Systems*, *10*(2), 155–170. (https://doi.org/10.1109/91.995118)

Juang, C.-F., & Hsu, C.-H. (2014). Structure and parameter optimization of fnns using multi-objective aco for control and prediction. *2014 ieee international conference on fuzzy systems (fuzz-ieee)* (pp. 928–933). (https://doi.org/10.1109/FUZZ-IEEE.2014.6891545)

Keshtkaran, M.R., & Pandarinath, C. (2019). Enabling hyperparameter optimization in sequential autoencoders for spiking neural data [Proceedings Paper]. H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alche Buc,

E. Fox, & R. Garnett (Eds.), *Advances in neural information processing systems 32 (nips 2019)* (Vol. 32).

Kim, Y., & Chung, M. (2019, NOV). An approach to hyperparameter optimization for the objective function in machine learning. *ELECTRONICS*, *8*(11). (https://doi.org/10.3390/electronics8111267)

Kim, Y., Reddy, B., Yun, S., Seo, C. (2017). Nemo: Neuro-evolution with multiobjective optimization of deep neural network for speed and accuracy. *Icml 2017 automl workshop.*

Kirkpatrick, S., Gelatt, C.D., Vecchi, M.P. (1983). Optimization by simulated annealing. *science*, *220*(4598), 671–680. (https://doi.org/10.1126/science.220.4598.671)

Knowles, J. (2006). Parego: A hybrid algorithm with on-line landscape approximation for expensive multiobjective optimization problems. *IEEE Transactions on Evolutionary Computation*, *10*(1), 50–66. (https://doi.org/10.1109/TEVC.2005.851274)

Koch, P., Wagner, T., Emmerich, M.T., Bäck, T., Konen, W. (2015). Efficient multi-criteria optimization on noisy machine learning problems. *Applied Soft Computing*, *29*, 357–370. (https://doi.org/10.1016/j.asoc.2015.01.005)

Kohavi, R., & John, G.H. (1995). Automatic parameter selection by minimizing estimated error. *Machine learning proceedings 1995* (pp. 304–312). Elsevier. (https://doi.org/10.1016/B978-1-55860-377-6.50045-1)

Kong, W., Dong, Z.Y., Luo, F., Meng, K., Zhang, W., Wang, F., Zhao, X. (2017). Effect of automatic hyperparameter tuning for residential load forecasting via deep learning. *2017 australasian universities power engineering conference (aupec)* (pp. 1–6). (https://doi.org/10.1109/AUPEC.2017.8282478)

Laskaridis, S., Venieris, S.I., Kim, H., Lane, N.D. (2020). Hapi: hardware-aware progressive inference. *2020 ieee/acm international conference on computer aided design (iccad)* (pp. 1–9).

Laumanns, M., Thiele, L., Deb, K., Zitzler, E. (2002). Combining convergence and diversity in evolutionary multiobjective optimization. *Evolutionary computation*, *10*(3), 263–282. (https://doi.org/10.1162/106365602760234108)

León, J., Ortega, J., Ortiz, A. (2019). Convolutional neural networks and feature selection for bci with multiresolution analysis. *International work-conference on artificial neural networks* (pp. 883–894). (https://doi.org/10.1007/978-3-030-20521-8_72)

Li, H., Zhang, Q., Tsang, E., Ford, J.A. (2004). Hybrid estimation of distribution algorithm for multiobjective knapsack problem. J. Gottlieb & G.R. Raidl (Eds.), *Evolutionary computation in combinatorial optimization* (pp. 145–154). (https://doi.org/10.1007/978-3-540-24652-7_15)

Li, L., Jamieson, K., DeSalvo, G., Rostamizadeh, A., Talwalkar, A. (2017). Hyperband: A novel bandit-based approach to hyperparameter optimization. *The Journal of Machine Learning Research*, *18*(1), 6765–6816. Retrieved from https://jmlr.org/papers/v18/16-558.html

Li, M., & Yao, X. (2019). Quality evaluation of solution sets in multiobjective optimisation: A survey. *ACM Computing Surveys (CSUR)*, *52*(2), 1–38. (https://doi.org/10.1145/3300148)

Li, S., Gong, W., Yan, X., Hu, C., Bai, D., Wang, L. (2019). Parameter estimation of photovoltaic models with memetic adaptive differential evolution. *Solar Energy*, *190*, 465–474. (https://doi.org/10.1016/j.solener.2019.08.022)

Liang, J., Meyerson, E., Hodjat, B., Fink, D., Mutch, K., Miikkulainen, R. (2019). Evolutionary neural automl for deep learning. *Proceedings of the genetic and evolutionary computation conference* (pp. 401–409). (https://doi.org/10.1145/3321707.3321721)

Liu, H., Cai, J., Ong, Y.-S. (2018). Remarks on multi-output gaussian process regression. *Knowledge-Based Systems*, *144*, 102–121. (https://doi.org/10.1016/j.knosys.2017.12.034)

Liu, J., Tunguz, B., Titericz, G. (2020). *Gpu accelerated exhaustive search for optimal ensemble of black-box optimization algorithms.* Workshop at NeurIPS 2020 Competition Track on Black-Box Optimization Challenge.

Loni, M., Sinaei, S., Zoljodi, A., Daneshtalab, M., Sjödin, M. (2020). Deepmaker: A multi-objective optimization framework for deep neural networks in embedded systems. *Microprocessors and Microsystems*, *73*,

102989. (https://doi.org/10.1016/j.micpro.2020.102989)

Loni, M., Zoljodi, A., Sinaei, S., Daneshtalab, M., Sjödin, M. (2019). Neuropower: Designing energy efficient convolutional neural network architecture for embedded systems. *International conference on artificial neural networks* (pp. 208–222). (https://doi.org/10.1007/978-3-030-30487-4_17)

López-Ibáñez, M., Dubois-Lacoste, J., Cáceres, L.P., Birattari, M., Stützle, T. (2016). The irace package: Iterated racing for automatic algorithm configuration. *Operations Research Perspectives*, *3*, 43–58. (https://doi.org/10.1016/j.orp.2016.09.002)

Lu, Z., Deb, K., Goodman, E., Banzhaf, W., Boddeti, V.N. (2020). Nsganetv2: Evolutionary multi-objective surrogate-assisted neural architecture search. *European conference on computer vision* (pp. 35–51). (https://doi.org/10.1007/978-3-030-58452-8_3)

Lu, Z., Whalen, I., Boddeti, V., Dhebar, Y., Deb, K., Goodman, E., Banzhaf, W. (2019). Nsga-net: neural architecture search using multi-objective genetic algorithm. *Proceedings of the genetic and evolutionary computation conference* (pp. 419–427). (https://doi.org/10.1145/3321707.3321729)

Luo, G. (2016). A review of automatic selection methods for machine learning algorithms and hyper-parameter values. *Network Modeling Analysis in Health Informatics and Bioinformatics*, *5*(1), 18. (https://doi.org/10.1007/s13721-016-0125-6)

Magda, M., Martinez-Alvarez, A., Cuenca-Asensi, S. (2017). Mooga parameter optimization for onset detection in emg signals. *International conference on image analysis and processing* (pp. 171–180). (https://doi.org/10.1007/978-3-319-70742-6_16)

Makarova, A., Shen, H., Perrone, V., Klein, A., Faddoul, J.B., Krause, A., ... Archambeau, C. (2021). *Overfitting in bayesian optimization: an empirical study and early-stopping solution.* Retrieved from https://www.amazon.science/publications/overfitting-in-bayesian-optimization-an-empirical-study-and-early-stopping-solution

Martinez-de Pison, F.J., Gonzalez-Sendino, R., Aldama, A., Ferreiro, J., Fraile, E. (2017). Hybrid methodology based on bayesian optimization and ga-parsimony for searching parsimony models by combining hyperparameter optimization and feature selection. *International conference on hybrid artificial intelligence systems* (pp. 52–62). (https://doi.org/10.1016/j

.neucom.2018.05.136)

McKinnon, K.I. (1998). Convergence of the nelder–mead simplex method to a nonstationary point. *SIAM Journal on optimization*, *9*(1), 148–158. (https://doi.org/10.1137/S1052623496303482)

Mei, J., Li, Y., Lian, X., Jin, X., Yang, L., Yuille, A., Yang, J. (2020). Atomnas: Fine-grained end-to-end neural architecture search. *International conference on learning representations*. (https://openreview.net/forum?id=BylQSxHFwr)

Meinshausen, N., & Ridgeway, G. (2006). Quantile regression forests. *Journal of Machine Learning Research*, *7*(6). Retrieved from http://jmlr.org/papers/v7/meinshausen06a.html

Mentch, L., & Hooker, G. (2016). Quantifying uncertainty in random forests via confidence intervals and hypothesis tests. *The Journal of Machine Learning Research*, *17*(1), 841–881. Retrieved from https://jmlr.org/papers/v17/14-168.html

Miettinen, K. (2012). *Nonlinear multiobjective optimization* (Vol. 12). Springer Science & Business Media.

Miettinen, K., & Mäkelä, M.M. (2002). On scalarizing functions in multiobjective optimization. *OR spectrum*, *24*(2), 193–213. (https://doi.org/10.1007/s00291-001-0092-9)

Mitchell, M. (1998). *An introduction to genetic algorithms*. MIT press.

Mitchell, T.M., et al. (1997). Machine learning. *Burr Ridge, IL: McGraw Hill*, *45*(37), 870–877.

Montgomery, D.C. (2017). *Design and analysis of experiments*. John wiley & sons.

Mostafa, S.S., Mendonça, F., Ravelo-Garcia, A.G., Juliá-Serdá, G.G., Morgado-Dias, F. (2020). Multi-objective hyperparameter optimization of convolutional neural network for obstructive sleep apnea detection. *IEEE Access*, *8*, 129586–129599. (https://doi.org/10.1109/ACCESS.2020.3009149)

Negrinho, R., Gormley, M., Gordon, G.J., Patil, D., Le, N., Ferreira, D. (2019). Towards modular and programmable architecture search. *Advances in neural information processing systems* (pp. 13715–13725). Retrieved from https://dl.acm.org/doi/abs/10.5555/3454287.3455517

Olsson, D.M., & Nelson, L.S. (1975). The nelder-mead simplex procedure for function minimization. *Technometrics*, *17*(1), 45–51. (https://doi.org/10.1080/00401706.1975.10489269)

Ounpraseuth, S.T. (2008). *Gaussian processes for machine learning*. Taylor & Francis.

Ozaki, Y., Tanigaki, Y., Watanabe, S., Onishi, M. (2020). Multiobjective tree-structured parzen estimator for computationally expensive optimization problems. *Proceedings of the 2020 genetic and evolutionary computation conference* (pp. 533–541). (https://doi.org/10.1145/3377930.3389817)

Parsa, M., Ankit, A., Ziabari, A., Roy, K. (2019). Pabo: Pseudo agent-based multi-objective bayesian hyperparameter optimization for efficient neural accelerator design. *2019 ieee/acm international conference on computer-aided design (iccad)* (pp. 1–8). (https://doi.org/10.1109/ICCAD45719.2019.8942046)

Pathak, Y., Shukla, P.K., Arya, K. (2020). Deep bidirectional classification model for covid-19 disease infected patients. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*. (https://doi.org/10.1109/TCBB.2020.3009859)

Pech, S., Kandler, G., Lukacevic, M., Füssl, J. (2019). Metamodel assisted optimization of glued laminated timber beams by using metaheuristic algorithms. *Engineering Applications of Artificial Intelligence*, *79*, 129–141. (https://doi.org/10.1016/j.engappai.2018.12.010)

Phillips, P.J., Flynn, P.J., Scruggs, T., Bowyer, K.W., Chang, J., Hoffman, K., ... Worek, W. (2005). Overview of the face recognition grand challenge. *2005 ieee computer society conference on computer vision and pattern recognition (cvpr'05)* (Vol. 1, pp. 947–954). (https://doi.org/10.1109/CVPR.2005.268)

Picheny, V. (2014). A stepwise uncertainty reduction approach to constrained global optimization. *Artificial intelligence and statistics* (pp. 787–795). Retrieved from http://proceedings.mlr.press/v33/picheny14.html

Ponweiser, W., Wagner, T., Biermann, D., Vincze, M. (2008). Multiobjective optimization on a limited budget of evaluations using model-assisted s-metric selection. *International conference on parallel problem solving from nature* (pp. 784–794). (https://doi.org/10.1007/978-3-540-87700-4_78)

Provost, F., Jensen, D., Oates, T. (1999). Efficient progressive sampling. *Proceedings of the fifth acm sigkdd international conference on knowledge discovery and data mining* (pp. 23–32). (https://doi.org/10.1145/312129.312188)

Qin, A.K., Huang, V.L., Suganthan, P.N. (2008). Differential evolution algorithm with strategy adaptation for global numerical optimization. *IEEE transactions on Evolutionary Computation*, *13*(2), 398–417. (https://doi.org/10.1109/TEVC.2008.927706)

Qin, H., Shinozaki, T., Duh, K. (2017). Evolution strategy based automatic tuning of neural machine translation systems. *Proceeding of international workshop on spoken language translation (iwslt)* (pp. 120–128).

Rajagopal, A., Joshi, G.P., Ramachandran, A., Subhalakshmi, R., Khari, M., Jha, S., ... You, J. (2020). A deep learning model based on multi-objective particle swarm optimization for scene classification in unmanned aerial vehicles. *IEEE Access*, *8*, 135383–135393. (https://doi.org/10.1109/ACCESS.2020.3011502)

Regis, R.G. (2014). Particle swarm with radial basis function surrogates for expensive black-box optimization. *Journal of Computational Science*, *5*(1), 12–23. (https://doi.org/10.1016/j.jocs.2013.07.004)

Richter, J., Kotthaus, H., Bischl, B., Marwedel, P., Rahnenführer, J., Lang, M. (2016). Faster model-based optimization through resource-aware scheduling strategies. *International conference on learning and intelligent optimization* (pp. 267–273). (https://doi.org/10.1007/978-3-319-50349-3_22)

Rojas-Gonzalez, S., & Van Nieuwenhuyse, I. (2020). A survey on kriging-based infill algorithms for multiobjective simulation optimization. *Computers & Operations Research*, *116*, 104869. (https://doi.org/10.1016/j.cor.2019.104869)

Russell, S., & Norvig, P. (2010). *Artificial intelligence: A modern approach* (3rd ed.). Prentice Hall.

Salt, L., Howard, D., Indiveri, G., Sandamirskaya, Y. (2019). Parameter optimization and learning in a spiking neural network for uav obstacle avoidance targeting neuromorphic processors. *IEEE Transactions on Neural Networks and Learning Systems*. (https://doi.org/10.1109/TNNLS.2019.2941506)

Sanz-García, A., Fernández-Ceniceros, J., Antonanzas-Torres, F., Pernia-Espinoza, A., Martinez-De-Pison, F. (2015). Ga-parsimony: A ga-svr approach with feature selection and parameter optimization to obtain parsimonious solutions for predicting temperature settings in a continuous annealing furnace. *Applied Soft Computing*, *35*, 13–28. (https://doi.org/10.1016/j.asoc.2015.06.012)

Schaffer, J.D. (1985). Multiple objective optimization with vector evaluated genetic algorithms. *Proceedings of the 1st international conference on genetic algorithms* (p. 93–100). USA: L. Erlbaum Associates Inc.

Shah, A., & Ghahramani, Z. (2016). Pareto frontier learning with expensive correlated objectives. *International conference on machine learning* (pp. 1919–1927). Retrieved from http://proceedings.mlr.press/v48/shahc16.html

Shinozaki, T., Watanabe, S., Duh, K. (2020). Automated development of dnn based spoken language systems using evolutionary algorithms. *Deep neural evolution* (pp. 97–129). Springer. (https://doi.org/10.1007/978-981-15-3685-4_4)

Sierra, M.R., & Coello, C.A.C. (2005). Improving pso-based multi-objective optimization using crowding, mutation and ∈-dominance. *International conference on evolutionary multi-criterion optimization* (pp. 505–519). (https://doi.org/10.1007/978-3-540-31880-4_35)

Silva, L.F., Santos, A.A.S., Bravo, R.S., Silva, A.C., Muchaluat-Saade, D.C., Conci, A. (2016). Hybrid analysis for indicating patients with breast cancer using temperature time series. *Computer methods and programs in biomedicine*, *130*, 142–153. (https://doi.org/10.1016/j.cmpb.2016.03.002)

Singh, D., Kumar, V., Kaur, M. (2020). Classification of covid-19 patients from chest ct images using multi-objective differential evolution-based convolutional neural networks. *European Journal of Clinical Microbiology & Infectious Diseases*, 1–11. (https://doi.org/10.1007/s10096-020-03901-z)

Sjöberg, A., Önnheim, M., Gustavsson, E., Jirstrand, M. (2019). Architecture-aware bayesian optimization for neural network tuning. *International conference on artificial neural networks* (pp. 220–231). (https://doi.org/10.1007/978-3-030-30484-3_19)

Smith, C., & Jin, Y. (2014). Evolutionary multi-objective generation of recurrent neural network ensembles for time series prediction. *Neurocomputing*, *143*, 302–311. (https://doi.org/10.1016/j.neucom.2014.05.062)

Smithson, S.C., Yang, G., Gross, W.J., Meyer, B.H. (2016). Neural networks designing neural networks: multi-objective hyper-parameter optimization. *Proceedings of the 35th international conference on computer-aided design* (pp. 1–8). (https://doi.org/10.1145/2966986.2967058)

Snoek, J., Larochelle, H., Adams, R.P. (2012). Practical bayesian optimization of machine learning algorithms. *Advances in neural information processing systems*, *25*.

Socha, K., & Dorigo, M. (2008). Ant colony optimization for continuous domains. *European journal of operational research*, *185*(3), 1155–1173. (https://doi.org/10.1016/j.ejor.2006.06.046)

Sopov, E., & Ivanov, I. (2015). Self-configuring ensemble of neural network classifiers for emotion recognition in the intelligent human-machine interaction. *2015 ieee symposium series on computational intelligence* (pp. 1808–1815). (https://doi.org/10.1109/SSCI.2015.252)

Srinivas, N., & Deb, K. (1994). Multiobjective optimization using nondominated sorting in genetic algorithms. *Evolutionary computation*, *2*(3), 221–248. (https://doi.org/10.1162/evco.1994.2.3.221)

Stamoulis, D., Cai, E., Juan, D.-C., Marculescu, D. (2018). Hyperpower: Power-and memory-constrained hyper-parameter optimization for neural networks. *2018 design, automation & test in europe conference & exhibition (date)* (pp. 19–24). (https://doi.org/10.23919/DATE.2018.8341973)

Stamoulis, D., Chin, T.-W., Prakash, A.K., Fang, H., Sajja, S., Bognar, M., Marculescu, D. (2018). Designing adaptive neural networks for energy-constrained image classification. *Proceedings of the international conference on computer-aided design* (pp. 1–8). (https://doi.org/10.1145/3240765.3240796)

Storn, R., & Price, K. (1997). Differential evolution - a simple and efficient heuristic for global optimization over continuous spaces. *Journal of global optimization*, *11*(4), 341–359. (https://doi.org/10.1023/A:1008202821328)

Sun, X.Y., Gong, D.W., Ma, X.P. (2009). Directed fuzzy graph-based surrogate model-assisted interactive genetic algorithms with uncertain individual's fitness. *2009 ieee congress on evolutionary computation* (pp. 2395–2402). (https://doi.org/10.1109/CEC.2009.4983240)

Talbi, E.-G. (2021). Automated design of deep neural networks: A survey and unified taxonomy. *ACM Computing Surveys (CSUR)*, *54*(2), 1–37. (https://doi.org/10.1145/3439730)

Tanabe, R., & Fukunaga, A. (2013). Success-history based parameter adaptation for differential evolution. *2013 ieee congress on evolutionary computation* (pp. 71–78). (https://doi.org/10.1109/CEC.2013.6557555)

Tanaka, T., Moriya, T., Shinozaki, T., Watanabe, S., Hori, T., Duh, K. (2016). Automated structure discovery and parameter tuning of neural network language model based on evolution strategy. *2016 ieee spoken language technology workshop (slt)* (pp. 665–671). (https://doi.org/10.1109/SLT.2016.7846334)

Tharwat, A. (2020). Classification assessment methods. *Applied Computing and Informatics*. (https://doi.org/10.1016/j.aci.2018.08.003)

Thornton, C., Hutter, F., Hoos, H.H., Leyton-Brown, K. (2013). Auto-weka: Combined selection and hyperparameter optimization of classification algorithms. *Proceedings of the 19th acm sigkdd international conference on knowledge discovery and data mining* (pp. 847–855). (https://doi.org/10.1145/2487575.2487629)

van Rijn, J.N., Abdulrahman, S.M., Brazdil, P., Vanschoren, J. (2015). Fast algorithm selection using learning curves. *International symposium on intelligent data analysis* (pp. 298–309). (https://doi.org/10.1007/978-3-319-24465-5_26)

Vanschoren, J. (2019). Meta-learning. *Automated machine learning: methods, systems, challenges* (pp. 35–61). Springer, Cham.

Victoria, A.H., & Maragatham, G. (2021). Automatic tuning of hyperparameters using bayesian optimization. *Evolving Systems*, 217-–223. (https://doi.org/10.1007/s12530-020-09345-2)

Wang, B., Sun, Y., Xue, B., Zhang, M. (2019). Evolving deep neural networks by multi-objective particle swarm optimization for image classification. *Proceedings of the genetic and evolutionary computation conference* (pp. 490–498). (https://doi.org/10.1145/3321707.3321735)

Wang, B., Xue, B., Zhang, M. (2020). Particle swarm optimization for evolving deep convolutional neural networks for image classification: Single- and multi-objective approaches. *Deep neural evolution* (pp. 155–184). Springer. (https://doi.org/10.1007/978-981-15-3685-4_6)

Wawrzyński, P. (2017). Asd+ m: Automatic parameter tuning in stochastic optimization and on-line learning. *Neural Networks*, *96*, 1–10. (https://doi.org/10.1016/j.neunet.2017.07.007)

Wistuba, M., Schilling, N., Schmidt-Thieme, L. (2018). Scalable gaussian process-based transfer surrogates for hyperparameter optimization. *Machine Learning*, *107*(1), 43–78. (https://doi.org/10.1007/s10994-017-5684-y)

Yang, L., & Shami, A. (2020). On hyperparameter optimization of machine learning algorithms: Theory and practice. *Neurocomputing*, *415*, 295–316. (https://doi.org/10.1016/j.neucom.2020.07.061)

Zames, G., Ajlouni, N., Ajlouni, N., Ajlouni, N., Holland, J., Hills, W., Goldberg, D. (1981). Genetic algorithms in search, optimization and machine learning. *Information Technology Journal*, *3*(1), 301–302.

Zhang, C., Lim, P., Qin, A.K., Tan, K.C. (2016). Multiobjective deep belief networks ensemble for remaining useful life estimation in prognostics. *IEEE transactions on neural networks and learning systems*, *28*(10), 2306–2318. (https://doi.org/10.1109/TNNLS.2016.2582798)

Zhang, Q., & Li, H. (2007). Moea/d: A multiobjective evolutionary algorithm based on decomposition. *IEEE Transactions on evolutionary computation*, *11*(6), 712–731. (https://doi.org/10.1109/TEVC.2007.892759)

Zitzler, E., Deb, K., Thiele, L. (2000). Comparison of multiobjective evolutionary algorithms: Empirical results. *Evolutionary computation*, *8*(2), 173–195. (https://doi.org/10.1162/106365600568202)

Zitzler, E., Laumanns, M., Thiele, L. (2001). Spea2: Improving the strength pareto evolutionary algorithm. *TIK-report*, *103*. (https://doi.org/10.3929/ethz-a-004284029)

Zitzler, E., & Thiele, L. (1999). Multiobjective evolutionary algorithms: a comparative case study and the strength pareto approach. *IEEE Transactions on Evolutionary Computation*, *3*(4), 257–271. (https://doi.org/10.1109/4235.797969)