Artificial Intelligence Review (2026) 59:5
https://doi.org/10.1007/s10462-025-11397-2

A literature review on automated machine learning

Edesio Alcobaça[1] · **André C. P. L. F. de Carvalho**[1]

Received: 21 January 2025 / Accepted: 8 September 2025

Abstract
AutoML represents a pivotal advancement in machine learning by simplifying and speeding model development. This paper provides a comprehensive survey of AutoML, tracing its evolution from early metalearning, hyperparameter optimization, and transfer learning techniques to the latest advancements in neural architecture search, automated pipeline design, and few-shot learning. It covers historical context, classical approaches, and modern applications while also addressing emerging topics. Key research directions are highlighted, focusing on enhancing model interpretability, improving generalization and robustness, expanding automated pipeline design, and ethical implications of AutoML technologies. This paper aims to provide a holistic view of the current state of AutoML, serving as a valuable resource for researchers, practitioners, and stakeholders seeking to understand and advance the capabilities of AutoML in both theoretical and practical contexts.

Keywords Automated machine learning · AutoML · Metalearning · Hyperparameter optimization · Transfer learning

1 Introduction

Machine learning (ML) has experienced unprecedented growth in recent years, advancing across various domains and fostering innovations and applications in numerous sectors (Provost and Fawcett 2013). One of the most significant developments in ML is the emergence of Automated Machine Learning (AutoML) (Feurer et al. 2015; Hutter et al. 2019). AutoML seeks to automate the end-to-end ML process, democratizing access to ML benefits and accelerating the development of data-driven solutions (Yao et al. 2018).

✉ Edesio Alcobaça
e.alcobaca@gmail.com

André C. P. L. F. de Carvalho
andre@icmc.usp.br

[1] Institute of Mathematics and Computer Sciences, University of São Paulo, Av. Trabalhador São Carlense, São Carlos, SP 13566-590, Brazil

Published online: 11 November 2025

The concept of AutoML is not entirely new, it has a rich and extensive history (Smith-Miles 2009). Its foundational pillars were established through classical metalearning, hyperparameter optimization and transfer learning (Rice 1976; Ritter 1991; Aha et al. 1991). Metalearning (MtL) and Transfer Learning (TL), commonly described as “learning to learn" (Brazdil et al. 2003), involve creating algorithms that adapt their learning strategies based on prior experiences on similar tasks. Classical hyperparameter optimization (HPO) usually occurs by systematically adjusting the values of hyperparameters that govern the learning process toward enhancing model performance.

This literature review provides a comprehensive overview of the current state of AutoML by examining its historical background, foundational methodologies, and recent advancements in the field. We highlight the following contributions: (i) use of systematic literature review approach (of literature reviews); (ii) historical background of algorithms selection; (iii) outline back classical definitions of MtL, HPO, and TL; (iv) connect the areas around meta-knowledge and AutoML definition (v) categorize and organize literature review; (vi) applications connected with of MtL, HPO, and TL; (vii) future research directions.

The paper is organized as follows. In Sect. 2, we describe the systematic literature review protocol employed. Section 3 presents a brief history of AutoML. In Sect. 4, we revisit classical definitions of MtL, HPO, and TL. Section 5 discusses how these approaches are unified under the concept of meta-knowledge. Section 6 introduces a comprehensive definition of AutoML that encompasses these foundational components. Section 8 outlines practical applications of AutoML across various domains. Finally, Sect. 9 highlights future research directions, and Sect. 10 concludes the paper.

2 Systematic literature review

A Systematic Literature Review (SLR) is a structured method for identifying, evaluating, and summarizing existing research on a specific topic (Kitchenham et al. 2009). This approach, initially developed in medicine and later widely adopted in software engineering, aims to reduce bias by following a clear, predefined protocol. Moreover, a SLR provides transparency and reproducibility of the study, making it easy to track and update (Kitchenham et al. 2009).

This study proposes a systematic literature review of existing literature reviews within the domain of AutoML. Conducting an SLR of other literature reviews enables a broader knowledge synthesis by integrating findings from multiple comprehensive reviews (Kitchenham et al. 2009; Petersen et al. 2015). It can also provide a more mature structure and view of a research domain (Petersen et al. 2008). Finally, it can offer a meta-perspective on methodological trends, application areas, and underexplored topics (Petersen et al. 2008).
The adopted methodology of this paper follows several structured steps, as illustrated in Fig. 1: (i) a set of research questions was defined to guide the review process; (ii) a search string was formulated to identify relevant studies; (iii) a systematic search for surveys and reviews was conducted using the Scopus and Web of Science databases; (iv) inclusion and exclusion criteria were applied to select appropriate studies; (v) data extraction was performed to collect the necessary information; and (vi) the research questions were addressed based on

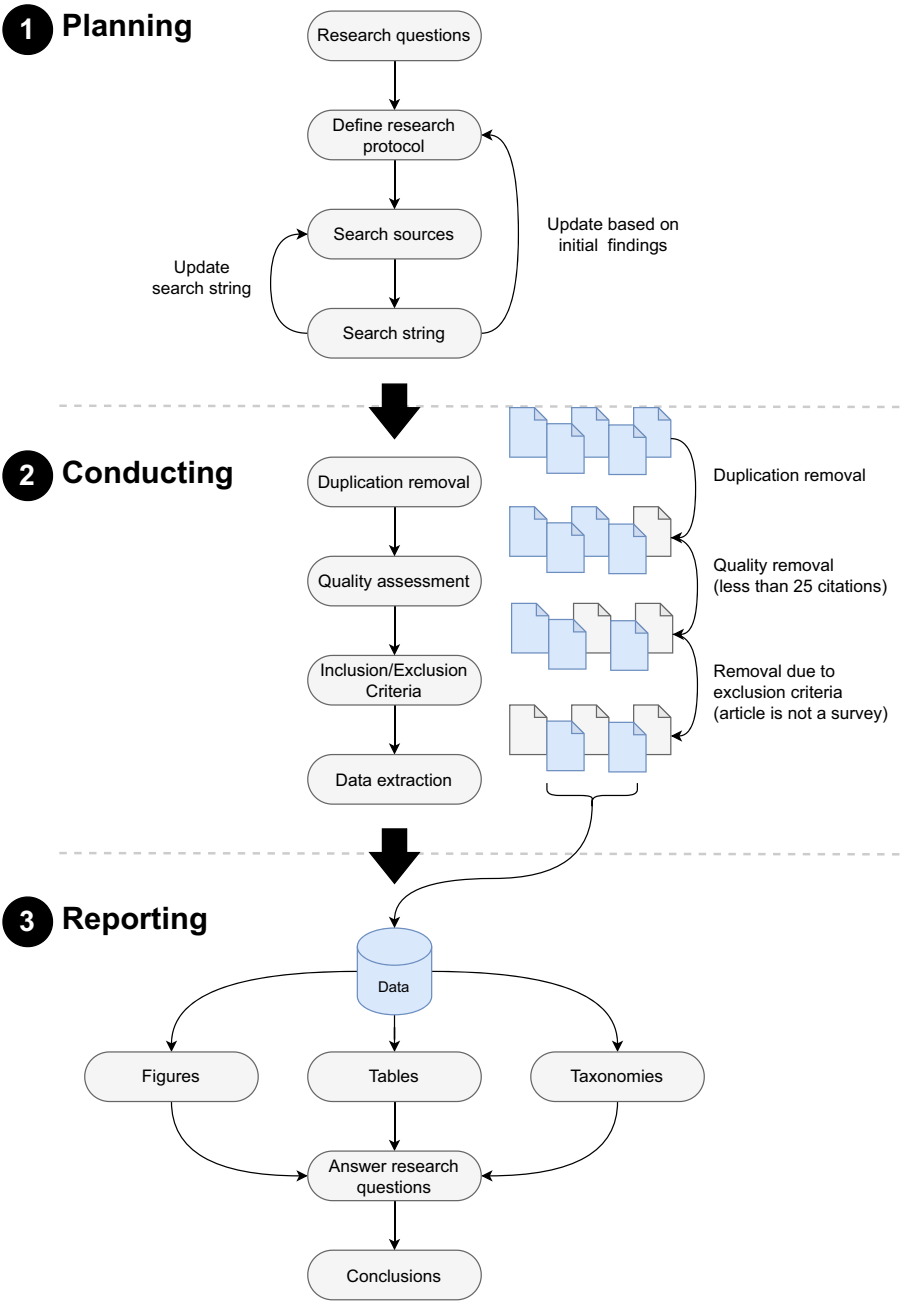

Fig. 1 Systematic literature review process

the extracted data. The search results are available on GitHub [1] while the search string used is shown below:

```
( “algorithm recommendation" OR “algorithm portfolio" OR “algorithm recommender"
OR “model recommendation" OR “model recommender" OR
“automated machine learning" OR “automated algorithm design"
OR “automatic machine learning" OR “automl" OR “pipeline design" OR
“pipeline optimization" OR “hyperparameter tuning" OR “algorithm configuration"
OR “hyper parameter optimization" OR “hyperparameter optimization" OR
“hyper parameter tuning" OR “meta-learning" OR “metalearning" OR
“transfer learning" ) AND ( “survey" OR “review" ) AND ( “machine learning"
OR “data mining" OR “data science" )
```

In terms of inclusion and exclusion criteria, we used the following: (i) only survey papers with at least 25 citations in the data sources were considered; (ii) specific application surveys were not considered (e.g., AutoML applied to healthcare). Figure 2 illustrates the conducting phase culminating in the 52 selected survey papers.

Therefore, the SLR was used as a tool to answer the following research questions:

Q1. What is the historical background of AutoML?
A: Answered in Sect. 3.
Q2. What are the classical definitions?
A: Answered in Sect. 4.
Q3. How does meta-knowledge bridge different areas of machine learning?
A: Answered in Sects. 5 and 6.
Q4. How can AutoML-related surveys be systematically categorized?
A: Answered in Sect. 7.
Q5. What are the main applications of AutoML?
A: Answered in Sect. 8.
Q6. What are the current and emerging research directions in the field of AutoML?
A: Answered in Sect. 9.

3 A brief history

The study of bias selection for Machine Learning (ML) algorithms,[2] strongly related to AutoML, has a long and fertile history, traced back to the middle of the 1970s (Rice 1976). Automatic algorithm selection (Brodley 1993), hyperparameter optimization (Ritter 1991),

[1] .

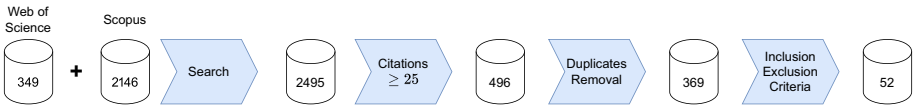

Fig. 2 Selection phase

[2] For the definition of bias, this text adopts the definition from Tom Mitchell: “any basis for choosing one generalization [hypothesis] over another, other than strict consistency with the observed training instances" (Mitchell 1980). For a good discussion regarding this definition, see Dietterich and Kong (1995).

algorithm recommendation (Brazdil et al. 2003), metalearning (Aha et al. 1991), transfer learning (Pratt et al. 1991) and AutoML (Thornton et al. 2013; Feurer et al. 2015; Olson et al. 2016) are some of the terminologies, and approaches, used by different research groups working in this area. Without taking into account their specificities, these groups were trying to answer the following research question: “Which algorithm is likely to present the best performance for a given new problem?", which was initially posed by John Rice (Rice 1976). Figure 3 shows a timeline with the distinct names adopted by scientific contributions from different groups that, on some level, are related to Rice’s initial question and consequently AutoML.

In the current context, this question can be modified to “which algorithms" or “which pre-trained models", because of the different aspects associated with the whole solution bias, such as preprocessing techniques, model architectures, learning algorithms, values of hyperparameters, and the whole learning pipeline. Moreover, the “given problem" can be understood as a learning problem, such as learning disease diagnosis, stock price, object recognition, and robot actions.

Recent studies have addressed new research questions related with Rice’s question, which include: "which hyperparameter values are more suitable for the induction of a model", "which neural architecture can benefit most from a deep learning algorithm when applied to a given task", and even "which is the best pipeline for an end-to-end ML solution". Besides, they have expanded the research questions by adding limitations such as time, memory, and other desired things (Brazdil et al. 2009; Hutter et al. 2019).

In 1976 Rice (1976) formally defined the algorithm selection problem proposing a framework called the abstract model, which was based on the approximation theory. This framework initially had three components: the problem space; the algorithm space; and the performance space (Rice 1976). Later in the text, Rice observed the need for a fourth component, the feature space, which was added to the framework. These four components can be summarized as follows:

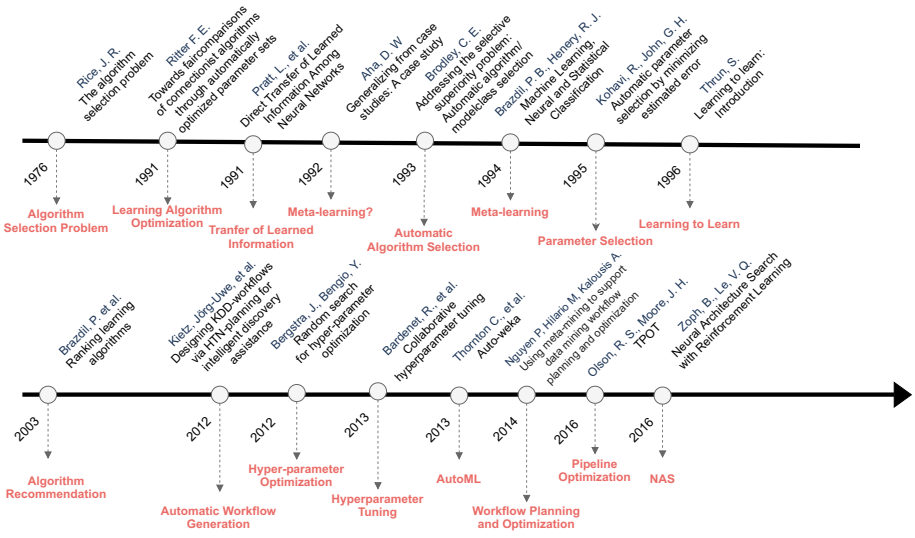

Fig. 3 Timeline with distinct names adopted in different scientific contributions

- *The Problem Space* (D): the set of problems.
- *The Feature/Characteristic Space* (C): the set of features (or characteristics) extracted from the problem d.
- *The Algorithm Space* (A): the set of algorithms that may deal with problem d.
- *The Performance Space* (P): The criterion used to assess the performance of a given algorithm in a specific problem.

According to Smith-Miles (2009), the algorithm selection problem posed by Rice can be formally described by Definition 1 (Smith-Miles 2009). Figure 4 depicts this framework, so-called the abstract model (see blue part).

Definition 1 (The Algorithm Selection Problem) For a given problem instance $d \in D$ with features $f_c(d) \in C$, find the selection mapping $f_m(f_c(d)) \in A$ in the algorithm space, such that the selected algorithm $\alpha \in A$ maximizes the performance mapping $f_p(\alpha(d)) \in P$.

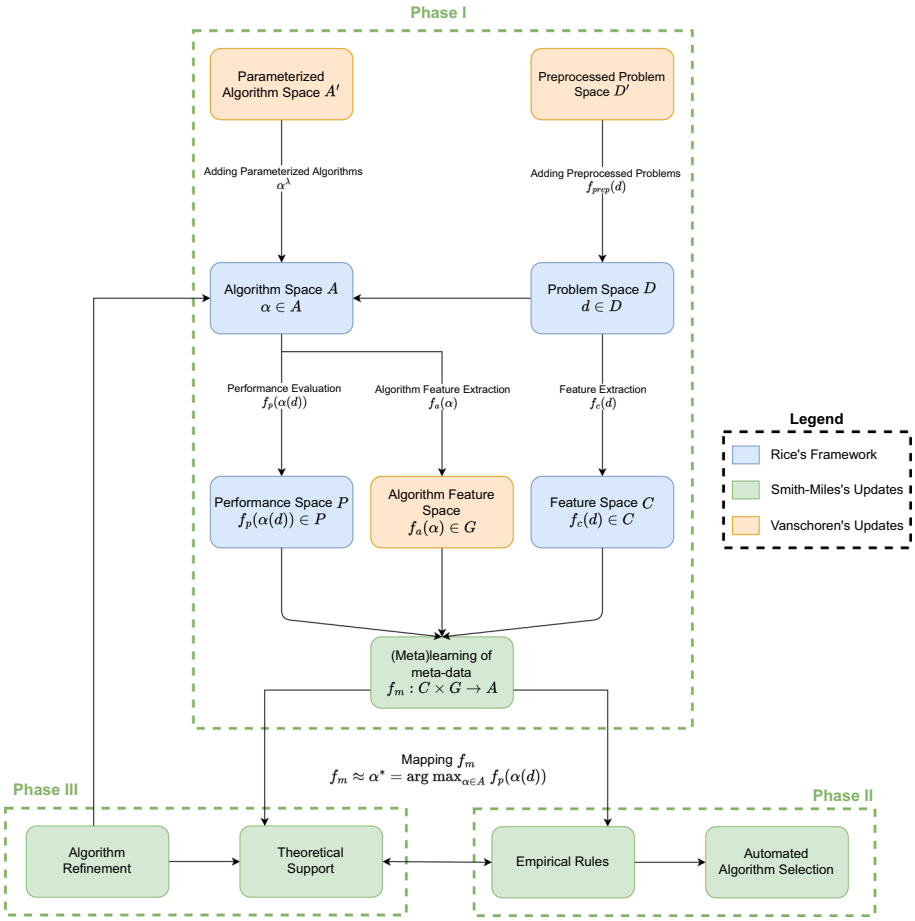

Fig. 4 Rice's framework with updates from Smith-Miles and Vanschoren. Adapted from Vanschoren (2010)

Therefore, the goal of the algorithm selection problem is to identify a mapping function $f_m : C \rightarrow A$ that selects the most appropriate algorithm for a given problem d. This function f_m depends on the problem characterization, which, in turn, is influenced by the domain. To define f_m, we need to provide features $f_c(d) \in C$, and designing these features depends on the application domain $d \in D' \subset D$. For example, $f_c(d)$ can be viewed as meta-features tailored for a specific class of problems $D' \subset D$. Developing suitable meta-features for a given problem is challenging and typically requires expert knowledge. As noted by Smith-Miles (2009), this difficulty has likely hindered the widespread adoption of Rice's framework (Smith-Miles 2009).

Subsequently, Rice's framework underwent some updates. Smith-Miles (2009) proposed three phases during the algorithm recommendation process, highlighted in green in Fig. 4. Phase I represents the meta-data acquisition to create the mapping function f_m. Phase II focuses on learning from the acquired meta-data and generated rules to create an automated method to select the algorithms (the mapping function f_m). Finally, Phase III aims to examine the rules and empirical results from a theoretical perspective view and generate insights to improve algorithms and even create new ones. After that, Vanschoren (2010) added three essential steps in the framework, which are depicted in orange in Fig. 4. Firstly, the *Preprocessed Problems Space* (preprocessed dataset) to include the transformation that occurs in the data mining process (e.g., feature selection and feature construction) or the adaptation of the problems to the algorithm input format (e.g., one-hot encoding in categorical features). Moreover, the *Parameterized Algorithm Space* permits the inclusion of algorithm hyperparameters. Finally, the *Algorithm Feature Space* makes features extraction possible from algorithms (e.g., type of data that algorithms can handle, interpretability of the learned model, runtime properties, resilience to noise, and resilience to redundant attributes).

According to Smith-Miles (2009), Aha (1992) was the first work to investigate the use of datasets' characterization to select algorithms Aha 1992; Smith-Miles 2009. The work was also the first to use the term *metalearning* (MtL), although it did not provide a definition for it. Aha (1992) posited that while many studies hypothesized that some algorithms performed better than others because they were better suited to specific classes of problems, there was a lack of experimental evidence to substantiate this hypothesis. Specifically, there was no clarity on the conditions and characteristics under which one algorithm outperforms another. To address this, Aha (1992) proposed an empirical method to generate rules describing when some algorithm outperforms others based on characteristics extracted from datasets, such as the number of classes, number of instances, and feature value range. Figure 5A shows the resulting rules produced to explain when it was better to use Instance-Based-1 (IB1), CN2, and the Quilan decision tree (C4) ML algorithm. Although Aha (1992) used dataset characteristics to select algorithms, they were previously used to investigate the concept learning of ML algorithms (Rendell and Cho 1990).

Brodley (1993) observed that while a given algorithm might be well-suited for certain tasks, it may not be optimal for all possible tasks, a phenomenon referred to by the author as the dilemma of 'selective superiority of algorithms'. This led Brodley (1993) to hypothesize that merging different biases could produce an algorithm ideal for any problem. In response, Brodley (1993) designed an algorithm that builds a hybrid classifier dynamically using a set of heuristics rules called Model Class Selection (MCS) (Brodley 1993). MCS used three different classes of algorithms: univariate decision trees, linear discriminant functions, and instance-based classifiers. Figure 5C depicts the bias difference of the algorithms used in

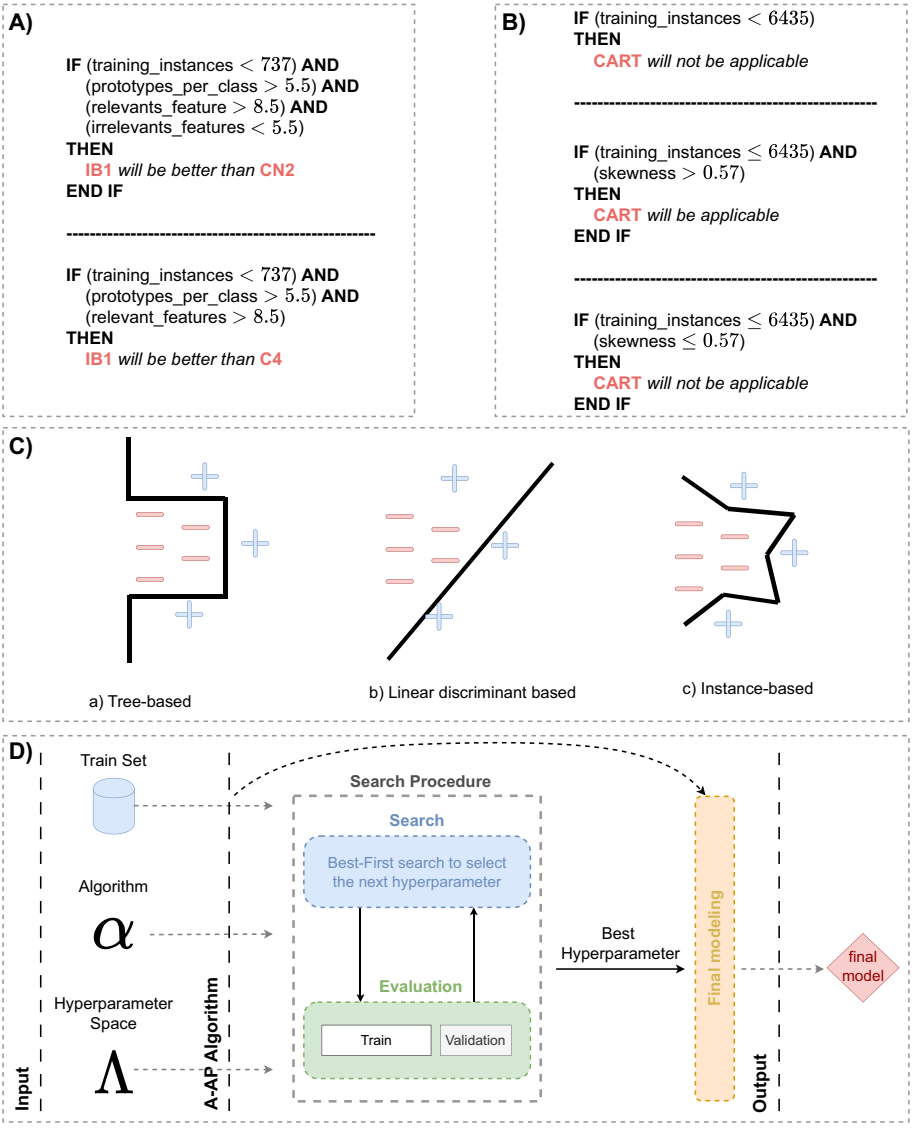

Fig. 5 **a** The rules induced to explain when IB1 will be better; **b** the rules induced for the recommendation of CART; **c** the bias difference visualized by the decision boundary; **d** the A-AP wrapper method. Adapted from Aha (1992), Brodley (1993) and Kohavi and John (1995)

MCS. Moreover, MCS used heuristics to guide the best-first search to combine models of these three different bias classes. The heuristic rules used in MCS include dataset characteristics such as the ones proposed by Aha (1992). Although MCS outperforms all the three different classes of algorithms investigated, it was not the best for all datasets used, showing that it also suffers from the selective superiority dilemma.

Instead of selecting the best algorithm, Pratt et al. (1991) went through on transfer learned information among neural networks. Inspired by symbolic representation, Pratt et al. (1991)

proposed to reuse information from a neural to help a second one learn a related task. Pratt et al. (1991) concluded that transferring weights from smaller networks trained on related tasks could speed up training by one order of magnitude compared to training a new network from scratch with random weights (Pratt et al. 1991).

Additionally, Ritter (1991) noticed that not only algorithms should be adequately selected, but also their hyperparameters (Ritter 1991). Ritter (1991) argued that the researchers were not achieving the optimal performance of the algorithms because many algorithms used standard values for the free hyperparameters or were manually tuned. According to Ritter (1991), default hyperparameters typically do not yield the best model performance, and manual tuning is both labor-intensive and prone to errors due to cognitive limitations, such as functional fixedness, limited memory, and human errors (Adamson 1952; Ritter 1991). To address this issue, Ritter (1991) proposed a method based on genetic algorithms (Holland 1992), which automatically selects the optimal hyperparameter settings (Ritter 1991). Chromosomes were sets of hyperparameter values of a neural network such as (learning rate, training regime, and learning grain size), while the genetic operators used were cross-over and mutation. The method outperformed when compared with specialist tuning by hand and default values provided by the algorithm developer.

Similar to Ritter (1991), Kohavi and John (1995) observed that hyperparameters should be tuned to achieve good performance (Kohavi and John 1995). Kohavi and John (1995) argued that although algorithms typically give rules of thumb for tuning their hyperparameters, it would be adequate to find values that work well for the particular dataset under analysis. To address this, Kohavi and John (1995) proposed the A-AP method, which worked as a wrapper that encapsulated a given model and received a dataset as input (Kohavi and John 1995). A-AP tuned the algorithm hyperparameters delivering the final best model by using cross-validation.

In contrast to the proposal put forward by Ritter (1991), which optimized learning proprieties (e.g., the number of epochs needed for the network to learn), the A-AP optimized the algorithm performance (e.g., accuracy). Figure 5D illustrates the A-AP method, which consists of two main components: search and evaluation. The search component employs Best-First Search to identify optimal hyperparameter values. The evaluation component utilizes cross-validation and accuracy metrics to assess the performance of models generated by different hyperparameter configurations. Additionally, a notable contribution of this approach was the formalization of hyperparameter optimization, originally termed the "parameter selection problem" (Kohavi and John 1995).

Many other projects were also proposed in the 1990s that directly or indirectly support AutoML, including the design of new ML algorithms, new benchmarking datasets, development of ML toolboxes to help the end-user, and methods for ML algorithm recommendation (Graner et al. 1993; Michie et al. 1994; Brazdil et al. 2003).

A key initiative was the Machine Learning Toolbox (MLT)[3] that developed the Consultant-MLT, a knowledge-based system (Craw et al. 1992) used to recommend ML algorithms from the MLT toolbox. The system collected a set of user's answers about the dataset, application domain, and preferences. The answers were used as input of predefined rules to recommend an algorithm (Brazdil et al. 2009). Next came Consultant-MTL-2 (Sleeman et al. 1995), an update of Consultant-MTL, which included recommending preprocessing steps. Thus, the objective of Consultant-MTL was to make machine learning accessible to non-

[3] ESPRIT project 2154, from 1989 to 1993 – https://cordis.europa.eu/project/id/1698

experts through algorithm recommendation, positioning it as one of the earliest examples of AutoML tools (Graner et al. 1993). Similarly, Consultant-MTL-2 represents one of the pioneering efforts to recommend entire machine learning pipelines.

Moreover, Comparative Testing of Statistical and Logical Learning (StatLog)[4] brought significant insights for ML (Henery and Taylor 1992; Michie et al. 1994). According to Michie et al. (1994), the objectives of the StatLog project were to provide critical performance measures of classification algorithms and indicate the imperfections to further improvements on developing new algorithms (Michie et al. 1994).

In this project, Brazdil and Henery proposed an experiment to create rules to recommend when to use an ML algorithm, extending the idea of Aha (1992) (Michie et al. 1994). The C4.5 decision tree algorithm was used to generate rules taking as predictive features dataset characteristics extracted from 22 datasets from the University of California - Irvine (UCI) repository. The dataset characteristics, so-called meta-features, were aggregated into three groups: general (e.g., the number of features, number of classes), statistical (e.g., mean of the skewness of the features, mean of the kurtosis of the features), and information-theoretic (e.g., the entropy of the classes, mean of entropy of each attribute). Figure 5B shows an example of rules generated when recommending the Classification And Regression Trees (CART) (Breiman et al. 1984) algorithm. The results of this work encouraged the starting of the field we know as metalearning (Henery and Taylor 1992; Brazdil et al. 2009).

The successes of Statlog inspired a successor project entitled METAL.[5] The goal of METAL was to create a metalearning assistant to provide user support during data mining tasks. Instead of using binary rules, METAL created methods for ranking ML algorithms according to their adequacy for given tasks based on dataset characteristics too (Brazdil et al. 2003). To illustrate this procedure, given an unseen dataset, a rank of the most appropriate algorithm was created, considering accuracy or even the computational time spent. The ranking approach was further incorporated into the Data Mining Advisor (DMA), a web-based tool designed to help practitioners automatically select ML algorithms for classification problems (Giraud-Carrier 2005).

The discussions in this section highlight the extensive and fruitful history of AutoML. The works reviewed offer early insights into concepts that are now central to modern research, such as metalearning (Aha 1992; Brazdil et al. 2003, 2009; Vanschoren 2018), meta-heuristics (Brodley 1993; Pappa et al. 2014), transfer learning (Pratt et al. 1991; Pan and Yang 2009), and optimization in machine learning (Ritter 1991; Kohavi and John 1995; Bergstra et al. 2011). These foundational studies laid the groundwork for subsequent research, driving advancements over the past three decades and paving the way for the evolution of contemporary AutoML.

4 Classical definitions

This section introduces the classical definitions of metalearning, hyperparameter optimization, and transfer learning.

[4] ESPRIT project 5170, from 1990 to 1993 – https://cordis.europa.eu/project/id/5170

[5] ESPRIT project 26.357, from 1998 to 2001 – https://cordis.europa.eu/project/id/FP4_26357

4.1 Classical metalearning

The term metalearning (MtL) was coined based on a branch of the metacognition field that studies how humans learn their learning process (Biggs 1985). Generally, when learning a new task, such as walking on two legs or some academic skill, we rarely start from scratch. Instead, we reuse approaches that previously worked well to guide us in the new task, avoiding spending time on trial-and-error, making the learning faster at each previous experience acquired (Lake et al. 2017). Thus, we are learning how to learn from previous tasks (Vanschoren 2018).

From the ML perspective, MtL will automatically speed up and improve new ML tasks (such as building a model) based on accumulated experience from previous tasks. This accumulated experience is called meta-knowledge, which can be acquired from different kinds of meta-data, such as measures of task's characteristics (so-called meta-features), algorithm hyperparameter settings, model learned parameters, model performance, pipeline structure, and network architecture (Brazdil et al. 2009; Vanschoren 2018). Therefore, MtL investigates how to learn the learning process using meta-knowledge to improve new learning tasks. One of the most known definitions was provided by Brazdil et al. (2009):

Definition 2 (Metalearning) "Metalearning is the study of principled methods that exploit meta-knowledge to obtain efficient models and solutions by adapting machine learning and data mining processes."

By reviewing various definitions of metalearning in the literature, Lemke et al. (2015) proposed two essential criteria for defining a metalearning system. The first criterion is that the learning subsystem should enhance its performance by adapting based on experience. The second criterion is that the system should accumulate and utilize meta-knowledge from its experiences to gain further insights. Based on Lemke et al. (2015) criteria, the definition of a metalearning system is as follows:

Definition 3 (Metalearning System) A metalearning system must:

1. Include a learning subsystem, which adapts with experience.
2. Gain experience by exploiting meta-knowledge extracted

   (a) in a previous learning episode on a single dataset, and/or
   (b) from different domains or problems.

In the classical approach, a MtL system acquires meta-knowledge from meta-features and performance measures from previous ML learning tasks (Brazdil et al. 2009). Meta-features are measures that capture the main characteristics of a dataset and its relations with the learning task. Meta-features and performance are combined to create meta-data as predictive and target attributes. Thus, a meta-dataset is created by accumulating meta-data from previous tasks meta-data.

The next step involves creating a meta-model, a process analogous to data mining where the meta-dataset undergoes preprocessing, modeling, and post-processing to develop a

model that maps meta-features to the target. For example, this meta-model might recommend or rank machine learning algorithms, hyperparameters, or preprocessing methods.

Meta-model design can be categorized into: the base-level and the meta-level. The base-level pertains to tasks directly related to the data itself, such as dataset handling, model induction, and evaluation. In contrast, the meta-level involves using meta-data to design the meta-model. The collection of datasets, induction of models and model evaluation are part of a base-level task, while taking the performances and meta-feature to induce a model that predicts the performance on a new unseen dataset is a meta-level task (Vanschoren 2018). Therefore, when a new task is introduced, its meta-features are extracted and fed into the meta-model, which then predicts the appropriate ML algorithms for the task. Figure 6 illustrates a typical MtL system.

Classical MtL application is generally one in which the meta-data is acquired from meta-features, and meta-knowledge is accumulated in the meta-model. Some classical application includes recommendation or ranking of ML algorithms (Brazdil et al. 2003; Lemke and Gabrys 2010), recommendation of hyperparameters (Soares et al. 2004), estimation of algorithm performance (Guerra et al. 2008; Reif et al. 2011, 2014), warm-start for optimization procedures (Reif et al. 2012; Gomes et al. 2012). We will describe more MtL applications in Sect. 8.

4.2 Classical hyperparameter optimization

Unlike MtL, classical hyperparameter optimization (HPO) aims to fit a suitable model for a given task instead of recommending algorithms (Bergstra et al. 2011; Yang and Shami 2020). This idea comes from the observation that ML algorithms usually underperformed when their hyperparameters were not tuned (Ritter 1991; Kohavi and John 1995). Most approaches to solve HPO problems come from the optimization field. Boyd and Vanden-

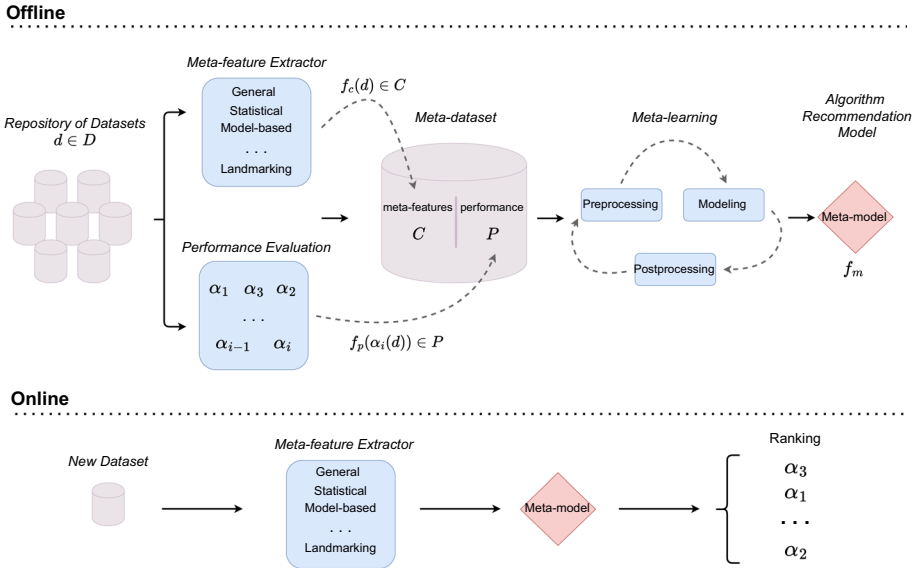

Fig. 6 MtL system for algorithm recommendation. Adapted from Brazdil et al. (2009)

berghe (2004) define an optimization problem (or mathematical optimization problem) as presented in Definition 4.

Definition 4 (Optimization Problem) Let the vector $\mathrm{x} = (x_1, \ldots, x_n)$ be the *optimization variable* of the problem, let the function $f_{\mathrm{obj}} : \mathbb{R}^n \to \mathbb{R}$ be the *objective function*, let the functions $g_i : \mathbb{R}^n \to \mathbb{R},\ i = 1, \ldots, m$ be the *constraint function* (or inequality) and the constants $b_1, \ldots, b_m$ be the limits (or bounds) for the constraints. An optimization problem can be defined as follows:

$$\arg\min_{\mathrm{x}} \quad f_{\mathrm{obj}}(\mathrm{x}) \tag{1a}$$

$$\text{subject to} \quad g_i(\mathrm{x}) \leq b_i, \qquad i = 1, \ldots, m. \tag{1b}$$

The vector x^* which obtains the minimum value for f_{obj} and satisfies the constraints is called optimal. Note that minimize $f_{\mathrm{obj}}(\mathrm{x})$ is equivalent to maximize $-f_{\mathrm{obj}}(\mathrm{x})$. The optimization problem is called *linear programming* (or linear problem) if the objective and constraint functions $f_{\mathrm{obj}}, g_1, \ldots, g_m$ are linear, otherwise it is called *nonlinear programming* (or nonlinear problem). Nonlinear problems typically assume that the objective function f_{obj} is convex or at least mathematically defined.

However, there are problems where the objective function is expensive or even impossible to compute, and the derivatives and convexity properties are unknown (Brochu et al. 2010). HPO is an example of such an optimization problem. Although it is possible to compute the objective function, it is unknown (so-called black-box function), with no guarantee of convexity nor derivatives. Moreover, it can have multiple local minima and maxima, and even noise (Feurer and Hutter 2019).

One of the first formalizations for the HPO problem was defined by Kohavi and John (1995) as follows:

Definition 5 (The Hyperparameter Optimization Problem) Let $D^{\mathrm{train}} \subset D$ be the training subset of the dataset $D = \{(\boldsymbol{x}, \boldsymbol{y}) \mid x_i \in \mathcal{X}, y_i \in \mathcal{Y}, i = 1, ..., n \in \mathbb{N}^+\}$ and $\alpha \in A$ be a machine learning algorithm where $\lambda \in \Lambda$ is the hyperparameter configuration belonging to the space of all possible settings Λ. The algorithm α produces an hypothesis function $h \in \mathcal{H}$, which approximates the real function *f*. Given a function $\mathcal{E}$ that computes the error of *h* when estimating *f*, where $h = \alpha(D^{\mathrm{train}}, \lambda)$, the optimal hyperparameter setting satisfies the following equation:

$$\lambda^* = \arg\min_{\lambda \in \Lambda} \ \mathcal{E}(\alpha(D_{\mathrm{train}}, \lambda), f) \tag{2}$$

The λ^* is the element in Λ that results in a minimal error. However, it is not possible to calculate λ^* because we do not know *f*, which is required to calculate $\mathcal{E}$. Thus, we should somehow estimate $\mathcal{E}$ using D^{train}. According to Kohavi and John (1995), an alternative can be a cross-validation or holdout procedure. The following definition uses k-fold cross-validation to estimate $\mathcal{E}$ and consequently λ^*.

Definition 6 (The Estimated Hyperparameter Setting) Let the k-fold cross-validation inner training set be $d_i^{\mathrm{t}} \subset D^{\mathrm{train}}$ and validation set be $d_i^{\mathrm{v}} \subset D^{\mathrm{train}}$, where $d_i^{\mathrm{t}} \cap d_i^{\mathrm{v}} = \emptyset$, $d_i^{\mathrm{t}} \cup d_i^{\mathrm{v}} = D^{train}$ and $i \leq k \mid i, k \in \mathbb{N}^+$. Given a loss function $\mathcal{L}$ we can approximate $\mathcal{E} \approx \hat{\mathcal{E}}^{\mathrm{cv}}$ as follows:

$$\mathcal{E} \approx \hat{\mathcal{E}}^{\mathrm{cv}} = \frac{1}{k} \sum_{i=1}^{k} \mathcal{L}(\alpha(d_i^{\mathrm{t}}, \lambda), d_i^{\mathrm{v}}) \tag{3}$$

Then, $\hat{\mathcal{E}}^{\mathrm{cv}}$ can be used to estimate λ^* as described below:

$$\lambda^* \approx \hat{\lambda}^* = \underset{\lambda \in \Lambda}{\arg\min} \ \frac{1}{k} \sum_{i=1}^{k} \mathcal{L}(\alpha(d_i^{\mathrm{t}}, \lambda), d_i^{\mathrm{v}}) \tag{4}$$

It is important to note that even if we know the f, the exhaustive search into Λ is impractical. This is because Λ may be prohibitively large for most machine learning algorithms.

More recently, the algorithm selection has been integrated with hyperparameter tuning, a process known as Combined Algorithm Selection and Hyperparameter Optimization (CASH). This development brings CASH closer to the concepts originally discussed by Rice and aligns it with MtL as both approaches involve selecting among various algorithm biases. To the best of our knowledge, the term AutoML was first introduced into the context of the CASH problem (Thornton et al. 2013). CASH can be defined as follows:

Definition 7 (Combined Algorithm Selection and Hyperparameter Optimization) Let $A = \{\alpha_1, \cdots, \alpha_n\}$ be a set of algorithms and $\{\Lambda_1, \cdots, \Lambda_n\}$ the respective associated algorithm hyperspaces. Given a loss function $\mathcal{L}(\alpha_j(d_i^t, \lambda), d_i^v)$, where $\lambda \in \Lambda_j$, the k-fold cross-validation inner training set $d_i^{\mathrm{t}} \in D^{\mathrm{train}}$ and validation set as $d_i^{\mathrm{v}} \in D^{\mathrm{train}}$ with $d_i^{\mathrm{t}} \cap d_i^{\mathrm{v}} = \emptyset$, $d_i^{\mathrm{t}} \cup d_i^{\mathrm{v}} = D^{train}$ and $i \leq k \mid i, j, k \in \mathbb{N}^+$, we can define CASH as:

$$\alpha^* \lambda^* \in \underset{\alpha_j \in A, \lambda \in \Lambda_j}{\arg\min} \ \frac{1}{k} \sum_{i=1}^{k} \mathcal{L}(\alpha_j(d_i^t, \lambda), d_i^v). \tag{5}$$

Several methods are used for hyperparameter optimization, including random search, grid search, Bayesian optimization, evolutionary algorithms, and swarm intelligence techniques (Bergstra et al. 2011; Pappa et al. 2014; Feurer and Hutter 2019; Yang and Shami 2020). Nevertheless, effectively addressing complex nonlinear problems remains a significant challenge (Boyd and Vandenberghe 2004). According to Boyd and Vandenberghe (2004), even relatively simple problems can be computationally intensive, difficult to solve, or intractable. A more detailed discussion of HPO algorithms and their applications will be provided in Sect. 8.

4.3 Classical transfer learning

The classical literature on transfer learning (TL) defines itself in terms of domain and task. The domain is seen in light of feature space and marginal distributions, while the task in light of label space and the decision function. Definitions 8 and 9 depict these definitions adapted from Pan and Yang (2009) and Zhuang et al. (2021).

Definition 8 (Domain)

Let $\mathcal{X}$ denote the feature space and $P(X)$ be the marginal probability distribution over $X \subset \mathcal{X}$. A domain is defined as:

$$\mathcal{D} = \{\mathcal{X}, P(X)\} \tag{6}$$

Definition 9 (Task) Let $\mathcal{Y}$ be a label space and let $h : \mathcal{X} \rightarrow \mathcal{Y}$ be a predictive function. A task is defined as:

$$\mathcal{T} = \{\mathcal{Y}, h\} \tag{7}$$

Definition 10 (Transfer Learning) Let $S = \{\left(\mathcal{D}_i^S, \mathcal{T}_i^S\right) \mid i = 1, ..., n \in \mathbb{N}^+\}$ be a set of source-domain pairs and let $T = \{\left(\mathcal{D}_i^T, \mathcal{T}_i^T\right) \mid i = 1, ..., m \in \mathbb{N}^+\}$ be a set of target domain-task pairs. Transfer learning is defined as the process of using knowledge acquired from source set (meta-knowledge) S to improve the performance of the predictive function h^T in the target domains T.

$$\hat{h}^T = \tau(h^T, S) \tag{8}$$

Note that by combining a source domain $\mathcal{D}_i^S$ with its associated task $\mathcal{T}_i^S$, we obtain a dataset $D = \{(\boldsymbol{x}, \boldsymbol{y}) \mid x_k \in \mathcal{X}_i^S, y_k \in \mathcal{Y}_i^S, k = 1, ..., n \in \mathbb{N}^+\}$ as previously defined. Similarly to metalearning definitions, we have the reuse of a previus knowledge, what we are calling meta-knowledge, being used in new learning tasks. Unlike hyperparameter optimization, which typically reuses meta-knowledge within the same task setting to guide the optimization, transfer learning leverages experience from related but potentially different tasks or domains to guide the learning process in a target task.

According to Pan and Yang (2009), transfer learning can be classified into three main categories: *inductive*, *transductive*, and *unsupervised*. Inductive TL is characterized by a difference between the source and target tasks, i.e., $\mathcal{T}^S \neq \mathcal{T}^T$. In this environment, the target domain has some labeled data, which enables the induction of a predictive model for the target task. A common application scenario includes fine-tuning a neural network pre-trained on a large dataset (e.g., ImageNet) for a different but related task with limited labeled data. In transductive TL, both the source and target tasks are the same (i.e., $\mathcal{T}^S = \mathcal{T}^T$), but the domains are different $\mathcal{D}^S \neq \mathcal{D}^T$. Moreover, the target domain typically lacks sufficient labeled data, while the source domain provides abundant labeled examples. This type of transfer is particularly relevant in cross-domain applications, such as sentiment analysis across different product categories or adapting visual recognition systems to variations in

lighting or sensor conditions. Finally, unsupervised TL considers the case where both the source and target tasks are unsupervised so no labeled data are available in either domain, such that $\mathcal{Y}^S = \mathcal{Y}^T = \emptyset$. The objective is to transfer structural knowledge (e.g., feature representations) from the source domain to improve unsupervised learning performance in the target domain.

In computer vision, transfer learning enables tasks like medical image classification and object detection by fine-tuning models pretrained on large datasets (Pan and Yang 2009; Arbane et al. 2023). In natural language processing, models like BERT are adapted for tasks such as sentiment analysis and named entity recognition (Arbane et al. 2023; Zhao et al. 2024). It also helps in areas like robotic learning, genomics, and environmental monitoring, allowing knowledge transfer across domains to improve model performance (Lu et al. 2023). In Sect. 8 we will describe more applications.

5 Meta-data and meta-knowledge

Even though MTL, TL and HPO, in their classical definition, appear to be different, they have more in common nowadays than ever before, mainly due to the emergence of new AutoML solutions. We will talk about this in light of two concepts, meta-data and meta-knowledge. Next, we will define and exemplify these concepts.

Definition 11 (Meta-data) Meta-data is data derived from the base-learning process that provides insights into the learning task.

Definition 12 (Meta-knowledge) Meta-knowledge is the knowledge gained from the process of learning how to learn. It is accumulated from meta-data and encompasses insights and patterns derived from previous learning experiences.

Definition 11 characterizes meta-data as the data produced during the learning process. Figure 7A illustrates various sources and types of meta-data that can be collected. Meta-data can originate from three distinct sources: different tasks; similar tasks; and the same task.

Meta-data from different tasks are exemplified by the data generated by classical ranking systems (Brazdil et al. 2003). Meta-data from similar tasks typically arises in transfer learning or few-shot learning (Oquab et al. 2014; Finn et al. 2017; Sun et al. 2019). Lastly, meta-data from the same task is commonly seen in hyperparameter optimization, where performance metrics and configurations are recorded and evaluated across iterations (Wu et al. 2019). For example, surrogate models used in Bayesian optimization track new configurations and their corresponding performance at each iteration (Brochu et al. 2010)

Furthermore, in Fig. 7A, we suggest several types of meta-data, which we categorized as follows: (i) dataset-based, similar to meta-features and data complexity (Rivolli et al. 2018); (ii) algorithm-based, similar to the hyperparameters stored by optimization process, as well as golden rules given by an algorithm developer (Wu et al. 2019; Yang and Shami 2020); (iii) model-based, similar to internal parameters and architectures which can be shared (Liu et al. 2018; Pham et al. 2018; Elsken et al. 2018; Wan et al. 2020); (iv) workflow-based, similar to combinations of pipelines and precedence rules (Olson et al. 2016; de Sá et al. 2017); (v) task-related measurements, similar to the cross-validation score and prediction

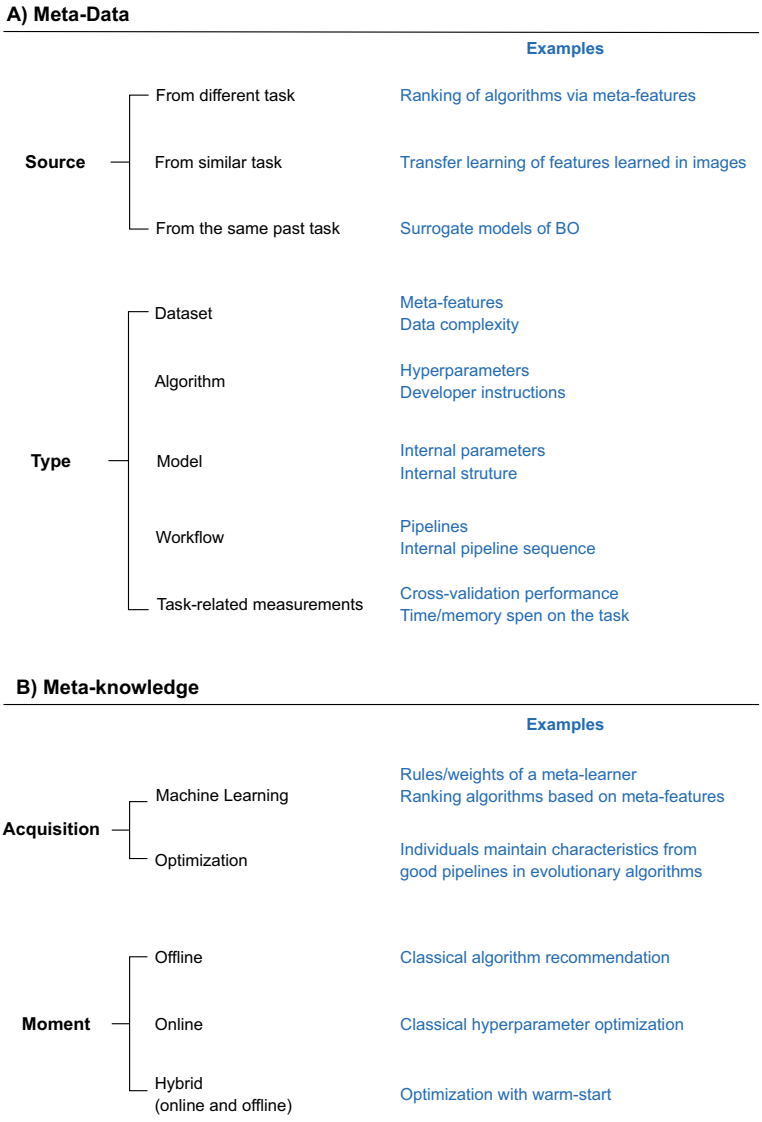

Fig. 7 The types and source of meta-data with examples in *blue* (**A**). The meta-knowledge acquisition type and the moment it occurs with examples in *blue* (**B**)

time. These meta-data can be captured and used to learn the base-learning process, create rules, transfer learning, or even choose the next hyperparameters to be evaluated (Oquab et al. 2014; Feurer et al. 2015; Finn et al. 2017).

In Definition 12, we present meta-knowledge as the knowledge acquired from learning how to learn. Figure 7B shows how the acquisition of meta-knowledge can be made and the moment in which it occurs. The acquisition can be made through (i) machine learning algorithms, which can store meta-knowledge in decision tree rules or even in weights of regressors and neural network layers (Aha 1992; Bensusan and Kalousis 2001; Brazdil et al. 2003;

Guerra et al. 2008; Reif et al. 2014); (ii) optimization procedures, as in the case of evolutionary algorithms in which each iteration keeps the best characteristics of the individuals alive in their descendants (Olson et al. 2016). Moreover, the acquisition can occur in different ways such as (i) offline, where the meta-data are collected before any meta-knowledge is acquired (Reif et al. 2012); (ii) online, where the acquisition of meta-data is made simultaneously with the acquisition of knowledge (Yang and Shami 2020); (iii) hybrid, where meta-knowledge is acquired both online and offline (Gomes et al. 2012; Feurer et al. 2015).

One can observe more similarities than differences when comparing the classical MtL, TL, and HPO approaches in light of the concepts of meta-data and meta-knowledge presented here. For example, in classical MtL approaches, we can obtain metadata from meta-feature extraction in an offline process applied to past tasks (Brazdil et al. 2009). The meta-knowledge is acquired by learning from the metadata through ML algorithms, which build meta-models. In the meta-models parameters/weights, we have the meta-knowledge accumulated that will be used to make a recommendation (Bensusan and Kalousis 2001; Brazdil et al. 2003; Guerra et al. 2008). In the case of TL, meta-knowledge is typically transferred through the model's parameters, which, after the training process, are used to speed up new learning tasks on similar or different tasks (Pan and Yang 2009; Zhuang et al. 2021).

Similarly, classical HPO also obtains meta-data through cross-validation in an online process applied over the same tasks (Bergstra and Bengio 2012). The meta-knowledge is obtained by learning from cross-validation through an optimization procedure. At each step, the best configurations and performance are updated based on the meta-knowledge acquired, i.e., the previously observed best configurations and performance (Kohavi and John 1995; Yang and Shami 2020).

Currently, the combination of classical HPO, TL and MtL approaches has produced new robust approaches (Feurer et al. 2015; Finn et al. 2017; Thornton et al. 2013). Next, we will introduce an AutoML definition based on the similarities of MtL, TL, and HPO through meta-knowledge acquisition. Thus, we will present AutoML as an umbrella term for methods/processes/algorithms/systems that learn to learn.

6 Automated machine learning definition

Nowadays, AutoML has gained significant attention due to its potential impact on both scientific research and industry. First, AutoML can contribute to the democratization of machine learning by providing powerful tools accessible to non-experts. As developing machine learning solutions typically demands substantial prior knowledge, AutoML facilitates research across various fields not traditionally associated with machine learning. Second, AutoML supports experts by automating labor-intensive tasks such as hyperparameter tuning and algorithm selection, thereby accelerating the development of improved products and technologies. Finally, AutoML can evolve into intelligent assistants that can guide non-experts and support professionals. In this envisioned future, AutoML systems would act as data science copilots, analyzing data, building models, and interpreting results, in a new era of artificial intelligence where machines enhance and extend human expertise in data-driven decision-making.

Based on the current literature and perspectives (Feurer et al. 2015; Barbudo et al. 2023; Hutter et al. 2019; Zöller and Huber 2021; Kohavi and John 1995; Pan and Yang 2009; He et al. 2021), we propose a comprehensive definition of AutoML. This aims to clarify the

concept of AutoML and unify various approaches with similar goals to advance research. Therefore, we define AutoML as follows:

Definition 13 (Automated Machine Learning) Automated Machine Learning consists of systems developed in order to design machine learning solutions automatically, learning to learn the learning process, with few human interactions. As objectives, AutoML aims to:

- Democratize machine learning
- Help data scientists in the laborious and repetitive steps of designing machine learning solutions
- Find the best possible machine learning solution given limited resources and restrictions
- Learn the learning process, accumulating meta-knowledge and being more efficient at each step

We consider a few human actions instead of none because we see AutoML as a tool that does not replace data scientists, but assists them. Data scientists can, e.g., define algorithm restrictions, pre-trained models, or limit available resources, such as time and memory. Therefore, some interactions are needed at some level.

The term “automatically" can be read as systems that, on some level, learn to learn from meta-data, accumulating meta-knowledge and being more efficient at each step. Therefore, MtL, HPO, and TL fit this definition well.

7 Literature review categorization

In this section, we systematically organize the existing literature reviews, proposing two primary levels: the *Classical Connection* and its corresponding *Subcategories*. The Classical Connection refers to the core traditional machine learning areas that form the conceptual basis for the surveyed reviews (MtL, TL, HPO). Within each classical field, additional subdivisions are introduced to capture more specific methodological emphases.

The first level of categorization identifies three classical machine learning domains associated with the surveyed literature: hyperparameter optimization, metalearning, transfer learning, and hybrid approaches (more than one domain). These categories reflect the historical development and foundational paradigms from which modern AutoML research has emerged.

Each classical domain is further subdivided into more detailed subcategories, as outlined below:

7.1 Metalearning

From Meta-features: Reviews in this subcategory emphasize the extraction of meta-features from datasets to guide the selection or configuration of algorithms.
From Model: This subcategory includes reviews that focus on extracting meta-knowledge directly from model parameters and optimization processes.

7.2 Hyperparameter optimization

Single Step Optimization: Reviews concentrating on the optimization of hyperparameters for a single algorithm without explicitly structuring the optimization across multiple pipeline stages.
Multiple Step Optimization (Pipeline Design): Reviews addressing optimization across multiple stages of a machine learning workflow, such as preprocessing, feature engineering, model selection, and postprocessing.
Multiple Step Optimization (Neural Architecture Search): Reviews focusing specifically on the optimization of deep neural network architectures.

7.3 Transfer learning

Meta-knowledge Transfer: Reviews focusing on the meta-knowledge transfer of models, features, or learned representations from source to target tasks.
Continuous Meta-knowledge Transfer: Reviews dealing with lifelong learning, continual learning, or incremental transfer learning, where meta-knowledge is accumulated and adapted progressively across multiple tasks.

7.4 Hybrid approaches

Hybrid reviews address combinations of two or more classical domains (MtL, TL, HPO).

Figure 8 shows the proposed categorization, while Table 1 in Appendix Appendix A shows the reviews organized according to the categorization with the publication year and author name. The columns *Classical Connection* and *Subcategory* reflect the described structure.

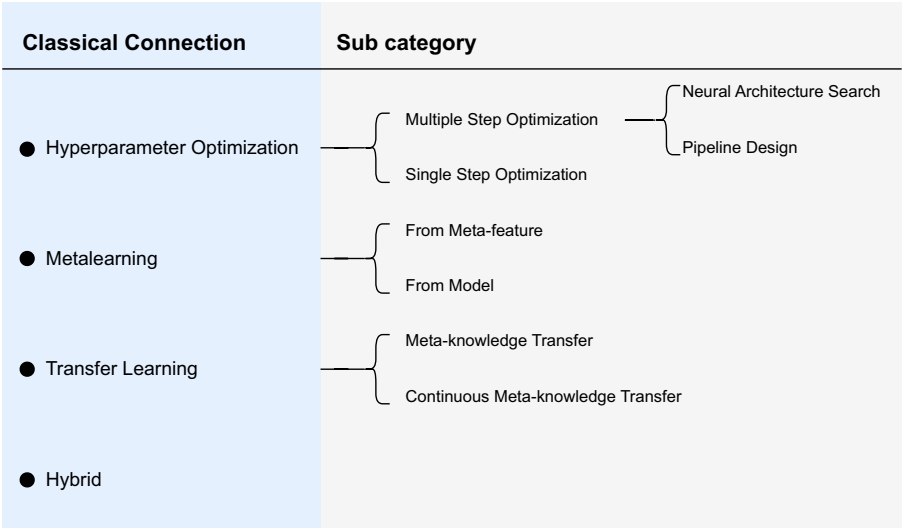

Fig. 8 Categorization of literature reviews according to classical connections

The categorization provides a valuable tool for both new researchers and practitioners seeking to familiarize themselves with the AutoML field, as well as for experienced researchers aiming to systematically position their work within the broader research landscape. By offering a structured overview, this categorization facilitates the identification of consolidated areas, emerging trends, and research gaps for further investigation. Section 8 presents a discussion on current applications of AutoML, while Sect. 9 explores future research directions in the field.

8 AutoML applications

This section explores AutoML applications. It begins with applications related to classical metalearning. Next, the section covers hyperparameter optimization in a single and multiple optimization step view. Finally, we finish with transfer learning applications.

8.1 Metalearning

In this sub-section, we describe applications related to classical metalearning.

8.1.1 From meta features

A way to learn from past experience in MtL is by inducing meta-models using meta-features (Brazdil et al. 2009). The meta-features can be extracted manually or systematically. Systematic meta-feature extraction was first proposed by Pinto et al. (2016) and then updated by Rivolli et al. (2018). It consists of applying pre-defined measures and summarization functions in the datasets used (Pinto et al. 2016; Rivolli et al. 2018; Alcobaça et al. 2020). The following categories of meta-features are commonly used in literature:

- *General:* also known as *simple* (Reif et al. 2014), general meta-features do not require significant computational resources, and they are easily extracted from the dataset (Brazdil et al. 2009). Examples are the number of instances, number of attributes, number of classes and the relative frequency of each distinct class.
- *Statistical:* captures statistical properties from data (Reif et al. 2014). These meta-features are used to characterize only numerical attributes (Brazdil et al. 2009). For example, the attribute correlation and kurtosis, and the geometric mean of each attribute.
- *Information-theoretic:* based on entropy, these measures are from the information theory field. They are appropriate to describe categorical attributes and their relationship with the target attribute (Segrera et al. 2008). Examples of these meta-features are Shannon's entropy of each predictive attribute, the mutual information between each attribute and target, and the concentration coefficient between each attribute.
- *Model-based:* extracts measures from a model induced in the training phase. Examples are the shape, depth, and size of a decision tree (Peng et al. 2002; Reif et al. 2014).
- *Landmarking:* uses the performance of fast and straightforward classifiers. Usually, they have different bias approaches to extract relevant information from data (Pfahringer et al. 2000). Examples are Naive Bayes, Linear Discriminant, 1-Nearest Neighborhood, and single-node trees.
- *Subsampling landmarking:* variants of the landmarking approach, subsampling land-

marking considers complex computational algorithms, evaluating their performance in a subset of the available data (Fürnkranz and Petrak 2001).
- *Relative landmarking:* variants of the landmarking approach, the relative landmarking instead of using the performance, compute the differences of the algorithms in absolute sizes, such as ranking (Soares et al. 2001; Vanschoren 2010).
- *Clustering:* extracts information based on external validation indexes. These meta-features are the number of clusters with a size smaller than a fixed value and the mean silhouette value (Pimentel and de Carvalho 2019).
- *Concept:* estimate the variability of class labels among examples and the example density (Vilalta and Drissi 2002).
- *Itemset:* use itemset approach for compute the correlation between binary attributes (Song et al. 2012).
- *Complexity:* estimate how difficult it is for a learning algorithm to separate the data points into their expected classes (Lorena et al. 2019).

Meta-models based on meta-features can be used for a wide range of tasks, such as recommending ML algorithms, their configurations, ranking the better ML algorithms and/or preprocessing technique, when tuning is necessary, and other tasks related to the data mining process (Smith-Miles 2009; Brazdil et al. 2009; Lemke et al. 2015; Vanschoren 2018). An example would be the induction of regressor meta-models to predict algorithm performance or hyperparameter performance, classifier meta-models to predict which algorithm will induce the model that performs better, and ranking meta-models to rank algorithms according to a performance measure (Brazdil et al. 2003; Soares et al. 2004; Reif et al. 2011, 2012, 2014; van Rijn et al. 2015; Mantovani et al. 2019).

MtL can also be used in a process called meta-mining or process-oriented MtL, in which meta-models are applied to the entire data mining process for pipeline generation (Hilario et al. 2011; Nguyen et al. 2014). In data preprocessing scenarios, meta-models can recommend a specific step, such as feature selection algorithms (Parmezan et al. 2017). More recently, Bilalli et al. (2018) proposed an automated approach for recommending preprocessing steps in which meta-models can suggest transformations to improve performance for a given dataset (Bilalli et al. 2018).

Meta-models can be used for decision-making, helping the user in the data mining process. They can, e.g., recommend tuning or not the hyperparameters of an algorithm (Mantovani et al. 2019), or even to predict the improvement obtained by tuning (Sanders and Giraud-Carrier 2017). In optimization scenarios, MtL can be used for warm-start optimization to predict reasonable initial solutions (Gomes et al. 2012; Wistuba et al. 2015) and a good initial pipeline composition (Feurer et al. 2015).

8.1.2 From model

Metalearning from a model uses meta-knowledge accumulated in the model's parameters, such as the neurons' weights, to learn new tasks, also known as the "learn to learn" paradigm. These methods are typically categorized into three main approaches: (i) metric-based, (ii) model-based, and (iii) optimization-based (Tian et al. 2022; Vettoruzzo et al. 2024; Ma et al. 2022).

In metric-based metalearning, the focus is on learning an embedding space where samples from the same class are close together, and those from different classes are well separated (Vettoruzzo et al. 2024). Similarity functions are then applied within this learned space to perform classification or other tasks. Representative methods in this category include prototypical networks, and siamese networks (Koch et al. 2015; Snell et al. 2017).

Model-based metalearning involves designing models with architectures that facilitate quick adaptation to new tasks by altering internal states or parameters (Vettoruzzo et al. 2024). These models often incorporate memory components or dynamic parameter updates. Examples include meta networks and memory-augmented neural networks (Santoro et al. 2016; Munkhdalai and Yu 2017).

In optimization-based metalearning, the objective is to learn a set of initial parameters-often referred to as meta-parameters-that can be rapidly adapted to new tasks using a small number of gradient descent steps (Vettoruzzo et al. 2024). This approach aims to generalize across tasks by optimizing for adaptability. Prominent methods include Model-Agnostic Meta-Learning (MAML) and META-SGD (Finn et al. 2017; Li et al. 2017).

8.2 Hyperparameter optimization

In this sub-section, we describe applications related to classical hyperparameter optimization.

8.2.1 Single step optimizations

The literature widely recognizes that HPO is essential in developing machine learning models once default hyperparameters lead to underperformance (Bischl et al. 2023; Karl et al. 2023; Kadhim et al. 2022; Parker-Holder et al. 2022; Al-Sahaf et al. 2019; Hutter et al. 2015). The optimization can be done considering only one algorithm (as shown in Definition 6) or over many algorithms simultaneously (as shown in Definition 7) (Kohavi and John 1995; Thornton et al. 2013; Yang and Shami 2020).

There is a wide range of methods for optimizing hyperparameters. The most common are model-free, such as random search and grid search (Witt 2005; Bergstra and Bengio 2012). The random search takes configurations randomly at each iteration, while grid search takes them from a user pre-defined grid. Although they are usually expensive, both can be fully parallelized (Bergstra and Bengio 2012).

Among the population-based methods, one can mention the evolutionary algorithms and methods based on swarm intelligence (Banzhaf et al. 1998; Gogna and Tayal 2013; Pappa et al. 2014). These methods maintain a population of hyperparameter configurations, trying to improve them at each iteration. These methods can benefit from parallelism, handling the population in different cores. Genetic algorithms and genetic programming are examples of metaheuristics, evolutionary algorithms, which create and improve a population of configuration candidates at each iteration, applying local perturbations (mutations) and combining different members (crossover) (Lessmann et al. 2005; Olson et al. 2016). Ant colony optimization and particle swarm optimization are examples of swarm intelligence methods that maintain and improve a population of configurations (agents) by interactions between agents and the environment (Shi and Eberhart 1998; Lorenzo et al. 2017; Cheng et al. 2018).

Unlike model-free methods, model-based ones maintain and update a model that takes advantage of previous iterations' configurations to predict the next best configuration (Bro-

chu et al. 2010; Snoek et al. 2012). An example is the Bayesian optimization that iteratively fits a surrogate model to predict the objective function with the configurations observed so far and then uses an acquisition function to determine the next configuration to deal with exploration and exploitation trading off (Eggensperger et al. 2013; Swersky et al. 2013; Wu et al. 2019).

Finally, one can also consider more than one objective when optimizing an algorithm. For example, it is desirable to embed resource limitation or multiple loss functions in the objective function, especially in real scenarios (Biswas et al. 2016; Horn and Bischl 2016; Cheng et al. 2018). Overall, the Pareto front is used to handle these multi-objective scenarios (Shah and Ghahramani 2016).

8.2.2 Neural architecture search

Neural Architecture Search (NAS) is a topic of AutoML that investigates methods for designing deep neural network architectures (Kang et al. 2023; Talbi 2022; Zhang et al. 2021; He et al. 2021; Jaafra et al. 2019). NAS methods usually work in a very high dimensional hyperparameter space due to the nature of the deep neural networks. The most common hyperparameters are the maximum number of layers, the type of operation of each layer (e.g., pooling, convolution, dilated convolutions), the conditional hyperparameters associated with the layer type (e.g., number of neurons, number of filters, kernel size, and strides) (Elsken et al. 2019).

According to Wistuba et al. (2019) and Elsken et al. (2019), two types of neural architecture search space can be found in the literature: global search space (e.g., single-branch and multi-branch) and cell-based (Elsken et al. 2019; Wistuba et al. 2019).

Single-branch space (so-called chain-structured space) considers only sequences of n layers $L_1, ..., L_n$, side by side, forming a chain where the output of L_i is the input of L_{i+1} (Baker et al. 2016). Multi-branch, in turn, increases the degree of freedom allowing more complex scenarios where it is possible to connect different layers and even skip connections. Thus, the input of layers L_i is $L_i^{\text{out}} = agg_i(L_1^{\text{out}}, \ldots, L_n^{\text{out}})$, where agg_i is an aggregation function that combines the outputs for layer i (Xie and Yuille 2017; Cai et al. 2018). This kind of search space can, for example represent Dense Nets, when considering agg_i as a concatenation function $L_1^{\text{out}} \frown L_2^{\text{out}} \frown \ldots \frown L_n^{\text{out}}$, and Residual Networks when agg_i is a sum of $L_{i-1} + L_j, j < i - 1$. Moreover, one can see single-branch spaces as a subcase of multi-branch by making agg_i as L_{i-1}^{out}. Thus, multi-branch search space has significantly more degrees of freedom than single-branch space (Wistuba et al. 2019).

Lastly, cell-based search space links different types of cells (or motifs), i.e., a group of well-arranged layers instead of single layers. It is motivated by the fact that hand-crafted architectures tend to present repetitive arrangements of layers (Zoph et al. 2018). Therefore, the different types of cells, their quantities and connections become hyperparameters to be tuned. If we consider a cell as a single layer, thus we can also see this search space within single-branch and multi-branch, increasing the possibilities (Wistuba et al. 2019).

With some adaptations, some of the methods used for hyperparameter optimizations can be applied to NAS. Random Search, Evolutionary Algorithms, and Bayesian Optimization (Xie and Yuille 2017; Cai et al. 2018; Elsken et al. 2019; Wan et al. 2020). Reinforcement learning and one-shot architecture search are also alternatives (Kang et al. 2023; Talbi 2022). In reinforcement learning approaches, an agent (i.e., search algorithm) learns iteratively to improve their behavior

(i.e., performance) by interacting with the environment (i.e., search space). When the agent acts in the environment, it generates some modifications in the architecture, which can be positively or negatively rewarded based on performance (Zoph and Le 2016; Baker et al. 2016; Zoph et al. 2018; Zhong et al. 2020).

Recent approaches have combined multi-objective optimizations such as performance and architecture size, fuzzy architecture components, linear genetic programming to encode multi-branch, and particle swarm-based lightweight neural architecture search (Lu et al. 2023; Gong and Ma 2024; Li et al. 2025; Stapleton et al. 2025)

8.2.3 Automated pipeline design

Automated Pipeline Design (AutoPD) investigates methods to design end-to-end machine learning solutions (Barbudo et al. 2023; Meisenbacher et al. 2022; Karmaker et al. 2022; Wever et al. 2021; Zöller and Huber 2021; Nagarajah and Poravi 2019). Unlike HPO, AutoPD addresses creating complete machine learning pipelines that involve choosing and tuning modeling and preprocessing algorithms (Zöller and Huber 2021). The number of hyperparameters and possibilities is also large in this scenario. For example, it is necessary to choose which algorithms to use, their respective hyperparameters, and their order of application (Hutter et al. 2019).

The search spaces in AutoPD describe how the pipelines will be generated. It can be in a single-branch or a multiple-branch, such as a directed acyclic graph (Feurer et al. 2015; Olson et al. 2016; das Dôres et al. 2018). With some adaptations, HPO and MtL methods have been used for AutoPD. RandomSearch, Evolutionary Algorithms, Multi-Armed Bandit, and Bayesian Optimization are examples (Hutter et al. 2019). Among the systems that implement these methods, we cite Auto-Weka, Auto-Sklearn, TPOT, and AutoBand (Thornton et al. 2013; Feurer et al. 2015; Olson et al. 2016; das Dôres et al. 2018).

Auto-Weka is based on Bayesian optimization using Sequential Model-based Algorithm Configuration (SMAC) and Tree-structured Parzen Estimator (TPE) (Thornton et al. 2013). It explores a hierarchical hyperparameter space and includes mechanisms to handle algorithm constraints and accelerate evaluation. SMAC outperforms grid search and default configurations, but both Bayesian methods demand tuning of meta-hyperparameters and are computationally expensive due to their stochastic nature. Hyperopt-Sklearn also employs Bayesian optimization (via TPE) on Scikit-learn components (Komer et al. 2014). Auto-Sklearn enhances Bayesian optimization (via SMAC) with metalearning and ensemble learning. It initializes optimization with prior knowledge from similar datasets and builds ensembles from top-performing pipelines. It reduces search space complexity but faces some level of overfitting (Fabris and Freitas 2019. TPOT, in contrast, uses genetic programming to evolve non-fixed-length pipelines (Olson et al. 2016). This enables multi-step preprocessing and complex model stacking. However, it can be prone to overfitting and high computational cost (Olson et al. 2016).

Other systems include RECIPE, a grammar-based genetic programming method that avoids invalid pipelines and alternates validation sets to reduce overfitting, though it suffers from low population diversity (de Sá et al. 2017). ML-Plan uses hierarchical task networks with MCTS to build pipelines via heuristic search and incorporates mechanisms to mitigate overfitting (Mohr et al. 2018). Auto-Band, based on the hyperband MAB strategy, demonstrates competitive performance but is limited by a fixed pipeline structure and reduced preprocessing diversity (das Dôres et al. 2018).

More recently, advances have been made in pipeline design for tasks beyond classification, including regression, graph-based learning, time series forecasting, multi-label classification, data cleaning, and clustering (Barbudo et al. 2023; Karmaker et al. 2022; Meisenbacher et al. 2022).

8.3 Transfer learning

In this sub-section, we describe applications related to classical transfer learning.

8.3.1 Meta-knowledge transfer

Fine-tuning is one of the most widely used transfer learning strategies (Zhao et al. 2024; Iman et al. 2023). In this approach, a model is initially trained on a large source dataset and then adapted to a specific target task by updating the weights using task-specific data (Iman et al. 2023). Fine-tuning typically involves freezing initial layers, responsible for capturing general representations, and retraining final layers that are more task-specific (Zhuang et al. 2021). This technique has been instrumental in natural language processing (e.g., fine-tuning BERT or GPT for text classification or question answering), computer vision (e.g., adapting ResNet to medical imaging tasks), and audio processing (e.g., fine-tuning wav2vec for accent-specific speech recognition) (Souza et al. 2020; Baevski et al. 2020; Pepino et al. 2021). In addition, the effectiveness of fine-tuning depends on the degree of similarity between source and target domains and the volume of available target data (Zhuang et al. 2021). In the case of low volume, using feature embedding representation or few-shot learning approaches can be preferable (Iman et al. 2023).

Feature-based transfer learning involves using pre-trained models as fixed feature extractors (Zhuang et al. 2021). The model's internal representations are learned from the source, typically from intermediate or last layers, and are used to generate embeddings for the input data. These embeddings can then be used as input for classifiers, regressors, and clustering algorithms. This method allows the efficient reuse of large models without updating their weights, making it particularly useful in resource-constrained environments (Iman et al. 2023; Zhuang et al. 2021). In computer vision, convolutional neural networks such as VGG and EfficientNet are frequently used to generate image embeddings. In neural language processing, sentence or word embeddings generated from BERT or LLM are used for semantic search, clustering, and similarity analysis (Zhao et al. 2024; Iman et al. 2023).

Zero-shot and few-shot learning aim to generalize learned knowledge to tasks with minimal or no labeled examples (Lu et al. 2023; Wang et al. 2021). Few-shot learning adapting to new tasks using only a handful of labeled examples, often through metalearning frameworks or prompt engineering in large language models (Gharoun et al. 2024). Applications range from multilingual machine translation and dialogue systems to medical image segmentation and rare disease diagnosis, where labeled data are scarce (Tian et al. 2024; Gharoun et al. 2024; Kadam and Vaidya 2020). Zero-shot learning relies on leveraging semantic information, such as textual prompts or label descriptions (Kojima et al. 2022; Kadam and Vaidya 2020), to make inferences about classes not seen during training. A prominent example is the Contrastive Language-Image Pre-Training (CLIP) model, which performs image classification by comparing image embeddings with textual class descriptions (Radford et al. 2021).

Domain adaptation addresses the challenge of distributional mismatch between source and target data. It aims to minimize the domain shift by aligning feature spaces or leveraging adver-

sarial training to learn domain-invariant representations (Iman et al. 2023). For instance, models trained on natural images may perform poorly on grayscale or low-quality images unless adapted. Techniques include fine-tuning with domain-specific corpora, using domain adversarial networks, or adapting representations through unsupervised or semi-supervised learning approaches (Ganin et al. 2016; Iman et al. 2023; Zhuang et al. 2021).

8.3.2 Continuous meta-knowledge transfer

Continual fine-tuning, or incremental learning, involves sequentially updating models as new tasks or data become available (De Lange et al. 2021; Biesialska et al. 2020; Parisi et al. 2019). Unlike static fine-tuning, the challenge here is to retain performance on previously learned tasks while integrating new knowledge, a problem known as catastrophic forgetting (Luo et al. 2023; Belouadah and Popescu 2019). Techniques such as elastic weight consolidation and knowledge distillation are commonly employed to mitigate this issue (De Lange et al. 2021). Applications include clinical decision systems that evolve with patient records over time, or adaptive recommendation engines that adjust to user preferences without retraining from scratch and graph lifelong learning (Febrinanto et al. 2023; Armstrong and Clifton 2022; Xie et al. 2022; Parisi et al. 2019).

Modular networks and progressive neural architectures address continual learning by structurally extending the model for each new task (Wang et al. 2024; Parisi et al. 2019). Instead of modifying shared parameters, new modules or branches are added to accommodate additional knowledge while preserving prior components (Liu et al. 2018; Parisi et al. 2019). This architecture is especially effective in robotics and multi-task learning settings, where new skills or environments are incrementally introduced (Devin et al. 2017). Progressive neural networks have been shown to outperform static models in tasks involving domain shifts, such as transferring control policies in reinforcement learning across different simulations (Liu et al. 2018; Ju et al. 2022).

Federated transfer learning combines the principles of decentralized learning and transfer learning to enable knowledge sharing across distributed environments without exchanging raw data (Wang et al. 2021). This is especially useful in privacy-sensitive domains, where institutions can collaboratively train models without violating data confidentiality (Li et al. 2021). Models are pre-trained on local data and aggregated using federated averaging or transfer-based strategies (Li et al. 2021).

9 Further research directions

As AutoML's landscape continues to evolve, several emerging research directions hold promise for advancing the field further. This chapter explores key research areas that will shape AutoML's future, addressing both opportunities for innovation and challenges that need to be overcome.

Enhancing AutoML interpretability and explainability: One of the ongoing challenges in AutoML is ensuring that automated models are interpretable and their decisions are explainable. As AutoML systems become more complex, particularly with the use of neural architecture search and other advanced techniques, understanding how models arrive at their predictions becomes increasingly difficult. Research in this area can

develop methods to improve the transparency of AutoML models (Coors et al. 2021; Eldeeb and Elshawi 2024). Enhancing interpretability is crucial for fostering trust in automated systems and ensuring their ethical application in sensitive domains (Burkart and Huber 2021).

Improving generalization: Ensuring that AutoML models generalize well across diverse datasets and real-world scenarios is another critical research direction. Current methods often stand out in controlled environments but may struggle with generalization when faced with new or unseen data distributions. There are some papers reporting overfiting which some tools can produce due to the large hyperparameter space (Thornton et al. 2013; Fabris and Freitas 2019). Research efforts could focus on developing techniques that enhance the robustness and adaptability of automated models. This includes methods for handling distributional shifts, improving the ability of models to learn from limited or noisy data, and designing AutoML systems that are resilient to adversarial attacks and other perturbations.

Addressing scalability and computational efficiency: As AutoML systems become more sophisticated, scalability and computational efficiency are increasingly important considerations. Research focusing on scaling computational resources efficiently is required, especially for hyperparameter tuning of pipelines and architectures. Techniques such as distributed computing, efficient sampling strategies, and resource-aware optimization algorithms are being explored to reduce the time and cost associated with AutoML processes. Developing scalable AutoML solutions that can handle large-scale data and complex models is essential for maintaining the practical applicability of these systems in real-world scenarios (Ribeiro et al. 2015).

Addressing security risks: As AutoML becomes increasingly autonomous, it also introduces new security vulnerabilities that the research community has not yet fully addressed. For instance, automated systems that ingest and process data without human supervision may be susceptible to data poisoning attacks, where maliciously crafted data can corrupt the model training process, leading to exploitable models. Similarly, AutoML-generated models may be vulnerable to adversarial examples, i.e., inputs designed to deceive the model into making incorrect predictions. For example, NAS-generated models are more vulnerable to attacks compared to manually designed ones, as they often favor architectures that converge quickly, which exhibit properties such as low gradient variance and high loss smoothness-traits associated with increased susceptibility to adversarial manipulation (Pang et al. 2022).

Online learning in AutoML: Typically, in real-world scenarios, data is generated continually over time. Thus, it is necessary to adapt AutoML methods to online learning scenarios, where models continuously evolve and update in response to new incoming data (Hoi et al. 2021). This includes developing techniques for online model selection, real-time hyperparameter optimization, and automated pipeline selection in environments where data is constantly changing (Celik et al. 2023). Moreover, key challenges involve computational efficiency, and handling concept drift problems.

Federated learning in AutoML: Federated learning enables decentralized model training across multiple devices without sharing raw data, ensuring privacy and reducing costs (Yang et al. 2019). When combined with AutoML, it could offer the benefit of optimizing model performance directly on distributed devices. This approach can lead to more efficient and scalable models, particularly in resource-constrained environments. Future

research should focus on creating federated AutoML frameworks that handle data heterogeneity, address computational limitations, and improve generalization across decentralized datasets (Preuveneers 2023).

Multimodal learning in AutoML: Multimodal machine learning focuses on the design of models that can handle diverse data types, such as text, images, and structured data (Shi et al. 2021; Tang et al. 2024). A key challenge for AutoML would be developing unified representations to combine different modalities and efficiently exploring large, complex search spaces involving multiple neural network architectures (Shi et al. 2021; Tang et al. 2024).

Enhancing usability and accessibility: Making AutoML tools more user-friendly and accessible to non-experts is a key research direction. This involves designing intuitive interfaces, simplifying the setup and configuration processes, and providing comprehensive documentation and support. Research in this area could lower the barrier to entry for users with limited machine learning experience, enabling a broader audience. Usability enhancements can also involve creating interactive visualizations, adaptive interfaces that guide users through the ML process and Large Language Models as an interface for data pipelines (Junior et al. 2024).

Ethical and societal implications: The ethical and societal implications of AI are increasingly coming into focus (Mittelstadt 2019). Research in this area addresses the potential biases and fairness issues associated with automated systems, as well as the broader impact on employment and decision-making. Ensuring that AutoML technologies are developed taking into account ethical principles and societal values is crucial for their integration into various areas.

10 Conclusion

This survey has offered a comprehensive analysis of AutoML, detailing its evolution from foundational concepts in MtL, HPO and TL to its current advanced methodologies and applications. The paper highlighted the historical evolution and classical definitions within the MtL, HPO, and TL, elucidating the connections between these fields through meta-data and meta-knowledge, culminating in an expanded definition of AutoML and its objectives.

Finally, the paper systematically organized the literature surveys, introduced the main applications of AutoML, from meta-models and simple random search to neural architecture search and pipeline design, and finished with a comprehensive research direction, including enhancements on interpretability, generalization, and ethical and societal implications.

Looking ahead, AutoML has the potential to revolutionize the development and application of machine learning solutions across diverse domains. As AutoML continues to evolve, it will present new opportunities and challenges, underscoring the necessity for ongoing research and innovation to realize its full potential.

Appendix A: Categorization of literature reviews

Table 1 presents the reviews organized according to the proposed categorization. The columns *Classical Connection* and *Subcategory* reflect the described structure. Additionally, we present the publication year and the names of the authors.

Table 1 Categorization of literature reviews according to classical connections with papers

Classical connection	Sub category	Year	Author
Hyperparameter optimization	Multiple step optimization (neural architecture search)	2023	Zheng et al.
		2023	Kang et al.
		2022	Talbi
		2021	Zhang et al.
		2021	He et al.
		2021	Elsken et al.*
		2021	Wistuba et al.*
		2019	Jaafra et al.
	Multiple step optimization (pipeline design)	2023	Barbudo et al.
		2022	Meisenbacher et al.
		2022	Karmaker et al.
		2021	Wever et al.
		2021	Zöller and Huber
		2019	Nagarajah and Poravi
	Single step optimization	2023	Morales-Hernández et al.
		2023	Bischl et al.
		2023	Karl et al.
		2022	Kadhim et al.
		2022	Parker-Holder et al.
		2020	Yang and Shami
		2019	Al-Sahaf et al.
		2015	Hutter et al.
Metalearning	From meta-features	2022	Rivolli et al.
		2020	Khan et al.
		2015	Lemke et al.
		2018	Vanschoren*
		2009	Smith-Miles*
	From model	2024	Vettoruzzo et al.
		2022	Tian et al.
		2022	Ma et al.
Transfer learning	Continuous meta-knowledge transfer	2024	Wang et al.*
		2023	Febrinanto et al.*
		2021	De Lange et al.*
		2020	Biesialska et al.*
		2019	Parisi et al.
	Meta-knowledge transfer	2024	Zhao et al.
		2023	Iman et al.
		2021	Zhuang et al.
		2021	Panigrahi et al.
		2021	Weber et al.
		2021	Agarwal et al.
		2020	Farahani et al.
		2020	Niu et al.
		2019	Liang et al.
		2016	Weiss et al.
		2009	Pan and Yang*
		2009	Taylor and Stone

Table 1 (continued)

Classical connection	Sub category	Year	Author
Hybrid	Multiple step optimization (neural architecture search)	2023	Kaveh and Mesgari
	Single step optimization	2024	Majid et al.
		2024	Li et al.
		2023	Bai et al.
		2022	Ünal and Başçiftçi
		2018	Kerschke et al.
		2017	Rodrigues et al.
		2014	Pappa et al.
	Meta-knowledge transfer	2024	Tian et al.
		2024	Gharoun et al.
		2023	Lu et al.
		2021	Wang et al.
		2020	Kadam and Vaidya
	From meta-features	2002	Vilalta and Drissi

* Some review papers discovered during our reading were not found due to limitations in the data source or search string. Therefore, we added them manually

Acknowledgements This research was supported by FAPESP (grant 2013/07375-0, grant 2018/14819-5) and Artificial Intelligence in the Remaking of Urban Environments (IARA -- grant 2020/09835-1). This study was financed in part by the Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brasil (CAPES) - Finance Code 001. This work was supported by the National Council for Scientific and Technological Development (CNPq).

Author contributions E.A. conducted the primary research, conceptualized the study, and drafted the main manuscript text. A.C.P.L.F.C provided critical feedback, contributed to the development of the manuscript structure, and assisted with the synthesis of key topics and the revision process. Both authors reviewed and approved the final manuscript.

Data availability No datasets were generated or analysed during the current study.

Declarations

Competing interests The authors declare no competing interests.

References

Adamson RE (1952) Functional fixedness as related to problem solving: a repetition of three experiments. J Exp Psychol 44(4):288

Agarwal N, Sondhi A, Chopra K, Singh G (2021) Transfer learning: survey and classification. Smart Innovations in Communication and Computational Sciences Proceedings of ICSICCS 2020:145–155

Aha DW (1992) Generalizing from case studies: A case study. In: Proceedings of the Ninth International Workshop on Machine Learning (ML 1992), Aberdeen, Scotland, UK, July 1–3, 1992, pp 1–10 . https://doi.org/10.1016/b978-1-55860-247-2.50006-1

Aha DW, Kibler DF, Albert MK (1991) Instance-based learning algorithms. Mach Learn 6:37–66. https://doi.org/10.1023/A:1022689900470

Alcobaça E, Siqueira F, Rivolli A, Garcia LPF, Oliva JT, Carvalho AC (2020) MFE: towards reproducible meta-feature extraction. J Mach Learn Res 21:111–1

Al-Sahaf H, Bi Y, Chen Q, Lensen A, Mei Y, Sun Y, Tran B, Xue B, Zhang M (2019) A survey on evolutionary machine learning. J R Soc N Z 49(2):205–228

Arbane M, Benlamri R, Brik Y, Alahmar AD (2023) Social media-based Covid-19 sentiment classification model using Bi-LSTM. Expert Syst Appl 212:118710

Armstrong J, Clifton DA (2022) Continual learning of longitudinal health records. In: 2022 IEEE-EMBS International Conference on Biomedical and Health Informatics (BHI), pp 01–06. IEEE

Baevski A, Zhou Y, Mohamed A, Auli M (2020) wav2vec 2.0: a framework for self-supervised learning of speech representations. Adv Neural Information Process Syst 33:12449–12460

Bai H, Cheng R, Jin Y (2023) Evolutionary reinforcement learning: a survey. Intell Comput 2:0025

Baker B, Gupta O, Naik N, Raskar R (2016) Designing neural network architectures using reinforcement learning. arXiv preprint arXiv:1611.02167

Banzhaf W, Nordin P, Keller RE, Francone FD (1998) Genetic programming. Springer, Cham

Barbudo R, Ventura S, Romero JR (2023) Eight years of automl: categorisation, review and trends. Knowl Inf Syst 65(12):5097–5149

Belouadah E, Popescu A (2019) Il2m: Class incremental learning with dual memory. In: Proceedings of the IEEE/CVF International Conference on Computer Vision, pp 583–592

Bensusan H, Kalousis A (2001) Estimating the predictive accuracy of a classifier. In: European Conference on Machine Learning, pp 25–36. Springer

Bergstra J, Bengio Y (2012) Random search for hyper-parameter optimization. J Mach Learn Res 13:281–305

Bergstra J, Bardenet R, Bengio Y, Kégl B (2011) Algorithms for hyper-parameter optimization. In: Advances in Neural Information Processing Systems 24: 25th Annual Conference on Neural Information Processing Systems 2011. Proceedings of a Meeting Held 12–14 December 2011, Granada, Spain, pp 2546–2554. http://papers.nips.cc/paper/4443-algorithms-for-hyper-parameter-optimization

Biesialska M, Biesialska K, Costa-Jussa MR (2020) Continual lifelong learning in natural language processing: a survey. arXiv preprint arXiv:2012.09823

Biggs JB (1985) The role of metalearning in study processes. Br J Educ Psychol 55(3):185–212

Bilalli B, Abelló A, Aluja-Banet T, Wrembel R (2018) Intelligent assistance for data pre-processing. Comput Standards Interfaces 57:101–109

Bischl B, Binder M, Lang M, Pielok T, Richter J, Coors S, Thomas J, Ullmann T, Becker M, Boulesteix A-L (2023) Hyperparameter optimization: foundations, algorithms, best practices, and open challenges. Wiley Interdiscipl Rev Data Mining Knowl Discov 13(2):1484

Biswas SK, Rauniyar A, Muhuri PK (2016) Multi-objective bayesian optimization algorithm for real-time task scheduling on heterogeneous multiprocessors. In: IEEE Congress on Evolutionary Computation, CEC 2016, Vancouver, BC, Canada, July 24–29, 2016, pp 2844–2851. https://doi.org/10.1109/CEC.2016.7744148

Boyd S, Vandenberghe L (2004) Convex optimization. Cambridge University Press, Cambridge

Brazdil P, Soares C, Costa JP (2003) Ranking learning algorithms: using IBL and meta-learning on accuracy and time results. Mach Learn 50(3):251–277. https://doi.org/10.1023/A:1021713901879

Brazdil P, Giraud-Carrier CG, Soares C, Vilalta R (2009) Metalearning - Applications to data mining. Cognitive technologies. Springer. https://doi.org/10.1007/978-3-540-73263-1

Breiman L, Friedman J, Stone CJ, Olshen RA (1984) Classification and regression trees. CRC Press, Boca Raton

Brochu E, Cora VM, Freitas N (2010) A tutorial on bayesian optimization of expensive cost functions, with application to active user modeling and hierarchical reinforcement learning. CoRR **abs/1012.2599** arXiv:1012.2599

Brodley CE (1993) Addressing the selective superiority problem: Automatic algorithm/model class selection. In: Proceedings of the Tenth International Conference on Machine Learning, pp 17–24

Burkart N, Huber MF (2021) A survey on the explainability of supervised machine learning. J Artif Intell Res 70:245–317

Cai H, Yang J, Zhang W, Han S, Yu Y (2018) Path-level network transformation for efficient architecture search. In: International Conference on Machine Learning, pp. 678–687. PMLR

Celik B, Singh P, Vanschoren J (2023) Online automl: an adaptive automl framework for online learning. Mach Learn 112(6):1897–1921

Cheng M-Y, Huang K-Y, Hutomo M (2018) Multiobjective dynamic-guiding PSO for optimizing work shift schedules. J Constr Eng Manag 144(9):04018089
Coors S, Schalk D, Bischl B, Rügamer D (2021) Automatic componentwise boosting: An interpretable automl system. arXiv preprint arXiv:2109.05583
Craw S, Sleeman D, Graner N, Rissakis M, Sharma S (1992) Consultant: Providing advice for the machine learning toolbox. In: Proceedings of the Research and Development in Expert Systems IX, 5–23
De Lange M, Aljundi R, Masana M, Parisot S, Jia X, Leonardis A, Slabaugh G, Tuytelaars T (2021) A continual learning survey: defying forgetting in classification tasks. IEEE Trans Pattern Anal Mach Intell 44(7):3366–3385
Devin C, Gupta A, Darrell T, Abbeel P, Levine S (2017) Learning modular neural network policies for multi-task and multi-robot transfer. In: 2017 IEEE International Conference on Robotics and Automation (ICRA), pp 2169–2176. IEEE
Dietterich TG, Kong EB (1995) Machine learning bias, statistical bias, and statistical variance of decision tree algorithms. Technical report, Citeseer
Dôres SCN, Soares C, Ruiz D (2018) Bandit-based automated machine learning. In: 2018 7th Brazilian Conference on Intelligent Systems (BRACIS), pp 121–126. IEEE
Eggensperger K, Feurer M, Hutter F, Bergstra J, Snoek J, Hoos H, Leyton-Brown K (2013) Towards an empirical foundation for assessing bayesian optimization of hyperparameters. In: NIPS Workshop on Bayesian Optimization in Theory and Practice, vol 10, p 3
Eldeeb H, Elshawi R (2024) Empowering machine learning with scalable feature engineering and interpretable automl. IEEE Transactions on Artificial Intelligence
Elsken T, Metzen JH, Hutter F (2019) Neural architecture search: a survey. J Mach Learn Res 20(1):1997–2017
Elsken T, Metzen JH, Hutter F (2018) Efficient multi-objective neural architecture search via lamarckian evolution. arXiv preprint arXiv:1804.09081
Elsken T, Metzen JH, Hutter F (2019) Efficient multi-objective neural architecture search via lamarckian evolution. In: 7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6–9, 2019. https://openreview.net/forum?id=ByME42AqK7
Fabris F, Freitas AA (2019) Analysing the overfit of the auto-sklearn automated machine learning tool. In: International Conference on Machine Learning, Optimization, and Data Science, pp 508–520. Springer
Farahani A, Pourshojae B, Rasheed K, Arabnia HR (2020) A concise review of transfer learning. In: 2020 International Conference on Computational Science and Computational Intelligence (CSCI), pp 344–351. IEEE
Febrinanto FG, Xia F, Moore K, Thapa C, Aggarwal C (2023) Graph lifelong learning: a survey. IEEE Comput Intell Mag 18(1):32–51
Feurer M, Hutter F (2019) Hyperparameter optimization. Automated machine learning. Springer, Cham, pp 3–33
Feurer M, Klein A, Eggensperger K, Springenberg JT, Blum M, Hutter F (2015) Efficient and robust automated machine learning. In: Advances in Neural Information Processing Systems 28: Annual Conference on Neural Information Processing Systems 2015, December 7–12, 2015, Montreal, Quebec, Canada, pp. 2962–2970. http://papers.nips.cc/paper/5872-efficient-and-robust-automated-machine-learning
Feurer M, Springenberg JT, Hutter F (2015) Initializing bayesian hyperparameter optimization via meta-learning. In: Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence, January 25-30, 2015, Austin, Texas, USA, pp 1128–1135. http://www.aaai.org/ocs/index.php/AAAI/AAAI15/paper/view/10029
Finn C, Abbeel P, Levine S (2017) Model-agnostic meta-learning for fast adaptation of deep networks. In: Proceedings of the 34th International Conference on Machine Learning, ICML 2017, Sydney, NSW, Australia, 6–11 August 2017, pp 1126–1135. http://proceedings.mlr.press/v70/finn17a.html
Fürnkranz J, Petrak J (2001) An evaluation of landmarking variants. In: Working Notes of the ECML/PKDD 2000 Workshop on Integrating Aspects of Data Mining, Decision Support and Meta-Learning, pp 57–68
Ganin Y, Ustinova E, Ajakan H, Germain P, Larochelle H, Laviolette F, March M, Lempitsky V (2016) Domain-adversarial training of neural networks. J Mach Learn Res 17(59):1–35
Gharoun H, Momenifar F, Chen F, Gandomi AH (2024) Meta-learning approaches for few-shot learning: a survey of recent advances. ACM Comput Surv 56(12):1–41
Giraud-Carrier CG (2005) The data mining advisor: meta-learning at the service of practitioners. In: Fourth International Conference on Machine Learning and Applications, ICMLA 2005, Los Angeles, California, USA, 15–17 December 2005. https://doi.org/10.1109/ICMLA.2005.65
Gogna A, Tayal A (2013) Metaheuristics: review and application. J Exp Theor Artif Intell 25(4):503–526
Gomes TAF, Prudêncio RBC, Soares C, Rossi ALD, Carvalho ACPLF (2012) Combining meta-learning and search techniques to select parameters for support vector machines. Neurocomputing 75(1):3–13. https://doi.org/10.1016/j.neucom.2011.07.005

Gong T, Ma Y (2024) PSO-based lightweight neural architecture search for object detection. Swarm Evol Comput 90:101684
Graner N, Sharma S, Sleeman DH, Rissakis M, Craw S, Moore C (1993) The machine learning toolbox consultant. Int J Artif Intell Tools 2(3):307–328. https://doi.org/10.1142/S0218213093000163
Guerra SB, Prudêncio RB, Ludermir TB (2008) Predicting the performance of learning algorithms using support vector machines as meta-regressors. In: International Conference on Artificial Neural Networks, pp 523–532. Springer
He X, Zhao K, Chu X (2021) Automl: a survey of the state-of-the-art. Knowl Based Syst 212:106622
Henery RJ, Taylor CC (1992) Statlog: an evaluation of machine learning and statistical algorithms. In: Dodge Y, Whittaker J (eds) Computational Statistics. Physica-Verlag HD, Heidelberg, pp 157–162
Hilario M, Nguyen P, Do H, Woznica A, Kalousis A (2011) Ontology-based meta-mining of knowledge discovery workflows. In: Meta-learning in Computational Intelligence, pp 273–315. Springer
Hoi SC, Sahoo D, Lu J, Zhao P (2021) Online learning: a comprehensive survey. Neurocomputing 459:249–289
Holland JH (1992) Genetic algorithms. Scientific American 267(1):66–73
Horn D, Bischl B (2016) Multi-objective parameter configuration of machine learning algorithms using model-based optimization. In: 2016 IEEE Symposium Series on Computational Intelligence (SSCI), pp 1–8. IEEE
Hutter F, Lücke J, Schmidt-Thieme L (2015) Beyond manual tuning of hyperparameters. KI-Künstliche Intelligenz 29:329–337
Hutter F, Kotthoff L, Vanschoren J (eds.) (2019) Automated machine learning—methods, systems, challenges, 1st edn. The Springer Series on Challenges in Machine Learning. Springer, https://doi.org/10.1007/978-3-030-05318-5
Iman M, Arabnia HR, Rasheed K (2023) A review of deep transfer learning and recent advancements. Technologies 11(2):40
Jaafra Y, Laurent JL, Deruyver A, Naceur MS (2019) Reinforcement learning for neural architecture search: a review. Image Vis Comput 89:57–66
Ju H, Juan R, Gomez R, Nakamura K, Li G (2022) Transferring policy of deep reinforcement learning from simulation to reality for robotics. Nat Mach Intell 4(12):1077–1087
Junior SB, Ceravolo P, Groppe S, Jarrar M, Maghool S, Sèdes F, Sahri S, Van Keulen M (2024) Are large language models the new interface for data pipelines? arXiv preprint arXiv:2406.06596
Kadam S, Vaidya V (2020) Review and analysis of zero, one and few shot learning approaches. In: Intelligent Systems Design and Applications: 18th International Conference on Intelligent Systems Design and Applications (ISDA 2018) Held in Vellore, India, December 6–8, 2018, vol 1, pp 100–112. Springer
Kadhim ZS, Abdullah HS, Ghathwan KI (2022) Artificial neural network hyperparameters optimization: a survey. Int J Online Biomed Eng 18(15):59–87
Kang J-S, Kang J, Kim J-J, Jeon K-W, Chung H-J, Park B-H (2023) Neural architecture search survey: a computer vision perspective. Sensors 23(3):1713
Karl F, Pielok T, Moosbauer J, Pfisterer F, Coors S, Binder M, Schneider L, Thomas J, Richter J, Lang M (2023) Multi-objective hyperparameter optimization in machine learning an overview. ACM Trans Evolutionary Learn Optimization 3(4):1–50
Karmaker SK, Hassan MM, Smith MJ, Xu L, Zhai C, Veeramachaneni K (2022) Automl to date and beyond: challenges and opportunities. ACM Comput Surveys 54(8):1–36
Kaveh M, Mesgari MS (2023) Application of meta-heuristic algorithms for training neural networks and deep learning architectures: a comprehensive review. Neural Process Lett 55(4):4519–4622
Kerschke P, Hoos HH, Neumann F, Trautmann H (2018) Automated algorithm selection: survey and perspectives. Evol Comput 27(1):3–45
Khan I, Zhang X, Rehman M, Ali R (2020) A literature survey and empirical study of meta-learning for classifier selection. IEEE Access 8:10262–10281
Kitchenham B, Brereton OP, Budgen D, Turner M, Bailey J, Linkman S (2009) Systematic literature reviews in software engineering—a systematic literature review. Inf Softw Technol 51(1):7–15
Koch G, Zemel R, Salakhutdinov R (2015) Siamese neural networks for one-shot image recognition. In: ICML Deep Learning Workshop, vol 2, pp 1–30. Lille
Kohavi R, John GH (1995) Automatic parameter selection by minimizing estimated error. In: Machine Learning, Proceedings of the Twelfth International Conference on Machine Learning, Tahoe City, California, USA, July 9–12, 1995, pp 304–312. https://doi.org/10.1016/b978-1-55860-377-6.50045-1
Kojima T, Gu SS, Reid M, Matsuo Y, Iwasawa Y (2022) Large language models are zero-shot reasoners. Adv Neural Inf Process Syst 35:22199–22213
Komer B, Bergstra J, Eliasmith C Hyperopt-sklearn: automatic hyperparameter configuration for scikit-learn. In: ICML Workshop on AutoML, vol 9. Citeseer

Lake BM, Ullman TD, Tenenbaum JB, Gershman SJ (2017) Building machines that learn and think like people. Behav Brain Sci 40:e253
Lemke C, Gabrys B (2010) Meta-learning for time series forecasting and forecast combination. Neurocomputing 73(10–12):2006–2016. https://doi.org/10.1016/j.neucom.2009.09.020
Lemke C, Budka M, Gabrys B (2015) Metalearning: a survey of trends and technologies. Artif Intell Rev 44(1):117–130
Lessmann S, Stahlbock R, Crone SF (2005) Optimizing hyperparameters of support vector machines by genetic algorithms. In: IC-AI, pp 74–82
Li Q, Wen Z, Wu Z, Hu S, Wang N, Li Y, Liu X, He B (2021) A survey on federated learning systems: vision, hype and reality for data privacy and protection. IEEE Trans Knowl Data Eng 35(4):3347–3366
Li N, Ma L, Yu G, Xue B, Zhang M, Jin Y (2024) Survey on evolutionary deep learning: principles, algorithms, applications, and open issues. ACM Comput Surv 56(2):1–34
Liang H, Fu W, Yi F (2019) A survey of recent advances in transfer learning. In: 2019 IEEE 19th International Conference on Communication Technology (ICCT), pp 1516–1523. IEEE
Liu C, Zoph B, Neumann M, Shlens J, Hua W, Li L-J, Fei-Fei L, Yuille A, Huang J, Murphy K (2018) Progressive neural architecture search. In: Proceedings of the European Conference on Computer Vision (ECCV)
Li N, Xue B, Ma L, Zhang M (2025) Automatic fuzzy architecture design for defect detection via classifier-assisted multiobjective optimization approach. IEEE Transactions on Evolutionary Computation
Li Z, Zhou F, Chen F, Li H (2017) Meta-sgd: Learning to learn quickly for few-shot learning. arXiv preprint arXiv:1707.09835
Lorena AC, Garcia LPF, Lehmann J, Souto MCP, Ho TK (2019) How complex is your classification problem?: a survey on measuring classification complexity. ACM Comput Surv 52(5):107–110734. https://doi.org/10.1145/3347711
Lorenzo PR, Nalepa J, Kawulok M, Ramos LS, Pastor JR (2017) Particle swarm optimization for hyperparameter selection in deep neural networks. In: Proceedings of the Genetic and Evolutionary Computation Conference, pp 481–488
Lu Z, Cheng R, Jin Y, Tan KC, Deb K (2023) Neural architecture search as multiobjective optimization benchmarks: problem formulation and performance assessment. IEEE Trans Evol Comput 28(2):323–337
Lu J, Gong P, Ye J, Zhang J, Zhang C (2023) A survey on machine learning from few samples. Pattern Recogn 139:109480
Luo Y, Yang Z, Meng F, Li Y, Zhou J, Zhang Y (2023) An empirical study of catastrophic forgetting in large language models during continual fine-tuning. arXiv preprint arXiv:2308.08747
Ma Y, Zhao S, Wang W, Li Y, King I (2022) Multimodality in meta-learning: a comprehensive survey. Knowl Based Syst 250:108976
Majid AY, Saaybi S, Francois-Lavet V, Prasad RV, Verhoeven C (2024) Deep reinforcement learning versus evolution strategies: a comparative survey. IEEE Transactions on Neural Networks and Learning Systems
Mantovani RG, Rossi AL, Alcobaça E, Vanschoren J, Carvalho AC (2019) A meta-learning recommender system for hyperparameter tuning: predicting when tuning improves SVM classifiers. Inf Sci 501:193–221
Meisenbacher S, Turowski M, Phipps K, Rätz M, Müller D, Hagenmeyer V, Mikut R (2022) Review of automated time series forecasting pipelines. Wiley Interdiscipl Rev Data Mining Knowl Discovery 12(6):1475
Michie D, Spiegelhalter DJ, Taylor CC (1994) Machine learning, neural and statistical classification. Ellis Horwood
Mitchell TM (1980) The need for biases in learning generalizations
Mittelstadt B (2019) Principles alone cannot guarantee ethical AI. Nat Mach Intell 1(11):501–507
Mohr F, Wever M, Hüllermeier E (2018) Ml-plan: automated machine learning via hierarchical planning. Mach Learn 107(8–10):1495–1515. https://doi.org/10.1007/s10994-018-5735-z
Morales-Hernández A, Nieuwenhuyse I, Rojas Gonzalez S (2023) A survey on multi-objective hyperparameter optimization algorithms for machine learning. Artif Intell Rev 56(8):8043–8093
Munkhdalai T, Yu H (2017) Meta networks. In: International Conference on Machine Learning, pp. 2554–2563. PMLR
Nagarajah T, Poravi G (2019) A review on automated machine learning (automl) systems. In: 2019 IEEE 5th International Conference for Convergence in Technology (I2CT), pp 1–6. IEEE
Nguyen P, Hilario M, Kalousis A (2014) Using meta-mining to support data mining workflow planning and optimization. J Artif Intell Res 51:605–644
Niu S, Liu Y, Wang J, Song H (2020) A decade survey of transfer learning (2010–2020). IEEE Trans Artif Intell 1(2):151–166

Olson RS, Bartley N, Urbanowicz RJ, Moore JH (2016) Evaluation of a tree-based pipeline optimization tool for automating data science. In: Proceedings of the Genetic and Evolutionary Computation Conference 2016. GECCO '16, pp 485–492. Association for Computing Machinery, New York, NY, USA. https://doi.org/10.1145/2908812.2908918

Olson RS, Urbanowicz RJ, Andrews PC, Lavender NA, Moore JH (2016) Automating biomedical data science through tree-based pipeline optimization. In: European Conference on the Applications of Evolutionary Computation, pp 123–137. Springer

Oquab M, Bottou L, Laptev I, Sivic J (2014) Learning and transferring mid-level image representations using convolutional neural networks. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp 1717–1724

Pan SJ, Yang Q (2009) A survey on transfer learning. IEEE Trans Knowl Data Eng 22(10):1345–1359

Pang R, Xi Z, Ji S, Luo X, Wang T (2022) On the security risks of {AutoML}. In: 31st USENIX Security Symposium (USENIX Security 22), pp 3953–3970

Panigrahi S, Nanda A, Swarnkar T (2021) A survey on transfer learning. In: Intelligent and Cloud Computing: Proceedings of ICICC 2019, vol 1, pp 781–789. Springer, Singapore

Pappa GL, Ochoa G, Hyde MR, Freitas AA, Woodward J, Swan J (2014) Contrasting meta-learning and hyper-heuristic research: the role of evolutionary algorithms. Genet Program Evolvable Mach 15(1):3–35. https://doi.org/10.1007/s10710-013-9186-9

Parisi GI, Kemker R, Part JL, Kanan C, Wermter S (2019) Continual lifelong learning with neural networks: a review. Neural Netw 113:54–71

Parker-Holder J, Rajan R, Song X, Biedenkapp A, Miao Y, Eimer T, Zhang B, Nguyen V, Calandra R, Faust A (2022) Automated reinforcement learning (autorl): a survey and open problems. J Artif Intell Res 74:517–568

Parmezan ARS, Lee HD, Wu FC (2017) Metalearning for choosing feature selection algorithms in data mining: proposal of a new framework. Expert Syst Appl 75:1–24

Peng Y, Flach PA, Soares C, Brazdil P (2002) Improved dataset characterisation for meta-learning. In: Discovery Science, 5th International Conference, DS 2002, Lübeck, Germany, November 24–26, 2002, Proceedings, pp 141–152. https://doi.org/10.1007/3-540-36182-0_14

Pepino L, Riera P, Ferrer L (2021) Emotion recognition from speech using wav2vec 2.0 embeddings. arXiv preprint arXiv:2104.03502

Petersen K, Feldt R, Mujtaba S, Mattsson M (2008) Systematic mapping studies in software engineering. EASE 8:68–77

Petersen K, Vakkalanka S, Kuzniarz L (2015) Guidelines for conducting systematic mapping studies in software engineering: an update. Inf Softw Technol 64:1–18

Pfahringer B, Bensusan H, Giraud-Carrier CG (2000) Meta-learning by landmarking various learning algorithms. In: Proceedings of the Seventeenth International Conference on Machine Learning (ICML 2000), Stanford University, Stanford, CA, USA, June 29–July 2, 2000, pp 743–750

Pham H, Guan M, Zoph B, Le Q, Dean J (2018) Efficient neural architecture search via parameters sharing. In: Dy J, Krause A (Eds) Proceedings of the 35th International Conference on Machine Learning. Proceedings of Machine Learning Research, vol 80, pp 4095–4104. PMLR. https://proceedings.mlr.press/v80/pham18a.html

Pimentel BA, Carvalho AC (2019) A new data characterization for selecting clustering algorithms using meta-learning. Inf Sci 477:203–219

Pinto F, Soares C, Mendes-Moreira J (2016) Towards automatic generation of metafeatures. In: Pacific-Asia Conference on Knowledge Discovery and Data Mining, pp 215–226. Springer

Pratt LY, Mostow J, Kamm CA (1991) Direct transfer of learned information among neural networks. Proceedings of the Ninth National Conference on Artificial intelligence 2:584–589

Preuveneers D (2023) Autofl: towards automl in a federated learning context. Appl Sci 13(14):8019

Provost F, Fawcett T (2013) Data science and its relationship to big data and data-driven decision making. Big Data 1(1):51–59

Radford A, Kim JW, Hallacy C, Ramesh A, Goh G, Agarwal S, Sastry G, Askell A, Mishkin P, Clark J (2021) Learning transferable visual models from natural language supervision. In: International Conference on Machine Learning, pp 8748–8763. PmLR

Reif M, Shafait F, Dengel A (2012) Meta-learning for evolutionary parameter optimization of classifiers. Mach Learn 87(3):357–380

Reif M, Shafait F, Goldstein M, Breuel T, Dengel A (2014) Automatic classifier selection for non-experts. Pattern Anal Appl 17(1):83–96

Reif M, Shafait F, Dengel A (2011) Prediction of classifier training time including parameter optimization. In: Annual Conference on Artificial Intelligence, pp 260–271. Springer

Rendell LA, Cho H (1990) Empirical learning as a function of concept character. Mach Learn 5:267–298. https://doi.org/10.1007/BF00117106

Ribeiro M, Grolinger K, Capretz MA (2015) Mlaas: Machine learning as a service. In: 2015 IEEE 14th International Conference on Machine Learning and Applications (ICMLA), pp 896–902. IEEE
Rice JR (1976) The algorithm selection problem. Adv Comput 15:65–118. https://doi.org/10.1016/S0065-2458(08)60520-3
Rijn JN, Abdulrahman SM, Brazdil P, Vanschoren J (2015) Fast algorithm selection using learning curves. In: Advances in Intelligent Data Analysis XIV - Proceedings of 14th International Symposium, IDA 2015, Saint Etienne, France, October 22–24, 2015, pp 298–309. https://doi.org/10.1007/978-3-319-24465-5_26
Ritter FE (1991) Towards fair comparisons of connectionist algorithms through automatically optimized parameter sets. In: Proceedings of the Annual Conference of the Cognitive Science Society, pp 877–881
Rivolli A, Garcia LP, Soares C, Vanschoren J, Carvalho AC (2022) Meta-features for meta-learning. Knowl Based Syst 240:108101
Rivolli A, Garcia LP, Soares C, Vanschoren J, Carvalho AC (2018) Towards reproducible empirical research in meta-learning. arXiv preprint arXiv:1808.10406
Rodrigues D, Papa JP, Adeli H (2017) Meta-heuristic multi-and many-objective optimization techniques for solution of machine learning problems. Expert Syst 34(6):12255
Sá AG, Pinto WJG, Oliveira LOV, Pappa GL (2017) Recipe: a grammar-based framework for automatically evolving classification pipelines. In: European Conference on Genetic Programming, pp 246–261. Springer
Sanders S, Giraud-Carrier C (2017) Informing the use of hyperparameter optimization through metalearning. In: 2017 IEEE International Conference on Data Mining (ICDM), pp 1051–1056. IEEE
Santoro A, Bartunov S, Botvinick M, Wierstra D, Lillicrap T (2016) Meta-learning with memory-augmented neural networks. In: International Conference on Machine Learning, pp. 1842–1850. PMLR
Segrera S, Pinho J, Moreno MN (2008) Information-theoretic measures for meta-learning. In: International Workshop on Hybrid Artificial Intelligence Systems, pp 458–465
Shah A, Ghahramani Z (2016) Pareto frontier learning with expensive correlated objectives. In: International Conference on Machine Learning, pp 1919–1927. PMLR
Shi Y, Eberhart RC (1998) Parameter selection in particle swarm optimization. In: International Conference on Evolutionary Programming, pp 591–600. Springer
Shi X, Mueller J, Erickson N, Li M, Smola A (2021) Multimodal automl on structured tables with text fields. In: 8th ICML Workshop on Automated Machine Learning (AutoML)
Sleeman DH, Rissakis M, Craw S, Graner N, Sharma S (1995) Consultant-2: pre- and post-processing of machine learning applications. Int J Hum-Comput Stud 43(1):43–63. https://doi.org/10.1006/ijhc.1995.1035
Smith-Miles K (2009) Cross-disciplinary perspectives on meta-learning for algorithm selection. ACM Comput Surv 41(1):6–1625. https://doi.org/10.1145/1456650.1456656
Snell J, Swersky K, Zemel RS (2017) Prototypical networks for few-shot learning. arXiv preprint arXiv:1703.05175
Snoek J, Larochelle H, Adams RP (2012) Practical bayesian optimization of machine learning algorithms. In: Advances in Neural Information Processing Systems 25: 26th Annual Conference on Neural Information Processing Systems 2012. Proceedings of a Meeting Held December 3–6, 2012, Lake Tahoe, Nevada, United States, pp 2960–2968. http://papers.nips.cc/paper/4522-practical-bayesian-optimization-of-machine-learning-algorithms
Soares C, Brazdil PB, Kuba P (2004) A meta-learning method to select the kernel width in support vector regression. Mach Learn 54(3):195–209
Soares C, Petrak J, Brazdil P (2001) Sampling-based relative landmarks: Systematically test-driving algorithms before choosing. In: Progress in Artificial Intelligence, Knowledge Extraction, Multi-agent Systems, Logic Programming and Constraint Solving, 10th Portuguese Conference on Artificial Intelligence, EPIA 2001, Porto, Portugal, December 17–20, 2001, Proceedings, pp 88–95. https://doi.org/10.1007/3-540-45329-6_12
Song Q, Wang G, Wang C (2012) Automatic recommendation of classification algorithms based on data set characteristics. Pattern Recogn 45(7):2672–2689
Souza F, Nogueira R, Lotufo R (2020) Bertimbau: pretrained bert models for Brazilian Portuguese. In: Brazilian Conference on Intelligent Systems, pp 403–417 . Springer
Stapleton F, Núñez DG, Sun Y, Galván E (2025) Surrogate-assisted evolution for efficient multi-branch connection design in deep neural networks. arXiv preprint arXiv:2506.20469
Sun Q, Liu Y, Chua T-S, Schiele B (2019) Meta-transfer learning for few-shot learning. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)
Swersky K, Snoek J, Adams RP (2013) Multi-task bayesian optimization. In: Advances in Neural Information Processing Systems 26: 27th Annual Conference on Neural Information Processing Systems 2013. Proceedings of a Meeting Held December 5–8, 2013, Lake Tahoe, Nevada, United States, pp 2004–2012. http://papers.nips.cc/paper/5086-multi-task-bayesian-optimization

Talbi E-G (2022) Automated design of deep neural networks: a survey and unified taxonomy. ACM Comput Surveys (CSUR) 54(2):1–37

Tang Z, Fang H, Zhou S, Yang T, Zhong Z, Hu T, Kirchhoff K, Karypis G (2024) Autogluon-multimodal (automm): Supercharging multimodal automl with foundation models. arXiv preprint arXiv:2404.16233

Taylor ME, Stone P (2009) Transfer learning for reinforcement learning domains: a survey. J Mach Learning Res 10(7):1–53

Thornton C, Hutter F, Hoos HH, Leyton-Brown K (2013) Auto-weka: combined selection and hyperparameter optimization of classification algorithms. In: The 19th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD 2013, Chicago, IL, USA, August 11–14, 2013, pp 847–855. https://doi.org/10.1145/2487575.2487629

Tian Y, Zhao X, Huang W (2022) Meta-learning approaches for learning-to-learn in deep learning: a survey. Neurocomputing 494:203–223

Tian S, Li L, Li W, Ran H, Ning X, Tiwari P (2024) A survey on few-shot class-incremental learning. Neural Netw 169:307–324

Ünal HT, Başçiftçi F (2022) Evolutionary design of neural network architectures: a review of three decades of research. Artif Intell Rev 55(3):1723–1802

Vanschoren J (2010) Understanding machine learning performance with experiment databases (het verwerven van inzichten in leerperformantie met experiment databanken); understanding machine learning performance with experiment databases. PhD thesis, Katholieke Universiteit Leuven, Belgium. https://lirias.kuleuven.be/handle/123456789/266060

Vanschoren J (2018) Meta-learning: a survey. arXiv preprint arXiv:1810.03548

Vettoruzzo A, Bouguelia M-R, Vanschoren J, Rögnvaldsson T, Santosh K (2024) Advances and challenges in meta-learning: a technical review. IEEE Trans Pattern Anal Mach Intell 46(7):4763–4779

Vilalta R, Drissi Y (2002) A perspective view and survey of meta-learning. Artif Intell Rev 18:77–95

Vilalta R, Drissi Y (2002) A characterization of difficult problems in classification. In: Wani MA, Arabnia HR, Cios KJ, Hafeez K, Kendall G (eds) Proceedings of the 2002 International Conference on Machine Learning and Applications - ICMLA 2002, June 24-27, 2002, Las Vegas, Nevada, USA, pp 133–138. CSREA Press, Las Vegas, Nevada

Wan A, Dai X, Zhang P, He Z, Tian Y, Xie S, Wu B, Yu M, Xu T, Chen K (2020) Fbnetv2: differentiable neural architecture search for spatial and channel dimensions. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 12965–12974

Wang Y, Yao Q, Kwok JT, Ni LM (2021) Generalizing from a few examples: a survey on few-shot learning. ACM Comput Surveys 53(3):1–34

Wang L, Zhang X, Su H, Zhu J (2024) A comprehensive survey of continual learning: theory, method and application. IEEE Transactions on Pattern Analysis and Machine Intelligence

Weber M, Auch M, Doblander C, Mandl P, Jacobsen H-A (2021) Transfer learning with time series data: a systematic mapping study. IEEE Access 9:165409–165432

Weiss K, Khoshgoftaar TM, Wang D (2016) A survey of transfer learning. J Big Data 3:1–40

Wever M, Tornede A, Mohr F, Hüllermeier E (2021) Automl for multi-label classification: overview and empirical evaluation. IEEE Trans Pattern Anal Mach Intell 43(9):3037–3054

Wistuba M, Rawat A, Pedapati T (2019) A survey on neural architecture search. arXiv preprint arXiv:1905.01392

Wistuba M, Schilling N, Schmidt-Thieme L (2015) Learning hyperparameter optimization initializations. In: 2015 IEEE International Conference on Data Science and Advanced Analytics, DSAA 2015, Campus des Cordeliers, Paris, France, October 19–21, 2015, pp 1–10. https://doi.org/10.1109/DSAA.2015.7344817

Witt C (2005) Worst-case and average-case approximations by simple randomized search heuristics. In: Annual Symposium on Theoretical Aspects of Computer Science, pp 44–56. Springer

Wu J, Chen X-Y, Zhang H, Xiong L-D, Lei H, Deng S-H (2019) Hyperparameter optimization for machine learning models based on Bayesian optimization. J Electr Sci Technol 17(1):26–40

Xie X, Sun F, Liu Z, Wu S, Gao J, Zhang J, Ding B, Cui B (2022) Contrastive learning for sequential recommendation. In: 2022 IEEE 38th International Conference on Data Engineering (ICDE), pp 1259–1273. IEEE

Xie L, Yuille A (2017) Genetic CNN. In: Proceedings of the IEEE International Conference on Computer Vision, pp 1379–1388

Yang L, Shami A (2020) On hyperparameter optimization of machine learning algorithms: theory and practice. Neurocomputing 415:295–316

Yang Q, Liu Y, Chen T, Tong Y (2019) Federated machine learning: concept and applications. ACM Trans Intell Syst Technol 10(2):1–19

Yao Q, Wang M, Chen Y, Dai W, Yi-Qi H, Yu-Feng L, Wei-Wei T, Qiang Y, Yang Y (2018) Taking human out of learning applications: a survey on automated machine learning. arXiv preprint arXiv:1810.13306

Zhang Z, Wang X, Zhu W (2021) Automated machine learning on graphs: a survey. arXiv preprint arXiv:2103.00742

Zhao Z, Alzubaidi L, Zhang J, Duan Y, Gu Y (2024) A comparison review of transfer learning and self-supervised learning: definitions, applications, advantages and limitations. Expert Syst Appl 242:122807

Zheng R, Qu L, Cui B, Shi Y, Yin H (2023) Automl for deep recommender systems: a survey. ACM Trans Information Syst 41(4):1–38

Zhong Z, Yang Z, Deng B, Yan J, Wu W, Shao J, Liu C-L (2020) Blockqnn: Efficient block-wise neural network architecture generation. IEEE Transactions on Pattern Analysis and Machine Intelligence

Zhuang F, Qi Z, Duan K, Xi D, Zhu Y, Zhu H, Xiong H, He Q (2021) A comprehensive survey on transfer learning. Proc IEEE 109(1):43–76

Zöller M-A, Huber MF (2021) Benchmark and survey of automated machine learning frameworks. J Artif Intell Res 70:409–472

Zoph B, Le QV (2016) Neural architecture search with reinforcement learning. arXiv preprint arXiv:1611.01578

Zoph B, Vasudevan V, Shlens J, Le QV (2018) Learning transferable architectures for scalable image recognition. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp 8697–8710