AutoML4Clust: Efficient AutoML for Clustering Analyses

Dennis Tschechlov, Manuel Fritz, Holger Schwarz
University of Stuttgart
Stuttgart, Germany
{dennis.tschechlov,manuel.fritz,holger.schwarz}@ipvs.uni-stuttgart.de

ABSTRACT

Data analysis is a highly iterative process. In order to achieve valuable analysis results, analysts typically execute many configurations, i.e., algorithms and their hyperparameter settings, based on their domain knowledge. While experienced analysts may be able to define small search spaces for promising configurations, especially novice analysts define large search spaces due to their lack of domain knowledge. In the worst case, they perform an exhaustive search throughout the whole search space, resulting in infeasible runtimes. Recent advances in the research area of AutoML address this challenge by supporting novice analysts in the combined algorithm selection and hyperparameter optimization (CASH) problem for supervised learning tasks. However, no such systems exist for unsupervised learning tasks, such as the prevalent task of clustering analysis. In this work, we present our novel AutoML4Clust approach, which efficiently supports novice analysts regarding CASH for clustering analyses. To the best of our knowledge, this is the first thoroughly elaborated approach in this area. Our comprehensive evaluation unveils that AutoML4Clust significantly outperforms several existing approaches, as it achieves considerable speedups for the CASH problem, while still achieving very valuable clustering results.

1 INTRODUCTION

Data analysis is a crucial discipline to extract knowledge and insights from data. Therefore, analysts apply data mining techniques, typically machine learning algorithms, to extract patterns from data and to gain insights about data. A fundamental primitive in data mining is clustering analysis, which is an unsupervised machine learning task being used in various application domains, e.g., computer vision, document clustering, for business purposes, or to study genome data in biology [11].

Throughout these manifold fields of application domains, analysts typically struggle with the selection of a promising clustering configuration, i.e., a clustering algorithm and its corresponding hyperparameter settings, that achieves valuable clustering results. Hence, analysts typically define a configuration space, i.e., a search space of clustering algorithms and their hyperparameter settings, in which they expect promising configurations. Yet, novice analysts lack in-depth domain knowledge and hence define very large configuration spaces. In the worst case, novice analysts cannot limit configuration spaces at all and perform an exhaustive search throughout all possible configurations. Since this exhaustive search is very time-consuming, novice analysts often explore only a few configurations, e.g., randomly selected, from the configuration space, which often leads to solely moderate results. Hence, novice analysts require support to achieve valuable clustering results in a reasonable amount of time.

For supervised learning tasks, recent advances in the research area of AutoML are able to support novice analysts in the combined algorithm selection and hyperparameter optimization (CASH) problem in an automated and efficient manner [5, 18]. These approaches greedily explore large configuration spaces by trading off exploration and exploitation strategies, thus avoiding a time-consuming or even infeasible exhaustive search.

In this work, we propose AutoML4Clust, an efficient AutoML approach to support novice analysts in the CASH problem for clustering analyses. To the best of our knowledge, this is the first thoroughly elaborated AutoML approach for efficient clustering analyses, capable of automatically selecting promising clustering algorithms and their hyperparameters in combination.

Our contributions include the following:

- We introduce AutoML4Clust, our novel approach to efficiently support novice analysts in the prevalent CASH problem for clustering analyses.
- We reveal that AutoML4Clust is generic, i.e., it can be instantiated with different AutoML concepts, clustering algorithms and clustering metrics.
- In our evaluation, we unveil that AutoML4Clust significantly outperforms several existing approaches, as it achieves speedups of up to 437x for the CASH problem, while still achieving valuable clustering results. Hence, AutoML4Clust efficiently supports novice analysts in the CASH problem for clustering analyses.

The remainder of this paper is structured as follows: We present related work in this area in Section 2. In Section 3, we present AutoML4Clust, our novel AutoML approach for clustering analyses. In Section 4, we unveil the results of our evaluation. Finally, we conclude this work in Section 5.

2 RELATED WORK

Based on the generally accepted separation of machine learning tasks, we separate related work into support for the CASH problem regarding supervised and unsupervised learning tasks. We distinguish two important groups of related work to support analysts regarding the CASH problem: (1) AutoML systems for supervised learning, and (2) methods for unsupervised clustering analyses that either explore the algorithm selection or the hyperparameter optimization.

We define a configuration c as the combination of an algorithm $a \in \mathcal{A}$ and its hyperparameters $h \in \mathcal{H}$. Hence, we define the configuration space as $\mathcal{CS} = \mathcal{A} \times \mathcal{H}$. In the following, we investigate related work based on the machine learning task and its ability to explore $\mathcal{CS}$.

2.1 AutoML Systems for Supervised Learning

AutoML systems arose in the area of supervised machine learning to support novice analysts in the combined algorithm selection and hyperparameter optimization problem [5, 18]. As class labels are already available in the datasets, $\mathcal{CS} = \mathcal{A} \times \mathcal{H}$ can be explored automatically. The common underlying procedure of existing

AutoML systems is as follows: Given a budget, training data and an optimization metric, they execute and evaluate different configurations from $\mathcal{CS}$, i.e., they execute classification algorithms with the specified hyperparameter settings, and return the configuration that yields the best value for the optimization metric. In order to steer the exploration towards valuable results, several hyperparameter optimization techniques are proposed, such as Bayes [1], Hyperband [13], or BOHB [3]. These optimization techniques proceed in a greedy manner, since they define a specific trade-off between exploration, i.e., exploring new regions in the $\mathcal{CS}$, and exploitation, i.e., exploiting regions in the $\mathcal{CS}$, where already executed configurations performed well.

2.2 Algorithm Selection and Hyperparameter Optimization for Clustering Analyses

Related work that supports novice analysts regarding CASH for clustering analyses can be divided into methods that consider algorithm selection and methods that consider hyperparameter optimization.

Algorithm Selection. For unsupervised learning tasks, several approaches were developed to support novice analysts with the selection of a promising clustering algorithm, i.e., $\mathcal{CS} = \mathcal{A}$ for certain problems [4, 16]. These methods are based on meta-learning, i.e., they learn from past experiences in order to select the most promising algorithm on a previously unseen dataset.

Existing approaches differ in the used (a) implementation of the meta-learning steps, (b) clustering algorithms, and (c) metrics for evaluating the clustering results. However, these approaches solely focus on the clustering algorithm, yet completely ignore the corresponding hyperparameters $\mathcal{H}$.

Hyperparameter Optimization. Regarding the hyperparameter optimization, i.e., $\mathcal{CS} = \mathcal{H}$ of clustering algorithms, two types of approaches exist [7]: While exhaustive methods execute all configurations from $\mathcal{CS}$, non-exhaustive methods only execute some configurations. However, non-exhaustive methods are designed to optimize the hyperparameters of specific algorithms, e.g., k-Means. Exhaustive methods could also be applied for $\mathcal{CS} = \mathcal{A} \times \mathcal{H}$, though this results in tremendous runtime as the whole configuration space has to be explored.

Summary. Summarizing related work, existing AutoML systems only focus on supervised learning algorithms, yet can explore valuable results, where $\mathcal{CS} = \mathcal{A} \times \mathcal{H}$. For clustering analyses, there are approaches that either conduct an exploration for valuable clustering algorithms ($\mathcal{CS} = \mathcal{A}$) or their hyperparameters ($\mathcal{CS} = \mathcal{H}$). However, they do not address the combination of both, i.e., the CASH problem for clustering analyses, where $\mathcal{CS} = \mathcal{A} \times \mathcal{H}$, which is a crucial problem for novice analysts. We identified only some experimental implementations[1] [2] that address this problem, however they use (a) one specific optimization technique, or (b) one specific clustering metric, without explaining, evaluating or justifying their choice regarding (a) and (b). In addition, they miss a clear scientific elaboration and a systematic evaluation in comparison to existing approaches in this area.

3 AUTOML4CLUST

In this section, we introduce our generic AutoML4Clust approach to support novice analysts regarding the CASH problem for clustering analyses, where we apply concepts from existing supervised AutoML systems on clustering. Existing AutoML systems solely focus on supervised learning tasks, whereas we focus on clustering analysis, which is an unsupervised learning task. The key difference between supervised and unsupervised learning tasks is that the input datasets for unsupervised learning tasks do not contain ground-truth labels. Therefore, it is not possible to evaluate the result based on an external metric. Consequently, existing AutoML systems and their components cannot be applied per se for clustering analyses.

Figure 1 presents the procedure of our AutoML4Clust approach. Similar to supervised AutoML systems, it draws on a configuration space $\mathcal{CS}$, which defines the set of configurations that can be selected, executed and evaluated during the optimizer loop, which is at the core of existing hyperparameter optimization techniques. To this end, we rely on the configuration space $\mathcal{CS} = \mathcal{A} \times \mathcal{H}$. When considering different families of clustering algorithms, e.g., k-center and density-based ones, $\mathcal{CS}$ has to be defined in a hierarchical way, i.e., by defining the algorithm as conditional root-level hyperparameter [18]. Our AutoML4Clust procedure is structured into three parts (cf. Figure 1): The inputs, the optimizer loop, and return best configuration. In the following, we present these three parts in detail and subsequently discuss the benefits of AutoML4Clust.

3.1 Inputs

AutoML4Clust requires three inputs prior to execution. These are a dataset $\mathcal{D}$, an internal metric $\mathcal{M}$, and a budget l. Here, $\mathcal{D}$ does not contain any additional information, e.g., class labels. Hence, $\mathcal{M}$ is an internal metric that evaluates the internal structure of a clustering result. In the literature, many different internal metrics with different objectives are proposed [11, 14]. Most of them measure the compactness and the separation of clusters in different variations. Subsequently, they consider a quotient of both. The budget l defines the resources that can be used by the system. A common choice for the budget is a time constraint to limit the runtime or the number of configurations to execute. In this work, we use the number of optimizer loops that are performed as budget. However, we note that choosing an appropriate kind of budget and also an appropriate value for the budget is a difficult task. A too large value may lead to a long runtime, whereas a too small one can lead to solely moderate results.

3.2 Optimizer Loop

In the optimizer loop, an optimizer such as Random [1], Bayes [18], Hyperband [13], or BOHB [3] is used to find well-performing configurations efficiently. Here, the optimizer performs l loops, where each loop consists of three steps:

i) *Selection:* $c \in \mathcal{CS}$, where the optimizer selects a configuration c from the configuration space $\mathcal{CS}$. The different aforementioned optimizers mostly differ in their greedy procedure, i.e., trading off exploration and exploitation, in order to select a configuration $c \in CS$ in each optimizer loop l_i. Yet, all optimizers require the definition of a black-box function $f : \mathcal{CS} \rightarrow \mathbb{R}$, which is subject to optimization. This function f assigns each configuration $c \in \mathcal{CS}$ a metric value and is implemented with the following steps ii) and iii).

ii) *Execution:* $\mathcal{R} \leftarrow c(\mathcal{D})$. Here, the previously selected configuration c is executed on $\mathcal{D}$. The result of the execution is $\mathcal{R}$, which can be any kind of clustering result, e.g., the resulting labels or the final centroids.

[1] https://git.io/JUNKu [2] https://git.io/JUNKz

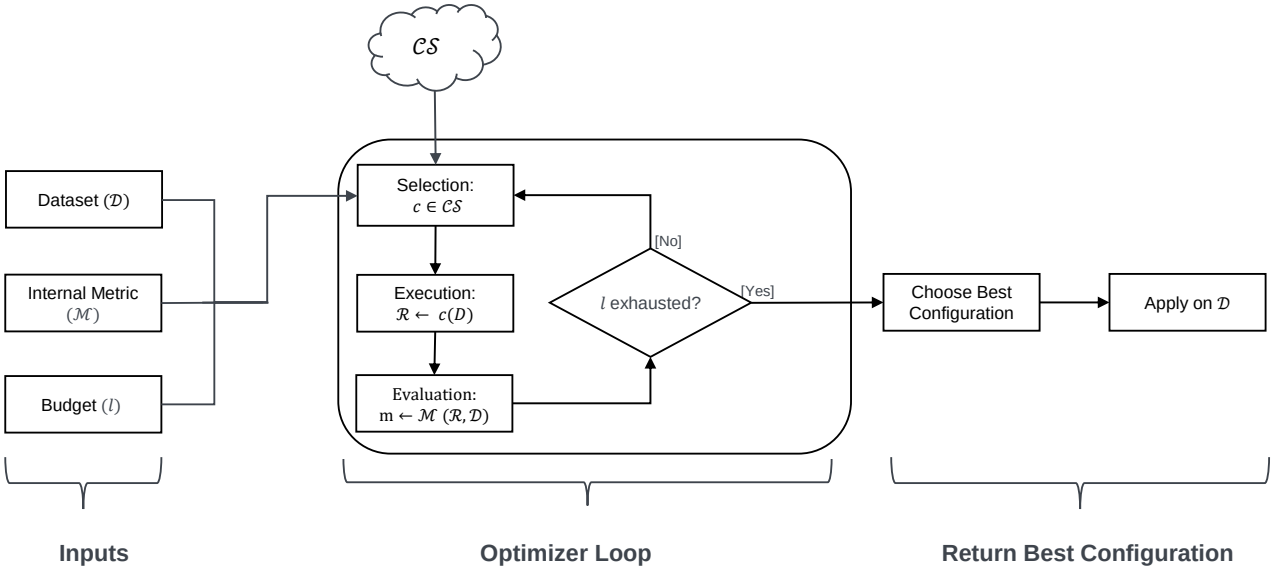

Figure 1: Procedure of AutoML4Clust.

iii) *Evaluation:* $m \leftarrow \mathcal{M}(\mathcal{R}, \mathcal{D})$, where the clustering result is evaluated. This means that the metric $\mathcal{M}$ is calculated based on the clustering result $\mathcal{R}$ and on the dataset $\mathcal{D}$.

3.3 Return Best Configuration

In the third step, the best configuration from all considered configurations is chosen. This is the configuration that achieves the best metric value regarding $\mathcal{M}$ from all configurations that are selected, executed and evaluated during each optimizer loop. This configuration is applied on the dataset $\mathcal{D}$ to finally obtain the best clustering result that is found by AutoML4Clust.

3.4 Discussion

Existing AutoML systems solely focus on supervised learning tasks in order to achieve valuable results, where $\mathcal{CS} = \mathcal{A} \times \mathcal{H}$, i.e., $\mathcal{CS}$ is typically very large. In contrast, AutoML4Clust addresses this problem for the prevalent unsupervised task of clustering analyses, where ground-truth labels are missing. Therefore, especially novice analysts are supported, which can neither limit $\mathcal{A}$ nor $\mathcal{H}$ to a manageable size and thus perform in the worst case an exhaustive exploration throughout $\mathcal{CS}$. Our proposed procedure draws on the latest fundamental concepts of existing AutoML systems, yet remains generic regarding the used clustering algorithms, metrics, and optimizers.

Regarding possible clustering algorithms, AutoML4Clust can use any kind of clustering algorithm by defining $\mathcal{CS}$ in a hierarchical way, similar to existing AutoML systems for supervised learning tasks. Regarding the metrics, AutoML4Clust draws on an internal metrics to asses the internal structure of a clustering result. Furthermore, several optimizers can be used, which follow the three steps (1) select a configuration, (2) execute the configuration, and (3) evaluate the result of the configuration.

Especially the combination of an internal metric and the used optimizer is of paramount importance: Since optimizers proceed in a greedy manner by defining a trade-off between exploration and exploitation throughout $\mathcal{CS}$, it is per se not clear which internal metric supports their behavior best. Yet, as prior work in the area of supervised learning has shown, these optimizers are able to efficiently explore large configuration spaces, while still achieving valuable results [3, 5, 18]. Therefore, we assume that AutoML4Clust similarly benefits from these optimizers. That is, we argue that AutoML4Clust is able to efficiently achieve valuable results within a predefined budget l. However, it is a very challenging task for novice analysts to specify such a suitable budget, since a too small budget leads to imprecise results, whereas a too large budget leads to long runtimes. Furthermore, it is not clear at all, if the used metric supports the greedy behavior of the optimizers within a reasonable budget.

4 EVALUATION

Since analysts are interested in fast and valuable results, the question remains how well different instantiations of our approach, i.e., combinations of optimizers and metrics, perform in order to achieve this goal. To this end, we compare in our evaluation how our novel AutoML4Clust approach performs (i) with different instantiations of optimizers and metrics for clustering analyses, and (ii) in contrast to existing approaches in this area. We first discuss the setup of our evaluation, before we investigate the accuracy of the results of different instantiations of AutoML4Clust in contrast to existing approaches. Subsequently, we analyze the runtime of AutoML4Clust in contrast to existing approaches. Finally, we show the practical feasibility of AutoML4Clust on real-world datasets regarding accuracy and runtime.

4.1 Setup

In the following, we describe the setup of our experiments. We focus on (i) the used hard- and software, (ii) the synthetic and real-world datasets that we use, (iii) the implementation details, and (iv) the performed experiment with its baselines.

Dataset	n	d	k_{act}
Statlog (Landsat Satellite)	6,435	35	7
ISOLET	7,797	617	26
Motion Capture Hand Postures	7,805	34	5
Avila	10,430	10	12
Pen-Based Recognition of Handwritten Digits	10,992	16	10

Table 1: Real-world datasets with their corresponding dataset characteristics n, d and k_{act}.

4.1.1 Hard- and Software. The experiments are performed on a virtual machine, which operates on Ubuntu 18.04. It has a 6-core CPU with 2.5 GHz and 32 GB RAM. Our implementation is based on Python 3.6 and on scikit-learn[3].

4.1.2 Datasets. We draw our evaluation on synthetically generated and real-world datasets. Regarding the synthetic datasets, we use the dataset generation tool from [7–9]. This tool generates datasets based on these four input characteristics:

1) n, which describes the number of entities, 2) d, which denotes the number of dimensions, where the values in each dimension lie in the interval $[-10, 10]$, 3) k_{act}, which is the actual number of clusters, where each cluster contains $\frac{n}{k_{act}}$ entities and 4) r, which is the ratio of noise, i.e., $\frac{r}{100} \cdot n$ additional entities are added uniformly at random to the dataset.

We generate datasets with $n \in [2{,}500; 7{,}500]$, $d \in [20; 40]$, $k_{act} \in [25; 75]$, and $r \in [0; 17; 50]$. We generate these datasets as a cross product of the above-mentioned characteristics, i.e., 24 synthetic datasets are used within our evaluation.

For the real-world datasets, we use 5 classification datasets from the UCI machine learning repository[4] with different dataset characteristics regarding n, d and k_{act}. Here, k_{act} describes the number of classes in the dataset. We removed the class labels from these datasets when applying instantiations of our AutoML4Clust approach and solely used them to evaluate the accuracy of our approach. In order to use these datasets for clustering, we removed any non-numeric and symbolic values, IDs, timestamps, class labels and empty values. Table 1 summarizes the datasets' characteristics. Note, that these datasets exhibit similar or even larger characteristics as the synthetic datasets regarding n and d.

4.1.3 Implementation. We evaluate AutoML4Clust with overall 12 instantiations, i.e., four optimizers and three internal metrics to unveil the best-performing combinations thereof. We provide our prototypical implementation of all AutoML4Clust instantiations in Python[5] with all versions of the used libraries[6].

Optimizers: We use the following four frequently used optimizers from the area of hyperparameter optimization (cf. Section 2.1): Random Search (RS) [1], Bayesian Optimization (BO) [1], Hyperband (HB) [13], and the combination of Bayesian Optimization and Hyperband (BOHB) [3]. We define the budget as number of optimizer loops that each optimizer performs.

Clustering algorithms: We focus on k-center clustering algorithms, due to their appealing runtime behavior and their popularity across researchers and practitioners [20]. We note that other kind of clustering algorithms, e.g., density-based ones like DBSCAN, have a runtime complexity of $O(n^2)$ or even higher, which makes them infeasible in practice for large datasets [10]. Therefore, we use k-Means [15], MiniBatch k-Means [17], k-Medoids [12], and GMM [2] as concrete instantiations of k-center clustering algorithms. We set the maximum number of clustering iterations for each algorithm to ten since Fritz et al. showed that even a few iterations already lead to valuable results [6].

Internal metrics: For the evaluation of the clustering result in each optimizer loop, we use three commonly used internal metrics that are implemented in scikit-learn: Calinski-Harabasz (CH), the Davies-Bouldin Index (DBI), and the Silhouettte (SIL).

4.1.4 CASH Experiment and Baselines. Based on $\mathcal{CS} = \mathcal{A} \times \mathcal{H}$, we define the CASH experiment analogue to related work in the area of AutoML [18]. We set $\mathcal{A}$ as described in Section 4.1.3. We set the search space $\mathcal{H}$ for the hyperparameter k of k-center clustering algorithms to $\mathcal{H} = \{2, \ldots, \frac{n}{10}\}$, i.e., the maximum k value is set in relation to the number of entities in the dataset. Since analysts perform in the worst-case an exhaustive search throughout $\mathcal{CS}$ due to the lack of more efficient approaches, we compare AutoML4Clust to an exhaustive search.

4.2 Accuracy Evaluation

In this section, we investigate the accuracy obtained by different instantiations of AutoML4Clust in contrast to the baselines. Therefore, we explain how we (i) examine a suitable budget to achieve valuable clustering results, (ii) compare the accuracy with the baselines, and (iii) discuss the effect of noisy data.

Since the actual labels of the datasets are known in our experiments, we use an external clustering metric to asses the accuracy of the achieved clustering result, similar to the accuracy from classification tasks. Therefore, we use the adjusted mutual information (AMI) [19], which is limited to $[0; 1]$, while values closer to one indicate a better matching of the predicted labeling with the actual labeling of the dataset.

4.2.1 Time to Accuracy. Figure 2 summarizes the accuracy results of the AutoML4Clust instantiations, i.e., the four optimizers and the three internal metrics at each optimizer loop l_i.

Budget: After about 60 optimizer loops, i.e., $l_i = 60$ (which is marked by the vertical line), the accuracy of AutoML4Clust does not further improve significantly for all instantiations. Hence, we argue that $l = 60$ is a suitable budget for AutoML4Clust to support novice analyst in achieving valuable results efficiently.

AutoML4Clust accuracy: AutoML4Clust achieves with every optimizer very accurate results, i.e., AMI values over 90%. Furthermore, we observe that more optimizer loops increase the accuracy.

AutoML4Clust instantiations: The AutoML4Clust instantiations with the CH metric achieve the highest accuracy. The reason for this is that the CH metric focuses on the intra- and inter-cluster compactness. Therefore, in contrast to DBI, it is less sensitive to sub-clusters, i.e., two or more clusters in a dataset that are very close to each other [14]. The most inaccurate results are obtained by instantiations with the SIL metric. This can be explained by the calculation of the SIL, as it can be highly influenced by the position of single entities. Due to its imprecise results, we do not present results for the SIL metric in the remaining experiments.

4.2.2 Accuracy Comparison. Figure 3 unveils the accuracy of AutoML4Clust in comparison to the respective baselines. We present the results of the AutoML4Clust instantiations with the four optimizers, the budget of $l = 60$, and the CH and DBI metrics.

[3] scikit-learn.org
[4] https://archive.ics.uci.edu/ml/datasets.php
[5] https://git.io/JTeix
[6] https://git.io/JTeXG

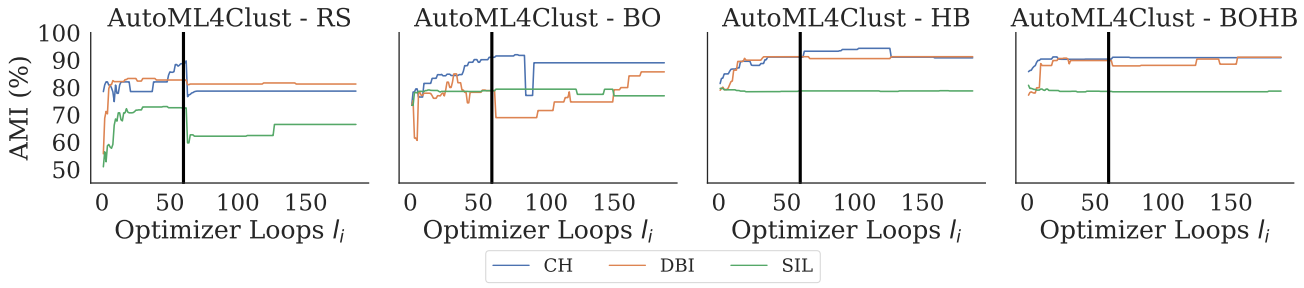

Figure 2: Accuracy of the AutoML4Clust instantiations over all synthetic datasets at each optimizer loop l_i for the CASH experiment. The vertical line at $l_i = 60$ marks where the accuracy does not further improve significantly.

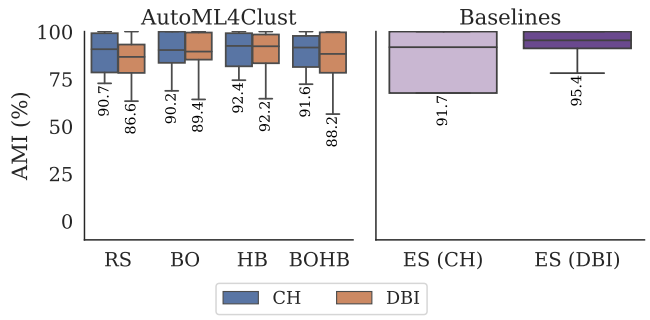

Figure 3: Accuracy of AutoML4Clust over synthetic datasets in contrast to exhaustive search (ES) with CH and DBI. Median values are shown at each box plot.

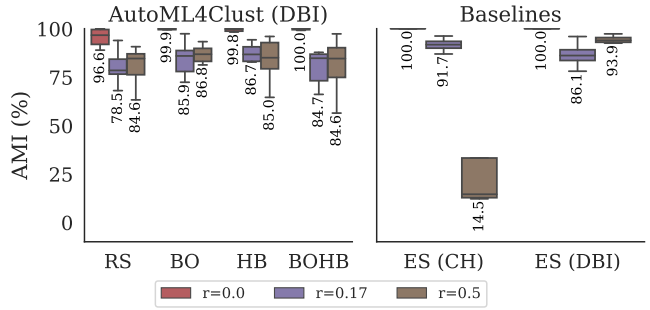

Figure 4: Comparison of the impact of noise (r) for AutoML4Clust and the baselines regarding the AMI results. Median values are shown at each box plot.

AutoML4Clust achieves similarly accurate results as the exhaustive search (ES), i.e., HB with CH achieve higher accuracy than ES (CH) and only deviates 3% from ES (DBI). Hence, the best result of AutoML4Clust deviates only 3% from the best result of ES. Therefore, AutoML4Clust supports novice analysts nearly as good as an exhaustive search, while being more efficient since it does not execute all configurations in $\mathcal{CS}$. Furthermore, we emphasize that AutoML4Clust achieves higher accuracy when using the CH metric. Regarding optimizers, we observe that AutoML4Clust achieves in most cases the best results with the HB optimizer and the CH metric. One possible reason is that BO and BOHB are more effective in higher dimensional configuration spaces. Yet, it achieves higher accuracy than RS since it discards poorly performing configurations early on [3].

4.2.3 Effect of Noisy Data. Throughout our experiments, we observed that noisy data have a significant impact on the baselines. However, noisy data are prevalent in real-world scenarios and should therefore be considered specifically. Figure 4 unveils the results for three different noise ratios $r \in [0; 0.17; 0.5]$. It can be seen that AutoML4Clust is robust against noise, i.e., it achieves AMI values typically over 85% even for $r = 0.5$ and can therefore almost compete with a time-consuming exhaustive search, which achieves an AMI value of 93.9% for DBI. The ES (CH) achieves accurate results for $r = 0$ and $r = 0.17$, while it performs poorly for $r = 0.5$. Hence, it is most affected by noise for $r = 0.5$. We observe a similar behaviour for the AutoML4Clust instantiations with the CH metric. The reason for the bad performance is that the calculation of the CH metric is essentially based on the compactness. Therefore, it is less robust against noise than the DBI metric [14]. However, we note that the results of AutoML4Clust with the CH metric (cf. Section 4.2.2) are still very accurate for $r \in [0; 0.17]$ and are only imprecise for $r = 0.5$, i.e., for highly noisy datasets.

4.3 Runtime Evaluation

Besides an accurate clustering result, the runtime is also crucial for analysts. Figure 5 summarizes the runtime results for all investigated AutoML4Clust instantiations for $l = 60$ and the corresponding baselines.

AutoML4Clust exhibits the highest runtimes with the SIL metric, since this metric has a runtime complexity of $O(n^2)$ [14]. The runtimes of the CH and DBI metrics differ only marginally, while the CH metric has the lower runtime in most cases. Regarding the optimizers, AutoML4Clust achieves faster results with the HB and BOHB optimizers than with the RS and BO optimizers. The reason is that HB and BOHB execute optimizer loops in parallel, while RS and BO execute them sequentially [3].

The results clearly show that AutoML4Clust is orders of magnitude faster than the time-consuming ES. It achieves the fastest results in 57 seconds, while the fastest results for the ES require roughly 6 hours. In comparison to ES (DBI), we even observe speedups of more than 437×. Hence, AutoML4Clust provides an efficient support for novice analysts regarding the CASH problem for clustering analyses.

4.4 Results on Real-World Datasets

In order to assess the practical feasibility of AutoML4Clust, we perform the same experiments as for the synthetic datasets, yet use real-world datasets. We focus on the CH and the DBI metric for these experiments, since the results on the synthetic datasets clearly showed that the SIL metric does not achieve valuable

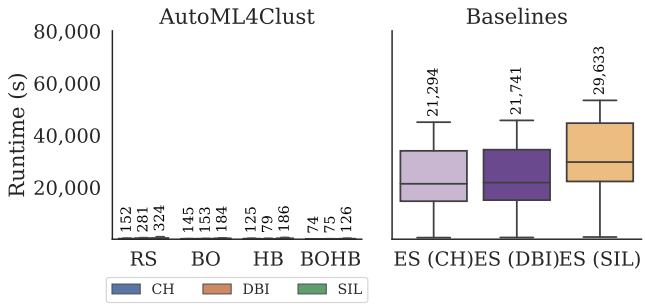

Figure 5: Runtime results of AutoML4Clust with all instantiations in contrast to the respective baselines. Median values are shown at each box plot.

Approach	Optimizer	Metric	AMI (%)	Runtime (s)
AutoML4Clust	RS	CH	**45.4**	781
		DBI	28.5	791
	BO	CH	22.5	879
		DBI	33.6	1,305
	HB	CH	**38.5**	**420**
		DBI	**33.9**	604
	BOHB	CH	22.0	**276**
		DBI	33.7	**598**
Baseline	ES	CH	13.8	76,211
	ES	DBI	33.5	77,196

Table 2: Median results on real-world datasets regarding AMI and the runtimes. We indicate the top-3 results for AMI and runtime in bold.

results and furthermore exhibits high runtimes. Table 2 summarizes the results on the real-world datasets, which are mostly very similar to the results on the synthetic datasets. We indicate the top three results for AMI and runtime in bold.

AutoML4Clust achieves higher accuracy than an exhaustive search, while also significantly outperforming it regarding runtime. The fastest results of AutoML4Clust requires less than 5 minutes, while the ES required with both metrics roughly 21 hours, i.e., AutoML4Clust achieves speedups of up to 276×. Furthermore, AutoML4Clust achieves with HB and CH up to 5% higher AMI values than the ES. The reason that AutoML4Clust can be more accurate than ES is that both optimize the internal metric value of CH or DBI and do not directly address the external metric AMI. Therefore, both approaches return the configuration with the best metric value, but not necessarily with the best accuracy, i.e., AMI value. We argue that HB and CH is a well-performing instantiation of AutoML4Clust, since it is the only instantiation that achieves one of the top-3 results with both, AMI and runtime.

Combining these observations with the results from synthetic datasets, we can state that the instantiation of the HB optimizer and the CH metric achieves in most cases the best results regarding accuracy and runtime.

5 CONCLUSION

In this work, we propose AutoML4Clust, an AutoML approach to support novice analysts efficiently with the combined algorithm selection and hyperparameter optimization (CASH) problem for clustering analyses. To the best of our knowledge, this is the first thoroughly elaborated AutoML approach for clustering analyses. AutoML4Clust remains generic, i.e., it can be instantiated with different optimizers and internal metrics. However, the concrete instantiation is crucial for an efficient exploration of large configuration spaces. Our evaluation reveals that specific instantiations of AutoML4Clust achieve similar or even more accurate results, while tremendously outperforming existing approaches regarding runtime on synthetic and real-world datasets.

Future work will address how clustering ensembles can be exploited to achieve even more valuable clustering results.

REFERENCES

[1] James Bergstra and Yoshua Bengio. 2012. Random Search for Hyper-parameter Optimization. *Journal of Machine Learning Research* 13, 1 (2012).

[2] Christopher M. Bishop. 2006. *Pattern Recognition and Machine Learning (Information Science and Statistics).* Springer-Verlag, Berlin, Heidelberg.

[3] Stefan Falkner, Aaron Klein, and Frank Hutter. 2018. BOHB: Robust and Efficient Hyperparameter Optimization at Scale. In *Proceedings of the 35th International Conference on Machine Learning.*

[4] Daniel G Ferrari and Leandro Nune de Castro. 2015. Clustering algorithm selection by meta-learning systems: A new distance-based problem characterization and ranking combination methods. *Information Sciences* (2015).

[5] Matthias Feurer, Aaron Klein, Katharina Eggensperger, Jost Springenberg, Manuel Blum, and Frank Hutter. 2015. Efficient and robust automated machine learning. In *Advances in neural information processing systems.*

[6] Manuel Fritz, Michael Behringer, and Holger Schwarz. 2019. Quality-driven early stopping for explorative cluster analysis for big data. *SICS Software-Intensive Cyber-Physical Systems* (2019).

[7] Manuel Fritz, Michael Behringer, and Holger Schwarz. 2020. LOG-Means: Efficiently Estimating the Number of Clusters in Large Datasets. *Proc. VLDB Endow.* 13, 12 (July 2020).

[8] Manuel Fritz and Holger Schwarz. 2019. Initializing k-Means Efficiently: Benefits for Exploratory Cluster Analysis. In *On the Move to Meaningful Internet Systems: OTM 2019 Conferences.*

[9] Manuel Fritz, Dennis Tschechlov, and Holger Schwarz. 2020. Learning from Past Observations: Meta-Learning for Efficient Clustering Analyses. In *Big Data Analytics and Knowledge Discovery*, Min Song, Il-Yeol Song, Gabriele Kotsis, A Min Tjoa, and Ismail Khalil (Eds.). Springer International Publishing, Cham, 364–379.

[10] Junhao Gan and Yufei Tao. 2015. DBSCAN revisited: Mis-claim, un-fixability, and approximation. In *Proceedings of the 2015 ACM SIGMOD international conference on management of data.*

[11] Anil K Jain. 2010. Data clustering: 50 years beyond K-means. *Pattern recognition letters* 8 (2010).

[12] L. Kaufman and P.J. Rousseeuw. 1987. Clustering by means of Medoids. In *Statistical Data Analysis Based on the L1–Norm and Related Methods.*

[13] Lisha Li, Kevin Jamieson, Giulia DeSalvo, Afshin Rostamizadeh, and Ameet Talwalkar. 2017. Hyperband: a novel bandit-based approach to hyperparameter optimization. *The Journal of Machine Learning Research* 1 (2017).

[14] Yanchi Liu, Zhongmou Li, Hui Xiong, Xuedong Gao, Junjie Wu, and Sen Wu. 2013. Understanding and enhancement of internal clustering validation measures. *IEEE Transactions on Cybernetics* 3 (6 2013).

[15] J MacQueen. 1967. Some methods for classification and analysis of multivariate observations. In *Proceedings of the fifth Berkeley Symposium on Mathematical Statistics and Probability.*

[16] Bruno Almeida Pimentel and André C.P.L.F. de Carvalho. 2019. A new data characterization for selecting clustering algorithms using meta-learning. *Information Sciences* (3 2019).

[17] D. Sculley. 2010. Web-scale k-means clustering. In *Proceedings of the 19th International Conference on World Wide Web, WWW '10.*

[18] Chris Thornton, Frank Hutter, Holger H. Hoos, and Kevin Leyton-Brown. 2013. Auto-WEKA: combined selection and hyperparameter optimization of classification algorithms. In *Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining - KDD '13.* ACM Press.

[19] Nguyen Xuan Vinh, Julien Epps, and James Bailey. 2010. Information theoretic measures for clusterings comparison: Variants, properties, normalization and correction for chance. *Journal of Machine Learning Research* (2010).

[20] Xindong Wu, Vipin Kumar, J Ross Quinlan, Joydeep Ghosh, Qiang Yang, Hiroshi Motoda, Geoffrey J McLachlan, Angus Ng, Bing Liu, S Yu Philip, et al. 2008. Top 10 algorithms in data mining. *Knowledge and information systems* 14, 1 (2008).