AutoCL: AutoML for Concept Learning

Jiayi Li[1][0009−0001−9475−8159], Sheetal Satheesh[1][0009−0009−0385−5825], Stefan Heindorf[1][0000−0002−4525−6865], Diego Moussallem[1][0000−0003−3757−2013], René Speck[1][0009−0001−0151−5538], and Axel-Cyrille Ngonga Ngomo[1][0000−0001−7112−3516]

Paderborn University, Paderborn, Germany
{jiayili, sheetal}@mail.uni-paderborn.de
{heindorf, diego.moussallem, rene.speck, axel.ngonga}@uni-paderborn.de

Abstract. Node classification in knowledge graphs aids in the discovery of new drugs, the identification of risky users in social networks, and the completion of missing type information in knowledge graphs. A crucial requirement for many stakeholders is to understand the models' predictions in these applications. To this end, concept learners have been proposed to learn concepts in description logic from positive and negative nodes in knowledge graphs in an interpretable way. However, learning concepts on large datasets is computationally expensive. Manually selecting important subgraphs and features to reduce runtime is tedious. While many feature selection approaches have been proposed to simplify the process for tabular data, they are not directly applicable to knowledge graphs and concept learning. In addition, current concept learners have a large number of hyperparameters that need to be optimized to achieve high predictive performance. In this paper, we propose AutoCL—an AutoML approach that is tailored to concept learning. AutoCL comprises methods for automatic feature selection and hyperparameter optimization for concept learners. We demonstrate its effectiveness with SML-Bench, a benchmarking framework for structured machine learning. Our approach leads to better predictive performance across concept learners, in terms of F_1-measure and *Accuracy* while reducing runtime on 6 of 8 datasets from SML-Bench.

Keywords: Knowledge bases; Concept learning; Automated machine learning; Feature selection; Hyperparameter optimization

1 Introduction

Artificial intelligence (AI) has made remarkable progress in recent years, with deep learning standing out as a prominent subfield. Deep learning models contain a series of linear and non-linear transformations that often comprise millions of parameters exceeding human comprehension capabilities. Due to their opaque decision-making processes, they are often referred to as "black-box" models. This lack of transparency poses significant challenges, particularly in critical domains where understanding the decision-making process is important.

In response, the field of explainable artificial intelligence (XAI) has emerged [42, 51]. XAI aims to explain the outcomes and inner workings of such models in a way that is understandable to humans. One promising avenue in this direction is the integration of knowledge graphs (KGs) [19]. KGs offer a structured representation of domain knowledge, providing clear semantics for entities and relations. By infusing deep learning models with domain knowledge from KGs via neuro-symbolic procedures, their functioning and outcomes might be explained.

Moreover, concepts in description logics (DLs) have been proposed as interpretable "white box" models. Given positive and negative examples of individuals in a knowledge base, the task of concept learning (CL) [17, 24, 34, 36] is to learn a concept such that many of the positive examples are entailed and many of the negatives are not. The learned concepts in turn can be applied to new unseen data to make new predictions. Concepts in DLs directly correspond to class expressions in the W3C Web Ontology Language (OWL) designed to represent rich and complex knowledge about things. OWL allows expressive concepts including logical operators, such as conjunctions, disjunctions, and negations, as well as data and cardinality restrictions [24].

Real-world KGs like DBpedia, Wikidata, and YAGO have numerous relations, but many of these relations are irrelevant to specific learning problems. For instance, the "author" relation is meant for books and publications, the "shares border with" relation is intended for countries, but both relations are hardly relevant for predicting family relations of individuals. Such superfluous relations can unnecessarily prolong the runtime of concept learners and if irrelevant features are picked up, this can jeopardize the interpretability of the results and consequently the generalizability to new unseen data [50]. Although many feature selection methods have been proposed and shown to enhance model interpretability [12, 13, 25, 21, 41], most of the approaches are tailored to tabular data and, to a lesser extent, to image and text data, but there is a lack of approaches specifically tailored to concept learning.

Moreover, contemporary concept learners like CELOE, OCEL [36], and EvoLearner [24] are equipped with a multitude of hyperparameters (HPs). To obtain the best predictive performance of the concept learners, it is necessary to carefully tune the HPs, called Hyperparameter Optimization (HPO).

In this paper, we propose AutoCL—an AutoML approach including FS and HPO for concept learning. We evaluate AutoCL on 8 datasets from the SML-Bench framework [55]. Our results show that AutoCL significantly improves upon the state-of-the-art concept learners, especially on large learning problems. We observe enhancements in both runtime and predictive performance.

To the best of our knowledge, no similar methods for FS and HPO for concept learning have been implemented at the time of writing. Therefore, we envisage that our findings lay the groundwork for future research in this area. Our contributions are summarized as follows:

1. We propose an AutoML approach for concept learning.
2. We perform automatic FS for concept learning both with a table-based and a graph-based approach.

3. We optimize the HPs of concept learners with the Covariance Matrix Adaptation Evolution Strategy (CMA-ES) [22].
4. We provide our source code to the research community.

The paper is structured as follows: In Section 2, we give an overview of the required background in DLs and concept learning. In Section 3, we introduce related work. Our approach is proposed in Section 4. Our experimental results are shown in Section 5 and we discuss the strengths and weaknesses of our AutoCL approach in Section 6. Finally, we summarize our work and give an outlook on future improvements of our approach in Section 7.

2 Background

DLs [3, 37] are a family of languages for knowledge representation (KR). DLs represent knowledge through concepts, objects, and roles. Concepts serve as formal descriptors of ideas within a specific domain; for instance, defining "Mother" as "woman has a child." Objects are members of concepts in the application domain and roles indicate binary relationships between objects. Information is stored in a knowledge base that is divided into two parts for description logic systems: the *TBox* and the *ABox*. The ABox contains assertions about objects. It relates objects to concepts and other objects via roles. The TBox describes the terminology by relating concepts and roles. Table 1 introduces the syntax of the DL $\mathcal{ALCQ}(\mathcal{D})$.

Table 1. Description logic constructs according to Lehmann et al. [37].

Construct	Syntax
$\mathcal{ALC}$	
abstract role	r
Boolean concrete role	b
numeric concrete role	d
atomic concept	A
top concept	$\top$
bottom concept	$\bot$
negation	$\neg C$
disjunction	$C \sqcup D$
conjunction	$C \sqcap D$
existential restriction	$\exists r.C$
universal restriction	$\forall r.C$

Construct	Syntax
$\mathcal{Q}$	
max. cardinality restriction	$\leq n\ r.C$
min. cardinality restriction	$\geq n\ r.C$
$\mathcal{D}$	
max. numeric restriction ($v \in \mathbb{R}$)	$d \leq v$
min. numeric restriction ($v \in \mathbb{R}$)	$d \geq v$
Boolean value restriction	$b = \mathit{true}$
Boolean value restriction	$b = \mathit{false}$

We revisit the definition of concept learning in DLs introduced by Lehmann et al. [37]. Assuming we have one target concept $Target$, a knowledge base $\mathcal{K}$, and sets P and N representing positive and negative examples of $Target$, the goal is to find a concept C such that $Target$ does not occur in C and for $\mathcal{K}' = \mathcal{K} \cup \{Target \equiv C\}$, we have $\mathcal{K}' \models P$ and $\mathcal{K}' \not\models N$.

Typically, the learning problem is to find a concept C which maximizes the F_1-measure or *Accuracy*, which are defined as follows:

$$Precision := \frac{|C_{\mathcal{P}} \cap P|}{|C_{\mathcal{P}} \cap P| + |C_{\mathcal{P}} \cap N|} \qquad Recall := \frac{|C_{\mathcal{P}} \cap P|}{|C_{\mathcal{P}} \cap P| + |C_{\mathcal{N}} \cap P|}$$

$$F_1 := 2 \times \frac{Precision \times Recall}{Precision \ + \ Recall} \qquad Accuracy := \frac{|C_{\mathcal{P}} \cap P| + |C_{\mathcal{N}} \cap N|}{|P| + |N|}$$

In this definition, $C_{\mathcal{P}}$ denotes the set of objects that are members of the concept C and $C_{\mathcal{N}}$ the set of objects that are not members of C [37, 33].

3 Related Work

Lehmann and Hitzler [37] learn concepts in $\mathcal{ALC}$ with a refinement operator. Their approach achieves high accuracy and surpasses previous inductive logic programs in certain cases.

DL-Learner [34], a framework widely utilized for inductive learning within the semantic web domain, serves as the foundation for various advanced algorithms such as CELOE [36] and OCEL [35]. CELOE is a noteworthy extension of OCEL that uses the same refinement operator but a different heuristic function with a soft syntactic bias that balances between predictive performance and short, readable concepts [33]. Another concept learning algorithm called DL-FOIL [17] constructs the solution as a disjunction of partial descriptions using refinement operators. The partial descriptions cover a part of the positive examples and eliminate as many negative and uncertain examples as possible. DL-FOCL1–3 [49] was created based on DL-FOIL. EvoLearner is a modern evolutionary concept learner for $\mathcal{ALCQ}(\mathcal{D})$. It differs from CELOE and OCEL as it utilizes evolutionary algorithms instead of inductive logic programming. OntoSample [4] speeds up concept learning by sampling a subset of the knowledge base. CLIP [33] accelerates concept learning by predicting the concept length before exploring the solution space. DRILL [15] employs reinforcement learning to traverse the space of potential refinement operations. Recent neurosymbolic approaches such as NCES [31] and NCES2 [32] directly synthesize concepts from sets of positive and negative examples.

Concept learners are machine learning models with important HPs, e.g. the fitness function in EvoLearner is controlled through one HP, whose optimal value depends on the underlying data [24]. AutoML is a promising solution for building a machine-learning system without human assistance and is being studied extensively. With the exponential growth of computing power, AutoML has become a hot topic in both industry and academia [23]. Many AutoML libraries have been proposed: AutoSklearn [18] is built on top of scikit-learn [47]; TPOT [44] supports neural networks via the Pytorch [45] backend; Auto-Keras [26] focuses on searching for deep learning models and supports multi-modal and multi-task models based on Keras.[1] AutoML techniques have been applied to many

[1] https://github.com/keras-team/keras

tasks, such as network compression, federated learning, image captioning [23], and even graph learning. The Auto-GNN framework [58] aims to find an optimal Graph Neural Network (GNN) architecture within a predefined search space. AutoGL is an AutoML pipeline based on PyTorch Geometric that enables feature engineering, model training and more [20].

Feature selection (FS) [9] is a key component of AutoML and involves the construction and selection of subsets of features for building a good predictor. FS algorithms [5, 11, 28, 57] have been proposed based on three categories: wrapper, filter, and embedded methods [21, 29, 30]. KGs have been integrated into FS methodologies in various ways to enhance model performance and interpretability. Peng et al. [48] proposed a hybrid feature selection model that integrates KG technology to obtain a feature text weight matrix, improving the efficiency and interpretability of the model. Li et al. [38] applied knowledge maps to generate new features. Atzmueller et al. [2] employed KGs together with an interactive visualization method for semi-automated FS. However, there is limited research on automatic FS for KGs and concept learners. While AutoGL includes FS methods for homogeneous GNNs, we introduce FS methods for concept learning and KGs with heterogeneous relations.

HPO as another key component of AutoML, plays a crucial role in the predictive performance of machine learning algorithms [56]. Machine learning models usually contain multiple hyperparameters that determine how a learning algorithm functions and affects the model parameters it learns. Before the actual training phase, we would like to find a set of HP values that achieve the best performance on the data in a reasonable amount of time. This process is called HPO or tuning. Several established approaches include Grid search [43], Random search [7], Model-based optimization (SMBO), Hyperband [39], and Bayesian Optimization and Hyperband [16]. In addition, open-source systems on HPO have emerged, such as Optuna [1], which allow users to construct the parameter search space dynamically and support an efficient sampling and pruning algorithm that allows for customization by the user. Ray-Tune [40] is a Python library for hyperparameter tuning. It scales from single machines to clusters, supports parallel execution, and integrates with various optimization algorithms for efficient and flexible hyperparameter search.

4 AutoCL

Similar to other advanced machine learning algorithms, concept learners require optimization of their features and HPs to achieve peak performance. However, current concept learners lack automated feature selection and hyperparameters are often manually adjusted. Improvements of concept learners were often accompanied by an increase in the number of HPs. While OCEL and CELOE are configured with 11 HPs, EvoLearner has more than 17 HPs.[2] Our research aims to address these challenges by introducing AutoCL, which comprises two main

[2] https://github.com/dice-group/AutoCL/blob/main/ontolearn/concept_learner.py

components: (1) features selection over KGs (Section 4.1), and automatic HPO (Section 4.2). The whole pipeline of our AutoCL approach is shown in Figure 1.

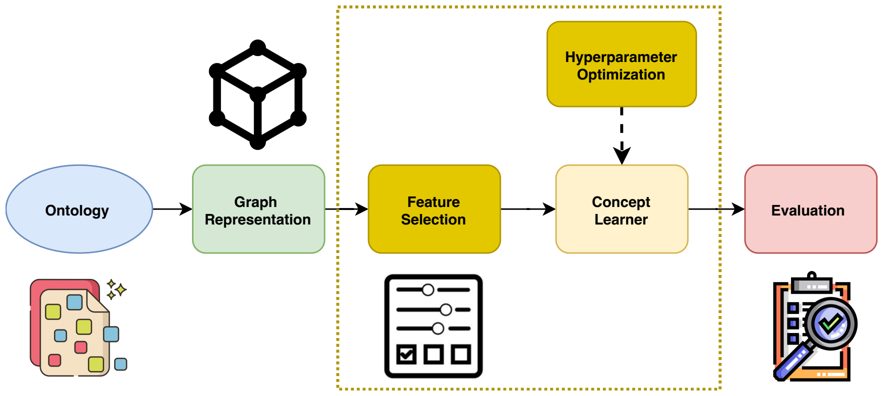

Fig. 1. Our AutoCL pipeline involves feature selection and hyperparameter optimization for concept learners.

4.1 Feature Selection

FS aims to identify a subset of relevant features from a dataset [27]. It can be formalized as follows [52]: Given a dataset D comprising M objects described by a feature set F, the aim is to discover an optimal subset of features, denoted as $F^* \subseteq F$, based on specific optimization criteria. In the context of KGs, we consider the relations of entities as features.

Consider the example of a family dataset depicted in Figure 2. It includes individuals of the concepts $Female \sqsubseteq Person$ (e.g., "Amy"), $Male \sqsubseteq Person$ (e.g., "Bob"), $Parent \sqsubseteq Person$ (e.g., "Tom") and $Mother \sqsubseteq Person$ (e.g., "Alice"). "Bob" has a `hasSibling` relationship with "Alice" and he is married to "Amy" as indicated by the `isMarried` relationship. "Amy" has a `hasSibling` relationship with "Tom", who is a parent.

Using the DL constructs from Table 1, "Bob" can be described as

$$Male \sqcap ((\exists \texttt{isMarried}.\exists \texttt{hasSibling}.Parent) \sqcup (\exists \texttt{hasSibling}.Mother))$$

For each individual in a dataset, each outgoing relationship that links to an object can be considered as a feature f with value v. For "Bob", the features are $f_1 =$ `hasSibling` with $v_1 =$ "Alice", $f_2 =$ `isMarried` with $v_2 = Amy$, and $f_3 =$ `hasType` with $v_3 = Male$. For "Amy", the features are $f_1 =$ `hasSibling` with $v_1 =$ "Tom" and $f_2 =$ `hasType` with $v_2 = Female$. For "Alice", the feature is $f_1 =$ `hasType` with $v_1 = Mother$. For "Tom", the feature is $f_1 =$ `hasType` with $v_1 = Parent$.

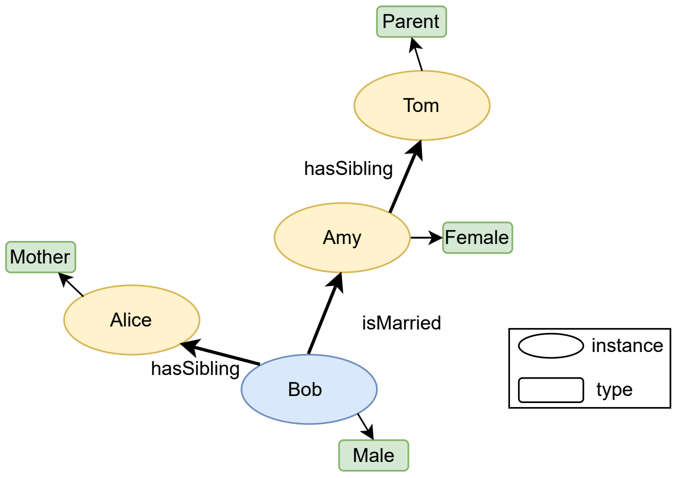

Fig. 2. Example of a family dataset with multiple individuals, relations, and types.

The most relevant features for the concept of an *Uncle* are `hasSibling`, `isMarried`, and `hasType`. For example, we can infer that "Bob" is an *Uncle* as he is *Male* and is married to "Amy" who has a sibling "Tom" who is a *Parent*.

Table-Based Feature Selection Approach (τ) Our initial approach involves transforming a graph representation into a tabular format for feature selection. This graph, denoted as G, comprises individuals I and relationship features F.

Transformation of Graph into Tabular Format The graph G is converted into a table format, where individuals are represented in the first column, and features are represented in the remaining columns. Each relationship feature f_i belongs to the set F. For each individual in I and each of its feature-value pairs (f_i, v_j) where $0 \leq i \leq n$, $0 \leq j \leq m$, $n, m \in \mathbb{N}$, a new column is created if it does not already exist for f_i with v_j. These columns are labelled as $f_i \cdot v_j$ and the value in each cell indicates whether that combination exists for the individual (1 if it exists, 0 otherwise).

The primary aim of our feature selection process is to identify a minimum of 5 distinct relationship features from the transformed tabular data.

Feature Selection Steps

1. Initially, the top $K = 5$ columns (feature-value pairs) are selected based on the χ^2 test [46].
2. As long as we have not obtained 5 distinct relationship features and we have not reached the maximum number of 100 iterations, we increment K by one and select the top K columns from the original table.

Graph-Based Feature Selection Strategy (γ) Our second idea is to use a graph-based wrapper method for FS. We run EvoLearner as the wrapper tool, employing it for retrieving features. EvoLearner operates directly on the graph, generating the top n distinct hypotheses (concepts in DL) based on the *Accuracy* metric. Subsequently, we extract all relations and atomic concepts from the top n hypotheses. This involves identifying the edges (relations) and nodes (atomic concepts) in the graph associated with the selected concepts.

Feature Selection Steps

1. Initially, select the top $n := 10$ distinct hypotheses using EvoLearner.
2. While the top n hypotheses do not contain at least 5 distinct relationship features and the maximum number of 100 iterations has not been reached, increment n by one and repeat the process.

4.2 Hyperparameter Optimization

HPO algorithms aim to find the best-performing configuration, denoted by $\boldsymbol{\lambda}$, within the hyperparameter space $\widetilde{\Lambda} \subset \Lambda$ for a given machine learning algorithm $\mathcal{A}_{\boldsymbol{\lambda}}$. This space encompasses all pertinent hyperparameters along with their corresponding ranges $\widetilde{\Lambda} = \tilde{\Lambda}_1 \times \tilde{\Lambda}_2 \times \ldots \times \tilde{\Lambda}_l$ where $\widetilde{\Lambda}_i$ represents a constrained subset of the domain of the ith hyperparameter. The HPO problem is defined as

$$\boldsymbol{\lambda}^* \in \arg\min_{\boldsymbol{\lambda} \in \widetilde{\Lambda}} c(\boldsymbol{\lambda}) = \arg\min_{\boldsymbol{\lambda} \in \widetilde{\Lambda}} \widehat{\mathrm{GE}}(\mathscr{I}, \mathscr{J}, \rho, \boldsymbol{\lambda})$$

where $\boldsymbol{\lambda}^*$ denotes the theoretical optimum, and $c(\boldsymbol{\lambda})$ is a shorthand for the estimated generalization error $\widehat{\mathrm{GE}}(\mathscr{I}, \mathscr{J}, \rho, \boldsymbol{\lambda})$ of a machine learner $\mathscr{I}_{\boldsymbol{\lambda}}$ with HP configuration $\boldsymbol{\lambda}$, target performance measure ρ, and resampling strategy $\mathscr{J} = \left(\left(\mathscr{I}_{\text{train},1}, \mathscr{I}_{\text{test},1}\right), \ldots, \left(\mathscr{I}_{\text{train},B}, \mathscr{I}_{\text{test},B}\right)\right)$, where $\mathscr{I}_{\text{train}}$ and $\mathscr{I}_{\text{test}}$ partition the available data into training and test sets and B denotes the number of splits [8].

The concept learner, which acts as a machine learner, has many HPs that can impact its performance. Traditional methods for HPO in machine learning models like grid search and random search are time-consuming [53]. To address this, we opted for the CMA-ES Sampler [1] from Optuna, after comparing it with five other samplers, including Random Sampler, Grid Sampler, Tree-structured Parzen Estimator Sampler [6], Non-dominated Sorting Genetic Algorithm II Sampler [14], and Quasi-Monte Carlo Sampler [10]. We choose the CMA-ES Sampler due to its effective performance in optimizing the HPs of concept learners in our experiment. Our HPO process utilizes the CMA-ES sampler to provide HPs to the concept learner and assesses them based on the returned quality score. Table 4 defines the search space of hyperparameters. To ensure stability and accuracy, our HPO process is run three times, with each run comprising `n_trials` := 100 iterations of the sampler to obtain the best HPO results via the quality score `quality_func`.

5 Evaluation

After introducing our experimental setup in Section 5.1, we evaluate the two FS methods presented in the previous section on state-of-the-art concept learners in Section 5.2. In Section 5.3, we perform HPO and in Section 5.4, we combine the two FS methods with HPO to form our complete AutoCL automation pipeline. Notations used in this evaluation section are detailed in Table 2.

Table 2. Overview of the notations used in Section 5.

Notation	Definition
LP	Learning problem
FS	Feature selection
T_{CL}	Time taken for concept learning
T_{FS}	Time taken for feature selection
τ	Table-based feature selection approach
γ	Graph-based feature selection approach
τ_{CMA-ES}	Table-based feature selection with HPO (sampler CMA-ES)
γ_{CMA-ES}	Graph-based feature selection with HPO (sampler CMA-ES)

5.1 Experimental Setup

Datasets Table 3 summarizes the 8 datasets from SML Bench [55], including *Carcinogenesis* (*CA*), *Family* (*FA*), *Hepatitis* (*HE*), *Mammographic* (*MA*), *Mutagenesis* (*MU*), *NCTRER* (*NC*), *Premier League* (*PR*), and *Pyrimidine* (*PY*). The *Lymphography* dataset was excluded due to its lack of relevant properties. Additionally, we added the *FA* dataset from DL-Learner [34] to SML-Bench, commonly used for evaluating structured machine learning approaches [54].

Table 3. Overview of the datasets in terms of their abbreviations, number of instances, axioms, atomic concepts, object properties and data properties.

Dataset	Abbr.	Instances	Axioms	Concepts	Obj. Pro.	Data Pro.
Carcinogenesis	*CA*	22,373	74,566	142	4	15
Family	*FA*	202	1,829	18	4	0
Hepatitis	*HE*	6,812	79,935	14	5	12
Mammographic	*MA*	975	6,80	19	3	2
Mutagenesis	*MU*	14,145	62,066	86	5	6
NCTRER	*NC*	10,209	103,070	37	9	50
Premier League	*PR*	11,209	2,153,818	9	13	202
Pyrimidine	*PY*	74	2,080	1	0	27

Baseline We apply our methods to three concept learners: EvoLearner, OCEL, and CELOE.

Features As described in the previous section, our raw data is saved in KGs, and the features are extracted from the ontologies. A different number of features are obtained for each dataset. For example, in the *FA* dataset, features such as `hasSibling` and `isMarried` are included. We chose to present the datasets in our GitHub repository because some datasets have over 100 features prior to FS.

Hyperparameters As shown in Table 4, we opted to optimize 8 of EvoLearner's 17 HPs: the maximum runtime in seconds (`max_runtime`), the number of random individuals considered in each tournament (`tournament_size`), the maximum height of a concept's abstract syntax tree (`height_limit`), the upper cardinality limit considered for cardinality restrictions (`card_limit`), whether to use data property restrictions (`use_data_properties`), whether to use inverse properties (`use_inverse`), the quality function (`quality_func`), and the splitting criterion for data properties (`value_splitter`).

For OCEL and CELOE, we considered 4 of their 11 HPs: the maximum runtime in seconds (`max_runtime`), the maximum number of concepts that are tested (`max_num_of_concepts_tested`), the maximum number of refinements (`iter_bound`), and the quality function (`quality_func`). Other hyperparameters had a negligible impact on performance, e.g., because the Ontolearn library only offered a single option for some of them at the time of writing.

Table 4. Overview of the range of hyperparameters.

HPs for EvoLearner	**Range**
max_runtime	1–25
tournament_size	1–25
height_limit	1–25
card_limit	2–10
use_data_properties	[True, False]
use_inverse	[True, False]
quality_func	[F1, Accuracy]
value_splitter	[BinningValueSplitter, EntropyValueSplitter]
HPs for OCEL and CELOE	**Range**
max_runtime	2–600
max_concepts	2–1000000000
iter_bound	2–1000000000
quality_func	[F1, Accuracy]

Dataset Split and Evaluation Metrics During HPO, we divided the data into 60% for training, 20% for validation, and 20% for testing to fine-tune the HPs in models on the validation set. This partitioning strategy helps prevent overfitting by tailoring HPs specifically to the validation set while ensuring that the chosen parameters generalize effectively to unseen data. In contrast, for FS, we use 80% for training and 20% for testing to train the model preferentially on a larger segment of the data to identify informative features. This simplified split is sufficient for selecting relevant features without the need for a separate validation set, as feature selection itself does not involve hyperparameter tuning. We evaluate the performance of the approaches in terms of *Accuracy*, F_1-measure mentioned in Section 2, and the time taken for learning in seconds.

Reproducibility Our three concept learners are implemented by the Ontolearn library in Python 3.8. For the FS, we use owlapy (0.41), owlready2 (0.41), Pandas (1.2.3), and the scikit-learn library (1.0.2) with its `SelectKBest` class. For HPO, we use the Optuna framework (3.0.3). Our experiments were run on a Linux-based virtual machine with 256 GB RAM and 64 CPUs. Our implementation is publicly available on GitHub at https://github.com/dice-group/AutoCL.

5.2 Feature Selection Results

Table 5 presents the evaluation results of our two FS methods τ and γ on the benchmarking datasets in terms of F_1-measure, *Accuracy*, concept learning time T_{CL}, and feature selection time T_{FS}.

Table-based Feature Selection We report the results of our initial table-based FS approach (τ). As shown in Table 5 (top), after table-based feature selection, EvoLearners results improved for 3 datasets in terms of F_1-measure and 2 datasets in terms of *Accuracy*, particularly on the *CA* and *HE* datasets.

The concept learning time was at least halved on the 6 datasets *CA*, *FA*, *HE*, *MA*, *NC*, and *PR*. Impressively, the large dataset *PR* demonstrates an extraordinary improvement, saving nearly 200 seconds for learning after τ. For OCEL, evaluation metrics improved for *CA*, *PR*, and PY. For *PR*, the learning time was reduced by over 4 minutes. Similar outcomes are seen in the CELOE experiments, where the *PR* dataset has a considerably shorter running time and higher quality scores than the original one. Furthermore, the datasets *FA*, *HE*, *PR*, and PY show improvements in both F_1-measure and *Accuracy*.

Graph-based Feature Selection We report the results of our second approach, graph-based FS (γ). When using the γ method on EvoLearner, the performance of F_1-measure and *Accuracy* for datasets *FA*, MU, NC, and *PR* remains at the optimal level of 1.00. For certain datasets, the learning time has notably decreased, particularly for the *PR*, with a learning time now at 3.86 seconds. Surprisingly, the datasets *MA* and PY improve F_1-measure and *Accuracy* while

Table 5. Evaluation of the concept learners EvoLearner (top), OCEL (middle), and CELOE (bottom) without (left), with table-based (middle), and graph-based (right) feature selection in terms of F_1-measure and *Accuracy*. The times for concept learning T_{CL} and feature selection T_{FS} are measured in seconds.

LP	**EvoLearner**			**EvoLearner(τ)**				**EvoLearner(γ)**			
	F1	**Acc.**	T_{CL}	**F1**	**Acc.**	T_{CL}	T_{FS}	**F1**	**Acc.**	T_{CL}	T_{FS}
CA	0.70±0.07	0.63±0.05	106	**0.75±0.06**	**0.72 ± 0.04**	**5.52**	167.98	0.73±0.04	0.70±0.02	6.00	**6.50**
FA	1.00±0.01	1.00±0.01	1.75	0.76±0.15	0.81±0.06	0.79	**0.47**	**1.00±0.00**	**1.00±0.00**	**0.73**	19.38
HE	0.80±0.03	0.83±0.04	44.35	**0.82±0.01**	**0.84±0.02**	5.68	13.04	0.79±0.02	0.83±0.01	**5.65**	**6.31**
MA	0.78±0.01	0.78±0.02	18.69	0.63±0.00	0.46±0.00	**5.16**	**0.10**	**0.80±0.01**	**0.83±0.02**	5.21	7.25
MU	**1.00±0.00**	**1.00±0.00**	3.10	0.56±0.24	0.83±0.07	5.63	377.02	**1.00±0.00**	**1.00±0.00**	**0.74**	**2.98**
NC	**1.00±0.00**	**1.00±0.00**	6.28	**1.00±0.00**	**1.00±0.00**	1.03	311.74	**1.00±0.00**	**1.00±0.00**	**0.84**	**3.91**
PR	**1.00±0.00**	**1.00±0.00**	205.86	0.99±0.02	0.97±0.02	6.58	232.08	**1.00±0.00**	**1.00±0.00**	**3.86**	**96.05**
PY	0.86±0.13	0.88±0.14	**1.21**	0.89±0.12	0.79±0.12	5.11	207.20	**0.95±0.05**	**0.96±0.06**	1.94	**1.92**

LP	**OCEL**			**OCEL(τ)**				**OCEL(γ)**			
	F1	**Acc.**	T_{CL}	**F1**	**Acc.**	T_{CL}	T_{FS}	**F1**	**Acc.**	T_{CL}	T_{FS}
CA	0.69±0.05	0.56±0.09	651.58	**0.75±0.06**	**0.72±0.04**	636.23	168.90	0.69±0.04	0.60±0.00	**610.29**	**6.40**
FA	0.87±0.13	0.83±0.17	**600.08**	0.59±0.02	0.63±0.06	600.04	**0.47**	**0.87±0.03**	**0.85±0.06**	600.86	16.66
HE	**0.81±0.04**	**0.81±0.05**	**602.10**	0.58±0.01	0.61±0.01	602.41	13.13	0.68±0.07	0.72±0.08	603.47	**6.09**
MA	0.81±0.06	0.84±0.03	**600.18**	0.63±0.00	0.46±0.00	600.27	**0.17**	**0.81±0.01**	**0.84±0.01**	607.81	9.00
MU	**0.93±0.7**	**0.96±0.04**	**600.85**	0.39±0.11	0.67±0.07	634.31	311.02	0.56±0.22	0.81±0.11	626.87	**2.02**
NC	**1.00±0.00**	**1.00±0.00**	**604.52**	0.95±0.03	0.94±0.02	734.75	331.22	0.99±0.01	0.99±0.01	757.99	**3.86**
PR	0.65±0.01	0.48±0.04	718.79	0.69±0.02	0.54±0.04	**458.00**	232.08	**1.00±0.00**	**1.00±0.00**	681.37	**98.00**
PY	0.73±0.07	0.71±0.03	**600.05**	**0.74±0.07**	**0.75±0.02**	600.11	367.20	0.53±0.17	0.54±0.13	620.63	**0.70**

LP	**CELOE**			**CELOE(τ)**				**CELOE(γ)**			
	F1	**Acc.**	T_{CL}	**F1**	**Acc.**	T_{CL}	T_{FS}	**F1**	**Acc.**	T_{CL}	T_{FS}
CA	0.65±0.05	0.61±0.04	603.82	0.57±0.04	**0.62±0.03**	603.92	163.95	**0.70±0.02**	0.60±0.03	**600.85**	**6.46**
FA	0.88±0.12	0.89±0.10	**84.57**	0.89±0.05	0.89±0.06	600.14	**0.57**	**1.00±0.00**	**1.00±0.00**	523.14	16.4
HE	0.80±0.04	0.79±0.03	**602.22**	**0.82±0.01**	**0.82±0.01**	607.74	13.0	0.72±0.10	0.78±0.05	600.72	**6.11**
MA	0.76±0.01	0.79±0.03	**600.16**	0.63±0.00	0.46±0.00	600.26	**0.08**	**0.81±0.02**	**0.84±0.01**	600.37	7.35
MU	**0.70±0.10**	**0.84±0.04**	602.74	0.00±0.00	0.71± 0.00	602.47	357.03	0.44±0.22	0.81±0.07	**601.28**	**2.98**
NC	0.96±0.02	0.96±0.02	**405.30**	0.97±0.02	0.96±0.02	727.92	313.12	**1.00±0.00**	**1.00±0.00**	600.66	**3.82**
PR	0.82±0.10	0.87±0.06	692.29	0.96±0.02	0.96±0.02	412.33	233.18	**1.00±0.00**	**1.00±0.00**	**101.67**	**93.1**
PY	0.80±0.05	0.75±0.12	**400.21**	0.80±0.03	0.79±0.05	600.92	207.20	**0.86±0.04**	**0.83±0.06**	600.23	**1.42**

the learning time is accelerated. The overall quality scores for *HE* are nearly the same as the original experiment, with the learning time reduced by nearly a factor of 8 compared to before. Additionally, on *CA*, the quality scores remain nearly the same; however, the learning time and overall runtime have been reduced, by nearly a factor of 17 compared to before. From OCEL, *CA* has reduced the learning time by nearly 40 seconds with improved *Accuracy* scores. Datasets such as *FA* and *MA* achieve slightly better quality scores in about the same running time as the original ones, and on the large dataset *PR*, we have an improvement of 53% in F_1-measure and almost 105% in *Accuracy* while improving the running time about 30 seconds. In CELOE, we found that the quality scores of *FA*, *MA*, *NC*, *PR*, and *PY* are better than the original experimental data, the *Accuracy* and F_1-measure for some of the data are even 1, and the learning time of *PR* is improved by 6 times. Not surprisingly, more than half of the datasets benefit from this γ method, especially in terms of two quality scores.

Although both τ and γ exhibit favourable effects on concept learners across most datasets especially some contain large axioms such as *PR*, γ not only demonstrates a superior learning rate compared to τ but also yields higher F_1-measure and *Accuracy* scores. To further compare the performance of the two feature methods themselves, we analyze in terms of T_{FS} for the feature selection process. Both methods are applied for feature processing on the same datasets, γ outperforms τ in terms of T_{FS} as evidenced on 6 datasets aside from *FA* and *MA*. This is probably because γ performs better in datasets with a higher number of atomic concepts, while τ works well in datasets with fewer atomic concepts.

5.3 Hyperparameter Optimization Results

Table 6 displays the best results for each learning problem including the best values for some HPs[3] before and after performing HPO on EvoLearner, OCEL, and CELOE within the search range in Table 4.

Table 6. Best results per learning problem obtained from EvoLearner, OCEL and CELOE before (1st row) and after (2nd row) HPO, in terms of `max_runtime`, `tournament_size` (EvoLearner only), F_1-measure, and *Accuracy*.

LP	**EvoLearner**				**OCEL**			**CELOE**		
	Maxt.	**Tour.**	F_1	**Acc.**	**Maxt.**	F_1	**Acc.**	**Maxt.**	F_1	**Acc.**
CA	-	7	0.73	0.67	-	0.69	0.56	-	**0.70**	**0.67**
	12	8	**0.78**	**0.74**	59	**0.69**	**0.57**	26	0.70	0.55
FA	-	7	**1.00**	**1.00**	-	**1.00**	**1.00**	-	**0.97**	**0.96**
	2	2	**1.00**	**1.00**	184	0.94	0.89	190	0.94	0.94
HE	-	7	0.80	0.83	-	**0.85**	**0.86**	-	0.81	0.81
	10	6	**0.82**	**0.85**	72	0.83	0.83	193	**0.83**	**0.83**
MA	-	7	0.79	0.81	-	**0.85**	**0.88**	-	**0.80**	**0.82**
	3	5	**0.80**	**0.85**	72	0.80	0.83	82	0.79	**0.82**
MU	-	7	**1.00**	**1.00**	-	**1.00**	**1.00**	-	0.80	0.89
	3	2	**1.00**	**1.00**	125	**1.00**	**1.00**	69	**1.00**	**1.00**
NC	-	7	**1.00**	**1.00**	-	0.98	0.98	-	0.98	0.98
	2	2	**1.00**	**1.00**	108	**1.00**	**1.00**	3	**1.00**	**1.00**
PR	-	7	**1.00**	**1.00**	-	0.67	0.52	-	0.93	0.94
	12	2	**1.00**	**1.00**	157	**1.00**	**1.00**	59	**1.00**	**1.00**
PY	-	7	0.86	0.86	-	0.71	0.70	-	**0.89**	0.87
	8	6	**1.00**	**1.00**	58	**0.88**	**0.87**	58	0.88	**0.87**

Through Table 6, we can observe an improvement in quality scores for EvoLearner on *CA*, *HE*, *MA*, and *PY*. In particular, for dataset *CA*, the F_1-measure improved by 6% and *Accuracy* improved by 10% after HPO. Similarly,

[3] All of the best HP settings are shown in our GitHub repository: https://github.com/dice-group/AutoCL/tree/main/HPO

excellent performance is also reflected in PY, the F_1-measure is almost 16% better. *FA*, *NC*, *MU*, and *PR* keep the best F_1-measure and *Accuracy* before and after HPO in some values of HP changed.

We explore how the HPs affect the evaluation metrics and observe from Table 6 that HPO reduces the running time of all datasets after compared with the default run time of 600 seconds, which proves that HPO accelerates the learning time of the concept learner. Some datasets like *CA*, *HE*, the best values of HPs including the `tournament_size`, all different from the default values. These HPs and their values interact with each other and have an impact on the decisions made by the concept learner. The results from Table 6 showed slightly higher quality scores after OCEL and CELEO with parameter selection when learning some of the data. In OCEL we can find that the optimal value of runtime is reduced on all datasets, and the F_1-measure as well as the *Accuracy* is improved on four of them, especially for the large dataset *PR*, where the F_1-measure and *Accuracy* are improved by at least 50%. When HPO was applied to CELOE, it produced similar results as on OCEL: the concept learner improved learning performance on more than half of the datasets while reducing the runtime and learning range. On the *CA*, *MA*, and *FA* dataset, F_1-measure and *Accuracy* still produced quality scores similar to the original data. In conclusion, we can conclude clearly that in our experiments, both when applying HPO to EvoLearner, OCEL and CELOE, the quality of concept learning can be effectively improved while reducing the search space in learning most of the datasets.

5.4 AutoCL Results

We conducted additional experiments by integrating different feature selection methods with the CMA-ES sampler to validate our final AutoCL approach. Table 7 presents the outcomes of comparing two pipelines: table-based feature selection with CMA-ES (τ_{CMA-ES}) and graph-based feature selection with CMA-ES (γ_{CMA-ES}).

After EvoLearner with τ_{CMA-ES}, the learning time was reduced when studying datasets *CA*, *HE*, *MA*, *NC*, and *PR*. There was a noteworthy enhancement in F_1-measure and *Accuracy* for *CA* and *HE*, and both quality scores were observed for *NC* and *PR*, similar to the original experimental results.

Following OCEL with τ_{CMA-ES}, the learning time reductions were observed across all 8 datasets. For instance, the learning time of *FA* decreased from the original 600.08 seconds to 92.26 seconds, maintaining comparable quality scores. Notably, most of the datasets such as *CA* and PY datasets exhibited improved quality scores alongside reduced learning time. Specifically, for *PR*, the time decreased by 10 minutes, with F_1-measure and *Accuracy* improving by almost 53% and 108%. With the incorporation of τ_{CMA-ES} into CELOE, all datasets have substantially shorter runtimes like in OCEL. Improvements in F_1-measure and *Accuracy* were observed for datasets *HE*, PY and *PR*. However, for datasets *MA*, and MU, there was a minor decline in quality scores.

Subsequently, after EvoLearner was incorporated with γ_{CMA-ES}, the time required to learn the majority of the datasets was lowered. Significant improve-

Table 7. Evaluation results of EvoLearner (top), OCEL (middle), and CELOE (bottom) without (left), with τ_{CMA-ES} (middle), and with γ_{CMA-ES} (right) in terms of F_1-measure and *Accuracy*. The concept learning times T_{CL} are measured in seconds.

LP	**EvoLearner**			**EvoLearner(τ_{CMA-ES})**			**EvoLearner(γ_{CMA-ES})**		
	$\boldsymbol{F_1}$	**Acc.**	$\boldsymbol{T_{CL}}$	$\boldsymbol{F_1}$	**Acc.**	$\boldsymbol{T_{CL}}$	$\boldsymbol{F_1}$	**Acc.**	$\boldsymbol{T_{CL}}$
CA	0.70±0.07	0.63±0.05	106	**0.75±0.06**	**0.72±0.04**	10.6	0.73±0.02	0.70±0.02	**10.52**
FA	1.00±0.01	1.00±0.01	1.75	**1.00±0.00**	**1.00±0.00**	1.97	**1.00±0.00**	**1.00±0.00**	**1.42**
HE	0.80±0.03	0.83±0.04	44.35	**0.83±0.01**	**0.84±0.00**	**9.18**	0.80±0.03	0.83±0.04	11.17
MA	0.78±0.01	0.78±0.02	18.69	0.63±0.00	0.46±0.00	**3.18**	**0.81±0.01**	**0.82±0.02**	6.18
MU	1.00±0.00	1.00±0.00	3.10	0.56±0.24	0.83±0.07	9.73	**1.00±0.00**	**1.00±0.00**	**0.58**
NC	1.00±0.00	1.00±0.00	6.28	**1.00±0.00**	**1.00±0.00**	**0.57**	**1.00±0.00**	**1.00±0.00**	2.61
PR	1.00±0.00	1.00±0.00	205.86	0.99±0.02	0.97±0.02	5.28	**1.00±0.00**	**1.00±0.00**	**1.81**
PY	0.86±0.13	0.88±0.14	**1.21**	0.67±0.11	0.71±0.13	5.46	**1.00±0.00**	**1.00±0.00**	2.31

LP	**OCEL**			**OCEL(τ_{CMA-ES})**			**OCEL(γ_{CMA-ES})**		
	$\boldsymbol{F_1}$	**Acc.**	$\boldsymbol{T_{CL}}$	$\boldsymbol{F_1}$	**Acc.**	$\boldsymbol{T_{CL}}$	$\boldsymbol{F_1}$	**Acc.**	$\boldsymbol{T_{CL}}$
CA	0.69±0.05	0.56±0.09	651.58	**0.69±0.02**	0.57±0.06	**128.07**	0.69±0.04	**0.60±0.00**	155.35
FA	0.87±0.13	0.83±0.17	600.08	**0.90±0.04**	**0.89±0.05**	92.26	0.89±0.07	0.87±0.06	**77.71**
HE	0.81±0.04	0.81±0.05	602.10	**0.82±0.00**	**0.82±0.00**	**141.13**	0.67±0.07	0.67±0.05	207.23
MA	0.81±0.06	0.84±0.03	600.18	0.63±0.00	0.46±0.00	**116.85**	**0.81±0.03**	**0.84±0.02**	192.75
MU	**0.93±0.7**	0.96±0.04	600.85	0.46±0.35	0.74±0.14	143.39	0.93±0.09	**0.96±0.01**	**156.88**
NC	**1.00±0.00**	**1.00±0.00**	604.52	0.97±0.03	0.96±0.04	**90.76**	1.00±0.01	1.00±0.01	100.39
PR	0.65±0.01	0.48±0.04	718.79	**1.00±0.00**	**1.00±0.00**	88.08	1.00±0.04	1.00±0.02	**54.55**
PY	0.73±0.07	0.71±0.03	600.05	**0.86±0.14**	**0.87±0.13**	**195.38**	0.76±0.12	0.80±0.05	229.92

LP	**CELOE**			**CELOE(τ_{CMA-ES})**			**CELOE(γ_{CMA-ES})**		
	$\boldsymbol{F_1}$	**Acc.**	$\boldsymbol{T_{CL}}$	$\boldsymbol{F_1}$	**Acc.**	$\boldsymbol{T_{CL}}$	$\boldsymbol{F_1}$	**Acc.**	$\boldsymbol{T_{CL}}$
CA	0.65±0.05	**0.61±0.04**	603.82	0.66±0.03	0.56±0.03	**233.59**	**0.70±0.02**	0.54±0.04	303.45
FA	**0.95±0.10**	0.94±0.06	284.57	0.89±0.06	**1.00±0.00**	179.71	0.90±0.03	0.85±0.10	**151.88**
HE	0.80±0.04	0.79±0.03	602.22	**0.82±0.01**	**0.80±0.04**	107.71	0.71±0.05	0.73±0.03	**54.86**
MA	0.76±0.01	0.79±0.03	600.16	0.63±0.00	0.46±0.00	99.31	**0.82±0.01**	**0.83±0.02**	**72.48**
MU	0.70±0.10	0.84±0.04	602.74	0.39±0.15	0.78±0.11	208.45	**0.89±0.03**	**0.92±0.03**	**111.07**
NC	0.96±0.02	0.96±0.02	405.30	0.95±0.01	0.96±0.12	218.09	**1.00±0.01**	**1.00±0.01**	**42.27**
PR	0.82±0.10	0.87±0.06	692.29	**1.00±0.00**	**1.00±0.02**	62.66	0.98±0.01	0.98±0.01	**9.27**
PY	0.80±0.05	0.75±0.12	400.21	**0.88±0.12**	0.75±0.04	106.05	0.79±0.03	**0.80±0.06**	**60.13**

ments in F_1-measure and *Accuracy* were seen when learning *CA*, *MA*, and *PY*, outperforming the results obtained in the original experiment. Across all datasets, OCEL, and CELOE consistently reduced learning time. Notably, there was a remarkable 53% improvement in F_1-measure, a 108% increase in *Accuracy*, and a surprising 11-minute reduction in learning time when OCEL learned the *PR*. Similarly, both F_1-measure and *Accuracy* improved during the *HE* learning process with OCEL which learning time was reduced by roughly 6 minutes. CELOE performance similar to OCEL, achieved comprehensive learning speedups when learning datasets such as *MA*, *MU*, *NC*, *PR* and so on, both F_1-measure and *Accuracy* show overall increases ranging from 5% to 30%.

Based on the information presented above, whether the concept learner employs τ_{CMA-ES} or γ_{CMA-ES} for learning most data sets, the running time will surely reduce. However, we can see from comparing the time values in Table 7 that the run time behind the γ_{CMA-ES} is shorter than the run time through the τ_{CMA-ES}, and there are 14 best time performance uses. Similarly, after checking with the best F_1-measure and *Accuracy*, the three concept learners acquired 24 times of optimal quality score performance through the operation after γ_{CMA-ES}, whereas only 21 times appeared in the τ_{CMA-ES} part. In terms of quality score and learning time, γ_{CMA-ES} performs better than τ_{CMA-ES}. As a result, integrating the γ_{CMA-ES} approach with the CMA-ES sampler in a pipeline as our final AutoCL technique seems reasonable. We achieve promising performance on 6 of the 8 datasets in SML-Bench across concept learners while reducing the learning time.

6 Discussion

Feature Selection Approach According to the two methods we proposed, the τ_{CMA-ES} method relies on a tabular method, which is a feature selection technique that involves the use of converted KGs to a tabular format in our experiment. Our γ_{CMA-ES} method enables FS directly on the KGs, which is convenient and reasonably interpretable. In Section 4.1, we have mentioned a stopping criteria for the FS process. Because our features contain predicate and subject pairs, the presence of multiple repetitions of a predicate adds complexity. In certain instances, after 100 iterations, the selected features may not yield distinct values. At the same time, some datasets like *FA*, may only have 4 or 3 features in total, which makes it impossible to attain 5 distinct features. Hence, a stopping criteria is necessary. In some cases, we may end up with only one inadequate feature for classification, which could lead to an F_1-measure score of 0 in some instances, but this is not typically the case.

Hyperparameter Optimization Approach In a pilot study, we encountered some incompatibilities of HPO frameworks with the Ontolearn library. This influenced our choice of the HPO framework and we ultimately selected Optuna because it allowed us to easily define, save, and analyze HPO processes. In the future, we plan to enhance the HPO by expanding the search space and considering additional concept learners.

AutoCL Pipeline Concept learners are a type of supervised learning that enhances the learning process by using the semantic knowledge and information offered by KGs. AutoCL can automate concept learning and enhance application efficiency by using the rich semantic information held in KGs.

The efficiency of AutoCL in real-world applications depends not only on the concept learners use for learning but also on the quality and quantity of the learning dataset. For example, the time required by AutoCL in the FS phase depends on the number of data attributes etc. When dealing with huge and complex data where performance is an issue, parallel processing can be explored to ensure good performance.

7 Conclusion

We investigated the use of AutoML for concept learning to improve the efficiency of concept learning. In particular, we developed a feature selection and hyperparameter optimization approach for concept learners. We evaluated our proposed approach, dubbed AutoCL, for state-of-the-art concept learners on the SML-Bench datasets. The evaluation results show that our approach can improve the performance of the learning algorithm. Additionally, we show that feature selection reduces the search space and improves efficiency while HPO improves the predictive performance of concept learners.

To further enhance the effectiveness of AutoCL, we intend to incorporate additional feature selection and hyperparameter optimization approaches in the future. Moreover, we plan to test it on large-scale KGs beyond the scope of the SML benchmark datasets. Potential applications include explainable anomaly detection, identification of cybersecurity threads, financial fraud detection, and manufacturing quality control.

8 Acknowledgement

This work has been supported by the German Federal Ministry of Education and Research (BMBF) within the projects COLIDE under the grant No. 01I521005D, KIAM under the grant No. 02L19C115, the European Union's Horizon Europe research and innovation programme within the project ENEXA under the grant No 101070305. This work has also been supported by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation): TRR 318/1 2021 – 438445824.

Disclosure of Interests.

The authors have no competing interests to declare that are relevant to the content of this article.

References

1. Akiba, T., Sano, S., Yanase, T., Ohta, T., Koyama, M.: Optuna: A next-generation hyperparameter optimization framework. In: KDD. pp. 2623–2631. ACM (2019)
2. Atzmueller, M., Sternberg, E.: Mixed-initiative feature engineering using knowledge graphs. In: K-CAP. pp. 45:1–45:4. ACM (2017)
3. Baader, F., Calvanese, D., McGuinness, D., Patel-Schneider, P., Nardi, D.: The description logic handbook: Theory, implementation and applications. Cambridge university press (2003)
4. Baci, A., Heindorf, S.: Accelerating concept learning via sampling. In: CIKM. pp. 3733–3737. ACM (2023)
5. Bennasar, M., Hicks, Y., Setchi, R.: Feature selection using joint mutual information maximisation. Expert Syst. Appl. **42**(22), 8520–8532 (2015)

6. Bergstra, J., Bardenet, R., Bengio, Y., Kégl, B.: Algorithms for hyper-parameter optimization. In: NIPS. pp. 2546–2554 (2011)
7. Bergstra, J., Bengio, Y.: Random search for hyper-parameter optimization. J. Mach. Learn. Res. **13**, 281–305 (2012)
8. Bischl, B., Binder, M., Lang, M., Pielok, T., Richter, J., Coors, S., Thomas, J., Ullmann, T., Becker, M., Boulesteix, A., Deng, D., Lindauer, M.: Hyperparameter optimization: Foundations, algorithms, best practices, and open challenges. WIREs Data. Mining. Knowl. Discov. **13**(2) (2023)
9. Blum, A., Langley, P.: Selection of relevant features and examples in machine learning. Artif. Intell. **97**(1-2), 245–271 (1997)
10. Caflisch, R.E.: Monte carlo and quasi-monte carlo methods. Acta numerica **7**, 1–49 (1998)
11. Chaudhuri, A., Sahu, T.P.: A hybrid feature selection method based on binary jaya algorithm for micro-array data classification. Comput. Electr. Eng. **90**, 106963 (2021)
12. Chen, J., Song, L., Wainwright, M.J., Jordan, M.I.: Learning to explain: An information-theoretic perspective on model interpretation. In: ICML. Proceedings of Machine Learning Research, vol. 80, pp. 882–891. PMLR (2018)
13. Covert, I., Lundberg, S.M., Lee, S.: Explaining by removing: A unified framework for model explanation. J. Mach. Learn. Res. **22**, 209:1–209:90 (2021)
14. Deb, K., Agrawal, S., Pratap, A., Meyarivan, T.: A fast elitist non-dominated sorting genetic algorithm for multi-objective optimisation: NSGA-II. In: PPSN. vol. 1917, pp. 849–858. Springer (2000)
15. Demir, C., Ngomo, A.N.: Neuro-symbolic class expression learning. In: IJCAI. pp. 3624–3632. ijcai.org (2023)
16. Falkner, S., Klein, A., Hutter, F.: BOHB: robust and efficient hyperparameter optimization at scale. In: ICML. Proceedings of Machine Learning Research, vol. 80, pp. 1436–1445. PMLR (2018)
17. Fanizzi, N., d'Amato, C., Esposito, F.: DL-FOIL concept learning in description logics. In: ILP. vol. 5194, pp. 107–121. Springer (2008)
18. Feurer, M., Klein, A., Eggensperger, K., Springenberg, J.T., Blum, M., Hutter, F.: Efficient and robust automated machine learning. In: NIPS. pp. 2962–2970 (2015)
19. Gaur, M., Faldu, K., Sheth, A.P.: Semantics of the black-box: Can knowledge graphs help make deep learning systems more interpretable and explainable? IEEE Internet Comput. **25**(1), 51–59 (2021)
20. Guan, C., Zhang, Z., Li, H., Chang, H., Zhang, Z., Qin, Y., Jiang, J., Wang, X., Zhu, W.: Autogl: A library for automated graph learning. CoRR **abs/2104.04987** (2021)
21. Guyon, I., Elisseeff, A.: An introduction to variable and feature selection. J. Mach. Learn. Res. **3**, 1157–1182 (2003)
22. Hansen, N.: The CMA evolution strategy: A comparing review. In: Towards a New Evolutionary Computation, Studies in Fuzziness and Soft Computing, vol. 192, pp. 75–102. Springer (2006)
23. He, X., Zhao, K., Chu, X.: Automl: A survey of the state-of-the-art. Knowl. Based Syst. **212**, 106622 (2021)
24. Heindorf, S., Blübaum, L., Düsterhus, N., Werner, T., Golani, V.N., Demir, C., Ngonga Ngomo, A.: Evolearner: Learning description logics with evolutionary algorithms. In: WWW. pp. 818–828. ACM (2022)
25. Huang, Q., Yamada, M., Tian, Y., Singh, D., Chang, Y.: Graphlime: Local interpretable model explanations for graph neural networks. IEEE Trans. Knowl. Data Eng. **35**(7), 6968–6972 (2023)

26. Jin, H., Song, Q., Hu, X.: Auto-keras: An efficient neural architecture search system. In: KDD. pp. 1946–1956. ACM (2019)
27. Jovic, A., Brkic, K., Bogunovic, N.: A review of feature selection methods with applications. In: MIPRO. pp. 1200–1205. IEEE (2015)
28. Kang, C., Huo, Y., Xin, L., Tian, B., Yu, B.: Feature selection and tumor classification for microarray data using relaxed lasso and generalized multi-class support vector machine. Journal of theoretical biology **463**, 77–91 (2019)
29. Khaire, U.M., Dhanalakshmi, R.: Stability of feature selection algorithm: A review. J. King Saud Univ. Comput. Inf. Sci. **34**(4), 1060–1073 (2022)
30. Kohavi, R., John, G.H.: Wrappers for feature subset selection. Artif. Intell. **97**(1-2), 273–324 (1997)
31. Kouagou, N.J., Heindorf, S., Demir, C., Ngomo, A.N.: Neural class expression synthesis. In: ESWC. vol. 13870, pp. 209–226. Springer (2023)
32. Kouagou, N.J., Heindorf, S., Demir, C., Ngomo, A.N.: Neural class expression synthesis in *ALCHIQ(D)*. In: ECML/PKDD. vol. 14172, pp. 196–212. Springer (2023)
33. Kouagou, N.J., Heindorf, S., Demir, C., Ngonga Ngomo, A.: Learning concept lengths accelerates concept learning in ALC. In: ESWC. pp. 236–252. Springer (2022)
34. Lehmann, J.: Dl-learner: Learning concepts in description logics. J. Mach. Learn. Res. **10**, 2639–2642 (2009)
35. Lehmann, J.: Learning OWL Class Expressions, Studies on the Semantic Web, vol. 6. IOS Press (2010)
36. Lehmann, J., Auer, S., Bühmann, L., Tramp, S.: Class expression learning for ontology engineering. J. Web Semant. **9**(1), 71–81 (2011)
37. Lehmann, J., Hitzler, P.: Concept learning in description logics using refinement operators. Mach. Learn. **78**(1-2), 203–250 (2010)
38. Li, L., Yang, H., Jiao, Y., Lin, K.Y.: Feature generation based on knowledge graph. IFAC-PapersOnLine **53**(5), 774–779 (2020)
39. Li, L., Jamieson, K.G., DeSalvo, G., Rostamizadeh, A., Talwalkar, A.: Hyperband: A novel bandit-based approach to hyperparameter optimization. J. Mach. Learn. Res. **18**, 185:1–185:52 (2017)
40. Liaw, R., Liang, E., Nishihara, R., Moritz, P., Gonzalez, J.E., Stoica, I.: Tune: A research platform for distributed model selection and training. CoRR **abs/1807.05118** (2018)
41. Liu, H., Motoda, H.: Computational methods of feature selection. CRC press (2007)
42. Mikriukov, G., Schwalbe, G., Hellert, C., Bade, K.: Evaluating the stability of semantic concept representations in cnns for robust explainability. In: XAI. Communications in Computer and Information Science, vol. 1902, pp. 499–524. Springer (2023)
43. Montgomery, D.C.: Design and analysis of experiments. John wiley & sons (2017)
44. Olson, R.S., Moore, J.H.: TPOT: A tree-based pipeline optimization tool for automating machine learning. In: AutoML@ICML. JMLR Workshop and Conference Proceedings, vol. 64, pp. 66–74. JMLR.org (2016)
45. Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., Desmaison, A., Köpf, A., Yang, E.Z., DeVito, Z., Raison, M., Tejani, A., Chilamkurthy, S., Steiner, B., Fang, L., Bai, J., Chintala, S.: Pytorch: An imperative style, high-performance deep learning library. In: NeurIPS. pp. 8024–8035 (2019)

46. Pearson, K.: X. on the criterion that a given system of deviations from the probable in the case of a correlated system of variables is such that it can be reasonably supposed to have arisen from random sampling. The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science **50**(302), 157–175 (1900)
47. Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., VanderPlas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., Duchesnay, E.: Scikit-learn: Machine learning in python. J. Mach. Learn. Res. **12**, 2825–2830 (2011)
48. Peng, X., Shuai, Y., Gan, Y., Chen, Y.: Hybrid feature selection model based on machine learning and knowledge graph. In: Journal of Physics: Conference Series. vol. 2079. IOP Publishing (2021)
49. Rizzo, G., Fanizzi, N., d'Amato, C.: Class expression induction as concept space exploration: From dl-foil to dl-focl. Future Gener. Comput. Syst. **108**, 256–272 (2020)
50. Santu, S.K.K., Hassan, M.M., Smith, M.J., Xu, L., Zhai, C., Veeramachaneni, K.: Automl to date and beyond: Challenges and opportunities. ACM Comput. Surv. **54**(8), 175:1–175:36 (2022)
51. Schwalbe, G., Finzel, B.: A comprehensive taxonomy for explainable artificial intelligence: a systematic survey of surveys on methods and concepts. Data Mining and Knowledge Discovery pp. 1–59 (2023)
52. Smetannikov, I., Filchenkov, A.: Melif: filter ensemble learning algorithm for gene selection. Advanced Science Letters **22**(10), 2982–2986 (2016)
53. Tang, H., Yu, J., Lin, B., Geng, Y., Wang, Z., Chen, X., Yang, L., Lin, T., Xiao, F.: Airport terminal passenger forecast under the impact of covid-19 outbreaks: A case study from china. Journal of Building Engineering **65**, 105740 (2023)
54. Tran, A.C., Dietrich, J., Guesgen, H.W., Marsland, S.: Parallel symmetric class expression learning. J. Mach. Learn. Res. **18**, 64:1–64:34 (2017)
55. Westphal, P., Bühmann, L., Bin, S., Jabeen, H., Lehmann, J.: Sml-bench - A benchmarking framework for structured machine learning. Semantic Web **10**(2), 231–245 (2019)
56. Wu, J., Chen, X.Y., Zhang, H., Xiong, L.D., Lei, H., Deng, S.H.: Hyperparameter optimization for machine learning models based on bayesian optimization. Journal of Electronic Science and Technology **17**(1), 26–40 (2019)
57. Zheng, W., Eilam-Stock, T., Wu, T., Spagna, A., Chen, C., Hu, B., Fan, J.: Multi-feature based network revealing the structural abnormalities in autism spectrum disorder. IEEE Trans. Affect. Comput. **12**(3), 732–742 (2021)
58. Zhou, K., Huang, X., Song, Q., Chen, R., Hu, X.: Auto-gnn: Neural architecture search of graph neural networks. Frontiers Big Data **5** (2022)