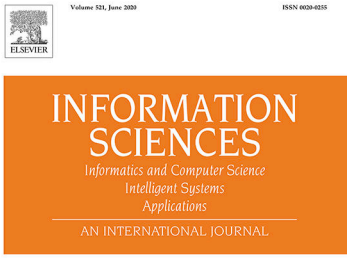

General-purpose Hierarchical Optimisation of Machine Learning Pipelines with Grammatical Evolution

Suilan Estevez-Velarde, Yoan Gutiérrez, Yudivián Almeida-Cruz, Andrés Montoyo

PII: S0020-0255(20)30698-8
DOI: https://doi.org/10.1016/j.ins.2020.07.035
Reference: INS 15673

Available online at www.sciencedirect.com
ScienceDirect

To appear in: *Information Sciences*

Received Date: 10 January 2020
Accepted Date: 10 July 2020

Please cite this article as: S. Estevez-Velarde, Y. Gutiérrez, Y. Almeida-Cruz, A. Montoyo, General-purpose Hierarchical Optimisation of Machine Learning Pipelines with Grammatical Evolution, *Information Sciences* (2020), doi: https://doi.org/10.1016/j.ins.2020.07.035

General-purpose Hierarchical Optimisation of Machine Learning Pipelines with Grammatical Evolution

Suilan Estevez-Velarde[a,*], Yoan Gutiérrez[b,c], Yudivián Almeida-Cruz[a], Andrés Montoyo[b,c]

[a] *School of Math and Computer Science, University of Havana, Cuba*
[b] *University Institute for Computing Research (IUII), University of Alicante, Spain*
[c] *Department of Software and Computing Systems, University of Alicante, Spain*

Abstract

This paper introduces Hierarchical Machine Learning Optimisation (HML-Opt), an AutoML framework that is based on probabilistic grammatical evolution. HML-Opt has been designed to provide a flexible framework where a researcher can define the space of possible pipelines to solve a specific machine learning problem, which can range from high-level decisions about representation and features to low-level hyper-parameter values. The evaluation of HML-Opt is presented via two different case studies, both of which demonstrate that it is competitive with existing AutoML tools on a variety of benchmarks. Furthermore, HML-Opt can be applied to novel problems, such as knowledge extraction from natural language text, whereas other techniques are insufficiently flexible to capture the complexity of these scenarios. The source code for HML-Opt is available online for the research community.

Keywords: AutoML, Grammatical evolution, Evolutionary computation, Supervised learning, Natural language processing

1. Introduction

The research process for solving a machine learning problem often involves experimenting with several different approaches (neural networks, supervised

*Corresponding author.
Email addresses: sestevez@matcom.uh.cu (Suilan Estevez-Velarde), ygutierrez@dlsi.ua.es (Yoan Gutiérrez), yudy@matcom.uh.cu (Yudivián Almeida-Cruz), montoyo@dlsi.ua.es (Andrés Montoyo)

classifiers, clustering algorithms, etc.). Designing an effective solution to any one of these problems often involves a large experimentation phase where researchers use different datasets, algorithms, and specific parameters. As an example, Figure 1 shows a hypothetical pipeline composed of several steps. In each step, different options are available. Suitable combinations of these options yield different values for the performance metric that is being evaluated. These hypothetical steps can range from applying some data preprocessing techniques to selecting specific algorithms and further determining the values of their hyperparameters.

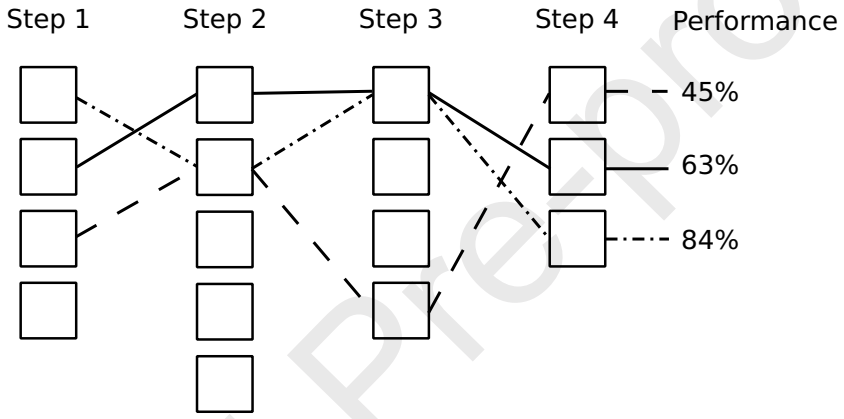

Figure 1: Illustrative example of an abstract pipeline, in which several steps can be performed in different ways, each combination obtaining a different performance (e.g, classification accuracy, recall, etc.)

Exploring all the possible combinations of algorithms and parameters for a given problem can be unfeasible, since the number of possibilities is often exponential with respect to the number of steps in a pipeline. Furthermore, when some algorithms have numerical (discrete or continuous) parameters (e.g., regularisation rate, number of neurons in a neural network layer), it is impossible to evaluate all the combinations. This problem is further complicated by the fact that each experiment may have a high computational complexity (e.g., training from scratch a neural network on a large dataset). Researchers are often compelled to select a small number of possibilities based on previous experience and domain knowledge about the problem at hand.

The automation of this lengthy experimentation process is denominated Automatic Machine Learning (AutoML). AutoML is an increasingly growing field that has been applied for finding optimal machine learning pipelines in a

variety of scenarios. As an example, in computer vision, where several neural network architectures have been extensively explored, Zoph et al. [44] applies reinforced learning to learn to build the optimal neural network architecture for any given image dataset. Likewise, general frameworks such as Auto-Sklearn [12], Auto-Weka [39], or Auto-Keras [15] have appeared, based upon existing machine learning frameworks, that automatically explore different combinations of algorithms available in such frameworks.

Current AutoML techniques focus mostly on a specific subset of algorithms, often tailored to a specific framework or tool-set. Solving complex problems, on the other hand, requires the combination of different tools that might not be available in a single framework. Besides, challenging machine learning problems are not restricted to finding the best architecture or hyper-parameter set for a given algorithm, but often involve higher-level decisions. For example, in natural language processing, prior to deciding which machine learning algorithm to use, the researcher must decide on representation (e.g., whether to use embeddings, and which ones specifically), features (e.g., whether to incorporate knowledge-based features), preprocessing (e.g., whether to remove stop-words or to apply stemming), etc. Advancing towards the automation of this process requires more expressive tools that allow a researcher to define a complex and problem-specific space of possible pipelines including everything from high-level decisions to low-level hyper-parameter ranges.

The main objective of this research is to present Hierarchical Machine Learning Optimisation (HML-Opt), an AutoML technique based on probabilistic grammatical evolution. This work extends and generalises previous research by Estevez-Velarde et al. [11]. The most important contributions of this research are:

- HML-Opt allows a researcher to define a large and complex space of possible pipelines to solve a specific problem, potentially combining different frameworks and technologies.
- HML-Opt automatically searches this space of pipelines, with given time frames and computational resources, to find an optimal or close to optimal architecture.
- Meaningful statistics and insights about the problem are extracted from the experimentation process.

- Freely available source code is provided for the research community[1].

This rest of the paper is organised as follows. Section 2 presents a brief review of the relevant concepts and related work. Section 3 introduces HML-Opt, describing its overall design and relevant implementation details. Section 4 presents two case studies involving different datasets, ranging from simple classification problems to state-of-the-art knowledge discovery problems. Finally, Section 5 discuses the main contributions and highlights of the research and the case study results, and Section 6 presents the main conclusions and future lines of research.

2. Related work

The idea of designing meta-algorithms to select the best algorithms for specific problem domains is a recurrent trend in artificial intelligence research, which has been motivated by several factors. These include: the complexity of parameter tuning in practical problems; the wide variety of existing algorithms with similar performance; and, the existence of no-free-lunch results. In the domains of continuous and combinatorial optimisation, hybrid strategies have been developed to select from different search techniques (e.g., different metaheuristics) for a specific problem [4]. Hyperheuristics [3] are high-level methodologies to select or generate strategies from an underlying space of heuristic components to solve a specific optimisation problem or class of problems. A similar formulation is developed in the field of automatic machine learning (AutoML), in which the underlying components are learning algorithms and the problem to solve is a supervised learning problem [14].

In the context of AutoML, several approaches attempt to find the best architecture for a given problem either through searching [30] or through meta-learning [23]. Different approaches exist for AutoML. Among the most relevant, we can cite: Bayesian optimisation, e.g., Auto-Sklearn [12], Auto-Weka [39], and Hyperot [22]; Neural Architecture Search (NAS), e.g., Auto-Keras [15], Zoph and Le [43], Zoph et al. [45]; Genetic Programming, e.g., Recipe [9], TPOT [31]; Hierarchical Task Networks, e.g., ML-Plan [29]; and, other metaheuristics applied to specific problem domains [35].

Several metaheuristics have been used as optimisation strategies for intelligently searching the best combination of algorithms for a given prob-

[1] https://github.com/knowledge-learning/hp-optimization

lem [19, 18]. Among the most used strategies are genetic algorithms [24] and evolutionary algorithms [35]. In this last group, grammatical evolution [10] has been employed for optimising the architecture of neural networks [5] as well as for directly finding the optimal weights [41, 37, 1]. This increases the complexity of the optimisation, because the search space is much larger. Furthermore, there are many efficient optimisation strategies specifically for tuning neural network weights using variants of gradient descent, such as SGD [2], RMSProp [40], and Adam [21]. For these reasons, our proposal attempts to optimise the architecture and high-level hyperparameters while the actual model parameters are tuned via specialised learning algorithms. Moreover, our proposal not only applies to neural networks, but in general to any machine learning pipeline that can be designed as a hierarchical process.

Hierarchical representations have shown to be effective for efficiently exploring this large space [26]. These representations impose a structure in the search space that helps to guide the optimisation by ensuring that semantically similar elements are closer in the search space. Since grammars provide an effective and simple method for researchers to define the hierarchical structure of a given space [9], our proposal uses grammatical evolution to impose a semantic structure. The majority of metaheuristic algorithms deal either with continuous or with discrete values independently because comparing solutions with a mixture of continuous and discrete values is harder [17, 42, 16]. HML-Opt has been designed to handle both continuous and discrete values, and factors in that solutions represent hierarchical pipelines. Therefore, differences between solutions at higher levels of the hierarchy have greater significance.

Table 1 shows the characteristics that define our proposal compared to some relevant alternatives in the literature. We focus fundamentally on AutoML approaches —row 4—, although we include a reference to Grammatical Evolution applied to probabilistic context-free grammars [20] because it is similar to our proposal in the theoretical formulation —first column—. We make a distinction of which techniques can deal with discrete and/or continuous hyperparameters —rows 5 and 6—. Also, we are interested in techniques that explicitly build a probabilistic description of the search space —row 3—. Finally, we highlight techniques that can be generalised across many machine learning problems over those that are specifically designed for one problem, such as supervised classification, or one learning paradigm, such as neural networks —row 7—.

Most of the AutoML approaches presented in Table 2 are black-box, in

	Features	Prob. GE	Auto-Sklearn	Auto-Weka	Real et al.	Recipe	TPOT	Auto-Keras	ML-Plan	Hyperopt	HML-Opt
1	Hierarchical	✓				✓			✓		✓
2	Metaheuristics	✓			✓	✓	✓				✓
3	Space description	✓	✓	✓						✓	✓
4	AutoML		✓	✓	✓	✓	✓	✓	✓	✓	✓
5	Continuous		✓	✓		?		✓		✓	✓
6	Discrete	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
7	General-purpose	✓				✓			✓	✓	✓
8	Year	2015	2015	2013	2017	2017	2018	2016	2018	2014	2019

Table 1: The characteristics of our proposal compared to related work. A question mark ("?") indicates that we could not find evidence for that characteristic in the corresponding paper.

the sense that they define a single generic machine learning pipeline which is parameterized by a fixed subset of algorithms and their hyperparameters. These approaches are difficult to apply to problems which are not directly framed as classic supervised learning from an input feature matrix (eg., NLP problems where text preprocessing must also be taken into consideration). A notable exception is the case of Hyperopt [22], which provides a general-purpose framework that can describe many different pipelines, and the authors provide one pipeline, specifically tailored for `scikit-learn` algorithms. Likewise, in the case of Recipe [9], the authors predefine different grammars with fixed sets of algorithms, making the proposal easy to adapt to other scenarios. A recent technique is ML-Plan [29], which proposes the use of Hierarchical Task Networks as a means for representing the search space, and two specific implementations, based either on `weka` or `scikit-learn`. HML-Opt is also designed as an extensible framework that can be tailored to any specific subset of algorithms, such as `sklearn` or `keras`. However, in contrast with other approaches, HML-Opt takes this extensibility a step further, by giving the user the possibility to completely define the space of possible pipelines in an intuitive, declarative language, i.e., with a context-free grammar. We illustrate this extensibility in Section 4.2 with an application to a domain-specific machine learning problem with key restrictions about the possible pipelines.

In terms of representation, the most relevant related works are Recipe [9]

and ML-Plan [29] which also define hierarchical search spaces. More specifically, Recipe also employs a grammar to define the set of possible pipelines. One key difference in our work is the use of probabilistic grammatical evolution, as opposed to genetic programming in Recipe, which provides a probabilistic model of the space of pipelines. In ML-Plan the search algorithm is a modified variant of Monte Carlo tree search designed to deal with the large computational cost of evaluating machine learning pipelines. Other AutoML proposals based on Bayesian optimisation (e.g., Auto-Sklearn and Auto-Weka) also define a probabilistic model of the pipelines, even though it is not explicitly used to gather insights of the optimisation process. HML-Opt leverages the use of grammars for defining a flexible and adaptable structure of pipelines, and the use of a probabilistic model which enables analysis and understanding of the optimisation process for a specific machine learning problem.

3. Hierarchical Machine Learning Optimisation

This section presents Hierarchical Machine Learning Optimisation (HML-Opt), a technique that searches for the optimal pipeline for a specific machine learning problem. The core of our proposal is a process that involves three different parts. First, the researcher designs a grammar that defines the space of all possible pipelines that are interesting to analyze for a given machine learning problem. (see Section 3.1). This grammar allows the generation of different pipelines which can be evaluated for the problem at hand (see Section 3.2). Finally, an iterative optimisation algorithm continuously generates and evaluates pipelines searching for the optimal combination (see Section 3.3). Figure 2 presents a high-level representation of this process.

In HML-Opt, the grammar that defines the search space is a probabilistic context-free grammar [6] provided by the researcher, that represents the possible choices for algorithms and hyperparameters in a machine learning pipeline. The grammar includes a probability distribution σ, which is initially uniform for each of the decisions. A pipeline is obtained by sampling the grammar using this probability distribution. The optimisation process is based on probabilistic grammatical evolution [20], designed to incrementally adjust the probability distribution based on the marginal distribution of the best pipelines in each iteration. With a sufficiently large number of iterations, this process converges towards generating the best pipelines [13].

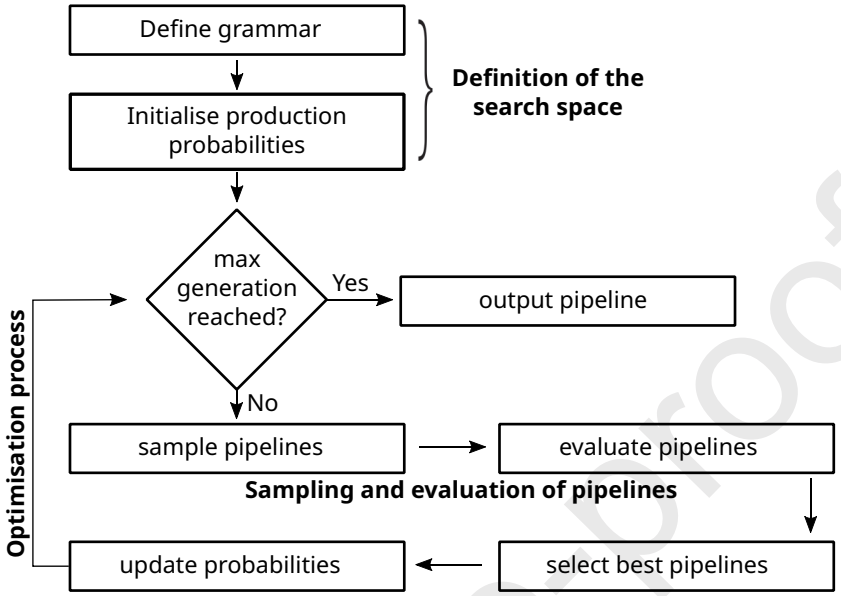

Figure 2: High-level representation of the Hierarchical Machine Learning optimisation process.

3.1. Definition of the search space

We define the search space for a specific machine learning problem as the set of all possible pipelines that solve the problem. Two different pipelines can use different algorithms for one specific step, or can use the same algorithm but with different values for some of its hyperparameters. Also, some steps can be combined under a higher-level step, forming a hierarchy of steps. For example, in NLP, a general "preprocessing" step can be composed of substeps such as "cleaning" and "dimensionality reduction", which are in turn composed of more specific steps, until reaching algorithms and hyperparameters. To represent all possible pipelines for a given problem, and capture their hierarchical nature, we propose the use of context-free grammars. The grammar also defines the importance of each step of the pipelines. High-level decisions (e.g., using linear or non-linear classifiers) are represented closer to the start symbol of the grammar, while low-level decisions (e.g., the value of a regularisation factor) are closer to the end of the grammar. This way the researcher imposes a structure in the search space in which different pipelines can be compared.

As an illustrative example, the grammar in Figure 3 defines a small

```
(1)  <Pipeline>:= <Prep> <Vect> <Reduct> <Class>
(2)  <Prep>     := yes | no
(3)  <Vect>     := TF (<tf-ngram>) | CV (<cv-ngram>)
(4)  <tf-ngram>:= i(1,3)
(5)  <cv-ngram>:= i(1,3)
(6)  <Reduct>   := none | SVD
(7)  <Class>    := NB | LR (<reg>, <penalty>) | SVM (<kernel>)
(8)  <reg>      := f(0.01,10)
(9)  <penalty>  := l1 | l2
(10) <kernel>   := linear | rbf
```

Figure 3: Grammar defining a set of pipelines for text classification. Each production represents a choice of the pipeline, which can be between different algorithms or their parameters.

search space to solve a text classification problem. The pipelines defined include: a text preprocessing phase —`Prep`—; different vectorial representations —`Vect`—; an optional dimensionality reduction —`Reduct`—; and, different classifiers —`Class`—. The preprocessing step —production 2— consists of an optional application of stop-words removal. The vectorisation step —production 3— consists of a bag-of-word representation either using counts —`CV`— or TF-IDF weights —`TF`—. In each case, the grammar also specifies an n-gram size, an integer value between 1 and 3 —productions 4 and 5—. Dimensionality reduction —production 6—, if applied, uses a standard singular value decomposition (SVD). For classification —production 7—, three algorithms are available, either naive bayes —`NB`—, logistic regression —`LR`— or support vector machines —`SVM`—. In the case of logistic regression —production 8—, two different regularisation functions are available, either L_1 or L_2, and in each case the regularisation factor is allowed to take values from 0.01 to 10 —production 9—. Finally, in the case of SVM, either a linear kernel or a radial basis function kernel (RBF) is used.

3.2. Sampling and evaluation of pipelines

The grammar defined has an attached probabilistic model σ, initialised as a collection of uniform distributions for each production of the grammar. Hence, every production has an assigned probability distribution π_i with parameters that change in time (as explained in Section 3.3). From this model σ a random pipeline is sampled using a top-down process. Starting at the root of the grammar (`Pipeline`), one of the available options is randomly

chosen according to the corresponding distribution π_0. In the exemplary grammar shown in Figure 3, the initial production has only one option. Then, for each symbol in the selected production, the same sampling process is applied recursively. For example, in the case of production 7 —`Class`—, considering initial uniform probabilities, one of the three options (i.e., naive Bayes, logistic regression or SVM) is randomly selected. If naive Bayes —`NB`— is selected, the sampling process ends. However, if logistic regression —`LR`— is selected, then the sampling process proceeds recursively to production 8. Figure 4 shows an example, illustrating the top-down sampling process and the final pipeline obtained for an arbitrary random selection.

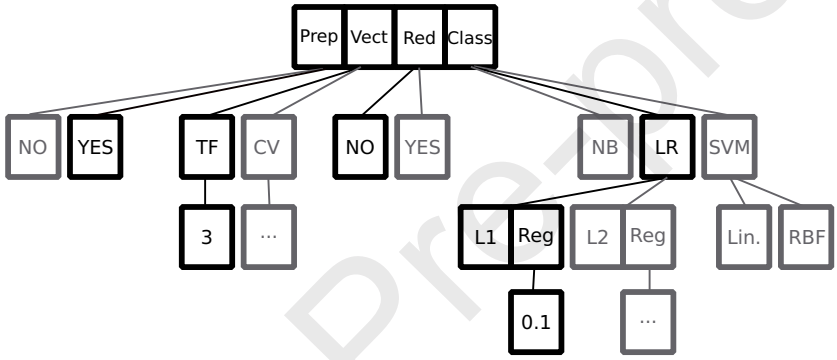

Figure 4: Sampling a pipeline from the exemplary grammar. The path that defines this specific pipeline is highlighted in **bold**.

During the top-down sampling process, each production of the grammar is interpreted with a different probabilistic distribution, according to its type. Productions can represent categorical decisions (i.e., one out of many), discrete or continuous values for hyper-parameters, or Boolean decisions (actually a special case of categorical decisions, but treated separately for convenience). According to the intended production type, the following distributions are used:

Boolean values: A Bernoulli distribution is used, with a parameter p that models the probability of obtaining `True`. A random value is generated by sampling from a random uniform value $u \in [0, 1]$, and the Boolean output is conditioned to $u \geq p$. In the previous example of text opinion mining, we can decide whether to delete stop-words.

Categorical values: Similar to the previous case, a multinomial Bernoulli

distribution is used, with parameters p_i, one for each option. When sampling, a random uniform value $u \in [0, 1]$ is generated and the p_i weights are accumulated until summing u, i.e., the result is the largest index k such that $\sum_{i=0}^{k} p_i \leq u$. This type of value is used for representing a selection between a finite number of options where order is irrelevant, and only their relative weights are considered. For example, in the case of text opinion mining, we can decide between different classifiers: logistic regression, naive Bayes or support vector machines.

Continuous values: A normal distribution with mean μ and variance σ is used, plus two values a, b that correspond to the interval limits. When sampling, a Gaussian variable $N(\mu, \sigma)$ is generated (via the inverse method) and clamped to the limits a, b defined in the production. This type of value is used for representing continuous values. For example, in the case of text opinion mining we can decide the value of the regularisation parameter for logistic regression.

Discrete values: A normal distribution is also used as in the previous case, and the end results are rounded to the nearest integer. This distribution encourages sampling to converge to a contiguous range of values as the search progresses, something that is not possible with a traditional multinomial uniform distribution. This type of value is used for representing a selection between a finite number of options ordered by their numeric value. For example, in the case of text opinion mining, we can decide between different n-gram lengths: uni-grams, bi-grams or tri-grams.

Every pipeline sampled from the grammar is assigned a fitness value according to the machine learning problem at hand. For example, in the case of supervised learning, each pipeline presumably involves a classification or regression algorithm at some steps. Thus, each pipeline will be be trained and evaluated on a suitable dataset, measuring precision, F_1, R^2, or another relevant metric. Even though statistically significant comparisons require several cross-validation iterations for each pipeline, a smaller number of evaluations per pipeline suffices during the optimisation process. This is due to the stochastic nature of the optimisation process explained in Section 3.3, which will tend to smooth away small random differences in pipeline fitness. Inspired by similar considerations in the design of ML-Plan [29], we recommend between 3 and 10 Monte Carlo cross-validation evaluations per pipeline.

3.3. Optimisation process

The optimisation process is based on a continuous optimisation technique with Grammatical Evolution for probabilistic context-free grammars by Kim and Ahn [20]. The process consists of a generation and evaluation loop using a grammar G suitable for the machine learning problem at hand, as defined in Section 3. In each iteration, a number of pipelines N is generated by sampling the grammar G according to the probabilities σ assigned to each production. These probabilities are initiated with a uniform distribution σ_0 across all productions. Every pipeline is evaluated (which may consist of a single training/test execution, or a more complex cross-validation), and the best performing pipelines are used to modify σ, so as to maximise the probability of generating them.

A learning factor $\alpha \in [0, 1]$ is used to interpolate between the previous distribution σ and the marginal distribution σ^* computed from the best pipelines. This learning factor controls the balance between the exploration and exploitation. A large α changes σ more rapidly, increasing exploitation, while a smaller value promotes more exploration. Recommended values for this parameter are 0.01 to 0.05. Algorithm 1 summarises the optimisation process for an arbitrary grammar G.

4. Results

This section presents two different evaluation scenarios to demonstrate the applicability and effectiveness of our proposal. The first scenario (Section 4.1) presents a comparison between HML-Opt and five AutoML frameworks in several standard machine learning problems. The purpose of this scenario is to demonstrate that HML-Opt can be used as an out-of-the-box AutoML tool that is competitive with the state-of-the-art. The second scenario (Section 4.2) applies HML-Opt to a complex knowledge discovery problem that includes not only selecting different algorithms, but also modelling high-level tasks that can be performed in different orders. The purpose of this scenario is to show the performance of HML-Opt in a real-life application, with a much larger search space, that includes NLP tasks, several shallow classifiers, neural networks, and different semantic features. Existing AutoML tools are not sufficiently expressive to model the space of possible pipelines required in this task. Hence, in this scenario, HML-Opt can only be compared with human-designed pipelines.

Algorithm 1 Grammatical Evolution for HML-Opt

$N \leftarrow$ size of population
$n \leftarrow$ number of individuals selected each iteration
$\alpha \leftarrow$ learning factor
$G \leftarrow$ grammar for specific machine learning pipeline
$\sigma_0 \leftarrow$ initial probabilities (uniform)
$f \leftarrow$ fitness function (training and evaluation of a pipeline)
Best $\leftarrow \square$

for each iteration i **do**
  $P_i \leftarrow$ sample population from G, σ_{i-1}
  for each solution $S \in P_i$ **do**
    $f(S) \leftarrow$ calculate fitness of S (evaluate pipeline)
  end for
  Best $\leftarrow \text{argmax}_{S \in P \cup \{Best\}} \{f(S)\}$
  $P_i^* \leftarrow$ best n individuals of P_i
  $\sigma_i^* \leftarrow$ compute marginal distribution of P^*
  $\sigma_i \leftarrow \alpha\sigma_i^* + (1-\alpha)\sigma_{i-1}$
end for
return Best

4.1. Comparison with other AutoML frameworks

One practical use of the AutoML paradigm is the design of out-of-the-box solutions that can be readily applied to a large number of similar problems. Using the `scikit-learn` library [33], we define a grammar that incorporates several different classifiers and preprocessing steps. This grammar is in part inspired by the design of Auto-Sklearn [12] and includes 4 data preprocessing steps, 7 different feature preprocessing techniques and 17 different classification algorithms. With this grammar, we implemented a `scikit-learn` compatible classifier that, in the same spirit of Auto-Sklearn, automatically chooses the best combination for a black-box classification problem. The grammar and complete source code to replicate the experimental setup for this example is available online[2]. Figure 5 shows a graphical representation of the pipeline space defined by the grammar, not taking into consideration

[2] http://bit.ly/hml-opt-sklearn

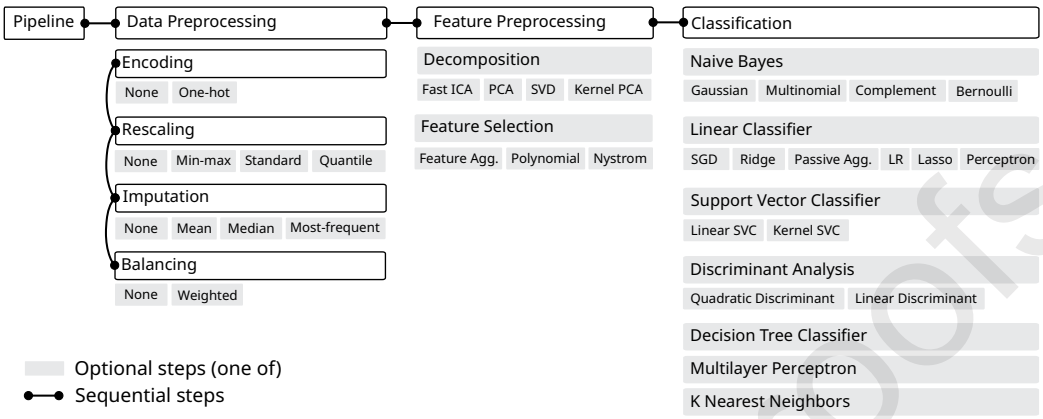

Figure 5: Graphical representation of the pipeline space defined by the *scikit-learn*-inspired grammar. Possible pipelines consist of three steps: data preprocessing, feature preprocessing and classification. Each step is recursively composed of sequential or optional sub-steps.

the hyperparameters of each technique.

Table 2 shows a comparison of our proposal, HML-Opt (Sklearn), to other similar AutoML frameworks, for different classic datasets. This experimental setup is based on the results reported by Mohr et al. [29], and closely follows their proposed methodology. Specifically, we report the mean accuracy across 20 independent executions of HML-Opt (Sklearn) in each dataset, with a random 70% of the data for optimisation and the remaining 30% as a hold-out test set, where the final accuracy is measured. Each execution has a timeout of one hour, and performs 5 Monte Carlo cross-validation evaluations (70%/30%) for each pipeline, with individual timeouts of 5 minutes per evaluation. In all cases the population size is 20 with a selection of the best 5 and a learning rate of 0.05.

The objective of this comparison is to show that our approach can be used in the same manner as other AutoML frameworks to design a high-performing black-box classifier, capable of obtaining competitive results. Three datasets from the UCI repository [25] are used in this evaluation: the *Cars* dataset [46] is a multivariate classification problem with $1,728$ examples and 6 categorical features; the *Credit G.* dataset[3] is also a multivariate classification problem with $1,000$ examples and 20 categorical or integer features; and, the *Wine*

[3] https://archive.ics.uci.edu/ml/datasets/statlog+(german+credit+data)

Dataset	Cars	Credit G.	Wine Q.
ML-Plan (Weka)	98.73 (±0.56)	74.46 (±1.28)	67.38 (±0.91)
Auto-WEKA	99.34 (±0.38)	73.50 (±2.32)	66.31 (±1.90)
ML-Plan (Sklearn)	99.66 (±0.51)	75.44 (±2.53)	67.47 (±1.31)
Auto-Sklearn-v	98.62 (±0.67)	74.05 (±1.89)	63.17 (±1.27)
Auto-Sklearn-we	98.74 (±0.53)	74.61 (±0.88)	64.13 (±1.18)
TPOT	99.63 (±0.33)	76.09 (±2.22)	67.06 (±1.09)
HML-Opt (Sklearn)	**99.92** (±0.11)	75.03 (±2.55)	59.99 (±2.74)

Table 2: Comparison of HML-Opt to other AutoML frameworks for three classic machine learning datasets in terms of mean and standard deviation of accuracy. Values for other frameworks were obtained from Mohr et al. [29]. Our results are highlighted in **bold** or underlined if they are statistically significantly **larger** or smaller than the rest, respectively, according to a Student's t-test with a p-value < 0.05.

Q. dataset [8] is a multivariate classification problem with $4,898$ examples and 12 continuous features.

Using the same underlying grammar and optimisation parameters for all datasets, our approach obtains the best result in the Cars dataset, a statistically similar result in the Credit German dataset and a lower result in the Wine Quality dataset. In the latter dataset, we noticed that the one hour timeout was not sufficient for the evolutionary optimisation to achieve convergence, resulting in less than optimal performance. Since the search space and underlying algorithms in HML-Opt (Sklearn) are very similar to two of the comparisons that also use `scikit-learn`, it can be inferred that the differences in performance are due to differences either in the grammar definition or in the exploration of the possible pipelines. These results demonstrate that HML-Opt can be used as an effective out-of-the-box AutoML tool in several machine learning scenarios.

4.2. Knowledge Extraction in Spanish eHealth text

In this section we present an example of the use of HML-Opt for solving a more complex machine learning problem as has been described by Estevez-Velarde et al. [11]. The problem selected is the eHealth Knowledge Discovery challenge presented in the TASS 2018 workshop [28]. The purpose of this case study is to show how HML-Opt can be extended beyond typical black-box machine learning problems and applied to complex domain problems, with the design of a suitable problem-specific grammar.

In the eHealth-KD challenge, participants were asked to submit a system that can identify relevant key phrases in a set of Spanish language documents, classify them into two categories, and then detect semantic relations among them. The problem is divided into three different subtasks, that can be performed sequentially, or combined in different ways: (A) detecting entities in text; (B) classifying them; and (C) detecting semantic relations among them. Figure 6 shows an example of sentences from the challenge, with their corresponding annotations. For this problem, a corpus with $1,173$ sentences is provided [34] with a total of $13,113$ examples, of which $3,310$ are used for evaluation, and the rest for training. Results for this challenge show that a variety of techniques obtain competitive results, including: classic machine learning; deep learning; natural language processing; and, knowledge-based approaches. An effective exploration of the space of suitable pipelines requires mixing technologies from different software frameworks and knowledge areas. This makes the eHealth-KD challenge a complex scenario where existing AutoML tools that provide out-of-the-box classifiers cannot be directly applied without considerable customisation.

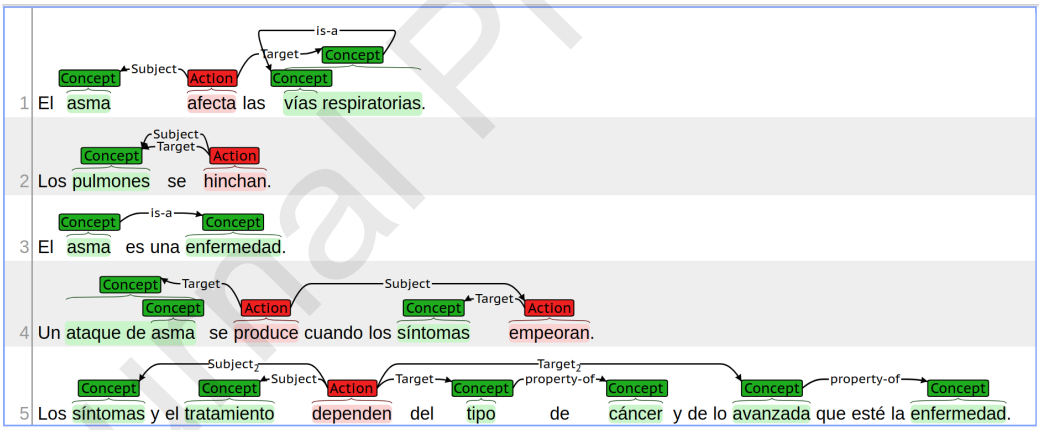

Figure 6: Example of annotated sentences in the eHealth-KD challenge. Adapted from Piad-Morffis et al. [34].

For this problem, we define a grammar that comprises several possible pipelines submitted to the official challenge evaluation. However, given the length of the grammar, it is not reported in full in this work although some insights on its characteristics are subsequently discussed . Our grammar considers all the 4 possible options for performing the three tasks sequentially

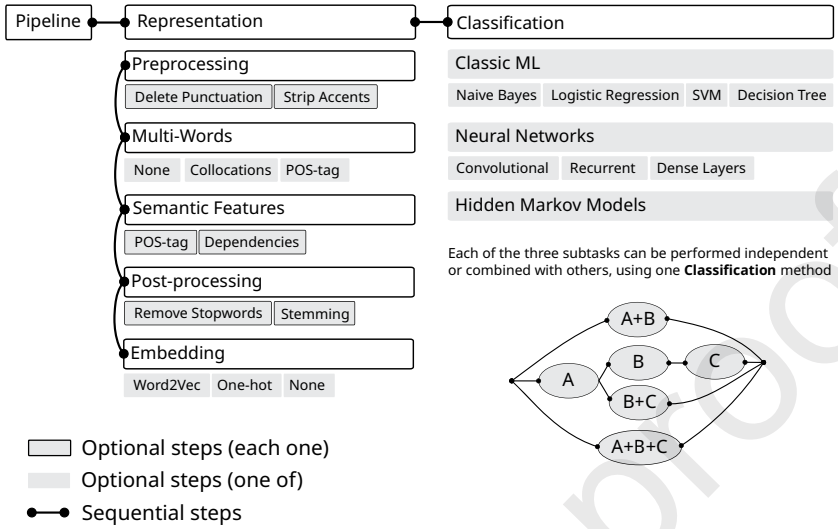

Figure 7: Graphical representation of the pipeline space defined by the eHealth-KD grammar. Possible pipelines consist of a representation step and a classification step where each subtask (i.e., A, B, and C) can be performed either independently or combined with each other.

combined. In terms of features, several options are defined, including syntactic features, word embedding techniques and knowledge-based features using the `nltk` and `gensim` libraries. For the classification part we include both classic machine learning techniques from `scikit-learn` as well as deep learning architectures from `keras` [7]. The grammar and complete source code for this case study is available online[4]. Figure 7 shows a graphical representation of the pipeline space defined by this grammar. Due to the hierarchical nature of the grammar, not all details can be captured in this figure.

The optimisation process was performed for a total of 60 generations, with a population size of 50 pipelines per generation, a selection factor of the best 20% and a learning factor of 0.05. The average time required to evaluate each pipeline was 5.17 minutes for a total of approximately 257 hours of computation in a single computational node of commodity hardware. Long-running pipelines were stopped after a 10 minute timeout. The best performing pipeline was encountered at the eighteenth generation.

Table 3 shows the best results obtained by HML-Opt for this problem.

[4] http://bit.ly/hml-opt-ehealthkd

For comparative purposes, the results obtained by other authors are taken from the TASS 2018 Overview [28]. We only evaluate HML-Opt on the proposed evaluation scenario that comprises all three subtasks. Hence, only authors that participate in this scenario are included. As Table 3 shows, in this problem HML-Opt obtains results competitive with the state-of-the-art.

Techniques	**System**	F_1
AutoML (NLP+LR+DT+SVM)	**HML-Opt (TASS)**	**75.4**
Bi-LSTM + CRF	Extended NeuroNER [36]	74.4
Knowledge-based	SINAI [27]	71.0
NLP + CRF + LR	UPF-UPC [32]	68.1
Convolutional NN	LABDA [38]	31.0

Table 3: Comparison of HML-Opt to other techniques used on the eHealth-KD dataset, specifically in the evaluation scenario No. 1. Following the dataset evaluation methodology, a single evaluation on the official test set was performed.

Interestingly, the final pipeline obtained by HML-Opt in this example is composed of classic algorithms: SVM, logistic regression and decision trees. In terms of representations, HML-Opt selected one pipeline that included one-hot encoding with bi-grams, stemming, and POS-tagging as an additional syntactic feature. By contrast, other authors rely on deep learning architectures, word and character embedding, and external knowledge bases. The simple but effective pipeline discovered by HML-Opt shows that an adequate optimisation and automatic parameter tuning can out-perform more complex but insufficiently tuned strategies that are hand-designed.

It is important to emphasise that in this case study it is impossible to compare with other AutoML tools since, to the extent of our knowledge, none of the existing technologies allow for the degree of versatility necessary in this NLP problem. To solve the aforementioned problem, we need to combine technologies from multiple machine learning frameworks, e.g., embeddings from `gensim`, structured features from `sklearn` and learning architectures from `keras`. HML-Opt leverages expert domain knowledge expressed in the form of a problem-specific grammar, and is thus able to combine a variety of tools and frameworks in a non-trivial manner to produce pipelines that are specifically designed for this problem.

5. Discussion

In this section, we present a high-level analysis of the experimental results and discuss key characteristics of HML-Opt. Unlike other proposals, our technique is not restricted to particular types of pipelines, such as neural networks, or shallow classifiers. Since the grammar defines the space of possible experimentation, anything can be included, such as natural language preprocessing techniques or knowledge bases. Our proposal optimises at many different levels, from high-level decisions about classes of algorithms to low-level decisions such as hyperparameters (e.g., regularisation factors).

The fundamental basis of our proposal is that researchers can define a problem-specific grammar that encodes their domain knowledge about a problem. This way, researchers define which pipelines are interesting to explore, while HML-Opt evaluates and searches for the best pipeline. However, for classic problems, or in a real life scenario, having access to an out-of-the-box classifier can be desirable. In these cases, our approach can be used similar to other AutoML frameworks, to predefine a set of "black-box" solutions, i.e., problem-specific grammars that are tailored for specific problems. Researchers or practitioners can start by using these black-box solutions, but still have the option to take a look inside the black-box and modify the grammar for their specific purposes, where necessary.

The next step in the automation of machine learning would be to automate the process of defining a suitable grammar for a specific problem. In our proposal, researchers have to manually design the appropriate grammar based on their knowledge and previous experience about the problem. However, analysing the case studies presented in Section 4, we observed that although the two grammars are different, they also share similarities. For example, they consist of an initial production that defines high-level steps, which in turn are defined into simpler steps, and so on, down to algorithms and hyperparameter values. Furthermore, they all define some sort of preprocessing and representation steps, even if they are named differently and composed of different sub-steps, and a final classification step. These and other regularities could lay the foundation for building a generic knowledge base of machine learning techniques that would help automate the definition of problem-specific grammars.

Approaching the AutoML problem from the perspective of its underlying optimisation model, two interesting considerations become apparent: the hierarchical nature of the solutions and the multi-objective nature of the

performance metric. Our proposal explicitly represents solutions as a hierarchical decision process, in the same way as other some alternative AutoML approaches (i.e., Recipe and ML-Plan). This allows modelling and accounting for the fact that some decisions have a greater influence on the pipeline optimisation, for instance, selecting between linear, non-linear or tree-based classifiers supersedes selecting specific hyperparameter values. We argue that as machine learning pipelines grow in complexity and involve algorithms from different libraries and technologies, a hierarchical conceptualisation will be even more relevant. With respect to the performance metric, most current AutoML approaches focus on a single objective function, e.g., accuracy, precision, F_1, in correspondence with the machine learning problem at hand. However, in practical scenarios, it may be necessary to balance different performance metrics, including also time and memory usage, and more subjective qualities such as explainability of the models. The simplest approach (taken by HML-Opt and others) of optimising a main metric subject to restrictions of time and memory is insufficient in a scenario where the end-user has to decide about practical issues such as deploying these pipelines in a production system. TPOT considers this problem from the multi-objective approach by jointly optimising both accuracy and model complexity (in terms of pipeline length). However, future approaches will need to consider this issue in-depth, potentially including the end-user in the pipeline evaluation loop.

Analysing the evolution of the performance of pipelines provides some insight on the complexity of a given machine learning problem. As an illustrative example, Figure 8 (left) shows the evolution of the average and optimal accuracy in the case study presented in Section 4.2. The average accuracy steadily increased, evidencing the optimisation process. The relatively low average accuracy at the beginning is in large part influenced by the existence of *invalid pipelines*. Invalid pipelines are encountered in two cases: when a pipeline exceeds a timeout predefined by the researcher, or when unpredictable run-time errors occur, such as out-of-memory errors provoked by an unfeasible combination of hyperparameters. These circumstances are often impossible to predict by the researcher beforehand and as such cannot be accounted for in the grammar. Invalid pipelines are scored "0" for fitness, to steer the optimisation process away from them. As Figure 8 (right) shows, the number of invalid pipelines decreases steadily, which is an indication that the probabilistic model is penalising the characteristics that appear consistently in invalid pipelines.

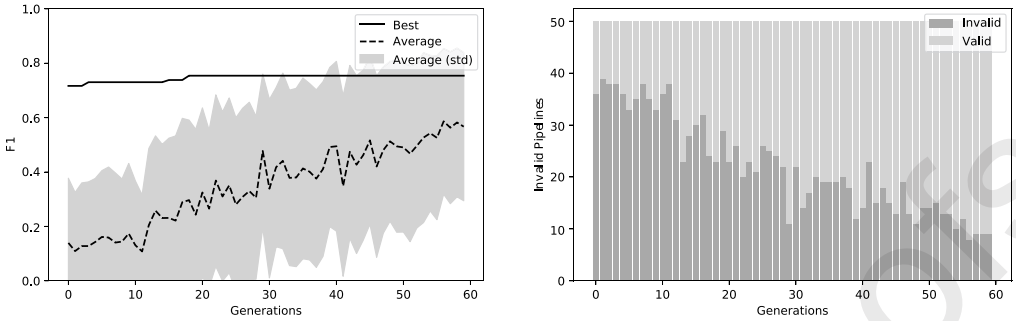

Figure 8: **Left:** Evolution of the average and optimal accuracy of pipelines in each iteration. **Right:** Proportion of invalid vs. valid pipelines found in each iteration. Results are from the case study presented in Section 4.2.

During the optimisation process a significant amount of data is generated in each iteration about the best pipelines and their characteristics. In principle, this information is used to update the probabilistic model so that new sampled pipelines are similar to the best pipelines already evaluated. However, this information can provide additional insights when aggregated for the whole optimisation process, by analysing the characteristics (i.e., the productions of the grammar) that consistently appear in the best pipelines.

Specifically for the case study presented in section 4.2, we analysed the classification algorithms used in each subtask by training a linear regression on all the valid pipelines evaluated during the optimisation process to predict the accuracy of a pipeline given its characteristics. The features used in the linear regression are obtained directly from each pipeline's derivation tree by considering all the decisions involved in a pipeline. Every path from the root to a leaf in the derivation tree becomes a feature. As an example, if one specific pipeline applies Logistic Regression with a regularisation factor of 0.3 for subtask A, then a feature `A.Classifier.LR.regularisation=0.3` is generated. Our grammar involves a total of 242 such features, which account for all possible decisions for each pipeline, from high-level task ordering to hyperparameter values.

The coefficients assigned to each characteristic indicate which characteristic has a positive or negative impact on the pipeline's fitness. Rather than exclusively analysing the best pipeline, these coefficients indicate general trends about which algorithms are consistently more effective in the majority of pipelines. For instance, Table 4 shows these coefficients only

for the classification algorithms used in each subtask, aggregated across all hyperparameter combinations. In this case, the best pipeline uses Logistic Regression, SVM and Decision Tree as classifiers (see Section 4.2). However, Table 4 shows that Neural Networks, on average, tend to outperform other algorithms, even though no specific pipeline using Neural Networks was the optimum.

	Coefficients by subtask		
Algorithm	**A**	**B**	**C**
Logistic Regression	0.0014	0.0074	-0.0127
Neural Networks	**0.0361**	**0.0320**	-0.0009
Support Vector Machines	-0.0157	-0.0221	0.0018
Decision Tree	0.0165	0.0223	**0.0024**
Naive Bayes	0.0222	0.0128	-
Markov Models	-0.0081	-	-

Table 4: Relevance of the classification algorithms in each subtask (A, B and C) in the case study presented in Section 4.2. The highest coefficient in each subtask is highlighted. Missing values correspond to combinations that were not evaluated in any valid pipeline.

A similar analysis can shed light on the relevance of the NLP intermediate tasks. Table 5 shows a subset of the features corresponding to the representation phase and their corresponding coefficients. The most relevant tasks appear to be the computation of POS-tag and dependency labels, while the use of embeddings, counter-intuitively seems to provide a marginal negative effect. Interestingly, the optimal solution found by HML-Opt does not necessarily comprise the components with the highest weights. For example, the optimal solution applies the `postag` strategy for multi-word representation and performs stemming but does not perform stop-words removal, directly contradicting what these coefficients suggest. Researchers can use these insights to better understand the characteristics of the problem and design more effective pipelines. However, caution must be exercised, since this analysis rests on the assumption that different decisions in a machine learning pipeline have an independent impact on the overall performance of the model, which can be characterised by a simple linear relationship.

The analysis presented in this section is only illustrative of the type of insights that can be obtained when both a probabilistic model and a hierarchical representation of a machine learning pipeline are available. This

NLP Subtask	Strategy	Coefficient
Embedding	none	**0.015**
Embedding	onehot	-0.030
Embedding	wordVec	0.006
MultiWords	collocations	-0.032
MultiWords	none	**0.014**
MultiWords	postag	0.009
PostProcessing.Stemming	no	**-0.003**
PostProcessing.Stemming	yes	-0.005
PostProcessing.StopWords	no	-0.010
PostProcessing.StopWords	yes	**0.001**
Preprocessing.DelPunt	no	**-0.001**
Preprocessing.DelPunt	yes	-0.008
Preprocessing.StripAccents	no	**0.001**
Preprocessing.StripAccents	yes	-0.010
SemanticFeatures.Dependencies	no	-0.013
SemanticFeatures.Dependencies	yes	**0.003**
SemanticFeatures.PosTag	no	-0.025
SemanticFeatures.PosTag	yes	**0.016**

Table 5: Overall relevance of the representation features for the case study presented in Section 4.2. The highest coefficient for each feature is highlighted.

information can also be used to design more intelligent search procedures that employ past experience in similar problems. For example, the probabilistic model can be initialised with a non-uniform distribution proportional to the weights obtained in previous experimentation. Furthermore, by designing grammars with productions that represent high-level characteristics of the pipelines (e.g., separating linear and non-linear classifiers), this analysis can also provide high-level insights on which characteristics are more important in specific problem domains. An in-depth analysis of this experimental setup has already been published [11].

In terms of performance, the cost of executing HML-Opt is dominated by the cost of training and evaluating each individual pipeline. Since pipelines

for machine learning problems are often computationally intensive, the additional cost of updating the probabilistic model and the bookkeeping associated with the optimisation process is negligible. Several strategies can help reduce the computational cost. For instance, since each generation performs N independent pipeline evaluations, HML-Opt can benefit from the implementation of parallel and distributed computing strategies. Another cost saving strategy consists of monitoring some convergence metrics of the underlying probabilistic model to provide an early stopping mechanism. However, we believe that a meta-learning strategy that learns from previous experience to seed the probabilistic model can provide even higher cost saving benefits by comparing the new dataset that requires solving to previously solved problems. Such a strategy would leverage every past execution of HML-Opt, potentially even by other researchers, as every pipeline evaluation ever made will work to more efficiently solve all future experiments.

6. Conclusions

This paper presents Hierarchical Machine Learning Optimisation (HML-Opt), an AutoML framework for automatically finding a close to optimal pipeline in a specific machine learning problem. This novel technique allows researchers to evaluate a much higher number of experimental setups than what is manually possible, given specific time frames and computational resources. Moreover, a key and innovative design feature of HML-Opt is to provide a declarative and expressive framework whereby a researcher can define the desired space of possible pipelines and explore it effectively and efficiently. This contrasts with existing alternatives that tend to force the researcher into a predefined set of algorithms or possible architectures for pipelines. Experimental evaluations indicate that our proposal is competitive with other approaches for AutoML, while at the same time applicable to novel scenarios that require integrating tools from different machine learning frameworks.

AutoML techniques like the one presented in this research are a staple in the current trend of democratising machine learning technologies. The tools continually evolve to provide more powerful ready-to-use machine learning pipelines. However, AutoML systems should not only attempt to replace human experts, but rather to serve as complementary tools that enable researchers and practitioners to quickly obtain better baselines and insights

on the most promising strategies. Armed with AutoML, we envision a future where people and computers work together in solving the most pressing problems of mankind.

Acknowledgements

Funding: This research has been supported by a Carolina Foundation grant in agreement with University of Alicante and University of Havana. Moreover, it has also been partially funded by both aforementioned universities, the Generalitat Valenciana (*Conselleria d'Educació, Investigació, Cultura i Esport*) and the Spanish Government through the projects LIVING-LANG (RTI2018-094653-B-C22) and SIIA (PROMETEO/2018/089, PROMETEU/2018/089).

This manuscript has been greatly improved by the valuable comments of anonymous reviewers.

[1] F. Assuncao, N. Lourenco, P. Machado, and B. Ribeiro. Automatic generation of neural networks with structured Grammatical Evolution. In *2017 IEEE Congr. Evol. Comput. CEC 2017 - Proc.*, pages 1557–1564. IEEE, 2017. ISBN 9781509046010. doi: 10.1109/CEC.2017.7969488.

[2] L. Bottou. Large-scale machine learning with stochastic gradient descent. In *Proc. COMPSTAT'2010*, pages 177–186. Springer, 2010.

[3] E.K. Burke, M. Gendreau, M. Hyde, G. Kendall, G. Ochoa, E. Özcan, and R. Qu. Hyper-heuristics: A survey of the state of the art. *J. Oper. Res. Soc.*, 64(12):1695–1724, 2013. ISSN 14769360. doi: 10.1057/jors.2013.71. URL https://doi.org/10.1057/jors.2013.71.

[4] F. Caraffini, F. Neri, and M. Epitropakis. HyperSPAM: A study on hyper-heuristic coordination strategies in the continuous domain. *Inf. Sci. (Ny).*, 477:186–202, 2019. ISSN 00200255. doi: 10.1016/j.ins.2018.10.033. URL http://www.sciencedirect.com/science/article/pii/S002002551830851X.

[5] A.R. Carvalho, F.M. Ramos, and A.A. Chaves. Metaheuristics for the feedforward artificial neural network (ANN) architecture optimization problem. *Neural Comput. Appl.*, 20(8):1273–1284, 2011. ISSN 09410643. doi: 10.1007/s00521-010-0504-3.

[6] Z. Chi. Statistical properties of probabilistic context-free grammars. *Comput. Linguist.*, 25(1):131–160, 1999. ISSN 08912017.

[7] F. Chollet. Keras. https://keras.io, 2015.

[8] P. Cortez, A. Cerdeira, F. Almeida, T. Matos, and J. Reis. Modeling wine preferences by data mining from physicochemical properties. *Decis. Support Syst.*, 47(4):547–553, 2009. ISSN 01679236. doi: 10.1016/j.dss.2009.05.016.

[9] A.G.C. de Sá, W.J.G.S. Pinto, L.O.V.B. Oliveira, and G.L. Pappa. RECIPE: A grammar-based framework for automatically evolving classification pipelines. In *Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics)*, volume 10196 LNCS, pages 246–261. Springer, 2017. ISBN 9783319556956. doi: 10.1007/978-3-319-55696-3_16.

[10] A.M. De Silva and P.H.W. Leong. Grammatical evolution. *SpringerBriefs Appl. Sci. Technol.*, 5(9789812874108):25–33, 2015. ISSN 21915318. doi: 10.1007/978-981-287-411-5_3.

[11] S. Estevez-Velarde, Y. Gutiérrez, A. Montoyo, and Y. Almeida-Cruz. AutoML strategy based on grammatical evolution: A case study about knowledge discovery from text. In *ACL 2019 - 57th Annu. Meet. Assoc. Comput. Linguist. Proc. Conf.*, pages 4356–4365, Florence, Italy, 2020. Association for Computational Linguistics. ISBN 9781950737482. URL https://www.aclweb.org/anthology/P19-1428.

[12] M. Feurer, A. Klein, K. Eggensperger, J.T. Springenberg, M. Blum, and F. Hutter. Efficient and robust automated machine learning. In *Adv. Neural Inf. Process. Syst.*, volume 2015-Janua, pages 2962–2970, 2015.

[13] M. Hauschild and M. Pelikan. An introduction and survey of estimation of distribution algorithms. *Swarm Evol. Comput.*, 1(3):111–128, 2011. ISSN 22106502. doi: 10.1016/j.swevo.2011.08.003.

[14] F. Hutter, L. Kotthoff, and J. Vanschoren, editors. *Automated Machine Learning: Methods, Systems, Challenges*. Springer, 2018.

[15] H. Jin, Q. Song, and X. Hu. Auto-Keras: Efficient Neural Architecture Search with Network Morphism, 2018. ISSN 00086363 (ISSN).

[16] H. Josiński, D. Kostrzewa, A. Michalczuk, and A. Świtoński. The expanded invasive weed optimization metaheuristic for solving continuous and discrete optimization problems. *Sci. World J.*, 2014, 2014. ISSN 1537744X. doi: 10.1155/2014/831691.

[17] J. Józefowska and J. Weglarz. On a methodology for discrete-continuous scheduling. *Eur. J. Oper. Res.*, 107(2):338–353, 1998. ISSN 03772217. doi: 10.1016/S0377-2217(97)00346-9.

[18] P. Kerschke and H. Trautmann. Automated algorithm selection on continuous black-box problems by combining exploratory landscape analysis and machine learning. *Evol. Comput.*, 27(1):99–127, 2018. ISSN 15309304. doi: 10.1162/evco_a_00236. URL https://doi.org/10.1162/evco_a_00236.

[19] K. Khan and A. Sahai. A Comparison of BA, GA, PSO, BP and LM for Training Feed forward Neural Networks in e-Learning Context. *Int. J. Intell. Syst. Appl.*, 4(7):23–29, 2012. ISSN 2074904X. doi: 10.5815/ijisa.2012.07.03.

[20] H.T. Kim and C.W. Ahn. A New Grammatical Evolution Based on Probabilistic Context-free Grammar. In *Proc. 18th Asia Pacific Symp. Intell. Evol. Syst. 2*, pages 1–12. Springer, 2015. doi: 10.1007/978-3-319-13356-0_1.

[21] D.P. Kingma and J.L. Ba. Adam: A method for stochastic optimization. *3rd Int. Conf. Learn. Represent. ICLR 2015 - Conf. Track Proc.*, 2015.

[22] B. Komer, J. Bergstra, and C. Eliasmith. Hyperopt-Sklearn: Automatic Hyperparameter Configuration for Scikit-Learn. In *Proc. 13th Python Sci. Conf.*, pages 32–37. Citeseer, 2014. doi: 10.25080/majora-14bd3278-006.

[23] K. Li and J. Malik. Learning to optimize. *5th Int. Conf. Learn. Represent. ICLR 2017 - Conf. Track Proc.*, 2019.

[24] Z. Li, X. Xiong, Z. Ren, N. Zhang, X. Wang, and T. Yang. An Aggressive Genetic Programming Approach for Searching Neural Network Structure Under Computational Constraints. *arXiv Prepr. arXiv1806.00851*, 2018. URL http://arxiv.org/abs/1806.00851.

[25] M. Lichman. UCI Machine Learning Repository [http://archive.ics.uci.edu/ml]., 2013. URL http://archive.ics.uci.edu/ml.

[26] H. Liu, K. Simonyan, O. Vinyals, C. Fernando, and K. Kavukcuoglu. Hierarchical representations for efficient architecture search. *6th Int. Conf. Learn. Represent. ICLR 2018 - Conf. Track Proc.*, 2018.

[27] P. López-Úbeda, M.C. Díaz-Galiano, M.T. Martín-Valdivia, and L.A. Ureña-López. SINAI in TASS 2018 Task 3. Classifying actions and concepts with UMLS on MedLine. In *CEUR Workshop Proc.*, volume 2172, pages 77–82, 2018.

[28] E. Martínez-Cámara, Y. Almeida-Cruz, M.C. Díaz-Galiano, S. Estévez-Velarde, M. García-Cumbreras, M. García-Vega, Y. Gutiérrez, A. Montejo-Ráez, A. Montoyo, R. Muñoz, A. Piad-Morffis, and J. Villena-Román. Overview of TASS 2018: Opinions, health and emotions. In *CEUR Workshop Proc.*, volume 2172 of *CEUR Workshop Proceedings*, pages 13–27, Sevilla, Spain, 2018. CEUR-WS.

[29] F. Mohr, M. Wever, and E. Hüllermeier. ML-Plan: Automated machine learning via hierarchical planning. *Mach. Learn.*, 107(8-10):1495–1515, sep 2018. ISSN 15730565. doi: 10.1007/s10994-018-5735-z. URL https://doi.org/10.1007/s10994-018-5735-z.

[30] R. Negrinho and G. Gordon. DeepArchitect: Automatically Designing and Training Deep Architectures. *arXiv Prepr. arXiv1704.08792*, 2017. URL http://arxiv.org/abs/1704.08792.

[31] R.S. Olson and J.H. Moore. TPOT: A Tree-Based Pipeline Optimization Tool for Automating Machine Learning. In *Work. Autom. Mach. Learn.*, pages 151–160, 2019. doi: 10.1007/978-3-030-05318-5_8.

[32] J.V. Palatresi and H.R. Hontoria. TASS2018: Medical knowledge discovery by combining terminology extraction techniques with machine learning classification. In *CEUR Workshop Proc.*, volume 2172, pages 89–95. CEUR-WS. org, 2018.

[33] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Van-

derplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[34] A. Piad-Morffis, Y. Gutiérrez, S. Estévez-Velarde, Y. Almeida-Cruz, A. Montoyo, and R. Muñoz. Analysis of eHealth knowledge discovery systems in the TASS 2018 Workshop. *Proces. Leng. Nat.*, 62(0):13–20, 2019. ISSN 19897553. doi: 10.26342/2019-62-1. URL http://journal.sepln.org/sepln/ojs/ojs/index.php/pln/article/view/5947.

[35] E. Real, S. Moore, A. Selle, S. Saxena, Y.L. Suematsu, J. Tan, Q.V. Le, and A. Kurakin. Large-scale evolution of image classifiers. In *34th Int. Conf. Mach. Learn. ICML 2017*, volume 6, pages 4429–4446, 2017. ISBN 9781510855144.

[36] R.M. Rivera-Zavala, P. Martínez, and I. Segura-Bedmar. A Hybrid Bi-LSTM-CRF model for knowledge recognition from ehealth documents. In *CEUR Workshop Proc.*, volume 2172, pages 65–70, 2018.

[37] K.O. Stanley and R. Miikkulainen. Evolving neural networks through augmenting topologies. *Evol. Comput.*, 10(2):99–127, jun 2002. ISSN 10636560. doi: 10.1162/106365602320169811. URL http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.28.5457&rep=rep1&type=pdfhttp://www.mitpressjournals.org/doi/10.1162/106365602320169811.

[38] V. Suárez-Paniagua, I. Segura-Bedmar, and P. Martínez. LABDA at TASS-2018 Task 3: Convolutional neural networks for relation classification in Spanish eHealth documents. In *CEUR Workshop Proc.*, volume 2172, pages 71–76, 2018.

[39] C. Thornton, F. Hutter, H.H. Hoos, and K. Leyton-Brown. Auto-WEKA: Combined selection and hyperparameter optimization of classification algorithms. In *Proc. ACM SIGKDD Int. Conf. Knowl. Discov. Data Min.*, volume Part F1288, pages 847–855. ACM, 2013. ISBN 9781450321747. doi: 10.1145/2487575.2487629.

[40] T. Tieleman and G. Hinton. Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude. *COURSERA Neural networks Mach. Learn.*, 4(2):26–31, 2012.

[41] I. Tsoulos, D. Gavrilis, and E. Glavas. Neural network construction and training using grammatical evolution. *Neurocomputing*, 72(1-3):269–277, 2008. ISSN 09252312. doi: 10.1016/j.neucom.2008.01.017.

[42] G. Waligóra. Discrete-continuous project scheduling with discounted cash flows-A tabu search approach. *Comput. Oper. Res.*, 35(7):2141–2153, 2008. ISSN 03050548. doi: 10.1016/j.cor.2006.09.022.

[43] B. Zoph and Q.V. Le. Neural architecture search with reinforcement learning. *5th Int. Conf. Learn. Represent. ICLR 2017 - Conf. Track Proc.*, 2019.

[44] B. Zoph, V. Vasudevan, J. Shlens, and Q.V. Le. Learning Transferable Architectures for Scalable Image Recognition. In *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit.*, pages 8697–8710, 2018. ISBN 9781538664209. doi: 10.1109/CVPR.2018.00907.

[45] B. Zoph, V. Vasudevan, J. Shlens, and Q.V. Le. Learning Transferable Architectures for Scalable Image Recognition. *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit.*, 2(6):8697–8710, 2018. ISSN 10636919. doi: 10.1109/CVPR.2018.00907.

[46] B. Zupan, M. Bohanec, I. Bratko, and J. Demsar. Machine learning by function decomposition. In *ICML*, pages 421–429, 1997.