Honda Research Institute EU
Honda Research Institute Europe GmbH
https://www.honda-ri.de/

Using Bayesian Optimization to Improve Hyperparameter Search in AutoML

Angus Kenny, Tapabrata Ray, Steffen Limmer, Hemant Kumar Singh, Tobias Rodemann, Markus Olhofer

2024

Preprint:

Copyright ACM 2024. This is the author's version of the work. It is posted here for your personal use. Not for redistribution. The definitive Version of Record was published in GECCO '24: Proceedings of the Genetic and Evolutionary Computation Conference, https://doi.org/10.1145/3638529.3654061.

Using Bayesian Optimization to Improve Hyperparameter Search in TPOT

Angus Kenny
The University of New South Wales
Canberra, Australia
angus.kenny@unsw.edu.au

Tapabrata Ray
The University of New South Wales
Canberra, Australia
t.ray@unsw.edu.au

Steffen Limmer
Honda Research Institute Europe
Offenbach, Germany
steffen.limmer@honda-ri.de

Hemant Kumar Singh
The University of New South Wales
Canberra, Australia
h.singh@unsw.edu.au

Tobias Rodemann
Honda Research Institute Europe
Offenbach, Germany
tobias.rodemann@honda-ri.de

Markus Olhofer
Honda Research Institute Europe
Offenbach, Germany
markus.olhofer@honda-ri.de

ABSTRACT

Automated machine learning (AutoML) has emerged as a pivotal tool for applying machine learning (ML) models to real-world problems. Tree-based pipeline optimization tool (TPOT) is an AutoML framework known for effectively solving complex tasks. TPOT's search involves two fundamental objectives: finding optimal pipeline structures (i.e., combinations of ML operators) and identifying suitable hyperparameters for these structures. While its use of genetic programming enables TPOT to excel in structural search, its hyperparameter search, involving discretization and random selection from extensive potential value ranges, can be computationally inefficient. This paper presents a novel methodology that heavily restricts the initial hyperparameter search space, directing TPOT's focus towards structural exploration. As the search evolves, Bayesian optimization (BO) is used to refine the hyperparameter space based on data from previous pipeline evaluations. This method leads to a more targeted search, crucial in situations with limited computational resources. Two variants of this approach are proposed and compared with standard TPOT across six datasets, with up to 20 features and 20,000 samples. The results show the proposed method is competitive with canonical TPOT, and outperforms it in some cases. The study also provides new insights into the dynamics of pipeline structure and hyperparameter search within TPOT.

CCS CONCEPTS

• **Computing methodologies** → **Search methodologies**; **Randomized search**.

ACM Reference Format:
Angus Kenny, Tapabrata Ray, Steffen Limmer, Hemant Kumar Singh, Tobias Rodemann, and Markus Olhofer. 2024. Using Bayesian Optimization to Improve Hyperparameter Search in TPOT. In *Genetic and Evolutionary Computation Conference (GECCO '24), July 14–18, 2024, Melbourne, VIC, Australia.* ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/3638529.3654061

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than the author(s) must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.
GECCO '24, July 14–18, 2024, Melbourne, VIC, Australia
© 2024 Copyright held by the owner/author(s). Publication rights licensed to ACM.
ACM ISBN 979-8-4007-0494-9/24/07
https://doi.org/10.1145/3638529.3654061

1 INTRODUCTION

In recent years, the volume and complexity of data in various domains have surged, necessitating more sophisticated approaches in data analysis. Automated machine learning (AutoML) [11] has emerged as a viable solution, streamlining the model selection and hyperparameter tuning processes, thereby making machine learning accessible to practitioners without expertise in the field.

AutoML tools select and combine multiple machine learning (ML) models and data transformation operators into so-called *pipelines* [19]. In these pipelines, each model processes data in sequence, with the output of one serving as the input to the next. This sequential arrangement harnesses the collective strengths of the models. Integral to each model within the pipeline are its unique hyperparameters, which need to be configured. These hyperparameters, governing diverse facets of the model's performance, can vary in nature, encompassing continuous, discrete, binary, or categorical types. Because each hyperparameter setting can drastically influence the overall effectiveness of the pipeline, and due to the large number of possible configurations, the task of optimizing these pipelines becomes highly complex. This complexity necessitates a strategic approach to both the selection of ML operators and the fine-tuning of their hyperparameters, ensuring that the final pipeline is not only structurally sound but also finely adjusted for optimal performance.

Amongst the array of AutoML tools, the tree-based pipeline optimization tool (TPOT) [20] has gained prominence for its unique genetic programming approach for evolving ML pipelines. Despite its effectiveness, TPOT faces challenges, particularly in the realm of hyperparameter optimization. Traditionally, TPOT relies on a discrete hyperparameter space from which values are randomly selected, which can lead to inefficiencies, especially when computational resources are limited. The approach proposed in this paper, named TPOT with Bayesian optimization and variable hyperparameter value sets (TPOT-BOV), aims to address these inefficiencies, by initiating its process with a constrained hyperparameter space, emphasizing the exploration of pipeline structures. As the search progresses, this method strategically expands the hyperparameter

search domain. This expansion, steered by insights from Bayesian optimization (BO), ensures a more focused and efficient search.

This paper outlines the challenges inherent in current AutoML approaches, specifically in TPOT, and details the architecture of the proposed approach, TPOT-BOV. It evaluates TPOT-BOV's performance through systematic evaluation against standard TPOT configurations across diverse datasets. The results of the study indicate that the proposed approach remains competitive with TPOT, outperforming it in some problem instances, while simultaneously improving the diversity of explored pipeline structures. The study also provides important insights into the nature of TPOT search, shedding light on the delicate balance between structural exploration and hyperparameter optimization. By demonstrating the effectiveness of combining genetic programming with BO, this research contributes to a deeper understanding of pipeline optimization strategies in AutoML, offering a framework for a more efficient and diverse pipeline generation scheme.

2 BACKGROUND AND RELATED WORK

Recent years have witnessed a surge in the application of ML in industrial settings. The process of selecting a suitable model for tasks like regression or classification entails extensive evaluations across a range of models, each with their unique combinations of hyperparameters. In response, AutoML has risen as a pivotal solution. AutoML automates the discovery of optimal pipelines, which encompasses data preprocessing, feature engineering, and model selection with appropriate hyperparameter tuning. A detailed examination of the evolution and functionality of AutoML tools is available in [13, 24]. Notable contributions in the field include Auto-Weka [16], TPOT [20], auto-sklearn [8], H2O [23] and AutoGluon [21]. These developments have primarily concentrated on two aspects: the representation and amalgamation of models within pipelines, and the optimization of model hyperparameters.

TPOT employs a tree-based structure to represent its pipelines, and harnesses genetic programming (GP) techniques from the DEAP Python library [6] to navigate the vast landscape of pipeline configurations. In this context, a candidate solution in the TPOT pipeline optimization process is represented as a GP tree, where the nodes and terminals are defined by TPOT's configuration dictionaries. These dictionaries are distinct for the two main classes, TPOTRegressor and TPOTClassifier, which are specifically tailored for regression and classification tasks, respectively. They delineate the set of operators, their possible hyperparameters, and the permissible values for these hyperparameters for each class. The search process in TPOT is iterative, executed over a predefined number of generations G, aiming to discover an ML pipeline that minimizes cross-validation (CV) error for regression or maximizes classification accuracy for classification problems. Concurrently, there is a secondary objective to minimize the number of operators in the GP tree, thereby curbing excessive complexity growth. Each generation within TPOT selects N parent solutions—where N represents the user-defined population size—by applying a non-domination sorting strategy based on these two objectives for its selection criteria. From these N parent solutions, N offspring solutions are generated and evaluated, meaning the total number of pipeline evaluations conducted throughout the search is $G \times N$. Offspring solutions are produced using a one-point GP crossover, or one of three mutation operations:

- the *insert* operation, which expands the pipeline by inserting a new operator at a chosen point;
- the *shrink* operation, which streamlines the pipeline by removing an operator, effectively simplifying the structure;
- the *replace* operation, which replaces either operators or hyperparameters within the pipeline.

These operations collectively enable TPOT to dynamically evolve and refine pipelines, balancing between complexity and performance. However, this approach can occasionally produce redundant or impractical pipelines, which are then excluded from evaluation. In this instance, it is possible for the total number of evaluated pipelines to fall short of the theoretical maximum $G \times N$, as these excluded evaluations are not replaced. This mechanism ensures that computational resources are not wasted on reassessing previously encountered hyperparameter configurations, but it can result in under-utilization of the full computational budget.

Regarding hyperparameters, TPOT employs random search within a discretized parameter space, selecting values from a pre-defined, static set of hyperparameter values. This method of hyperparameter selection, while straightforward, can turn out to be inefficient in practice. It relies on an arbitrary discretization of the hyperparameter space and a uniform random selection from this discretized space, a strategy shown by Bergstra et al. [4] to be surprisingly effective in certain contexts for quickly identifying high-performance parameter settings. However, this approach does not account for the potentially complex relationships between hyperparameters and their impact on model performance. The static nature of the hyperparameter value set, and its method of random selection from it, means that TPOT's search is prone to wasting computational budget on the evaluation of suboptimal hyperparameter value combinations, rather than focusing on those areas of the search space that hold the most promise for improving model performance. This can lead to inefficiencies, particularly in scenarios where computational resources are limited, or the search space is exceedingly large.

More sophisticated hyperparameter optimization strategies include evaluating configurations through methods like successive halving [14] or Bayesian optimization (BO) [15]. Typically, these methods involve the generation of Gaussian process models, drawing from historical data [12]. An alternative approach is the tree-structured Parzen estimator (TPE) [3, 5], capable of handling various hyperparameter types, including continuous, discrete, and categorical. A comprehensive review of recent advancements in BO can be found in [26]. The Optuna framework [1] exemplifies a state-of-the-art open-source platform for hyperparameter optimization, incorporating BO through multiple models and samplers, including TPE. Initially featuring an independent TPE sampler, Optuna has since integrated a multivariate sampler, enhancing the efficiency of the optimization process significantly.

3 PROPOSED METHOD: TPOT-BOV

Through its tree-structure representation of pipelines and use of GP during optimization, TPOT is particularly adept at searching the space of possible pipeline structures. A pipeline's structure is

defined as the unique combination of machine learning operators it comprises, without considering the values of its hyperparameters. This paper adopts a "bar" notation convention to indicate pipeline structure representation. For example, if p and q are unique AutoML pipelines, their structural representations are denoted as $\bar{p}$ and $\bar{q}$, respectively. In the case where p and q differ in the values of their hyperparameters alone (i.e., $\bar{p} = \bar{q}$), both pipelines are said to have identical structure.

When TPOT introduces a new operator to a pipeline via mutation, it randomly selects its hyperparameter values from a predetermined range of options. For continuous hyperparameter domains, TPOT discretizes these spaces into a set number of discrete points, which can sometimes lead to a very large number of possible values for a single hyperparameter. The recombination operator in TPOT's GP framework ensures that hyperparameter values from high-performing pipelines are retained within the population. However, the selection of new hyperparameter values required by any modifications to these pipelines is made at random. Given the multitude of pipeline operators which can comprise a pipeline — each with their own unique set of hyperparameters — AutoML problems typically present very large hyperparameter search spaces. Random sampling of these spaces, given the vast number of potential combinations, is unlikely to be the most efficient strategy, especially when working within a constrained computational budget.

The method proposed in this paper, TPOT-BOV, aims to address this inefficiency in TPOT's hyperparameter tuning methods, by systematically restricting the domains of the hyperparameters. With fewer options to choose from, it is less likely that TPOT will produce new, previously unevaluated pipelines through swapping hyperparameter values alone. This means TPOT is encouraged to capitalise on its strengths; investing more of its computational budget into finding new pipeline structures, as opposed to randomly sampling hyperparameter values for existing ones.

Over the course of TPOT-BOV search, the restricted hyperparameter domains are expanded using BO. When a promising pipeline structure is identified, a tree-structured Parzen estimator (TPE) sampler is constructed, using all previously evaluated pipelines that share that structure as data points. This sampler is employed to suggest new elements to add to the overall hyperparameter value set. Initially, when there are very few evaluated pipelines to consider, the suggestions made by the sampler will be arbitrary. However, as the search progresses, more historical information is available and the suggestions become increasingly more productive.

Two variants of TPOT-BOV are proposed and evaluated in this paper: TPOT-BOVR and TPOT-BOVM. Both are structurally similar, but differ in how their initial restricted hyperparameter value sets are constructed. Pseudo-code for TPOT-BOV is provided in Algorithm 1, and the different methods of restricted hyperparameter value set construction are described below.

Initially, the restricted hyperparameter value set V^r is constructed from the full hyperparameter value set V^f, defined in the default TPOT configuration dictionary (Line 1). How V^r is constructed depends on the TPOT-BOV variation being considered. In the case of TPOT-BOVR, a single value is selected for each hyperparameter, at random, from V^f. In the case of TPOT-BOVM, $n = \left\lceil \frac{h}{10} \right\rceil$ values are selected from V^f, where h is the number of possible values, for each hyperparameter. The selection for TPOT-BOVM is performed using one-dimensional Latin hypercube sampling [18], in order to ensure a relatively even spread across the full range of possible values.

Algorithm 1: TPOT-BOV

Input: D: training data; G: TPOT generations; N: population size; V^f: full TPOT hyperparameter (HP) value set; ρ: GP parameters.
Output: S: set of evaluated pipelines

1: $V^r \leftarrow$ construct restricted HP value set from V^f
2: $T \leftarrow$ TPOTRegressor with pop size N and GP parameters ρ
3: $S \leftarrow$ fit T on D using V^r for 0 generations (evaluate initial pop)
4: $g \leftarrow 0$
5: **while** $|S| < N \times G$ **and** $g < G$ **do**
6: $\quad P \leftarrow$ partition active population in T by pipeline structure
7: $\quad$ **for** $p \in P$ **do**
8: $\quad\quad M \leftarrow$ TPE sampler constructed from pipelines in $S_{\bar{p}}$
9: $\quad\quad V' \leftarrow |p|$ HP value sets suggested from V^f by M
10: $\quad\quad V^r \leftarrow V^r \cup V'$
11: $\quad$ **end for**
12: $\quad S' \leftarrow$ fit T on D using V^r for 1 generation
13: $\quad g \leftarrow g + 1$ **if** $|S'| = |S|$ **else** 0 **end if**
14: $\quad S \leftarrow S'$
15: **end while**
16: **return** S

A TPOTRegressor[1] object T is instantiated, with population size N and with GP parameters ρ (Line 2), and initialized on the training data D by fitting it for 0 generations (Line 3), using the restricted hyperparameter value set V^r. This produces a set S of N evaluated pipelines, which is maintained as an archive of unique evaluated pipelines, throughout the search and updated whenever a new pipeline is evaluated.

Until the termination criteria is met, the current active population for T is derived by applying a non-dominated (ND) sorting to S, minimizing both CV and pipeline complexity, from which the top N solutions are selected, across potentially multiple ND fronts. This group is then partitioned by structure, resulting in P (Line 6). For each set of pipelines p in P, a TPE sampler M is constructed, using the pipelines in $S_{\bar{p}}$ as historical data (Line 8). Here, p is the set of pipelines in the active TPOT population sharing the structural representation $\bar{p}$, and $S_{\bar{p}}$ is the set of all previously evaluated pipelines which share the structural representation $\bar{p}$. The sampler M uses Bayesian optimization to suggest new sets of hyperparameter values V', which are added to V^r, expanding the search space (Line 10). The number of new hyperparameter value sets is determined by the number of pipelines in p. This ensures that pipeline structures that are dominant in the current active population will be explored more thoroughly.

Having updated the restricted hyperparameter value set, T is fit on D using V^r again for one generation, producing an updated set

[1]As discussed in Section 2, TPOT is capable of handling both classification and regression tasks. This study, however, concentrates exclusively on regression tasks due to their more frequent occurrence in the authors' routine work. It is important to note that the underlying mechanisms of pipeline search and optimization in TPOT are fundamentally the same for both task types. Consequently, it is anticipated that the TPOT-BOV methodology proposed in this paper would be equally applicable and effective for classification tasks. Nevertheless, this assertion has not been empirically verified within the scope of the current study.

of evaluated pipelines, S' (Line 12), and a check is performed to ensure new pipelines have been evaluated this generation. If the updated set of pipelines is the same size as the previous generation's set, a "stagnancy" counter g is incremented, otherwise g is reset to zero (Line 13). This process continues until there are $N \times G$ pipelines in S, or until G generations have elapsed without any new pipeline evaluations. This alternative stopping condition exists in case the active TPOT population converges to a single structure, in which case it may be difficult to find better unique structures with a limited hyperparameter search space for the other operators.

3.1 Conceptual parallels with existing methods

There are plenty of examples in the literature where Bayesian optimization (BO) has been used as a strategy to tune hyperparameters in machine learning [22, 25, 27]. Typically, these methods are designed to operate under the assumption of a fixed pipeline structure. While these methods are useful for their efficiency in navigating the large, complex hyperparameter spaces typical of AutoML problems, their reliance on a pre-specified architecture inherently limits their capability to explore and optimize in the space of possible pipeline structures. This is a crucial limitation, as the choice of pipeline's structure often has a profound impact on its performance. TPOT-BOV distinguishes itself from these methods by not employing BO for direct hyperparameter optimization. Instead, it utilizes BO to intelligently direct the TPOT framework into promising regions of the hyperparameter search space. This strategy enables TPOT-BOV to maintain its primary focus on discovering and evaluating a diverse range of pipeline structures, taking advantage of the strengths of TPOT, while simultaneously benefiting from the guided exploration offered by BO.

TPOT-BOV also shares some conceptual similarities with a more general class of methods, referred to here as adaptive grid-based search (AGS) methods [3, 10], notably in its initial employment of a coarse hyperparameter grid, which is subsequently refined based on the identification of promising areas within the hyperparameter space. This approach mirrors the denser sampling around high-performing regions characteristic of AGS methods. However, a notable divergence is observed in the refinement mechanism. While most AGS methods typically adopt a greedy approach, focusing solely on previously effective areas of the hyperparameter space, TPOT-BOV employs Bayesian optimization (BO) in its decision-making. This allows for a more balanced exploration of the hyperparameter space, combining exploitation of known high-performing regions with the exploration of less certain areas.

It is in this progressive refinement that a resemblance is drawn to column generation (CG) techniques, used in large-scale optimization problems [7]. TPOT-BOV starts with a restricted hyperparameter search space, analogous to the smaller subset of variables initially considered in CG. The expansion of this space in TPOT-BOV is guided by BO, akin to the role of the dual pricing problem in CG, which determines new variables for inclusion in the restricted problem set. Thus, BO serves a similar function, suggesting new hyperparameters for consideration based on their potential to improve model performance.

The innovative aspect of TPOT-BOV is underscored through this dual analogy with AGS methods and CG. The integration of these methods' strategic strengths with the predictive capabilities of BO offers a more effective and efficient method for hyperparameter optimization in AutoML — while retaining the flexibility to explore new pipeline structures.

4 NUMERICAL EXPERIMENTS

4.1 Test problems

The study uses six extensively researched datasets from existing literature [9], to represent a wide range of real-world regression scenarios. These datasets, detailed in Table 1, were specifically selected for their varied feature sets. Categorical features in the data are encoded as integer values.

Table 1: Six data sets used in this study, obtained from the OpenML repository (https://openml.org/).

Problem	Features	Samples
quake	3	2,178
socmob	5	1,156
abalone	8	4,177
brazilian_houses	12	10,692
house_16h	16	22,784
elevators	18	16,599

4.2 Experimental design

The codes used for the experiments conducted in this study[2] were implemented in Python, with the TPOT Python library [20] providing the baseline TPOT implementation and Optuna Python library [1] providing the Bayesian optimisation tools.

To observe their statistical behaviour, TPOT-BOVR and TPOT-BOVM were compared based on the results obtained from 21 runs, for each data set, with the same set of 21 random seeds, against the TPOT baseline, (called TPOT-BASE here) with default TPOT parameters. These include population size $N = 100$, evolved for $G = 100$ generations, with mutation rate of 0.9, crossover rate of 0.1, evaluation timeout of 5 minutes and 5-fold cross-validation of mean squared error during fitting. As discussed in Section 2, TPOT terminates its search after a pre-determined number of generations, resulting in a theoretical maximum of $G \times N$ pipeline evaluations. Due to the restricted hyperparameter set employed by TPOT-BOV, the likelihood of encountering previously evaluated hyperparameter combinations is significantly greater than it is for TPOT-BASE. Therefore the TPOT-BOV experiments use a maximum of $N \times G$ pipeline evaluations as its principal stopping criteria — as opposed to a generational scheme — in this case $100 \times 100 = 10,000$ evaluations.

The CV error, complexity (number of operators) and generation evaluated was recorded for all pipelines, for every run. Along with direct comparisons of the best, median and mean CV errors obtained across all runs, the pairwise win/tie/loss results are reported, using the Mann-Whitney-Wilcoxon statistical significance test [17]. Here, method A is said to "win" over method B (and B "lose" to A), if the

[2]Source code at http://www.mdolab.net/Ray/Research-Data/gecco2024tpot.zip.

Table 2: Experimental CV error for best pipeline in each run of 21 runs. Values in parentheses represent (number of features/sample points) for a given data set.

	TPOT-BASE	TPOT-BOVR	TPOT-BOVM
quake (3/2178)			
best	**3.5183e-02**	3.5284e-02	3.5314e-02
median	3.5482e-02	3.5484e-02	**3.5450e-02**
mean	3.5460e-02	3.5458e-02	**3.5447e-02**
std dev	9.0973e-05	7.4179e-05	6.3582e-05
socmob (5/1156)			
best	1.4051e+02	**1.2491e+02**	1.3677e+02
median	**1.6492e+02**	1.6660e+02	1.6871e+02
mean	**1.6217e+02**	1.6479e+02	1.6714e+02
std dev	1.1222e+01	1.3241e+01	1.4325e+01
abalone (8/4177)			
best	4.2582e+00	4.2596e+00	**4.2456e+00**
median	4.2830e+00	4.2858e+00	**4.2775e+00**
mean	4.2937e+00	4.2863e+00	**4.2798e+00**
std dev	2.6742e-02	1.6073e-02	2.0540e-02
brazilian_houses (12/10692)			
best	**4.0771e+01**	**4.0771e+01**	**4.0771e+01**
median	4.0819e+01	4.0825e+01	**4.0817e+01**
mean	4.0814e+01	4.0820e+01	**4.0811e+01**
std dev	2.1863e-02	2.3144e-02	1.9234e-02
house_16h (16/22784)			
best	**8.6236e+08**	8.6910e+08	8.6258e+08
median	9.0858e+08	9.1108e+08	**8.9873e+08**
mean	9.1004e+08	9.0994e+08	**8.9830e+08**
std dev	1.5189e+07	1.2778e+07	2.0383e+07
elevators (18/16599)			
best	3.4869e-06	**3.4815e-06**	3.4839e-06
median	3.5326e-06	3.5262e-06	**3.5236e-06**
mean	3.5543e-06	3.5456e-06	**3.5547e-06**
std dev	6.0378e-08	5.7580e-08	7.6093e-08

median CV error of A is less than that of B, with p-value ≤ 0.05. If p-value > 0.05, it cannot be said with any confidence that the samples are drawn from different distributions, and the result is deemed a "tie".

5 RESULTS AND DISCUSSION

The results from the experiments conducted are presented in Table 2. This table shows that TPOT-BOVM provides consistent improvement over TPOT-BOVR and TPOT-BASE. On nearly all problem instances, TPOT-BOVM achieved the best median and mean CV error across all 21 runs — with the exception of `socmob`. However, the results of the pairwise Mann-Whitney-Wilcoxon (MWW) statistical test, given in Table 3 suggest that TPOT-BOVM only performed better — with statistical significance — than TPOT-BASE in two of the six instances, and better than TPOT-BOVR in three of them. TPOT-BOVR did not outperform either TPOT-BASE or TPOT-BOVM with any significance.

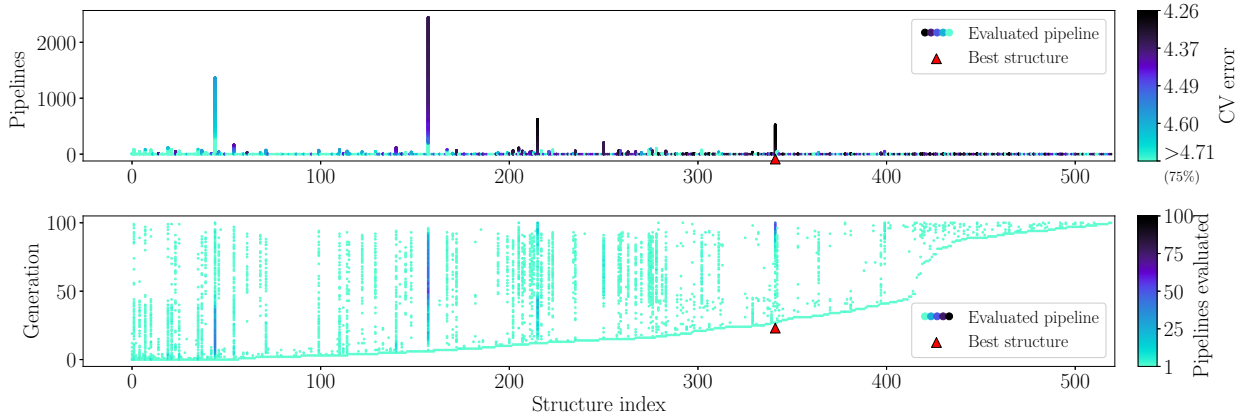
Table 3: Pairwise win/tie/loss count for MWW statistical test, with p-value ≤ 0.05 considered significant.

	TPOT-BASE	TPOT-BOVR	TPOT-BOVM
TPOT-BASE	0/6/0	0/6/0	**2/4/0**
TPOT-BOVR	0/6/0	0/6/0	**3/3/0**
TPOT-BOVM	0/4/2	0/3/3	0/6/0

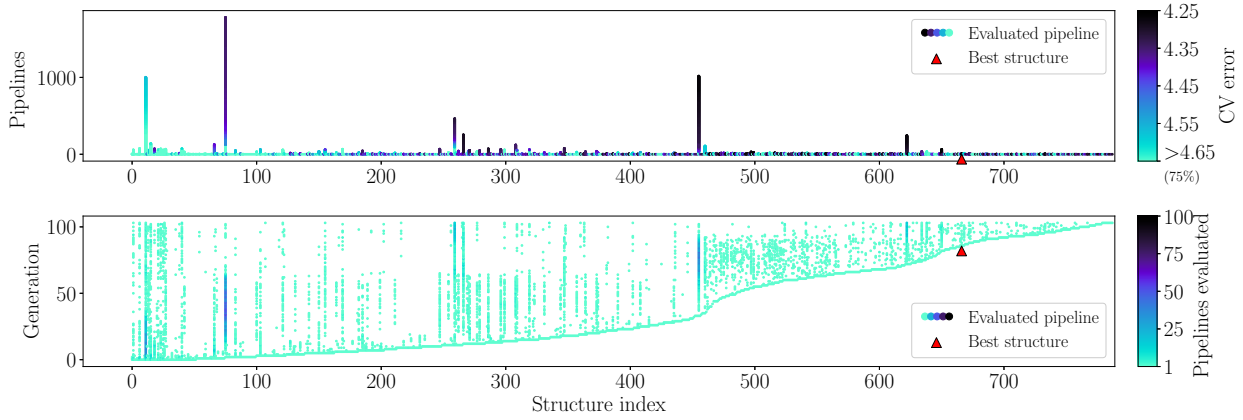
While Tables 2 and 3 suggest that TPOT-BOV does not definitively surpass TPOT-BASE in achieving lower cross-validation (CV) errors in all instances, this was not the primary goal of TPOT-BOV. Sections 2 and 3 established that random selection from large hyperparameter spaces in TPOT leads to inefficiencies during search. The primary objective of TPOT-BOV was to drive TPOT's exploration towards structural search by constraining the hyperparameter space, thereby capitalizing on the strengths of its genetic programming (GP) aspects. Table 4 presents statistics on the number of unique pipeline structures evaluated during the search process. These statistics reveal that the TPOT-BOV variants consistently explore more structures compared to TPOT-BASE. Furthermore, this increase in structural exploration did not significantly affect the maximum or average complexity of the pipelines, across all explored structures.

A closer examination of a single run on the `abalone` problem, as given in Figure 1, offers a detailed comparison between TPOT-BASE and TPOT-BOVM. Each sub-figure in Figure 1 is divided into two plots. The horizontal axes in both plots represents the index of every unique structure that was evaluated over the course of the run, in the order they were first evaluated. The top plot in each sub-figure is a histogram of the frequency of the unique hyperparameter combinations that were evaluated for each pipeline structure, over the course of the search. The points in this plot are ordered and coloured by CV error. Because the range of CV errors for evaluated pipelines can vary by many orders of magnitude, the colour mapping has a threshold at the 75th percentile, with all pipelines that have a worse CV error than this threshold value coloured the same. The lower plot in each sub-figure illustrates which pipeline structures had hyperparameter combinations evaluated in each generation, represented on the vertical axis. The colour of each point denotes the proportion of the offspring population that was evaluated from that pipeline structure, in that generation. Darker points indicate the constituency of the offspring population was dominated by one, or a few, pipeline structures.

When selecting pipelines to use in offspring production, TPOT performs an ND sorting on the set of evaluated pipelines, with CV error and number of operators as its minimization criteria. The motivation behind minimizing the number of operators is to keep the complexity of the pipeline structures "in check". This is an important consideration for GP-based methods, as recombination of tree-structures can result in runaway complexity growth, known as "bloat" [2]. An unfortunate consequence of controlling pipeline complexity growth in this way is that there is a severe impact on the diversity of the parent population, selected in each generation. Furthermore, preferring pipeline structures simply because they have low complexity, can result in an inefficient use of much of the computational budget — especially in the case of complex machine



(a) TPOT-BASE



(b) TPOT-BOVM

Figure 1: Evaluated pipeline structures for TPOT-BASE (top) and TPOT-BOVM (bottom), on `abalone` dataset with same initial random seed. Upper plot in each figure indicates the number of pipelines evaluated for each structure, with points ordered and coloured by CV error. Lower plot in each figure shows how many pipelines were evaluated from each structure, per generation. Red triangle indicates best structure.

learning problems, where a simple pipeline of one or two operators is unlikely to have sufficient expressive power, regardless of how well the hyperparameters are optimized.

The frequency plots in Figure 1 demonstrate that both TPOT-BASE and TPOT-BOVM tend to focus most of their attention on just a few structures, and Table 5 provides details to the top five most evaluated pipeline structures for each method. Interestingly, the most evaluated structure is the same for both methods, achieving identical CV errors. Similarly, the 2nd ranked structure for TPOT-BASE, and 3rd ranked for TPOT-BOVM, are also identical, with the same (very poor) CV errors. In both cases, over 10% of the computational budget is consumed testing different hyperparameter combinations, however TPOT-BOVM spends considerably fewer evaluations on each than TPOT-BASE, indicated by a smaller number in the "Pipes" column. These findings suggest that TPOT's selection mechanism can lead to inefficient searches, with considerable resources expended on evaluating suboptimal solutions simply because they are less complex. By restricting the hyperparameter value set, TPOT-BOVM marginally improves on this, but is still limited by TPOT's selection mechanism.

The plots agree with the general trend presented in Table 4, illustrating a significant difference in the number of structures explored by TPOT-BOVM over TPOT-BASE. Notably, TPOT-BASE predominantly explores new structures in the initial 50 generations, with less than 100 of its total 520 evaluated structures discovered in the second half of the search. In contrast, TPOT-BOVM examines a substantial number of new structures — more than 300 out of 788 total — in the final 50 generations. This suggests that the strategy of restricting the hyperparameter value set was successful in exploring new structures, over the entire course of the search. Furthermore,

Table 4: Evaluated pipeline structure statistics averaged over 21 runs. Values in parentheses represent (number of features/sample points) for a given data set.

	TPOT-BASE	TPOT-BOVR	TPOT-BOVM
quake (3/2178)			
mean structures	545.24	555.19	576.43
std dev	225.03	259.45	203.96
max. complexity	6.62	6.10	6.76
mean complexity	3.06	3.00	3.13
socmob (5/1156)			
mean structures	918.43	1035.05	1113.86
std dev	251.03	454.76	554.10
max. complexity	7.67	8.43	8.57
mean complexity	3.46	3.59	3.65
abalone (8/4177)			
mean structures	739.52	876.24	821.76
std dev	143.27	235.55	216.91
max. complexity	7.81	8.43	8.19
mean complexity	3.45	3.72	3.53
brazilian_houses (12/10692)			
mean structures	1055.19	1205.81	1263.00
std dev	391.31	507.13	467.54
max. complexity	9.19	8.33	9.19
mean complexity	3.72	3.58	3.71
house_16h (16/22784)			
mean structures	1123.00	1405.50	1286.39
std dev	427.40	425.69	413.46
max. complexity	10.45	9.85	9.61
mean complexity	4.18	4.21	4.01
elevators (18/16599)			
mean structures	1545.95	1613.35	1655.14
std dev	559.62	593.77	661.96
max. complexity	12.05	11.65	11.67
mean complexity	4.73	4.70	4.77

Table 5: Statistics of top 5 most evaluated pipeline structures in Figure 1. Table shows structural representation, number of pipelines evaluated and the best CV of any pipeline evaluated with that structure. {D} indicates input data.

Pipeline structure	Pipes	CV
TPOT-BASE:		
{RandomForestRegressor{PCA{D}}}	2447	4.35
{RandomForestRegressor{D}}	1368	4.57
{RandomForestRegressor{SGDRegressor{PCA{D}}}}	623	4.28
{RandomForestRegressor{SGDRegressor{SGDRegressor{PCA{D}}}}}	526	4.26
{RandomForestRegressor{PCA{PCA{D}}}}	207	4.35
TPOT-BOVM:		
{RandomForestRegressor{PCA{D}}}	1781	4.35
{RandomForestRegressor{SGDRegressor{PCA{ZeroCount{D}}}}}	1018	4.27
{RandomForestRegressor{D}}	1003	4.57
{ExtraTreesRegressor{PCA{D}}}	464	4.32
{RandomForestRegressor{SGDRegressor{PCA{D}}}}	254	4.29

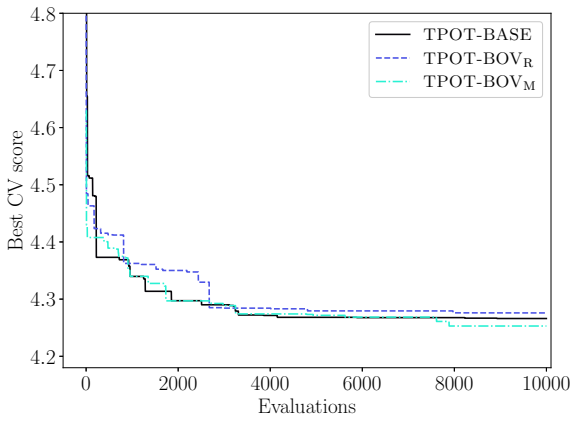


Figure 2: CV error convergence for example run detailed in Figure 1.

due to the reduced concentration on evaluating many hyperparameter combinations for just a few structures, more of the budget can be used to explore different hyperparameter combinations in more diverse pipeline structures. This is indicated by the fewer dark regions in the TPOT-BOVM tracking plots, and the denser pattern of dots, especially in the latter stages of the search.

Figure 2 provides a convergence plot for the observed instance discussed above. This plot indicates small difference in the convergence behavior between all three methods, with TPOT-BASE and TPOT-BOVM, in particular, displaying nearly identical convergence behaviour. Where the two eventually diverge is in the latter half of the search. The structure tracking plots in Figure 1 show that TPOT-BOVM evaluated significantly more new structures over this period, and it appears that maintaining this bias towards exploration of structural space was pivotal in TPOT-BOVM's eventual outperformance of TPOT-BASE. This finding highlights that while restricting the hyperparameter space does not significantly affect convergence, it enables more effective structural exploration.

One final aspect to consider is that TPOT-BOV uses a single, shared hyperparameter value set for the entire population. The growth of this hyperparameter value set over the course of the observed instance is given in Figure 3. This plot clearly shows that TPOT-BOV significantly restricts the size of the hyperparameter set over the course of the search, with both variants finishing with restricted hyperparameter sets just over half the size of the full set V^f. Interestingly, despite TPOT-BOVR starting the search with a smaller restricted hyperparameter value set compared to TPOT-BOVM, it concludes with a larger one. This outcome can possibly be attributed to TPOT-BOVM's initial distribution of values across the entire range, which facilitates the earlier identification of promising areas within the search space. This assertion is supported by the convergence plot in Figure 2, where TPOT-BOVM demonstrates a notably

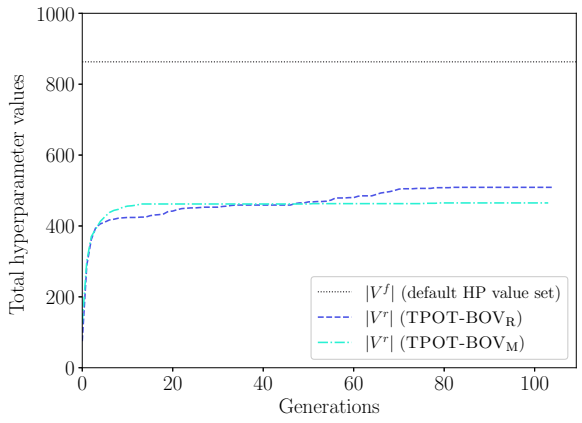


Figure 3: Growth of restricted hyperparameter value set V^r for example run detailed in Figure 1. Dotted line indicates size of the default TPOT hyperparameter value set V^f.

quicker convergence than TPOT-BOVR in the initial stages. One key aspect of the multivariate TPE sampler used in making suggestions to expand the restricted hyperparameter value set, is that it considers variables not in isolation, but as part of a collective. As a result, if the set of suggested values contains at least one unique element, the entire suggestion is deemed unique. This approach to suggestions, coupled with the early identification of promising regions, implies that TPOT-BOVM is likely to repeatedly suggest similar values for some hyperparameters. Such repetition does not contribute to an increase in the size of the overall hyperparameter value set, potentially explaining the relative slower growth of TPOT-BOVM's set compared to that of TPOT-BOVR.

While the obtained results suggest that TPOT-BOV represents a promising advance in refining hyperparameter search strategies within TPOT, it may not fully capture the intricacies of hyperparameter effectiveness across different pipeline structures. The practice of utilizing a unified hyperparameter value set for all pipeline configurations overlooks the fact that certain hyperparameters may yield significant performance benefits in one structure but prove less effective in others. Because the search starts with a population of diverse pipeline structures with tightly constrained hyperparameter ranges, BO's recommendations often contribute to expanding the hyperparameter set, as depicted in Figure 3. Initially, this diversity in pipeline structures leads to a rapid increase in the hyperparameter value set, as BO is more likely to suggest additions that are not already present. However, as the search progresses, identifying new and unique hyperparameter values that have not yet been included becomes increasingly challenging. This difficulty is further compounded when all structures draw from a common hyperparameter pool, resulting in a saturation point where the value set expansion slows significantly due to the high likelihood of BO suggestions already being represented in the set.

A potential improvement to this strategy could be to maintain distinct hyperparameter value sets for each pipeline structure, allowing each to draw from its own tailored set of values suggested by BO. For new pipeline structures generated via mutation, the parent's hyperparameter value set could be inherited. For recombination, a simple union of the parents' hyperparameter value sets might suffice. While this idea extends beyond the scope of this study, it offers an intriguing direction for future research.

6 CONCLUSION AND FUTURE WORK

This paper introduced TPOT-BOV, a novel approach enhancing the TPOT framework by incorporating BO for restricted hyperparameter search. When selecting new values for its hyperparameters, TPOT chooses randomly from — sometimes very large — hyperparameter spaces. This approach to hyperparameter tuning, combined with its method of parent population selection, can result in much of the computational budget being used inefficiently. TPOT-BOV addresses the inefficiencies in TPOT's hyperparameter tuning by initially constraining the hyperparameter search space and gradually expanding it based on insights gained from BO.

Two variants of TPOT-BOV were proposed, distinguished by the method used to construct the initial restricted hyperparameter search space. The performance of these variants was assessed by evaluating the cross-validation error obtained when trained on six data sets from the literature. These performance results were compared against standard TPOT, on the same data sets, using the same set of initial random seeds. The experiments demonstrated that TPOT-BOV, particularly the TPOT-BOVM variant, successfully balances the exploration of new pipeline structures and hyperparameter spaces, achieving competitive results compared to standard TPOT in various instances — sometimes significantly outperforming it. Through analysis of these experiments, significant insights into the dynamics of AutoML search were gained, particularly in the realm of structural exploration and hyperparameter optimization.

Future research can build upon TPOT-BOV in several directions. One potential area is the exploration of separate hyperparameter sets for different pipeline structures, allowing each pipeline to draw from a unique, tailored set of parameter values. This could further refine the search process, potentially leading to more efficient and effective pipeline optimization. While TPOT must discretize continuous hyperparameter search spaces to operate, BO does not suffer from such restrictions. Another direction of research could capitalize on this fact, by allowing TPOT-BOV to suggest values from a continuous hyperparameter distribution, instead of a discrete one, thereby increasing the granularity of potential solutions. Moreover, extending the methodology to broader sets of problems, including classification tasks, and exploring its scalability to larger datasets would be valuable. Finally, BO could be integrated into the mutation operators of TPOT itself, allowing it to more intelligently select hyperparameter values when new pipeline operators are first introduced, as opposed to relying on random selection from a restricted value set. Such explorations would contribute to the ongoing evolution of AutoML tools, aiming for more efficient, robust, and versatile automated machine learning solutions.

ACKNOWLEDGMENT

The authors acknowledge the financial support from Honda Research Institute Europe for this work.

REFERENCES

[1] Takuya Akiba, Shotaro Sano, Toshihiko Yanase, Takeru Ohta, and Masanori Koyama. 2019. Optuna: A next-generation hyperparameter optimization framework. In *ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. 2623–2631.

[2] Eva Alfaro-Cid, JJ Merelo, F Fernández de Vega, Anna Isabel Esparcia-Alcázar, and Ken Sharman. 2010. Bloat control operators and diversity in genetic programming: A comparative study. *Evolutionary Computation* 18, 2 (2010), 305–332.

[3] James Bergstra, Rémi Bardenet, Yoshua Bengio, and Balázs Kégl. 2011. Algorithms for hyper-parameter optimization. *Advances in Neural Information Processing Systems* 24 (2011), 1–9.

[4] James Bergstra and Yoshua Bengio. 2012. Random search for hyper-parameter optimization. *Journal of Machine Learning Research* 13, 2 (2012), 281–305.

[5] James Bergstra, Daniel Yamins, and David Cox. 2013. Making a science of model search: Hyperparameter optimization in hundreds of dimensions for vision architectures. In *International Conference on Machine Learning*. 115–123.

[6] François-Michel De Rainville, Félix-Antoine Fortin, Marc-André Gardner, Marc Parizeau, and Christian Gagné. 2012. DEAP: A Python framework for evolutionary algorithms. In *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO)*. 85–92.

[7] Guy Desaulniers, Jacques Desrosiers, and Marius M Solomon. 2006. *Column generation*. Vol. 5. Springer Science & Business Media.

[8] Matthias Feurer, Aaron Klein, Katharina Eggensperger, Jost Springenberg, Manuel Blum, and Frank Hutter. 2015. Efficient and robust automated machine learning. In *Advances in Neural Information Processing Systems 28 (2015)*. 2962–2970.

[9] Pieter Gijsbers, Marcos LP Bueno, Stefan Coors, Erin LeDell, Sébastien Poirier, Janek Thomas, Bernd Bischl, and Joaquin Vanschoren. 2022. AMLB: an AutoML benchmark. *arXiv preprint arXiv:2207.12560* (2022).

[10] Frank Hutter, Holger H Hoos, and Kevin Leyton-Brown. 2011. Sequential model-based optimization for general algorithm configuration. In *Learning and Intelligent Optimization: 5th International Conference, LION 5, Rome, Italy, January 17-21, 2011. Selected Papers 5*. Springer, 507–523.

[11] Frank Hutter, Lars Kotthoff, and Joaquin Vanschoren. 2019. *Automated machine learning: methods, systems, challenges*. Springer Nature.

[12] Donald R Jones, Matthias Schonlau, and William J Welch. 1998. Efficient global optimization of expensive black-box functions. *Journal of Global optimization* 13, 4 (1998), 455–492.

[13] Shubhra Kanti Karmaker, Md Mahadi Hassan, Micah J Smith, Lei Xu, Chengxiang Zhai, and Kalyan Veeramachaneni. 2021. AutoML to date and beyond: Challenges and opportunities. *ACM Computing Surveys (CSUR)* 54, 8 (2021), 1–36.

[14] Zohar Karnin, Tomer Koren, and Oren Somekh. 2013. Almost optimal exploration in multi-armed bandits. In *International Conference on Machine Learning*. 1238–1246.

[15] Angus Kenny, Tapabrata Ray, Steffen Limmer, Hemant Kumar Singh, Tobias Rodemann, and Markus Olhofer. 2023. Hybridizing TPOT with Bayesian optimization. In *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO)*. 502–510.

[16] Lars Kotthoff, Chris Thornton, Holger H Hoos, Frank Hutter, and Kevin Leyton-Brown. 2019. Auto-WEKA: Automatic model selection and hyperparameter optimization in WEKA. In *Automated Machine Learning*. Springer, Cham, 81–95.

[17] Henry B Mann and Donald R Whitney. 1947. On a test of whether one of two random variables is stochastically larger than the other. *The Annals of Mathematical Statistics* (1947), 50–60.

[18] Michael D McKay, Richard J Beckman, and William J Conover. 2000. A comparison of three methods for selecting values of input variables in the analysis of output from a computer code. *Technometrics* 42, 1 (2000), 55–61.

[19] Andreas C Müller and Sarah Guido. 2016. *Introduction to machine learning with Python: a guide for data scientists*. " O'Reilly Media, Inc.".

[20] Randal S Olson, Nathan Bartley, Ryan J Urbanowicz, and Jason H Moore. 2016. Evaluation of a tree-based pipeline optimization tool for automating data science. In *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO)*. 485–492.

[21] Oleksandr Shchur, Caner Turkmen, Nick Erickson, Huibin Shen, Alexander Shirkov, Tony Hu, and Yuyang Wang. 2023. AutoGluon-TimeSeries: AutoML for probabilistic time series forecasting. In *International Conference on Automated Machine Learning (ICML)*.

[22] Jasper Snoek, Hugo Larochelle, and Ryan P Adams. 2012. Practical bayesian optimization of machine learning algorithms. In *Advances in Neural Information Processing Systems*. 2951–2959.

[23] P. Stetsenko. 2020. *Machine learning with Python and H2O*. http://docs.h2o.ai/h2o/latest-stable/h2o-docs/booklets/PythonBooklet.pdf (accessed January 2024).

[24] Anh Truong, Austin Walters, Jeremy Goodsitt, Keegan Hines, C Bayan Bruss, and Reza Farivar. 2019. Towards automated machine learning: Evaluation and comparison of AutoML approaches and tools. In *2019 IEEE 31st International Conference on Tools with Artificial Intelligence (ICTAI)*. IEEE, 1471–1479.

[25] A Helen Victoria and Ganesh Maragatham. 2021. Automatic tuning of hyperparameters using Bayesian optimization. *Evolving Systems* 12 (2021), 217–223.

[26] Xilu Wang, Yaochu Jin, Sebastian Schmitt, and Markus Olhofer. 2023. Recent Advances in Bayesian Optimization. *ACM Computing Surveys (CSUR)* 55, 13s, Article 287 (jul 2023), 36 pages. https://doi.org/10.1145/3582078

[27] Jia Wu, Xiu-Yun Chen, Hao Zhang, Li-Dong Xiong, Hang Lei, and Si-Hao Deng. 2019. Hyperparameter optimization for machine learning models based on Bayesian optimization. *Journal of Electronic Science and Technology* 17, 1 (2019), 26–40.