AutoML from Software Engineering Perspective: Landscapes and Challenges

Chao Wang*†, Zhenpeng Chen‡, Minghui Zhou*†§
*School of Computer Science, Peking University, Beijing, China
†Key Laboratory of High Confidence Software Technologies, Ministry of Education, Beijing, China
‡University College London, London, United Kingdom
wchao@pku.edu.cn, zp.chen@ucl.ac.uk, zhmh@pku.edu.cn

***Abstract*—Machine learning (ML) has been widely adopted in modern software, but the manual configuration of ML (e.g., hyper-parameter configuration) poses a significant challenge to software developers. Therefore, automated ML (AutoML), which seeks the optimal configuration of ML automatically, has received increasing attention from the software engineering community. However, to date, there is no comprehensive understanding of how AutoML is used by developers and what challenges developers encounter in using AutoML for software development. To fill this knowledge gap, we conduct the first study on understanding the use and challenges of AutoML from software developers' perspective. We collect and analyze 1,554 AutoML downstream repositories, 769 AutoML-related Stack Overflow questions, and 1,437 relevant GitHub issues. The results suggest the increasing popularity of AutoML in a wide range of topics, but also the lack of relevant expertise. We manually identify specific challenges faced by developers for AutoML-enabled software. Based on the results, we derive a series of implications for AutoML framework selection, framework development, and research.**

***Index Terms*—AutoML, software engineering, application, challenge**

I. Introduction

The vast success of machine learning (ML) paves its way into a wide range of software applications, including autonomous driving [1], medical treatments [2], speech recognition [3], and natural language processing [4]. These software applications (i.e., ML software) integrate ML models trained using a large data corpus with ML programs. In ML programs, software developers need to configure various settings, such as the way of data processing, the choice of ML models, and the hyperparameters of models.

Problematic configuration of these settings could make ML software miss the functional requirements (e.g., correctness), resulting in a lot of ML testing and repair efforts that need to be invested later [5]. Moreover, manually configuring these settings can be challenging for software developers, as it requires a high level of domain knowledge. Therefore, how to find the optimal configurations of ML programs automatically, known as *automated ML (AutoML)*, has been an urgent and significant Software Engineering (SE) concern [6], [7], [8], [9].

To help developers achieve AutoML, major software companies have rolled out software frameworks, such as `nni` from `Microsoft` [10] and `autogluon` from `Amazon` [11]. These frameworks consider AutoML as a search-based SE problem whose starting point is a default model configuration. They provide off-the-shelf APIs to support the automated mutation of the starting point and exploration of configuration space to find the optimal one for the problem.

Despite the widespread interest in AutoML, there is no comprehensive understanding of two fundamental questions: how AutoML is used by developers (**RQ1**) and what challenges developers encounter in using AutoML for software development (**RQ2**). For RQ1, the freshness of AutoML leaves unclear basic problems in the AutoML software ecosystem, e.g., whether AutoML is widely used in ML practice, what kinds of repositories make use of AutoML frequently, and whether different kinds of repository topics have different preferences for AutoML frameworks. The exploration of the use of different AutoML frameworks in different kinds of applications could provide insights for software developers in making decisions on framework selection. For RQ2, AutoML poses specific programming challenges to developers (e.g., how to customize the search space), which are frequently complained about and asked on GitHub and developers' Q&A forums [12], [13], [14]. Moreover, recent SE studies have revealed that the AutoML practice enabled by existing frameworks is fault-prone [8], time-consuming [6], and resource-intensive [8]. Through uncovering the specific challenges and software faults that developers frequently encounter about AutoML in software development, people could design automated assistance tools (e.g., for testing and debugging) and improve existing frameworks to help developers tackle these challenges in a more targeted manner.

To fill the knowledge gap, we conduct the first comprehensive study on analyzing the use and challenges of AutoML in software development practice. In this study, we target whoever uses AutoML frameworks, including application developers, data scientists, and so on. We collect 5,748 AutoML downstream repositories that use AutoML frameworks (and sample 1,554 repositories for further analysis), 769 AutoML-related issues from Stack Overflow (SO), and 1,437 relevant GitHub issues. Based on the collected data, we answer the following sub-questions of RQ1 and RQ2.

RQ1.1 (popularity): How popular is AutoML in software applications? We analyze the AutoML downstream reposito-

§ Corresponding Author

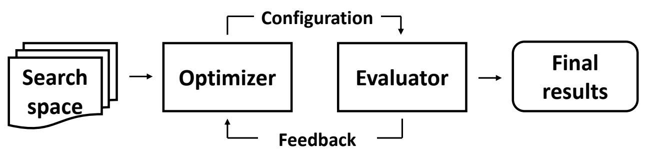


Fig. 1. Basic AutoML workflow

ries and the AutoML-related SO questions in recent years. The results indicate the increasing popularity of AutoML in software development and demonstrate the timeliness of this study.

RQ1.2 (applications): What software applications use AutoML? We manually label the topics of AutoML downstream repositories and observe that AutoML has been used in a wide range of repository topics. In addition, we find that different kinds of applications have different preferences for AutoML frameworks.

RQ2.1 (difficulty): How difficult is AutoML for software developers? We use widely-adopted SE metrics to measure the difficulty of AutoML based on the relevant SO questions. The results suggest that AutoML lacks experts compared to other SE topics.

RQ2.2 (challenges): What challenges do developers frequently encounter when using AutoML in software development? We identify developers' challenges from the SO posts and GitHub issues. We manually identify 26 specific challenges with AutoML in software development. The resulting taxonomies indicate that developers face a wide spectrum of challenges in AutoML, and call for actions by SE researchers and AutoML framework vendors to tackle them.

As an additional contribution to the research community, we make our code scripts and dataset publicly available for researchers to replicate and build upon [15].

II. Background and Related Work

AutoML workflow. ML techniques are frequently used to derive actionable insights from datasets. Traditional ML model development is time-consuming, resource-intensive, and requires expertise to compare and train models, thus AutoML is leveraged to automate the process. A simplified basic workflow [16] for AutoML approaches is shown in Figure 1. The optimizer generates model configurations based on given search space and feedback from the evaluator and passes the configuration to the evaluator. The evaluator measures the model with assigned configuration on specific metrics and gives feedback to the optimizer. The configuration includes the constructed features, ML algorithms, model hyperparameters, and so on. Overall, the configuration that achieves the best performance within the given time and resource constraints will be considered the final result. Studies on AutoML are dedicated to improving the efficiency of the evaluator and the effectiveness of the optimizer [17], [18], [19].

AutoML frameworks. As AutoML technologies grow mature, AutoML frameworks emerge rapidly. Existing AutoML frameworks can be classified into two categories, open-source frameworks and commercial frameworks. Open-source frameworks contain frameworks from two sources, including auto-sklearn [17], autokeras [20], and TPOT [21] that come from academia and are based on existing ML frameworks; and nni [22], H2O AutoML [23], and autogluon [24] that come from industry. Open-source frameworks offer great flexibility with extensive customization, while developers have to provide their own computation resources. Commercial frameworks provide paid AutoML solutions, including cloud providers like Google Cloud AutoML [25] and Amazon SageMaker Autopilot [26], and AutoML platforms targeting business users like DataRobot [27]. Commercial frameworks usually provide end-to-end services where developers only upload the dataset and then easily obtain the trained model.

Related work. As we aim to understand the use and challenges in AutoML, we discuss related work that enables AutoML with efficient algorithms and helps developers to better leverage AutoML. To automate the process of AutoML, many studies focus on feature engineering [28], [29], [30], model selection [17], [18], [19], and neural architecture search [31], [32], [33], [34], which support the emergence of AutoML frameworks. With the growing popularity of AutoML frameworks, researchers study how to assist developers to use AutoML [6], [7], [8], [9], [12], [35]. Cambronero et al. [7] propose AMS to assist developers in strengthening the search space via mining complementary information from related code corpus and API documents. Gao et al. [8] design DnnSAT to help developers with configuration, which estimates needed resources via constraint solving. Nguyen et al. [6] accelerate the neural architecture search by replacing the initial model with models mined from GitHub repositories. Different from existing studies targeting specific challenges encountered by developers, we conduct the first comprehensive study on AutoML-related challenges and identify 26 categories of challenges that provide in-depth insights.

III. Data collection

We construct the dataset of our interest from three data sources. First, we use the most commonly-used data sources for studying challenges in development, i.e., SO and GitHub, to collect AutoML-relevant issues that developers cannot resolve and thus post online for technological advice. The issues derived from the two sources are demonstrated to be representative and align well with the real-world challenges encountered by software practitioners [36], [37]. Second, to investigate how AutoML is used by developers in the wild, we collect relevant software repositories from World of Code (WoC) [38], an infrastructure that contains massive git repositories.

A. Mining Stack Overflow

We collect AutoML-related questions from the entire SO dataset downloaded from its official Data Dump [39] on December 6, 2021, which covers more than 20 million posts

TABLE I
STATISTICS OF SELECTED FRAMEWORKS

Project	#Stars	Creation date	#Issues	#Selected issues	#Downstream repositories
Microsoft/nni [22]	12.1k	2018-06-01	1,450	498	321
EpistasisLab/tpot [21]	8.8k	2015-11-03	833	270	3,662
Keras-team/autokeras [20]	8.6k	2017-11-19	790	112	652
Automl/auto-sklearn [17]	6.6k	2015-07-02	811	425	930
awslabs/autogluon [24]	4.9k	2019-07-29	570	132	183

from July 31, 2008, to December 5, 2021. Each SO question is tagged with one to five tags to indicate its topics.

To obtain AutoML-related SO questions, we first need to identify AutoML-related tags. In line with previous work [40], [41], we construct a set of AutoML-related tags with the following steps: (1) We construct the initial set of tags with the general keyword "automl", i.e., $T_{ini} = \{automl\}$. (2) We collect a subset of posts P_{sub} that contain any tag in T_{ini}, and we construct the candidate tag set T_{can} with all tags of posts in P_{sub}. (3) We filter out tags in the T_{can} with two metrics *significance* and *relevance*, which are widely adopted in previous work [40], [41], [42]. We calculate the *significance* and *relevance* for each tag t in T_{can} as follows (where $pwitht$ refers to posts with t):

$$Significance(t) = \frac{\mid \{p \mid p \in \mathcal{P}_{sub} \wedge pwitht\} \mid}{|\mathcal{P}_{sub}|} \quad (1)$$

$$Relevance(t) = \frac{\mid \{p \mid p \in \mathcal{P}_{sub} \wedge pwitht\} \mid}{\mid \{p \mid p \in \mathcal{P}_{all} \wedge pwitht\} \mid} \quad (2)$$

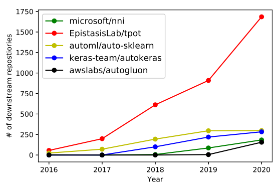
A tag t is significantly relevant to AutoML if Significance(t) and Relevance(t) are higher than specific thresholds. We use the lowest thresholds among all previous studies [40], [41] to collect the maximum number of relevant tags. We select only the tags whose significance is higher than 0.005 and whose relevance is higher than 0.05. As a result, we extract six additional tags, *"google-cloud-automl"*, *"google-cloud-automl-nl"*, *"h2o.ai"*, *"tpot"*, *"auto-keras"* and *"neuraxle"*. We manually check the description of all retrieved tags and find that all of them are AutoML frameworks, covering both commercial and open-sourced frameworks. We also check other popular AutoML framework names to expand the tag set, e.g., "nni" and "autogluon ", and these tags only contain a small number of posts so we do not include them. We extract questions tagged with "*automl*" or any of the six tags from the SO dataset and thus obtain 769 AutoML-related SO questions.

B. Mining GitHub Issues

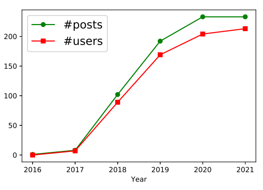
Following previous work [43] that selects frameworks by the star count filtering and manual check of repository descriptions, we first identify popular AutoML frameworks on GitHub, and then mine issues in their official repositories to uncover the challenges that developers encounter when using them. Developers raise issues (with detailed descriptions such as fault symptoms and running environment) in GitHub repositories to seek technical advice when they face difficulties. The detailed descriptions of issues as well as the following discussions allow us to distill developers' challenges from them easily. Thus, following previous work in SE [44], [45], [46], [47] that summarize challenges in software development from GitHub issues, we employ GitHub issues as a data source for the following analysis.

1) Identification of AutoML frameworks: To understand how AutoML is used by developers, we target open-source AutoML frameworks on GitHub that are representative and widely used. We do not investigate the use of commercial frameworks because developers interact with the platform UI directly in most cases and we cannot track their use. To identify representative open-source AutoML frameworks, we retrieved the top 10 projects sorted with "most stars" under the "automl" topic, and manually check their description and documentation. We identify 5 projects that are actually AutoML frameworks, shown in Table I with statistics. These frameworks provide a complete solution for developers to obtain the optimal model for the specific dataset, and support typical AutoML tasks including feature selection, model selection, and parameter tuning. They support traditional ML techniques, e.g., `tpot` and `auto-sklearn` (both based on Scikit-learn [48]), as well as deep learning, e.g., `autokeras` (based on Keras [49]), `autogluon` (supporting PyTorch [50] model tuning), and `nni` (supporting neural architecture search for both PyTorch [50] and TensorFlow [51]), which indicates the representativeness of selected frameworks.

2) Extraction of GitHub Issues: We use the official GitHub API [52] to mine all the issues of the selected frameworks on January 1, 2022. Developers open issues on GitHub for various reasons, e.g., bug reports and feature requests, so we filter out irrelevant issues with three rules. First, in line with previous work [44], [45], we filter out issues via labels. We manually inspect all issue labels in selected frameworks, and identify all labels that correspond to specific development tasks, e.g., "bug", "todo", and "help wanted". We excluded issues with identified labels. However, not all issues are tagged with corresponding labels, so we further filter task issues related to the pull requests and commits. Then, we obtain all pull requests and commits of selected frameworks via a web crawler, and filtered out all issues mentioned in the pull requests or commits, which are likely to be bound up with specific developing tasks. Next, following previous work [44], we extract all closed issues with at least one response to ensure the quality of selected issues. Overall, as shown in Table I, we collect 1,437 issues.



(a) The number of new downstream repositories per year



(b) The number of new AutoML questions and involved users on SO per year

Fig. 2. The popularity trend of AutoML

C. *Mining World of Code*

To understand the use of AutoML in real-world software applications, we use the World of Code (WoC) [38] to collect the downstream repositories of the selected AutoML frameworks. WoC is an infrastructure storing massive open-source Git (version control system) data and it mines Git objects of open source repositories across major code hosting platforms, including GitHub, GitLab, Bitbucket, and so on. Specifically, we use version T of WoC [53], which contains code of more than 140 million distinct repositories by February 2021. Following the definition of software chain proposed by Tan et al. [54], we define the downstream repositories as repositories that import specific frameworks. We extract downstream repositories of each AutoML framework via WoC, which identified downstream repositories by import statements. Forked repositories are excluded to avoid duplication. Overall, we collect 5,748 downstream repositories, and the number for each framework is shown in Table I.

IV. RQ1.1: Popularity

A. *Methodology*

To illustrate the popularity trend of AutoML usage, we employ two metrics to measure the popularity trend: the number of repositories using AutoML, and the number of developers using AutoML. For the number of repositories, we measure the popular trend of downstream repositories of AutoML frameworks. We consider the committed time of the first AutoML framework-related import statement as the creation time and calculate the number of created downstream repositories per year. For the number of developers using AutoML, it is tricky to count developers directly. Thus, following previous work [55], [56], [57], we target users involved in AutoML-related SO questions, which can reflect the amount of AutoML users. We calculated the number of AutoML-related SO questions per year, as well as involved SO users within these questions.

B. *Results*

Figure 2(a) shows the popularity trend of downstream repositories, which illustrates that all AutoML frameworks have continuous growth in the number of downstream repositories. Both `nni`, a new framework, and `tpot`, a mature framework, show more than 100% growth of new downstream repositories in 2020 compared to 2019. The growth trend of downstream repositories with `autokeras` stabilizes in recent years because it is out of maintenance for a period of time in 2019 [58]. The overall rapidly growing trend suggests that AutoML is increasingly used in ML development, and demonstrates the timeliness of this study.

Figure 2(b) shows the trend of AutoML-related SO questions and involved users. The figure illustrates that, with the emergence of extensive studies on AutoML and the maturity of AutoML frameworks, AutoML-related questions and users increased rapidly in 2018 and 2019. We can observe a more than 1.8x increase in the number of involved users in 2019 compared to 2018. The increasing trend suggests that a growing number of developers are using AutoML in ML development. Meanwhile, the increase also suggests that developers often encounter challenges with AutoML frameworks, and seek solutions online, again demonstrating the timeliness of this study.

Findings 1: The numbers of repositories using AutoML and developers involved in AutoML are both showing rapid growth, suggesting the growing willingness of developers to use AutoML in ML development.

V. RQ1.2: Applications

A. *Methodology*

As explained in Section III-C, we collect 5,748 downstream repositories of AutoML frameworks. Following the threshold of sampling in previous work [55], [59], [60], ensuring a 95% confidence level and a 5% confidence interval, we randomly sample 350 downstream repositories for each framework. For frameworks with less than 350 downstream repositories (`nni` and `autogluon`), we select all their downstream repositories. In total, we collect 1,554 downstream repositories. Next, we present the procedures of the taxonomy construction.

1) Pilot Construction: To get familiar with the dataset, two authors, who have three and five years of ML experience, respectively, carefully read all introductions and README files in all downstream repositories. They follow the standard open coding procedure [61] to construct the pilot taxonomy for randomly sampled 30% of the dataset. The detailed procedures are described below.

Two authors read and reread the introduction and the README file of sampled downstream repositories. They generate short descriptions as initial codes to indicate the **topic** that shows the purpose of the repository.

With initial codes, two authors construct the taxonomy for downstream repository topics with the following steps. (1) Iteratively group similar initial codes into categories. (2) Consider each category, whether it contains sub-categories, and how these sub-categories interact and relate to the main category. (3) Define the final categories. Each repository is assigned to one topic category. Repositories without sufficient information to understand the circumstance would be labeled with “unclear”.

TABLE II
THE TAXONOMY OF AUTOML DOWNSTREAM REPOSITORY TOPICS

Inner-category	**Category**	**Category description**	**#Downstream repositories**
Software application (35.3%)	NLP application	Applications on natural language processing tasks, including semantic analysis, named entity recognition, text classification, and so on.	36 (4.55%)
	CV application	Applications on computer vision tasks, including object detection, image classification, face parsing, and so on.	34 (4.30%)
	AutoML implementation	Applications that implement and reproduce algorithms in AutoML studies.	14 (17.70%)
	Other CS-related application	Applications on classic CS-related topics, including recommendation systems (13), audio processing (5), time-series data procession (3), and so on.	36 (4.55%)
	Game	Applications that are games or just for entertainment.	7 (0.88%)
	Biology and Medicine	Applications related to challenges in Biology or Medicine.	46 (5.82%)
	Business and Marketing	Applications related to challenges in Business and Marketing.	40 (5.06%)
	Other practical challenge	Applications related to other practical challenges that cannot fit in other categories, including air quality prediction, grain production prediction, traffic simulator, and so on.	66 (8.34%)
Software education (23.89%)	Learning project	Projects that store course materials, tutorials, or source code for homework.	189 (23.89%)
Software toolkit (9.61%)	Library and toolkit	Projects that are libraries, wrappers, or toolkits providing easily accessible API for ML or AutoML.	76 (9.61%)
Crowd-sourced competition (8.47%)	Competition source code	Projects that stores source code of solutions for competitions like Kaggle and hackathons.	67 (8.47%)
Paper experiment (7.84%)	Paper in CS	Projects that store source code for research papers of different domains in Computer Science(CS), including AutoML (18), computer vision (9), model theory (7), security (4), code analysis (3), and so on.	53 (6.70%)
	Paper in other subjects	Projects that store source code for research papers in other subjects, including Biology (6) and Physics (3).	9 (1.14%)
Benchmark (3.54%)	Benchmark project	Projects that compare the performance of different AutoML frameworks.	28 (3.54%)
Others (11.38%)	Other project	Projects including personal project collection, templates code, toy projects, and so on.	90 (11.38%)
Total			791 (100.00%)

2) Reliability Analysis: Two authors independently label the rest 70% of the dataset based on taxonomies generated in the pilot construction. Repositories that cannot fit in any category in the taxonomy are labeled with "pending". The Cohen's Kappa is employed to measure the inter-rater agreement during the labeling. The Kappa value is 0.95, indicating the substantial agreement and the reliability of the coding procedure.

The conflicts in independent labeling are discussed by two authors. If they cannot reach an agreement, the conflict would be resolved by a non-author arbitrator, who has rich experience in both ML and AutoML. All repositories labeled with "pending" are also discussed by two authors and the arbitrator, to see if they can be assigned to existing categories, or if new categories need to be added. One category is added at the end.

Overall, we identify topics of 1,554 repositories and construct the taxonomy for the topics of AutoML downstream repositories.

B. Results

Table II presents the taxonomy of AutoML downstream repository topics, which contains 15 categories. Two types of repositories are excluded in the taxonomy: projects that cannot be accessed due to removal or turning to private (152), and projects that cannot be classified due to the lack of informative introductions and README files (611). We also merge categories that account for a relatively small percentage, e.g., we merge the categories like applications for recommendation systems, audio processing, time-series data procession, and so on, into one category.

The taxonomy illustrates that AutoML frameworks are frequently used in Computer Science (CS) studies and software development, suggesting the usefulness of AutoML in assisting ML development. Meanwhile, AutoML frameworks are used to conduct research in other subjects like Biology and Physics and address interdisciplinary challenges including Biology, Business, Environment, and so on, where developers may not have rich ML background knowledge. AutoML frameworks are often introduced in ML courses and tutorials, and frequently used in homework and competitions to

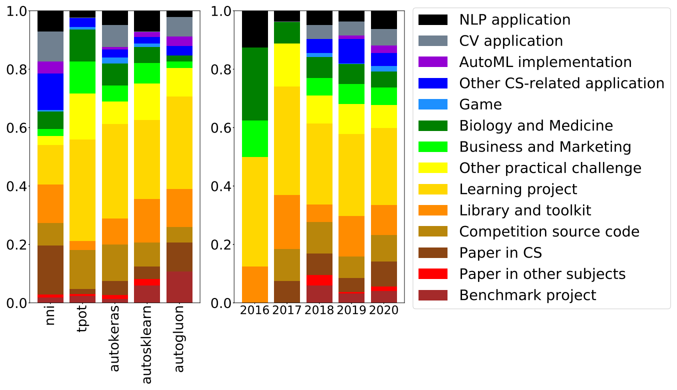


Fig. 3. The topics of downstream repositories of different AutoML frameworks and in different years

efficiently build the model, suggesting the extensive interest AutoML has received from software educators and beginners.

Figure 3 presents the distribution of downstream repository topics within different AutoML frameworks, indicating that AutoML frameworks vary in terms of application scenarios. As a new AutoML framework developed and promoted by Microsoft, providing support for the implementation of various state-of-the-art AutoML algorithms, `nni` is widely used in CS-related applications (34%) and studies (17%). Conversely, `tpot` has a high proportion of applications on interdisciplinary applications (38%), learning projects (35%), and competitions (13%). Meanwhile, the selection of frameworks varies from domain to domain. For example, developers tend to choose `nni`, `autokeras`, and `autogluon` for CV applications, which support automation on deep learning techniques that sufficiently support CV tasks.

Figure 3 also presents the evolution of downstream repository topics, which shows the distribution of newly created downstream repositories by topic categories per year, illustrating that the distribution fluctuates over time since 2018. Over time, the repositories that use AutoML have become more diverse, starting with easy tasks using tabular datasets, and then gradually applying to cutting-edge domains. The emergence of AutoML studies since 2018 and the implementation of state-of-the-art AutoML algorithms may play a driving role in the evolution [62]. Notably, learning projects always take the largest share whose percentage exceeds 20% every year, suggesting that beginners constantly have a strong interest and need to learn and use AutoML.

Findings 2: (1) The AutoML is widely used in practice, ranging from research, interdisciplinary applications (e.g., application in Biology), to education and programming competitions like Kaggle. (2) Different frameworks vary in prevalence in different domains, e.g., `nni`, `autokeras`, and `autogluon` have a relatively higher proportion on CV applications. (3) The diversity of AutoML downstream repository topics increased over time, and the learning projects have always been proportionally dominant, suggesting the strong need from beginners.

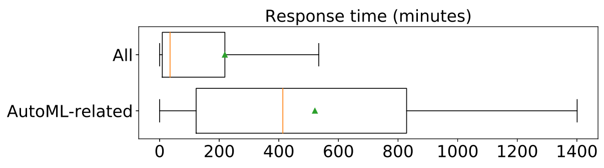


Fig. 4. The distribution of response time

VI. RQ2.1: Difficulty

A. Methodology

To measure the difficulty of AutoML challenges, following previous work [41], [42], [55], [56], [63], [64], [65], [66], we use two widely used metrics, the percentage of questions without any accepted answer ($\%no\_acc$) and the response time for a SO question to receive an accepted answer. For $\%no\_acc$, we calculated the percentage among AutoML-related SO questions and all SO questions, and compare them with other topics reported in previous work. For response time, we calculate the time interval from when the question is posted to when the accepted answer is created, for both AutoML-related questions and all SO questions with accepted answers. The above two indicators suggest the degree of scarcity of relevant expertise.

B. Results

The percentage of AutoML-related SO questions without accepted answers ($\%no\_acc$) is 73.0%, which is much higher than the percentage of all SO questions ($\%no\_acc$ = 51.4%). AutoML-related SO questions also have a higher $\%no\_acc$ than other well-studied topics in SE, including deep learning software deployment ($\%no\_acc$ = 70.7% [55]), serverless computing($\%no\_acc$ = 61.5% [64]), big data ($\%no\_acc$ = 60.5% [41]), mobile ($\%no\_acc$ = 55.0% [65]), and concurrency ($\%no\_acc$ = 43.8% [63]).

Figure 4 shows the distribution of response time of AutoML-related questions and all questions on SO. The figure illustrates that AutoML-related questions have a much longer response time, which is about 413.5 minutes in the median, compared with 35 minutes for all SO questions in the median. The median value of response time is also higher than other topics reported in previous work, including deep learning deployment (405 minutes [55]), big data (198 minutes [41]), serverless computing (190 minutes [64]), mobile (55 minutes [65]), and concurrency (42 minutes [63]).

AutoML-related questions have a higher $\%no\_acc$ and a longer response time than other popular topics with several potential factors: the complex nature of AutoML-related questions, the confusing formalization or description from inexperienced questioners, and the lack of experts and domain knowledge for such a novel topic. Whatever the reason, the higher value of the two metrics reveals the high difficulty of asking for a technical solution online with AutoML-related challenges, and the lack of expertise, which again rationalizes the necessity of our study that systematically and comprehensively organizes AutoML-related challenges.

Findings 3: Compared to other topics, AutoML-related questions present a significantly higher percentage without accepted answers, and a significantly longer time to receive an accepted answer, suggesting the difficulty in answering AutoML-related questions and the lack of relevant expertise.

VII. RQ2.2: CHALLENGES

A. Methodology

To understand the challenges in AutoML use, we manually analyze AutoML-related questions. We collect 208 SO questions with accepted answers from all 769 questions retrieved in Section III-A following previous work [40], [55], and all 1,437 GitHub issues collected in Section III-B2. In total, we collect 1,645 posts (i.e., GitHub issues and SO questions) for the taxonomy construction. The size of the dataset is comparable and even larger than previous work [37], [55], [67], [64], [68], [69] that manually analyze GitHub issues and SO questions.

Following the same procedure described in Section V-A1, two authors randomly sample 30% of posts and generate initial codes for them. For each post, they generate initial codes to indicate the **challenge** that describes the topic of the problem developers encountered. Posts that are not exactly questions proposed by developers would be labeled with “false positive”, e.g., discussions on the function design. Posts with questions that are actually triggered by confirmed bugs are also labeled with “false positive”.

Two authors labeled the rest 70% of posts following the same schema in Section V-A2. Note that each post can be assigned to multiple relevant categories of challenges. The Kappa value for the challenges is 0.81, indicating the substantial agreement and the reliability of the coding procedure. One category is added within the reliability analysis. Overall, we identify the challenges of 1,645 posts.

We further investigate the variation in challenges across AutoML frameworks and the evolution of AutoML challenges. For five selected open-source AutoML frameworks, we calculate the distribution of GitHub issues in each framework by challenges categories. We also consider `Google Cloud AutoML`(`GCA`), which is a representative commercial AutoML framework, and 47% (98/208) of SO questions in our dataset have corresponding labels. We calculate the distribution for `GCA` with the above 98 SO questions. To investigate the evolution of AutoML challenges, we calculate the distribution of challenge categories per year.

B. Results

1) Challenge categories: After excluding “false positive” (298) and “Unclear” (190) posts, we have 1,157 posts with 1,170 labels for the taxonomy of challenges. Figure 5 illustrates the hierarchical taxonomy of challenges, indicating a wide range of challenges that developers encounter. The taxonomy has 6 inner categories and 26 leaf categories and the percentage of labels in each category is presented in Figure 5. We further classified 26 challenge categories into three types: challenges with practical solutions (marked with green background in Figure 5), challenges that require more engineering support (marked with a yellow background in Figure 5), and challenges that require more theoretic support (marked with red background in Figure 5). A detailed description of each category can be found in the supplementary material [15]. In the following section, we describe and exemplify each challenge category except for resolved challenges as follows.

(1) Environment. This inner category covers challenges that developers encounter with setting up the environment.

GPU scheduling. Developers encounter challenges in the configuration of GPU, including introducing GPU to the trial [70] and scheduling the trial on multiple GPUs [71], accounting for 3.18% of all challenges.

Compatibility with ML frameworks. The rapid evolution of ML frameworks [54] leads to challenges for the use and maintenance of AutoML framework. Framework vendors need a strategy for dependency maintenance to balance the update cost and users’ needs [72], while related knowledge remains sparse. Framework users need to use the specific version of ML frameworks to ensure compatibility [73], which may lead to functional limitations of the trial.

(2) Data preparation. This inner category covers challenges developers encountered when preparing the dataset, including challenges in *data cleaning*, *data adaption*, and *data type/format* conversion.

(3) Trial. This inner category covers challenges that developers encounter in the setup and procedure of AutoML trials, which accounts for 23.57% and covers AutoML-specific challenges.

Metric selection/customization. The metric used to assess the effectiveness of candidate models is essential to the trial, which should be selected appropriately and receive great concerns (5.56%). The improper metric may lead to unsatisfied performance (e.g., using accuracy for an imbalanced dataset [74]), or even faults (e.g., using precision in multiclass classification [75]). Despite various metrics being implemented by AutoML frameworks, developers still have a demand for customizing the metric for specific purposes, e.g., the Sharpe ratio [76]. Developers propose questions on how to customize the metric [77], and encounter errors if the customized metric is not implemented correctly [78], [79].

Search space customization. The set of search space plays a significant role in AutoML algorithms, which frequently bothers developers (5.91%). Developers lacking sufficient expertise may encounter difficulties in defining a suitable search space, who are unclear about which variables can be searched [80], and how to specify the scope [13]. Overly broad, restricted, or unrealistic search space may lead to poor performance of the obtained model or even faults, e.g., the inapplicable preprocessor setting in search space brings the error [81].

Intermediate results presentation. Despite AutoML frameworks assisting developers on ML, AutoML remains the

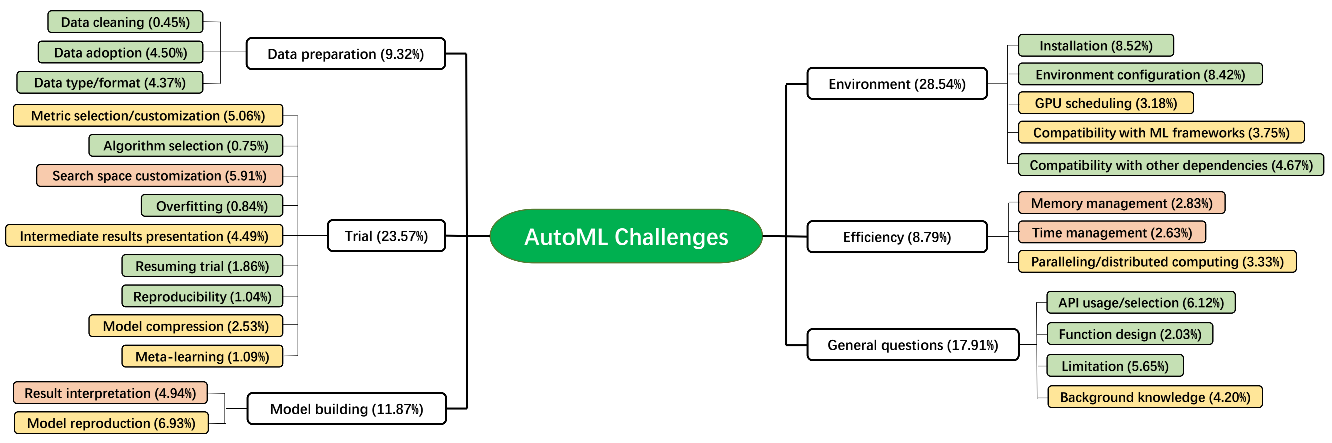

AutoML Challenges
Data preparation (9.32%)
Data cleaning (0.45%)
Data adoption (4.50%)
Data type/format (4.37%)
Trial (23.57%)
Metric selection/customization (5.06%)
Algorithm selection (0.75%)
Search space customization (5.91%)
Overfitting (0.84%)
Intermediate results presentation (4.49%)
Resuming trial (1.86%)
Reproducibility (1.04%)
Model compression (2.53%)
Meta-learning (1.09%)
Model building (11.87%)
Result interpretation (4.94%)
Model reproduction (6.93%)
Environment (28.54%)
Installation (8.52%)
Environment configuration (8.42%)
GPU scheduling (3.18%)
Compatibility with ML frameworks (3.75%)
Compatibility with other dependencies (4.67%)
Efficiency (8.79%)
Memory management (2.83%)
Time management (2.63%)
Paralleling/distributed computing (3.33%)
General questions (17.91%)
API usage/selection (6.12%)
Function design (2.03%)
Limitation (5.65%)
Background knowledge (4.20%)

Fig. 5. The taxonomy of challenges, including challenges with practical solutions (green background), challenges that require more engineering support (yellow background), and challenges that require more theoretic support (red background)

"black box" for many developers. Thus, to obtain the final optimal model, developers seek the support of monitoring the entire search process, which accounts for a relatively large percentage (4.49%). Developers want to collect information about which models are tested in the trial and how they perform [82], [83] to rationalize the obtained optimal model.

Model compression. Model compression is a new research subfield of AutoML and related algorithms are implemented by AutoML frameworks like `nni`, which help developers to reduce the model size and accelerate model inference without losing performance significantly. However, developers are not familiar with model compression, which leads to challenges. Developers are unclear about some basic concepts like what models are supported [84] and the difference between algorithms [85]. They also often fail to compress the model due to erroneous operation, e.g., forgetting to unwrap the original model [86]. Despite a new subfield, considerable related questions (2.63%) concern model compression, demonstrating its high difficulty.

Meta-learning. Meta-learning is also a new subfield of AutoML, where the metadata of datasets is used for model selection and algorithm tuning. Developers are unfamiliar with meta-learning, so they wonder how meta-learning works [87] and how to set relevant configurations [88]. Another common issue is that developers are unaware of the involvement of meta-learning in AutoML, where they obtain the same results with random seeds because meta-learning is used to warm start the optimization procedure [89].

(4) Model building. After the trial, developers need to reproduce the optimal model with results for further inference, where challenges within this inner category exist.

Results interpretation. ML frameworks already have sophisticated tools for model analysis and visualization, whereas AutoML frameworks do not, which leads to numerous questions (4.94% of all categories) about how to understand given results and retrieve more information for the optimal model. Developers encounter challenges with how to retrieve and understand the architecture of the best model, e.g., in `tpot`, the obtained model is present in the form of pipelines, which leads to confusion [90] and misunderstanding [91] on complex pipelines with multiple steps. Apart from the architecture, developers also call for more information to evaluate the obtained model and assess whether more trials are needed, e.g., CV scores [92], confusion matrix [93], and feature importance [94].

Model reproduction. The relatively high percentage (6.93%) of this category suggests that developers often encounter challenges with model building and inference. In some cases, the optimal model is already fitted with test data. Developers have questions about how to save the model properly and reload it for further training and prediction, e.g., developers wonder how to save the model that can be loaded later [95]. In other cases, AutoML frameworks only provide the architecture of the optimal model. Developers seek ways to automatically convert the architecture string to modeling code in ML frameworks such as `scikit-learn`, which they do it manually now [96], [97], [98]. Incorrect operations in model reproduction lead to faults, e.g., using the unfitted model directly [99].

(5) Efficiency. Efficiency plays a significant role in AutoML frameworks and the optimal solution is required to be obtained with limited resources and time. This inner category includes challenges related to memory and time management, as well as paralleling and distributed computing.

Memory management. AutoML trials are instinctively memory-consuming, thus AutoML frameworks provide the ability to limit memory usage for trials, and usually set default limitations on memory. Inexperienced developers have difficulty in estimating the amount of memory needed for trials and relevant knowledge is lacking [100]. Some are even unaware of the default memory limitation [101], which leads to errors. Moreover, developers have trouble with large datasets in AutoML, and they are recommended to increase the memory limitation and running time [102] or use the subset of the dataset [103].

Time management. Developers have to limit the duration of trials so that the search can end in a limited time. AutoML frameworks provide APIs for time management, including setting time limitations on each iteration and the whole search. How to estimate the trial time and properly set the limitation is challenging to inexperienced developers, accounting for 2.63% of all categories. Related expertise is still missing, e.g., developers ask for the rule of thumb choosing time limitation on each trial, whereas the best practice is still unclear according to the feedback [104]. Unreasonable time limits can drive issues, where overly long time limitations lead to substantial time costs, and short time limitations may cause unexpected faults [105].

Parallel/distributed computing. Parallel computing and distributed computing are widely used in AutoML frameworks to improve efficiency. Developers encounter challenges in parallel computing and distributed computing, including resource sharing [106], [107] and related configuration, accounting for 3.33% of all categories.

(6) General questions. This inner category includes questions developers have on general principles.

Background knowledge. This category includes questions about the general procedure of conducting the AutoML trial, e.g., "*What's the input and the output of nni?*" [108], and the background theory, e.g., "*How TPOT tune the parameters?*" [109]. The relatively high percentage (4.20%) suggests that not all users clearly understand how AutoML works. These questions are proposed by developers without expertise in AutoML or even ML, and can be easily solved with more examples and explanations in the document [110].

Findings 4: We build a taxonomy of challenges in using AutoML frameworks with 26 categories, illustrating a wide spectrum of challenges developers face in AutoML. The frequent categories range from general challenges like *Installation* (8.52%), *Environment configuration* (8.42%), and *API usage/selection* (6.12%), to AutoML specific challenges like *Model reproduction* (6.93%), *Search space customization* (5.91%), and *Metric selection/customization* (5.06%).

2) Distribution of challenges among frameworks: Figure 6 illustrates the distribution of challenges categories in different frameworks, suggesting that developers encounter different challenges within different AutoML frameworks.

Commercial AutoML frameworks pose different difficulties for developers compared with open-source frameworks. As a representative commercial platform that provides AutoML cloud solutions, `GCA` has a distinctive workflow from open-source frameworks, which leads to a distinct pattern of challenges. Developers mainly have challenges with how to properly prepare the dataset, how to correctly interact with the platform, how to save and deploy the trained model, and how to adapt the model for inference, instead of how to conduct the trial.

The variance also exists between different open-source AutoML frameworks. In `nni`, developers often have challenges

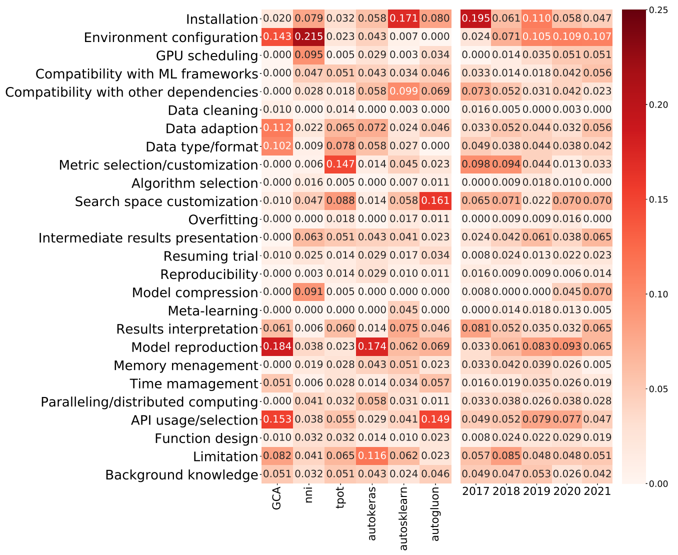


Fig. 6. The distribution of challenges among different AutoML frameworks and years

in *Environment configuration* (21.5%) due to its complex architecture, and almost all *Model compression* posts are related to `nni` because only `nni` implements model compression algorithms. Meanwhile, developers often encounter challenges of *Metric selection/customization* (14.7%) and *Search space customization* (8.8%) in `tpot`; *Model reproduction* (17.4%) and *Limitation* (11.6%) in `autokeras`; *installation* (17.1%), *Compatibility with other dependencies* (9.9%), and *Results interpretation* (7.5%) in `autosklearn`; *Search space customization* (16.1%) and *API usage/selection* (14.9%) in `autogluon`. While, some categories are widespread across open-source AutoML frameworks, including *Search space customization*, *Model reproduction*, and *Data adaption*, where best practices and supporting tools are missing.

Findings 5: The distribution of challenges varies from commercial frameworks to open-source frameworks, while widespread categories across all frameworks exist, including *Search space customization*, *Model reproduction*, and *Data adaption*.

3) Evolution of AutoML challenges: Figure 6 also illustrates the evolution of AutoML challenges, showing the distribution of challenges reported per year. Categories like *Installation* and *Compatibility with other dependencies* have a downward trend due to the increasing maturity of the AutoML frameworks. In contrast, some categories show an increasing trend, including *Environment configuration* due to the increasing architecture complexity of AutoML frameworks, *GPU scheduling* due to the more frequent involvement of GPU in AutoML trials, and *Model compression* because it receives increasing attention in recent years as a new topic. We can observe that developers constantly encounter challenges of *Search space customization*, *Results interpretation*, and *Model* *reproduction* over time, where more support is needed.

Findings 6: Categories like *Search space customization* and *Model reproduction* have been constantly reported by developers over time. *GPU scheduling* and *Model compression* even have an increasing trend in recent years.

VIII. Implication

Based on our results, we propose suggestions for AutoML framework selection, AutoML framework development, and potential research topics.

A. AutoML framework selection

Our results suggest that the usage of different frameworks varies in prevalence in different domains, e.g., developers tend to select `nni`, `autokeras`, and `autogluon` for CV applications. Developers can make a proper choice on AutoML framework selection based on their needs with our results, e.g., for a CV application, `nni` may be a better choice than `tpot` according to the distribution of their downstream repository topics.

Meanwhile, developers can benefit from the distribution of challenges encountered within different frameworks, which demonstrates differences in environment configuration, results analysis, and model reproduction across different frameworks. Developers may select the AutoML framework accordingly based on their demands, e.g., if developers are interested in how the specific model configuration is identified, it is more appropriate for them to select frameworks with fewer challenges on intermediate results presentation.

B. AutoML framework development

Our results suggest that AutoML frameworks are used in increasingly diverse domains. Thus, we suggest more domain-specific AutoML support in frameworks to efficiently serve developers with various purposes to meet extensive application scenarios of AutoML, e.g., providing APIs for typical usages like object detection applications that cannot be directly served by existing frameworks.

Our proposed taxonomy of challenges in AutoML development presents challenges that require additional engineering efforts. We suggest framework vendors provide more support for these challenges, especially those challenges with a higher proportion, e.g., providing clearer visualization so that developers can better access the progress and make optimal decisions for the next move. Our results also suggest that the challenges vary in different frameworks. There is no silver bullet for addressing challenges across all AutoML frameworks, and we suggest framework vendors tailor support for framework-specific frequent challenges according to our results, where the priority of challenges may vary in different frameworks.

We present the evolution of challenges developers encountered in using AutoML frameworks. We suggest framework vendors offer additional support for persistent and increasing challenges based on the evolution of challenges, e.g., providing automation on model construction in related ML frameworks, providing more AutoML-specific GPU support to accelerate AutoML, and providing more principle documentation on state-of-the-art techniques.

C. Potential research topics

Our taxonomy presents challenges that cannot be easily solved and require more theoretical support. Researchers may target such challenges, e.g., the automatic setup of AutoML trials, including search space setting, time estimation, and memory estimation, where SE researchers have made a preliminary attempt [7]. We also encourage researchers to mine software repositories for the embedding of prior experience rules to facilitate automated solutions to persistent challenges, e.g., recommend the search space based on configurations of similar projects mined from open-source platforms.

IX. Threats to validity

In this section, we discuss threats to the validity of our study.

Selection of frameworks. Our mining of downstream repositories and GitHub issues is based on the five selected open-source frameworks, which may lead to bias. To mitigate the bias on selection, we select widely-used representative AutoML frameworks, which have numerous stars on GitHub and considerable downstream repositories. They also cover all AutoML domains and implement state-of-the-art algorithms. Moreover, we introduce SO questions in the analysis of AutoML challenges to complement the commercial AutoML framework perspective.

Construction of the tag set. We mine SO questions based on the constructed tag set. We cannot guarantee we collect all AutoML-related tags in SO, which may lead to bias in our dataset. To mitigate the bias, we follow previous work [40], [41], [42], constructing the tag set with two widely adopted metrics *significance* and *relevance*, and we use the lowest thresholds in previous work [40], [41]. We collect as many tags as possible with the lowest thresholds and manually check all selected tags to ensure accuracy.

Limitations of data sources. To understand the challenges in AutoML development, we mine related SO questions and GitHub issues following previous work [40], [44], [55]. Thus, we may overlook important insights from other common sources, e.g., discussion forums, mailing lists, and technical support of commercial frameworks. Our results may miss challenges developers encountered in commercial AutoML frameworks. We plan to extend our study to comprehensive data sources and validate our results in future work.

Subjectivity of researchers. The subjectivity of researchers may lead to bias in manual labeling. To mitigate the bias, each data item is independently labeled by two authors and all conflict cases are resolved by an experienced arbitrator. Moreover, we have a high inter-rater agreement in the reliability analysis, indicating the reliability of the coding schema and procedure.

X. Conclusion

We have presented a comprehensive study on the use and challenges of AutoML. We demonstrate the increasing trend of AutoML use, but also the high difficulty and lack of expertise in AutoML. An investigation of 1,554 repositories using AutoML frameworks identify a wide range of AutoML application topics. A manual analysis of 1,437 GitHub issues and 208 SO questions identifies 26 challenges in AutoML. We also demonstrate the diversity of application scenarios and challenges between different AutoML frameworks, as well as their evolution. We also demonstrate the diversity of AutoML frameworks on usage and challenges. We believe that our findings will be valuable to both practitioners and researchers, and will facilitate the more effective utilization of AutoML.

XI. Data Availability

We make the replication package publicly available [15]. It contains 1) the coding books and labels for thematic analysis, 2) the dataset containing all retrieved downstream repositories, SO posts, and GitHub issues, and 3) the scripts to reproduce the results in the paper.

Acknowledgement

Chao Wang and Minghui Zhou are supported by the National Natural Science Foundation of China under Grant Nos. 61825201 and 62142201. Zhenpeng Chen is supported by the ERC advanced fellowship under Grant No. 741278 (EPIC: Evolutionary Program Improvement Collaborators).

References

[1] M. Bojarski, D. Del Testa, D. Dworakowski, B. Firner, B. Flepp, P. Goyal, L. D. Jackel, M. Monfort, U. Muller, J. Zhang *et al.*, "End to end learning for self-driving cars," *arXiv preprint arXiv:1604.07316*, 2016.

[2] L. Ma, C. Zhang, Y. Wang, W. Ruan, J. Wang, W. Tang, X. Ma, X. Gao, and J. Gao, "Concare: Personalized clinical feature embedding via capturing the healthcare context," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 34, no. 01, 2020, pp. 833–840.

[3] O. Abdel-Hamid, A.-r. Mohamed, H. Jiang, L. Deng, G. Penn, and D. Yu, "Convolutional neural networks for speech recognition," *IEEE/ACM Transactions on audio, speech, and language processing*, vol. 22, no. 10, pp. 1533–1545, 2014.

[4] Z. Chen, S. Shen, Z. Hu, X. Lu, Q. Mei, and X. Liu, "Emoji-powered representation learning for cross-lingual sentiment classification," in *Proceedings of The World Wide Web Conference, WWW 2019*, 2019, pp. 251–262.

[5] J. M. Zhang, M. Harman, L. Ma, and Y. Liu, "Machine learning testing: Survey, landscapes and horizons," *IEEE Transactions on Software Engineering*, vol. 48, no. 2, pp. 1–36, 2022.

[6] G. Nguyen, J. Islam, R. Pan, and H. Rajan, "Manas: Mining software repositories to assist automl," in *Proceedings of the IEEE/ACM 44th International Conference on Software Engineering, ICSE 2022*, 2022.

[7] J. P. Cambronero, J. Cito, and M. C. Rinard, "Ams: Generating automl search spaces from weak specifications," in *Proceedings of the 28th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*, 2020, pp. 763–774.

[8] Y. Gao, Y. Zhu, H. Zhang, H. Lin, and M. Yang, "Resource-guided configuration space reduction for deep learning models," in *2021 IEEE/ACM 43rd International Conference on Software Engineering (ICSE)*. IEEE, 2021, pp. 175–187.

[9] R. K. Saha, A. Ura, S. Mahajan, C. Zhu, L. Li, Y. Hu, H. Yoshida, S. Khurshid, and M. R. Prasad, "Sapientml: Synthesizing machine learning pipelines by learning from human-written solutions," in *Proceedings of the IEEE/ACM 44th International Conference on Software Engineering, ICSE 2022*, 2022.

[10] mircosoft, "nni," [EB/OL], https://github.com/microsoft/nni.

[11] awslabs, "autogluon," [EB/OL], https://github.com/awslabs/autogluon.

[12] D. Xin, E. Y. Wu, D. J.-L. Lee, N. Salehi, and A. Parameswaran, "Whither automl? understanding the role of automation in machine learning workflows," in *Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems*, 2021, pp. 1–16.

[13] https://github.com/EpistasisLab/tpot/issues/701, on Feb 2022.

[14] https://stackoverflow.com/questions/63182360, on Feb 2022.

[15] "Supplementary material," https://zenodo.org/record/7726459.

[16] Q. Yao, M. Wang, Y. Chen, W. Dai, Y.-F. Li, W.-W. Tu, Q. Yang, and Y. Yu, "Taking human out of learning applications: A survey on automated machine learning," *arXiv preprint arXiv:1810.13306*, 2018.

[17] M. Feurer, A. Klein, J. Eggensperger, Katharina Springenberg, M. Blum, and F. Hutter, "Efficient and robust automated machine learning," in *Advances in Neural Information Processing Systems 28 (2015)*, 2015, pp. 2962–2970.

[18] L. Kotthoff, C. Thornton, H. H. Hoos, F. Hutter, and K. Leyton-Brown, "Auto-weka: Automatic model selection and hyperparameter optimization in weka," in *Automated Machine Learning*. Springer, Cham, 2019, pp. 81–95.

[19] L. Li, K. Jamieson, G. DeSalvo, A. Rostamizadeh, and A. Talwalkar, "Hyperband: A novel bandit-based approach to hyperparameter optimization," *The Journal of Machine Learning Research*, vol. 18, no. 1, pp. 6765–6816, 2017.

[20] H. Jin, Q. Song, and X. Hu, "Auto-keras: An efficient neural architecture search system," in *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. ACM, 2019, pp. 1946–1956.

[21] T. T. Le, W. Fu, and J. H. Moore, "Scaling tree-based automated machine learning to biomedical big data with a feature set selector," *Bioinformatics*, vol. 36, no. 1, pp. 250–256, 2020.

[22] Q. Zhang, Z. Han, F. Yang, Y. Zhang, Z. Liu, M. Yang, and L. Zhou, "Retiarii: A deep learning exploratory-training framework," in *Proceedings of the 14th USENIX Symposium on Operating Systems Design and Implementation (OSDI 20)*, 2020, pp. 919–936.

[23] "Automl: Automatic machine learning - h2o.ai documentation," https://docs.h2o.ai/h2o/latest-stable/h2o-docs/automl.html, on Feb 2022.

[24] N. Erickson, J. Mueller, A. Shirkov, H. Zhang, P. Larroy, M. Li, and A. Smola, "Autogluon-tabular: Robust and accurate automl for structured data," *arXiv preprint arXiv:2003.06505*, 2020.

[25] "Cloud automl custom machine learning models," https://cloud.google.com/automl, on Feb 2022.

[26] "Automl - automated machine learning - amazon web services," https://aws.amazon.com/sagemaker/autopilot/, on Feb 2022.

[27] "Datarobot ai cloud - the next generation of ai," https://www.datarobot.com/, on Feb 2022.

[28] Q. Meng, D. Catchpoole, D. Skillicom, and P. J. Kennedy, "Relational autoencoder for feature extraction," in *2017 International Joint Conference on Neural Networks (IJCNN)*. IEEE, 2017, pp. 364–371.

[29] J. M. Kanter and K. Veeramachaneni, "Deep feature synthesis: Towards automating data science endeavors," in *2015 IEEE international conference on data science and advanced analytics (DSAA)*. IEEE, 2015, pp. 1–10.

[30] F. Nargesian, H. Samulowitz, U. Khurana, E. B. Khalil, and D. S. Turaga, "Learning feature engineering for classification." in *Ijcai*, 2017, pp. 2529–2535.

[31] H. Pham, M. Guan, B. Zoph, Q. Le, and J. Dean, "Efficient neural architecture search via parameters sharing," in *International conference on machine learning*. PMLR, 2018, pp. 4095–4104.

[32] H. Liu, K. Simonyan, and Y. Yang, "Darts: Differentiable architecture search," *arXiv preprint arXiv:1806.09055*, 2018.

[33] M. Tan, B. Chen, R. Pang, V. Vasudevan, M. Sandler, A. Howard, and Q. V. Le, "Mnasnet: Platform-aware neural architecture search for mobile," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019, pp. 2820–2828.

[34] B. Zoph and Q. V. Le, "Neural architecture search with reinforcement learning," *arXiv preprint arXiv:1611.01578*, 2016.

[35] A. Crisan and B. Fiore-Gartland, "Fits and starts: Enterprise use of automl and the role of humans in the loop," in *Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems*, 2021, pp. 1–15.

[36] N. Humbatova, G. Jahangirova, G. Bavota, V. Riccio, A. Stocco, and P. Tonella, "Taxonomy of real faults in deep learning systems," in *Proceedings of the 42nd International Conference on Software Engineering, ICSE 2020*, 2020, pp. 1110–1121.

[37] E. Aghajani, C. Nagy, M. Linares-Vásquez, L. Moreno, G. Bavota, M. Lanza, and D. C. Shepherd, "Software documentation: the practitioners' perspective," in *Proceedings of the 42nd International Conference on Software Engineering, ICSE 2020*, 2020, pp. 590–601.

[38] Y. Ma, C. Bogart, S. Amreen, R. Zaretzki, and A. Mockus, "World of code: an infrastructure for mining the universe of open source vcs data," in *2019 IEEE/ACM 16th International Conference on Mining Software Repositories (MSR)*. IEEE, 2019, pp. 143–154.

[39] "Stack exchange data dump : Stack exchange, inc. : Free download, borrow, and streaming : Internet archive," https://archive.org/details/stackexchange Accessed April 4, 2010.

[40] Y. Lou, Z. Chen, Y. Cao, D. Hao, and L. Zhang, "Understanding build issue resolution in practice: symptoms and fix patterns," in *Proceedings of the 28th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*, 2020, pp. 617–628.

[41] M. Bagherzadeh and R. Khatchadourian, "Going big: A large-scale study on what big data developers ask," in *Proceedings of the 2019 27th ACM joint meeting on european software engineering conference and symposium on the foundations of software engineering*, 2019, pp. 432–442.

[42] X.-L. Yang, D. Lo, X. Xia, Z.-Y. Wan, and J.-L. Sun, "What security questions do developers ask? a large-scale study of stack overflow posts," *Journal of Computer Science and Technology*, vol. 31, no. 5, pp. 910–924, 2016.

[43] H. B. Braiek, F. Khomh, and B. Adams, "The open-closed principle of modern machine learning frameworks," in *2018 IEEE/ACM 15th International Conference on Mining Software Repositories (MSR)*. IEEE, 2018, pp. 353–363.

[44] Z. Chen, H. Yao, Y. Lou, Y. Cao, Y. Liu, H. Wang, and X. Liu, "An empirical study on deployment faults of deep learning based mobile applications," in *2021 IEEE/ACM 43rd International Conference on Software Engineering (ICSE)*. IEEE, 2021, pp. 674–685.

[45] A. Di Franco, H. Guo, and C. Rubio-González, "A comprehensive study of real-world numerical bug characteristics," in *2017 32nd IEEE/ACM International Conference on Automated Software Engineering (ASE)*. IEEE, 2017, pp. 509–519.

[46] J. Han, E. Shihab, Z. Wan, S. Deng, and X. Xia, "What do programmers discuss about deep learning frameworks," *Empirical Software Engineering*, vol. 25, no. 4, pp. 2694–2747, 2020.

[47] H. Li, F. Khomh, M. Openja *et al.*, "Understanding quantum software engineering challenges an empirical study on stack exchange forums and github issues," in *2021 IEEE International Conference on Software Maintenance and Evolution (ICSME)*. IEEE, 2021, pp. 343–354.

[48] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[49] F. Chollet *et al.* (2015) Keras. [Online]. Available: https://github.com/fchollet/keras

[50] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Kopf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala, "Pytorch: An imperative style, high-performance deep learning library," in *Advances in Neural Information Processing Systems 32*. Curran Associates, Inc., 2019, pp. 8024–8035. [Online]. Available: http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf

[51] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard *et al.*, "Tensorflow: A system for large-scale machine learning," in *Proceedings of the 12th USENIX Symposium on Operating Systems Design and Implementation (OSDI 16)*, 2016, pp. 265–283.

[52] "Github rest api," https://docs.github.com/en/rest, on Feb 2022.

[53] "World of code(woc)," https://worldofcode.org/, on Feb 2022.

[54] X. Tan, K. Gao, M. Zhou, and L. Zhang, "An exploratory study of deep learning supply chain," in *2022 44th International Conference on Software Engineering (ICSE '22)*. ACM, 2022.

[55] Z. Chen, Y. Cao, Y. Liu, H. Wang, T. Xie, and X. Liu, "A comprehensive study on challenges in deploying deep learning based software," in *Proceedings of the 28th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*, 2020, pp. 750–762.

[56] M. Alshangiti, H. Sapkota, P. K. Murukannaiah, X. Liu, and Q. Yu, "Why is developing machine learning applications challenging? a study on stack overflow posts," in *2019 ACM/IEEE International Symposium on Empirical Software Engineering and Measurement (ESEM)*. IEEE, 2019, pp. 1–11.

[57] K. Gao, Z. Wang, A. Mockus, and M. Zhou, "On the variability of software engineering needs for deep learning: Stages, trends, and application types," *IEEE Transactions on Software Engineering*, vol. 49, no. 2, pp. 760–776, 2023.

[58] https://github.com/automl/auto-sklearn/issues/752, on Feb 2022.

[59] C. Wang, H. He, U. Pal, D. Marinov, and M. Zhou, "Suboptimal comments in java projects: From independent comment changes to commenting practices," *ACM Transactions on Software Engineering and Methodology*, 2022.

[60] K. Liu, Y. Han, J. Zhang, Z. Chen, F. Sarro, M. Harman, G. Huang, and Y. Ma, "Who judges the judge: An empirical study on online judge tests," in *ACM SIGSOFT International Symposium on Software Testing and Analysis (ISSTA) 2023*, 2023.

[61] C. B. Seaman, "Qualitative methods in empirical studies of software engineering," *IEEE Transactions on software engineering*, vol. 25, no. 4, pp. 557–572, 1999.

[62] X. He, K. Zhao, and X. Chu, "Automl: A survey of the state-of-the-art," *Knowledge-Based Systems*, vol. 212, p. 106622, 2021.

[63] S. Ahmed and M. Bagherzadeh, "What do concurrency developers ask about? a large-scale study using stack overflow," in *Proceedings of the 12th ACM/IEEE international symposium on empirical software engineering and measurement*, 2018, pp. 1–10.

[64] J. Wen, Z. Chen, Y. Liu, Y. Lou, Y. Ma, G. Huang, X. Jin, and X. Liu, "An empirical study on challenges of application development in serverless computing," in *Proceedings of the 29th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*, 2021, pp. 416–428.

[65] C. Rosen and E. Shihab, "What are mobile developers asking about? a large scale study using stack overflow," *Empirical Software Engineering*, vol. 21, no. 3, pp. 1192–1223, 2016.

[66] M. Ahasanuzzaman, M. Asaduzzaman, C. K. Roy, and K. A. Schneider, "Classifying stack overflow posts on api issues," in *2018 IEEE 25th international conference on software analysis, evolution and reengineering (SANER)*. IEEE, 2018, pp. 244–254.

[67] T. Zhang, C. Gao, L. Ma, M. Lyu, and M. Kim, "An empirical study of common challenges in developing deep learning applications," in *2019 IEEE 30th International Symposium on Software Reliability Engineering (ISSRE)*. IEEE, 2019, pp. 104–115.

[68] S. Beyer, C. Macho, M. Di Penta, and M. Pinzger, "Automatically classifying posts into question categories on stack overflow," in *2018 IEEE/ACM 26th International Conference on Program Comprehension (ICPC)*. IEEE, 2018, pp. 211–21 110.

[69] E. Aghajani, C. Nagy, O. L. Vega-Márquez, M. Linares-Vásquez, L. Moreno, G. Bavota, and M. Lanza, "Software documentation issues unveiled," in *2019 IEEE/ACM 41st International Conference on Software Engineering (ICSE)*. IEEE, 2019, pp. 1199–1210.

[70] https://github.com/EpistasisLab/tpot/issues/1102, on Feb 2022.

[71] https://github.com/microsoft/nni/issues/4105, on Feb 2022.

[72] https://github.com/automl/auto-sklearn/issues/768, on Feb 2022.

[73] https://github.com/microsoft/nni/issues/431, on Feb 2022.

[74] https://github.com/automl/auto-sklearn/issues/633, on Feb 2022.

[75] https://github.com/EpistasisLab/tpot/issues/700, on Feb 2022.

[76] https://github.com/automl/auto-sklearn/issues/1167, on Feb 2022.

[77] https://github.com/EpistasisLab/tpot/issues/301, on Feb 2022.

[78] https://github.com/EpistasisLab/tpot/issues/557, on Feb 2022.

[79] https://github.com/microsoft/nni/issues/4342, on Feb 2022.

[80] https://github.com/microsoft/nni/issues/3149, on Feb 2022.

[81] https://github.com/automl/auto-sklearn/issues/1064, on Feb 2022.

[82] https://github.com/EpistasisLab/tpot/issues/652, on Feb 2022.

[83] https://github.com/EpistasisLab/tpot/issues/736, on Feb 2022.

[84] https://github.com/microsoft/nni/issues/2130, on Feb 2022.
[85] https://github.com/microsoft/nni/issues/4336, on Feb 2022.
[86] https://github.com/microsoft/nni/issues/4230, on Feb 2022.
[87] https://github.com/automl/auto-sklearn/issues/606, on Feb 2022.
[88] https://github.com/automl/auto-sklearn/issues/607, on Feb 2022.
[89] https://github.com/automl/auto-sklearn/issues/587, on Feb 2022.
[90] https://github.com/EpistasisLab/tpot/issues/698, on Feb 2022.
[91] https://github.com/EpistasisLab/tpot/issues/360, on Feb 2022.
[92] https://github.com/automl/auto-sklearn/issues/268, on Feb 2022.
[93] https://github.com/automl/auto-sklearn/issues/1255, on Feb 2022.
[94] https://github.com/EpistasisLab/tpot/issues/459, on Feb 2022.
[95] https://github.com/automl/auto-sklearn/issues/184, on Feb 2022.
[96] https://github.com/EpistasisLab/tpot/issues/516, on Feb 2022.
[97] https://github.com/automl/auto-sklearn/issues/750, on Feb 2022.
[98] https://stackoverflow.com/questions/48064517/
auto-machine-learning-python-equivalent-code, on Feb 2022.
[99] https://github.com/automl/auto-sklearn/issues/361, on Feb 2022.
[100] https://github.com/automl/auto-sklearn/issues/520, on Feb 2022.
[101] https://github.com/automl/auto-sklearn/issues/674, on Feb 2022.
[102] https://github.com/automl/auto-sklearn/issues/520, on Feb 2022.
[103] https://github.com/EpistasisLab/tpot/issues/346, on Feb 2022.
[104] https://github.com/automl/auto-sklearn/issues/57, on Feb 2022.
[105] https://github.com/EpistasisLab/tpot/issues/1008, on Feb 2022.
[106] https://github.com/awslabs/autogluon/issues/656, on Feb 2022.
[107] https://github.com/microsoft/nni/issues/2820, on Feb 2022.
[108] https://github.com/microsoft/nni/issues/1371, on Feb 2022.
[109] https://github.com/EpistasisLab/tpot/issues/743, on Feb 2022.
[110] https://github.com/automl/auto-sklearn/issues/352, on Feb 2022.