ATMSeer: Increasing Transparency and Controllability in Automated Machine Learning

Qianwen Wang
Hong Kong University of Science and Technology
qwangbb@connect.ust.hk

Yao Ming
Hong Kong University of Science and Technology
yao.ming@connect.ust.hk

Zhihua Jin
Zhejiang University
jnzhihuoo1@gmail.com

Qiaomu Shen
Hong Kong University of Science and Technology
qshen@connect.ust.hk

Dongyu Liu
Hong Kong University of Science and Technology
dliuae@connect.ust.hk

Micah J. Smith
MIT
micahs@mit.edu

Kalyan Veeramachaneni
MIT
kalyanv@mit.edu

Huamin Qu
Hong Kong University of Science and Technology
huamin@cse.ust.hk

ABSTRACT

To relieve the pain of manually selecting machine learning algorithms and tuning hyperparameters, automated machine learning (AutoML) methods have been developed to automatically search for good models. Due to the huge model search space, it is impossible to try all models. Users tend to distrust automatic results and increase the search budget as much as they can, thereby undermining the efficiency of AutoML. To address these issues, we design and implement ATMSeer, an interactive visualization tool that supports users in refining the search space of AutoML and analyzing the results. To guide the design of ATMSeer, we derive a workflow of using AutoML based on interviews with machine learning experts. A multi-granularity visualization is proposed to enable users to monitor the AutoML process, analyze the searched models, and refine the search space in real time. We demonstrate the utility and usability of ATMSeer through two case studies, expert interviews, and a user study with 13 end users.

CCS CONCEPTS

• **Human-centered computing** → **Visualization systems and tools**; **Information visualization**;

KEYWORDS

Automated Machine Learning, Data Visualization

CHI 2019, May 4–9, 2019, Glasgow, Scotland UK

ACM ISBN 978-1-4503-5970-2/19/05…$15.00
https://doi.org/10.1145/3290605.3300911

ACM Reference Format:
Qianwen Wang, Yao Ming, Zhihua Jin, Qiaomu Shen, Dongyu Liu, Micah J. Smith, Kalyan Veeramachaneni, and Huamin Qu. 2019. ATMSeer: Increasing Transparency and Controllability in Automated Machine Learning . In *CHI Conference on Human Factors in Computing Systems Proceedings (CHI 2019), May 4–9, 2019, Glasgow, Scotland UK*. ACM, New York, NY, USA, 12 pages. https://doi.org/10.1145/3290605.3300911

1 INTRODUCTION

To ease the difficulty of developing machine learning (ML) models, automated machine learning (AutoML) methods have been proposed [12, 27, 39]. Instead of searching algorithms and tuning hyperparameters manually, AutoML automatically iterates through various machine learning algorithms and optimizes hyperparameters in a predefined search space (*i.e.*, a set of feasible machine learning models). AutoML has received considerable research attention and gained widespread popularity. A plethora of systems for AutoML, such as ATM [33], SigOpt [30], and Google Cloud AutoML [10] have been developed in recent years.

Unfortunately, these AutoML systems usually work as black boxes. Due to the lack of transparency in AutoML (*e.g.*, *what models have been searched?*), users tend to question the automatic results. *Did the AutoML sufficiently explore the search space? Did the AutoML run long enough? Did the AutoML miss some suitable models?* Concerns like these may make users reluctant to apply the results of AutoML in critical applications [17], such as disease diagnosis and stock market prediction. Meanwhile, when AutoML returns unsatisfying results, users are unable to reason and thus improve the results. They can only increase the computational budget (*e.g.*, running time) as much as possible, which undermines the efficiency of AutoML.

These issues can be alleviated by involving end users in AutoML, enabling them to reason the AutoML results and to modify the AutoML configurations. However, two challenges need to be addressed. First, it can be difficult for users

to analyze AutoML results. An AutoML process generates a series of (usually a few hundred) models selected based on a specific search strategy. These models have different algorithms, hyperparameter configurations, and performance scores. It is non-trivial to organize and present this data in an intuitive way so that users can easily understand and analyze it. Second, it can be challenging for users to modify the search space of an AutoML process. AutoML can return unsatisfying models due to various reasons, such as insufficient budget, large search space, and limitations of AutoML algorithms [24, 27, 33]. At the same time, the search space usually has a complicated hierarchical structure. Effective interactions are required to help users modify an AutoML process by combining their observation of the current process with their prior knowledge.

In this paper, we present ATMSeer [1], an interactive visualization tool that helps users analyze the searched models and refine the search space. Instead of opening the black box of AutoML and explaining the search decisions, ATMSeer offers a visual summary of the searched models to increase the **transparency** of AutoML. Users are allowed to explore the models searched by an AutoML process at three levels of detail (*i.e.*, the algorithm level, the hyperpartition level, and the hyperparameter level) based on the breadth (*e.g.*, has it searched all machine learning algorithms) and the depth (*e.g.*, has it extensively searched algorithms that can lead to good performance) . Meanwhile, ATMSeer enables users to interactively modify the search space in real time to increase the **controllability** of AutoML. Through the visual summary of the searched models from three levels, users are able to understand the behavior of different models, which helps them propose alternative models and modify the search space. An in-situ search space configuration is embedded in the three-level visualization to facilitate the switch between analysis of the results and modification of the search space.

In this work, we integrate ATMSeer with the ATM AutoML framework proposed by Swearingen et al. [33]. However, ATMSeer is not algorithm specific and can integrate with a variety of AutoML frameworks.

The main contributions of this paper are as follows:

- A summary of the workflow for using AutoML tools and the requirements for analyzing an automated model search process.
- An interactive visualization tool that enables users to monitor, analyze, and refine an AutoML process.
- An evaluation of ATMSeer through two case studies, interviews with two AutoML experts, and a user study with 13 end users.

[1] https://github.com/HDI-Project/ATMSeer

2 RELATED WORK

Choosing Machine Learning Models

There is no one machine learning model that works the best for every problem [6, 35]. To achieve high performance for a particular problem, users typically choose models based on their understanding of the algorithms, their observation of the data, and a time-consuming trial-and-error process.

Many efforts have been made to provide guidance for choosing models. On the one hand, some research provides theoretical guidance by summarizing the pros and cons of different machine learning algorithms [3, 11]. For example, Kotsiantis et al. [11] conclude that support vector machines have a high tolerance for irrelevant features but require a large sample size. On the other hand, experiments on a large number of datasets also provide empirical guidance for choosing models [1, 4, 6, 7]. For example, by evaluating 179 classifiers on 121 datasets, Fernández-Delgado et al. [6] find that random forests are most likely to be a good classifier, followed by support vector machines and neural networks.

While these work provides useful guidance, they fail to provide detailed instruction for a particular problem (*e.g.*, the exact model for a dataset). ATMSeer aims to provide guidance to solve particular problems. Given a dataset, ATMSeer automatically tries different models and allow users to easily observe and analyze these models through an interactive visualization.

Visualizing Automated Machine Learning

Visualization has long been used to facilitate human interaction in the model tuning process [19, 20]. Recently, efforts have been taken to visualize automated machine learning.

For example, MLJar [22] enables users to easily define a search space and analyze searched models with no coding required. Google Vizer [9] provides parallel coordinates to support the analysis of searched models. For one algorithm, users can observe the range of each hyperparameters, the correlation between hyperparameters, and the relationship between performance and hyperparameters. SigOpt [30] provides an interface that enables users to join in the optimization loop of a model. Users repeatedly observe suggested hyperparameter values, experiment with these values with their own model, analyze the experiment results, and finally report results back to SigOpt.

However, these works only support the analysis of one type of model (*e.g.*, neural networks) at a time. In contrast, ATMSeer supports the analysis of machine learning models generated with various algorithms (14 machine learning algorithms are supported in ATMSeer). Moreover, we provide a multi-granularity visualization of searched models to facilitate the analysis of the AutoML process.

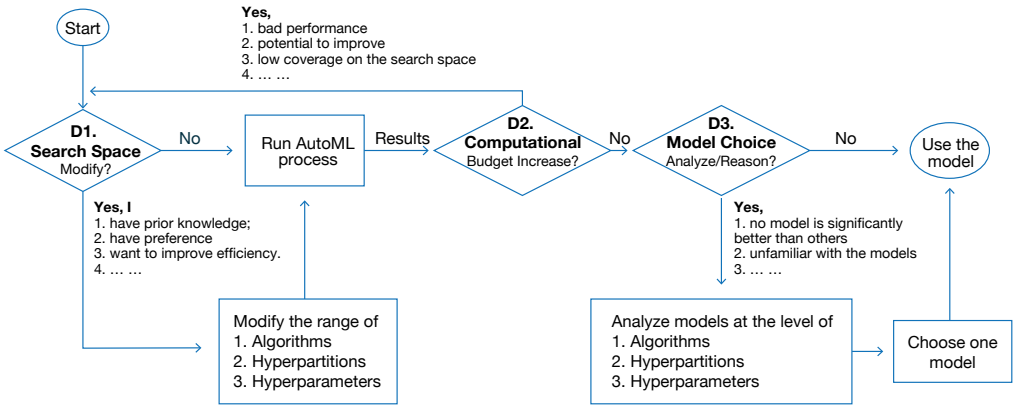

Figure 1: A workflow of using AutoML.

Visualizing Machine Learning Models

In recent years, there is a trend for combining visualization with machine learning to help people understand, diagnose, improve, and apply machine learning models.

Various visual analytics tools have been developed for opening the black box of different machine learning models, including generative models [14, 34], reinforcement learning [36], convolutional neural networks [13, 15, 37], and recurrent neural networks [21, 32]. These tools provide guidance for model developers to understand, diagnose, and refine machine learning models. However, in these tools, the requirements of model users are not thoughtfully considered.

To assist in applying machine learning models, some visual analytics tools analyze model behavior on the data instance level without opening up the algorithm black box [2, 28, 38]. For example, Squares [28] reveals the model mistakes at the instance level and connects summary statistics (*e.g.*, accuracy) with individual data instances, thereby helping practitioners analyze model performance. However, these tools focus on performance analysis and cannot be directly applied to AutoML, in which the configurations (*i.e.*, algorihtm, hyperpartition, hyperparameter) of many searched models need to be analyzed.

3 SYSTEM REQUIREMENTS AND DESIGN

Goals & Target Users

The main goal of ATMSeer is to help people efficiently search, analyze, and choose machine learning models for their own tasks. The target users of ATMSeer have a certain level of expertise in machine learning, but previously suffered from a time-consuming and error-prone manual search when developing machine learning models.

Data Abstraction

An AutoML process can be regarded as training a sequence of models on a given dataset. At each step, given the performance of previous models, the AutoML algorithm selects a new model to train and evaluate. Each model in an AutoML process can be treated as a multivariate data point with four types of attributes: algorithm (categorical variable), hyperpartition (set of categorical variables), hyperparameter (set of numerical variables), and performance (numerical variable).

User Interview

We conducted semi-structured interviews with six participants to understand how they choose machine learning models and what opportunities exist to improve the experience. We recruited participants through reaching out to personal contacts. Three participants were from diverse backgrounds (*i.e.*, biology, urban planning, finance) with experience in developing machine learning models for their domain problems and three participants were machine learning experts.

The interview consisted of three parts and lasted approximately 45 minutes for each participant. First, the participants were asked to describe their experience in developing machine learning models. Second, we introduced and discussed AutoML with them, and asked for their expectations of and concerns about AutoML. Third, the participants were asked to use a pilot system to solve a classification problem and comment on their experiences. Three participants used their own data and the other three used example data provided by us. Details of the pilot system are provided in the supplementary material.

The Workflow

Based on the interview, we identify three factors that the participants most care about: search space (*e.g.*, *"How many algorithms will be searched?"*), computational budget (*e.g.*, *"How long will the process run?"*), and model choice (*e.g.*, *"Which model is the best among the searched models?"*). The three factors correspond to the three key decisions (D1–D3) during the use of AutoML and demonstrate the necessity of human involvement. We connect the three decisions according to how the participants use the pilot system and their current practice of developing models, thereby summarizing a workflow of using AutoML, as shown in Figure 1.

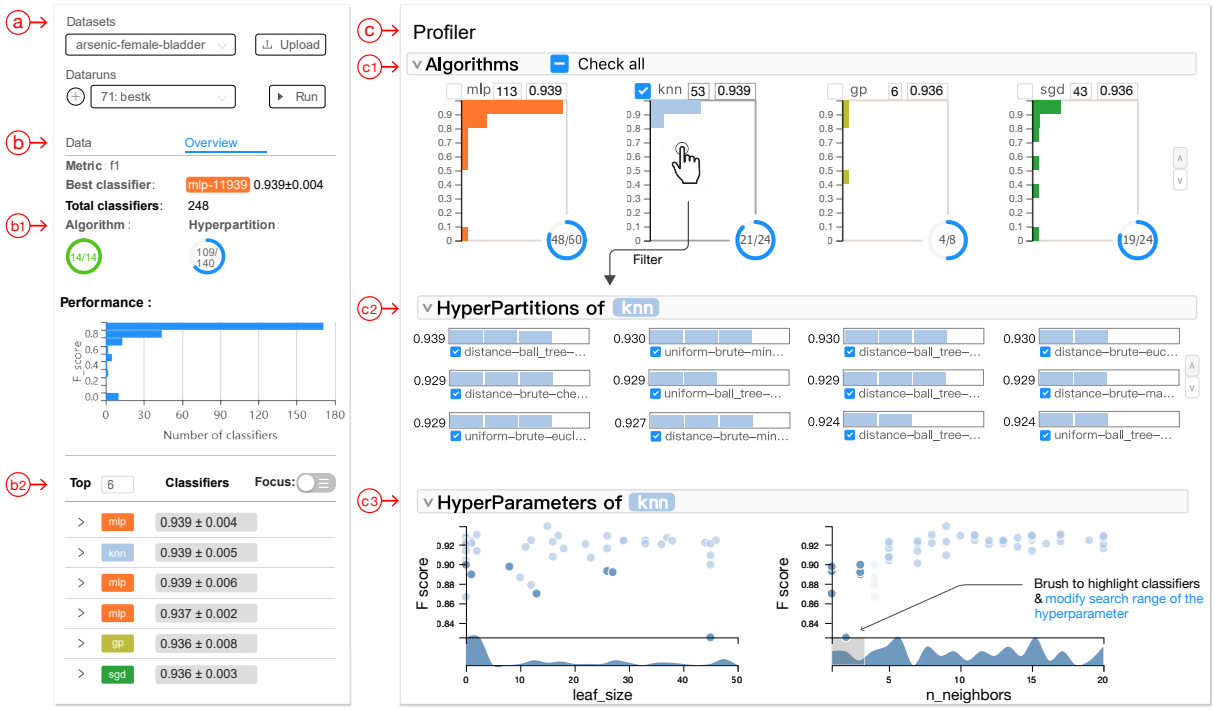

Figure 2: The Interface of ATMSeer. The user creates/resume AutoML process using the control panel (a), observe the high-level statistics of an AutoML process in the overview panel (b), and analyze the process in different granularities with the AutoML profiler (c).

D1. Modify Search Space. To incorporate human knowledge and improve the search efficiency, AutoML systems usually allow users to configure settings [7, 23, 33]. Participants stated that they modified the search space based on their prior knowledge (*e.g.*, *"the k-nearest neighbors algorithm usually has a good performance on my protein structure dataset and I want to try this algorithm first"*) or their observations of the ongoing search (*e.g.*, *"the random forest algorithm is performing well and is more stable than other algorithms"*).

D2. Adjust Computational Budget. AutoML tries to find a suitable model by searching through a large set of available machine learning models with limited computational budget (*e.g.*, running time). There exists a trade-off between the model performance and the computational budget of AutoML. The participants decided whether to continue an AutoML process based on the performance of the searched models, the potential for subsequent performance improvement, and their acceptable expenses for the AutoML services.

D3. Reason Model Choice. By default, AutoML returns the model with the best performance score. However, instead of directly using the model with the highest performance score, participants expressed the need to reason the model choice according to the search space (*e.g.*, *"maybe a good algorithm/hyperparameter hasn't been searched"*) or some domain-specific requirements (*e.g.*, *"I prefer models that are robust to the change of hyperparameters"*).

Design Requirements

We then distilled the following design requirements to assist in making decisions (D1–D3).

R1. Offer an overview of the AutoML process. An overview of all searched models can help users learn basic information about the process, such as the number of searched models and how the best performance changes over time [D2].

R2. Connect models with the search space. Users should be able to analyze models in the context of the search space. This enables users to reasone the model choice [D3] and to modify the search space [D1].

R3. Offer guidance for modification. Guidance should be provided to assist users in modifying the search space [D1].

R4. Allow in-situ search space configuration. The configuration of search space is usually complex and difficult to memorize. Users should be allowed to switch seamlessly between the observation of the current process and the modification of the search space [D1].

R5. Support multi-granularity analysis. The search space usually has a hierarchical structure (*i.e.*, algorithms, hyperpartitions, hyperparameters). A multi-granularity analysis of searched models should be supported to help users monitor and analyze the searched models [D3].

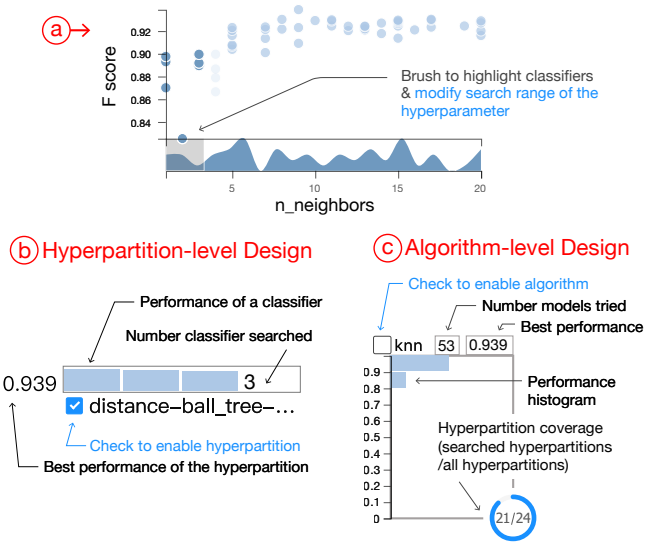

Figure 3: Detail of ATMSeer interface elements in the (a) hyperparameter-level, (b) hyperpartition-level, and (c) algorithm-level views.

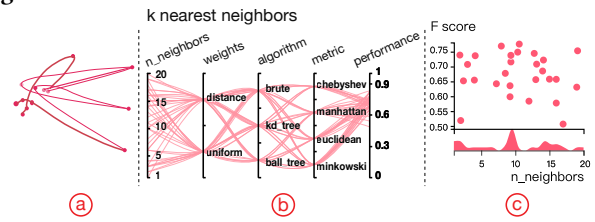

Figure 4: Alternative designs of the hyperparameter-view: a PCA mapping (a), parallel coordinates (b), and scatter plots (c).

4 ATMSEER

This section describes the design and implementation choices of ATMSeer, an interactive visualization tool that enables users to refine the search space of AutoML and analyze automatically-generated results.

System Overview

ATMSeer is implemented as a client-server application. The server coordinates three components: AutoML, data storage and model storage. The server provides the client with various APIs to create, configure, start, and pause an AutoML process, and to summarize the recorded data in AutoML processes. As the client, the visual interface provides graphical controls for AutoML processes and maps the summary data to visualization.

Interface

The interface of ATMSeer consists of three parts (Figure 2):

- the control panel (a), which allows users to upload a new dataset or select an existing dataset and create or resume an AutoML process.
- the overview panel (b), which shows high-level statistics of the dataset and the AutoML process.
- the three-level AutoML Profiler (c) for analyzing the AutoML process at different granularities.

AutoML Overview. The overview panel (Figure 2(b)) summarizes high-level information (**R1**) of an AutoML process in two aspects: general summary (b1) and top models (b2). In addition to the best performance score and the total number of models, two coverage metrics show the percentage of searched algorithms and hyperpartitions. Next, the performance distribution summarizes the performance of all tried models in a histogram. The top k models are listed for the users to compare and choose. Users can focus their analysis on the top models by enabling the *"focus mode"*, which highlights the corresponding algorithms and hyperpartitions in the detail views.

AutoML Profiler. After an AutoML process exhausts its budget, the user must carefully decide whether to resume the process with an increased budget (**D2**) and/or a refined search space (**D1**). It can also be challenging for users to decide which model to choose and if the chosen model is really the best (**D3**). How well does each type of algorithm perform? How many models have been tried for each algorithm? Can AutoML find better models with an increased budget and better configuration? Are some highly-ranked models likely to achieve better generalization performance than others? The AutoML Profiler summarizes an AutoML process at three levels of granularity to help users answer these questions. From macro to micro, these include algorithm-level, hyperpartition-level, and hyperparameter-level. The latter two levels, which are only shown on demand, are designed for advanced users who need to evaluate or configure an AutoML process at a finer granularity.

Algorithm-level View (Figure 2(c1)) visualizes the performance distribution of each machine learning algorithm as a histogram. As shown in (Figure 3(c)), important statistics of the algorithms, such as best performance and hyperpartition coverage, are displayed with each histogram. The algorithms are sorted according to their best performance in descending order. The algorithm-level view enables users to compare different algorithms (**R5**) with respect to performance distribution and the number of tried models. Users can evaluate the robustness and performance of each algorithm and gain intuition for modification (**R3**).

Hyperpartition-level View (Figure 2(c2)) summarizes different hyperpartitions of selected algorithms. A hyperpartition is a configuration of an algorithm with fixed non-tunable hyperparameters (only numeric hyperparameters can be tuned). Different hyperpartitions of an algorithm can have very different properties (*e.g.*, SVM with linear kernel vs. polynomial kernel). The hyperpartition-level view is designed to help advanced users to analyze the search space

(**R2**) and compare hyperpartitions (**R5**). For a selected algorithm, its hyperpartitions are visualized as a list of progress bars. Models are pushed into their corresponding progress bars according to their hyperpartitions as small boxes, whose height denotes the performance of the model (Figure 3(b)).

Hyperparameter-level View (Figure 2(c3)) shows the relation between performance and hyperparameters of a selected algorithm. For each tunable hyperparameter, a scatter plot is presented to compare hyperparameter values and performance scores. Each model is visualized as a point in each of the scatter plots. We also include an area plot showing the distribution of the hyperparameter below each scatter plot to help users evaluate the coverage of the hyperparameter space (*i.e.*, which values have been extensively tried and which have not) (**R2**). It also helps users learn how each hyperparameter influences the performance: at what values a model gets a generally good performance and where it does not. This information can be used as important hints for improving the configuration of the search space (**R3**).

During the development of ATMSeer, we experimented with two common multivariate visualization techniques as design alternatives (Figure 4): principal component analysis (PCA) mapping and parallel coordinates. We gathered preliminary feedback from three target users that we interviewed (Section 3). The PCA mapping was rejected by all users, since it loses the detail of hyperparameter values, which are important for analyzing and comparing models. Parallel coordinates visualize each hyperparameter in the high-dimensional coordinate space as a vertical line and were well-accepted by the users. However, we found that users needed to perform intensive interactions with the parallel coordinates during the analysis. We also noticed that most users are only interested in investigating the relationship between performance and a single hyperparameter at a time, possibly resulting from the fact that high-dimensional relationships are perceptually challenging for humans. Compared with parallel coordinates, multiple scatter plots are intuitive, simple, and preferred by all users. As a result, we adopt scatter plots as the final design for the hyperparameter-level view.

Interaction Design

Real-time Control. The ATMSeer interface is updated dynamically, which allows users to monitor and analyze AutoML processes in real-time. The users can also perform a "run-reconfigure-run" workflow – they can pause and reconfigure an AutoML process and then restart it from its previous state.

In-Situ Search Space Configuration. A well-chosen search space could improve the search efficiency of AutoML. Configuring the search space also helps integrate a user's prior knowledge into the AutoML algorithm. However, configuring the search space can be a challenging task due to its complex hierarchical structure (*i.e.*, different algorithms, categorical and numeric hyperparameters). As shown in Figure 3(a), ATMSeer provides an in-situ configuration which embeds in the three-level Profiler. It allows users to easily modify the search space at the same place they observe and analyze the search models (**R4**).

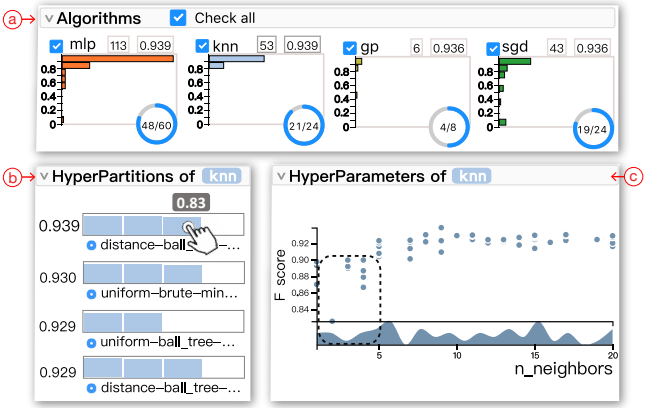

Figure 5: The use case of ATMSeer to select and understand models. (a): The top 4 algorithms have similar best performances but KNN has a more stable performance distribution. (b): There is no obvious relationship between the choice of KNN hyperpartition and the model performance. (c): A small value of `n_neighbors` leads to low performance.

5 CASE STUDY

The case studies are conducted in collaboration with two ML experts (denoted as E1 and E2) that we interviewed in Section 3. We use F1-score with 10-fold cross validation as the performance metric. The machine learning algorithms used in the case studies include support vector machine (SVM), extra-trees (ET), linear models with SGD training (SGD), *k*-nearest neighbors (KNN), random forest (RF), multi-layer perceptron (MLP), and Gaussian process (GP).

Select and analyze models (D2, D3)

In this case, we illustrate how ATMSeer helps users select and analyze the searched models. E1 wants to find a model for the `arsenic-female-bladder` dataset [25] using ATMSeer. This dataset classifies 559 female patient records as positive (bladder cancer) or negative (healthy). ATMSeer first searched 250 models for this dataset. Observing that the best performance score is 0.939 and the algorithm coverage is 100% in the overview panel (Figure 2(b2)), E1 is satisfied with the AutoML results. E1 then decides to stop the search (**D2**) and to choose a model from the already searched models.

E1 first examines the top 10 models in the leaderboard. The top 10 models have similar performance scores (from 0.936 to 0.939) and belong to four different algorithms (*i.e.*,

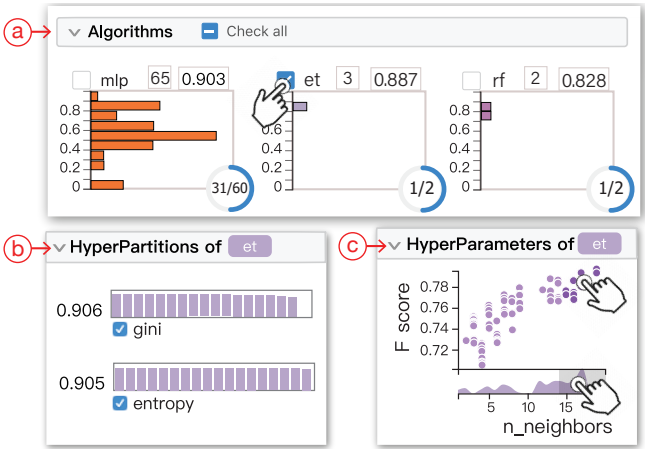

Figure 6: Using ATMSeer to modify the search space of an AutoML process. (a): Only ET algorithm is selected. (b): The best performance of the two ET hyperpartitions increases to 0.906 and 0.905. (c): The range of the `max_feature` is set to $[0.7, 1.0]$ and the best performance increases to 0.922.

MLP, KNN, GP, SGD). Since these models have similar performance scores, E1 thinks it would be better to choose a model by comparing the characteristics of the algorithms. E1 then compares these four algorithms in the algorithm-level visualization. He finds that the performance distribution of KNN is concentrated on the top, implying that KNNs have generally good performance on this dataset. He decides that KNN should be a good choice (**D3**).

E1 also wants to learn why some KNNs have less satisfying performance (*i.e.*, F1-score < 0.9) to increase his confidence in using the model. He clicks KNN in the algorithm-level view to reveal more information in the hyperpartition-level view (Figure 5(b)). He notices that the difference between the best performance of each hyperpartition is not significant. Meanwhile, one hyperpartition can have both strongly performing models and poorly performing models (*e.g.*, a model in the first hyperpartition has a performance of 0.83). E1 then clicks the hyperparameter-level view to observe more detailed information and finds that the choice of hyperparameter directly influences the performance. As shown in Figure 5(c), most poorly-performing models have a small "number of neighbors".

Refine Search Space (D1, D2)

Next, we illustrate how ATMSeer helps users modify an AutoML process and improve the performance. E2 wants to find a model for the Friedman Dataset `fri_c3_1000_10` [8], a synthetic binary classification problem with 1000 instances and 10 features. With strong domain knowledge, E2 wants to have more control over the AutoML process. He first tries all algorithms to analyze which one is better for this dataset. After searching 150 models, the algorithm coverage

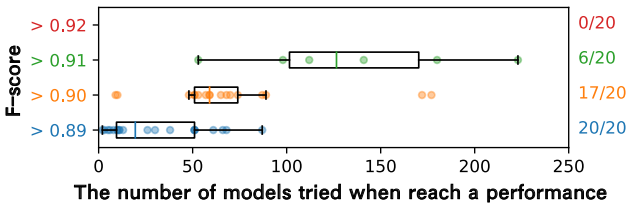

Figure 7: The performance of 20 AutoML processes. X-axis represents the number of models tried when a process first reaches a certain performances. Among 20 processes, all 20 processes achieves a best performance of over 0.89, 17 over 0.90, 6 over 0.91, and none over 0.92.

reaches 100% and he suspends the process. After observing the algorithm-level view (Figure 6(a)), E2 expresses a preference for the second best algorithm, ET. ET is tried for only three times and its performances is comparable to the best algorithm, MLP. In addition, ET has a concentrated performance distribution between 0.8 and 0.9. Thus, E2 modifies the algorithm-level configuration to focus on searching ET to see if further improvements could be achieved. After searching another 30 models, the best performance increases from 0.887 to 0.906. Opening the hyperpartition-level view (Figure 6(b)), E2 finds the performance distributions of ET-Entropy and ET-Gini are similar. This is consistent with his prior knowledge that ETs with Gini or Entropy measure have similar performance in general [26].

E2 then checks the hyperparameter-level view (Figure 6(c)) and finds that the value of `max_features`[2] directly influences the performance. Based on his observation of the hyperparameter-level view, he concludes that choosing `max_features` between 0.7 to 1 leads to higher performance, which, however, conflicts with his experience: *"The empirical good value of* `max_features` *is around* $\frac{\sqrt{n}}{n}$ *[n is the number of features] [16], which means I would set it to around* 0.3 *for this dataset."* E2 comments that *"this makes sense since empirical values are usually not optimal."* E2 then modifies the range of `max_features` to $[0.7, 1.0]$ (Figure 6(c)) and searches another 50 models. The best performance score increases to 0.922. Since no further performance improvement occurs in the last 20 searched models, E2 stops the search process and chooses the best ET model.

Interested in assessing how the involvement of humans improves an AutoML process, we run 20 AutoML processes without human interference for comparison. We let each AutoML process independently search 250 models. As shown in Figure 7, among 20 processes, 17 reach a performance score of over 0.90 and 6 reach 0.91, but none reaches 0.92. In comparison, the expert achieves the best performance of 0.922 within the 210th model, which shows that human

[2]In ET, `max_features` is a node splitting criterion given by the ratio of the size of random feature subsets to the total number of features.

involvement has the potential of improving both the performance and the efficiency of AutoML. This improvement might be caused by the fact that the human could modify the search space in a more aggressive way (*i.e.*, only choose one algorithm and reduce the range of a hyperparameter to 30% of its original range).

6 EXPERT INTERVIEW

To evaluate ATMSeer, we conduct interviews with two closely collaborating experts (E1 and E2, the same experts we collaborated with in Section 5) with particular expertise in AutoML. E1 is a co-author of an AutoML framework [33]. We collected their feedback about ATMSeer through weekly meetings over more than two months. Based the discussion, we summarize three main usage scenarios of ATMSeer.

Knowledge Distillation from AutoML

ATMSeer can help people better understand and apply machine learning algorithms. AutoML enables quick experimentation with a large number of models, whose results could provide useful knowledge to ML researchers and practitioners. As shown in Figure 8(a), ATMSeer shows that some algorithms (*e.g.*, ET, RF, KNN) tend to have a stable performance distribution while other algorithms (*e.g.*, SGD, MLP, SVM) are more prone to generating poorly-performing models. For the same algorithm, the strongly-performing hyperpartitions (Figure 8(c)) and hyperparameters (Figure 8(b)) vary from dataset to dataset. These findings can inform users of the importance of hyperparameter tuning for certain algorithms.

E2 commented that being able to match prior knowledge about machine learning to the visualizations produced by ATMSeer creates confidence in the underlying AutoML process and increases the likelihood of adopting AutoML. He also believed that ATMSeer could function as an educational tool for machine learning, which helps people better understand the behavior of unfamiliar algorithms.

Human-Machine Interaction in AutoML

Both experts appreciated the human-machine interaction introduced in ATMSeer. They believed such interaction can improve an AutoML process and enhance user experience. They commented that *"human observation and prior knowledge sometimes can be more efficient than AutoML algorithms, especially when there is a large search space and limited computational budget."* E1 said that *"users with more domain knowledge, such as myself, are usually critical of automated methods and like to be in control. I don't like getting a score back and hearing 'trust me.' "*

Diagnosis of AutoML

Even though ATMSeer is initially developed for AutoML end users, our expert interviews suggest that ATMSeer can also help AutoML developers diagnose AutoML algorithms.

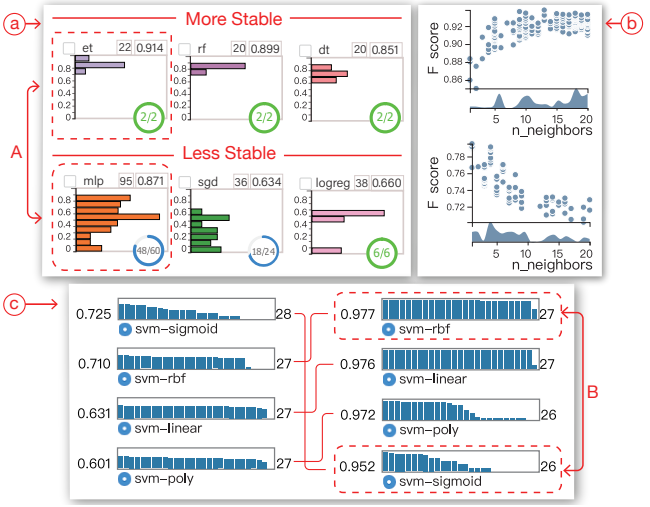

Figure 8: (a): Some algorithms have more stable performance distributions than other algorithms (using Friedman Dataset fri_c3_500_10 [8]). (b): The suitable range of n_neighbors in KNN varies from dataset to dataset (top: Quake Dataset [31]; bottom: Machine CPU Dataset from UIC[5]). (c): The suitable SVM hyperpartition for different datasets varies (top: Arsenic Female Bladder Dataset [25], bottom: Ringnorm Dataset [29]).

For example, using ATMSeer, the experts identified that the AutoML method proposed by Swearingen et al. [33] seems to be biased in favor of certain algorithms. As shown in Figure 8(a), MLP was searched many more times than ET even though their best performance was similar and the mode of the performance distribution of MLP was much lower than that of ET. E1 quickly identified the reason for this phenomenon using ATMSeer. In the ATM framework, each hyperpartition is modeled as separate arm in a multi-armed bandit framework and competes with others for the opportunity to be searched. In this case, algorithms with a small number of hyperpartitions (*e.g.*, RF, ET) tend to be searched less, motivating a hierarchical extension.

Another issue the experts identified was that the search frequencies of strongly-performing and poorly-performing hyperpartitions were sometimes similar. As shown in Figure 8(b), SVM with RBF kernel, having a higher best performance score and a more stable performance distribution, is likely better for the given dataset compared with SVM with Sigmoid kernel. However, SVM-RBF wasn't tried notably more times than SVM-Sigmoid (27 vs. 26). E2 thought this reflected an inappropriate reward function in the underlying AutoML algorithm: the search frequency of a hyperpartition (*i.e.*, the reward) was determined by the average performance of its best k models ($k = 5$ in this case). The search frequencies of SVM-RBF and SVM-Sigmoid were similar because

their best k models were similar. E2 thought this made him reconsider the design of the reward function.

7 USER STUDY

The visual analysis of AutoML processes is a relatively new problem. To the best of our knowledge, there are no similar tools for comparison. At the same time, we found that comparison with a baseline system[3] was unfair. Since the baseline system failed to provide detailed information about the AutoML results, users are unable to make informative decisions and modify the search process. Therefore, instead of an unfair benchmarking, we believe it more interesting and important to investigate user behavior under the characterized workflow with ATMSeer.

Participants and Apparatus

We recruited 13 graduate students by email (11 male and 2 female, age 22–30, $\mu = 24.46$, $\sigma = 2.66$), denoted as P1-P13. All participants had experience in machine learning or data science but none of them had prior experience with AutoML. Each participant was rewarded with a gift card worth $10. All studies were conducted with a 24-inch 1920 × 1200 monitor, a mouse, and a keyboard.

Datasets & Tasks

The participants were asked to perform two tasks that mimic the workflow described in Section 3.

- **T1:** try their best to find a model with good performance for an given dataset using AutoML.
- **T2:** analyze a given AutoML process and answer 13 questions related to **D1**-**D3**.

For **T1**, we use the German Credit Dataset [5], which classifies 1000 loan applicants as good or bad credit risks based on 20 features. For **T2**, to ensure a fair comparison across participants, we pre-run an AutoML process for 200 models on the artificially-generated Friedman `fri_c3_500_10` dataset [8]. In our 13 question survey, **Q1-Q8** are objective questions that investigate users' understanding of the three-level profiler (Table 1); **Q9-Q13** are subjective questions that investigate the information users refer to when making decisions about increasing the computational budget, modifying the search space of algorithms, modifying the search space of hyperpartitions, modifying the search space of hyperparameters, and choosing a model, respectively.

Procedure

The study began with a tutorial session, in which the tasks and the usage of ATMSeer were introduced to the participants. When performing the tasks, participants were free to ask questions and were encouraged to think-aloud. We deemed **T1** as complete when the participant was satisfied with the AutoML results and chose a model for the given dataset. In **T2**, users were allowed to skip questions that they did not know. The click activities of participants were automatically recorded. Finally, participants completed four usability questions using a 5-point Likert scale (1 for strongly disagree and 5 for strongly agree). A post-study interview was conducted to collect more detailed feedback from the participants. Each user study session lasted about 40 minutes.

[3] https://github.com/HDI-Project/ATMSeer/tree/dev-zhihua-baseline

Usability

The result of the usability questionnaire is summarized in Figure 9. Overall, most participants agreed that ATMSeer was easy to learn and easy to use. Most participants (strongly) agreed that they were confident in their selected model (84.6%) and were willing to use ATMSeer in the future (92.3%). P11 disagreed that ATMSeer was easy to use, commenting that *"I cannot remember the meaning of every hyperparameter and am not familiar with every algorithm."* P11 and P13 disagreed that they were confident in the selected model and desired additional validation using their familiar tools.

For the objective questions (Q1–Q8) in **T2**, participants answered fluently and correctly most of the time (99/104). This indicated the usability of ATMSeer in enabling users to analyze the searched models. Among the five errors/missing responses, one was a careless mistake (the participant mistook SVM as SGD). The other four came from the hyperparameter-level questions, which we found to have been caused by a low familiarity with the ML models, according to the post-study interview. For example, two participants skipped Q8 because that they were unfamiliar with SGD classifiers and were not confident about the correctness of their observations.

Revisiting the Workflow

The analysis of user behavior helps us to reflect on the workflow in Section 3.

D1: Modify Search Space. Among 13 participants, 10 follow a coarse-to-fine strategy to refine the search space. Specifically, after searching some models, participants first refined the search space at a coarse level (*e.g.*, choose 2 to 3 algorithms) based on their observation and continued the search. After searching more models, participants then refined the search space at a finer level (*e.g.*, modified the range of a hyperparameter). Participants expressed different preferences for refining the different levels of the search space and were less interested in fine modifications (Table 1).

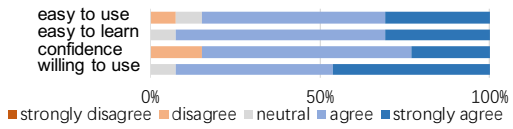

Figure 9: Results of the usability questionnaire.

	Algorithm	Hyperpartition	Hyperparameter
Clicks	**51% (13/13)**	**28% (10/13)**	**21% (7/13)**
Questions	Q1.find the best performed algorithm: **13/13**	Q4.find the algorithm with most hyperpartitions: **13/13**	Q7.find the suitable range of "n neighbors" in KNN: **11/13**
	Q2.find the most stable algorithm: **13/13**	Q5.find the worst performed hyperpartition in MLP: **12/13**	Q8.find the hyperparameter that influences the performance of SGD: **11/13**
	Q3.find the most frequently searched algorithm: **13/13**	Q6.find the best performed hyperpartition in SVM: **13/13**	

Table 1: Results on the interaction with and the understanding of three levels of information.

D2: Adjust Computational Budget. According to the interview, whether to continue the search with an increased budget is a decision made based on multiple reasons, including the *performance histogram* (considered by 9 participants), the *number of searched models* (by 5), the *algorithm coverage* (by 9), and the *best performance* (by 5).

We also found that people with strong domain knowledge (*i.e.*, self-reported as expert or advanced) were more willing to increase the computational budget so that they could further modify the search space. Six advanced participants searched on average 117 models (σ = 62.24) while seven novice participants searched 97.5 models (σ = 62.49).

D3: Reason Model Choice. While all participants preferred the models with good performance, nine participants also expressed preferences for familiar models. Three participants commented that ATMSeer helped them understand machine learning models, thereby improving their familiarity with a model and their willingness to use it.

8 DISCUSSION & FUTURE WORK

General Applicability

ATMSeer is initially designed for machine learning experts. However, based on our expert interviews and user studies, we identify other potential usage scenarios of ATMSeer, including learning machine learning models and debugging AutoML algorithms. For beginners in machine learning, ATMSeer enables them to observe how the choice of algorithms, hyperpartitions, and hyperparameters influences model performance. For AutoML designers, ATMSeer enables them to analyze the results of an AutoML process and identify possible bugs and opportunities for improvement. We will conduct further investigation in future work.

Limited Evaluation

One limitation of this work is that the user studies are only conducted with 13 participants — predominantly graduate students. Further investigation would help validate whether the insights and results can be applied more generally. Nevertheless, we are encouraged by the fact that ATMSeer was appreciated by these users and got positive feedback.

Scalability of the Visualization

Existing AutoML systems support 8–15 algorithms in general [7, 22]. A typical AutoML process would search 100 to 400 models in total [7, 22, 33]. ATMSeer uses a categorical color scheme to encode different machine learning algorithms, which most users could distinguish. The visualization can fluently support the analysis an AutoML process with over 1000 models. Thus, the effectiveness of ATMSeer is guaranteed for most real-world machine learning tasks.

Future Work

We envision improving ATMSeer in several directions. First, we plan to further validate ATMSeer in real-world applications with a larger and more diverse group of users. ATMSeer is open-sourced. We will further evaluate it and improve it based on the future feedback. Second, we intend to support intelligent control of AutoML processes. One possible direction is to combine ATMSeer with human-in-the-loop reinforcement learning [18] and automatically detect the critical points (*e.g.*, stuck in a local optimum) in an AutoML process where human involvement is needed.

9 CONCLUSION

In this work, we presented ATMSeer, an interactive visualization tool that supports machine learning experts in analyzing the automatic results and in refining the search space of AutoML. A workflow of using AutoML was proposed based on the interview with machine learning experts. Three key decisions in this workflow — updating the search space, modifying the computational budget, and reasoning the model choice — were identified to guide the design and implementation of ATMSeer. We next proposed a multi-granularity visualization with in-situ configuration to enable users to examine an AutoML process in real time at the algorithm level, hyperpartition level, and hyperparameter level. A series of evaluations demonstrated the utility and usability of ATMSeer. The user study suggested that users followed a coarse-to-fine strategy when using ATMSeer and that users with a higher level of expertise in machine learning were more willing to interact with ATMSeer.

ACKNOWLEDGMENTS

The authors would like to thank all the participants involved in the studies and the reviewers for their constructive comments and valuable suggestions. This work is supported by The Hong Kong Bank Foundation under Grant No.HKBF17RG0.

REFERENCES

[1] Nesreen Ahmed, Amir Atiya, Neamat Gayar, and Hisham El-Shishiny. 2010. An Empirical Comparison of Machine Learning Models for Time Series Forecasting. *Econometric Reviews* 29, 5-6 (2010), 594–621.

[2] Saleema Amershi, Max Chickering, Steven M Drucker, Bongshin Lee, Patrice Simard, and Jina Suh. 2015. Modeltracker: Redesigning performance analysis tools for machine learning. In *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems*. ACM, 337–346.

[3] George EP Box, J Stuart Hunter, and William Gordon Hunter. 2005. *Statistics for experimenters: design, innovation, and discovery*. Vol. 2. Wiley-Interscience New York.

[4] Rich Caruana and Alexandru Niculescu-Mizil. 2006. An empirical comparison of supervised learning algorithms. In *Proceedings of the 23rd International Conference on Machine Learning*. ACM, 161–168.

[5] Dua Dheeru and Efi Karra Taniskidou. 2017. UCI Machine Learning Repository. http://archive.ics.uci.edu/ml

[6] Manuel Fernández-Delgado, Eva Cernadas, Senén Barro, and Dinani Amorim. 2014. Do we need hundreds of classifiers to solve real world classification problems? *The Journal of Machine Learning Research* 15, 1 (2014), 3133–3181.

[7] 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*. 2962–2970.

[8] Jerome H Friedman. 2002. Stochastic gradient boosting. *Computational Statistics & Data Analysis* 38, 4 (2002), 367–378.

[9] Daniel Golovin, Benjamin Solnik, Subhodeep Moitra, Greg Kochanski, John Karro, and D. Sculley. 2017. Google Vizier: A Service for Black-Box Optimization. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD '17)*. ACM, New York, NY, USA, 1487–1495.

[10] Google. 2017. Cloud AutoML. https://cloud.google.com/automl/. Accessed: 2018-08-07.

[11] Sotiris B Kotsiantis, I Zaharakis, and P Pintelas. 2007. Supervised machine learning: A review of classification techniques. *Emerging Artificial Intelligence Applications in Computer Engineering* 160 (2007), 3–24.

[12] Lisha Li, Kevin Jamieson, Giulia DeSalvo, Afshin Rostamizadeh, and Ameet Talwalkar. 2017. Hyperband: A novel bandit-based approach to hyperparameter optimization. *The Journal of Machine Learning Research* 18, 1 (2017), 6765–6816.

[13] Dongyu Liu, Weiwei Cui, Kai Jin, Yuxiao Guo, and Huamin Qu. 2018. DeepTracker: Visualizing the Training Process of Convolutional Neural Networks. *To appear in ACM Transactions on Intelligent Systems and Technology* (2018).

[14] Mengchen Liu, Jiaxin Shi, Kelei Cao, Jun Zhu, and Shixia Liu. 2018. Analyzing the training processes of deep generative models. *IEEE Transactions on Visualization and Computer Graphics* 24, 1 (2018), 77–87.

[15] Mengchen Liu, Jiaxin Shi, Zhen Li, Chongxuan Li, Jun Zhu, and Shixia Liu. 2017. Towards better analysis of deep convolutional neural networks. *IEEE Transactions on Visualization and Computer Graphics* 23, 1 (2017), 91–100.

[16] Gilles Louppe. 2014. Understanding random forests: From theory to practice. *arXiv preprint arXiv:1407.7502* (2014).

[17] Niklas Luhmann. 2018. *Trust and power*. John Wiley & Sons.

[18] Travis Mandel, Yun-En Liu, Emma Brunskill, and Zoran Popovic. 2017. Where to Add Actions in Human-in-the-Loop Reinforcement Learning.. In *AAAI*. 2322–2328.

[19] Joe Marks, Brad Andalman, Paul A Beardsley, William Freeman, Sarah Gibson, Jessica Hodgins, Thomas Kang, Brian Mirtich, Hanspeter Pfister, Wheeler Ruml, et al. 1997. Design galleries: A general approach to setting parameters for computer graphics and animation. In *Proceedings of the 24th Annual Conference on Computer Graphics and Interactive Techniques*. ACM Press/Addison-Wesley Publishing Co., 389–400.

[20] Sean McGregor, Hailey Buckingham, Thomas G Dietterich, Rachel Houtman, Claire Montgomery, and Ronald Metoyer. 2015. Facilitating testing and debugging of Markov Decision Processes with interactive visualization. In *2015 IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC)*. IEEE, 53–61.

[21] Yao Ming, Shaozu Cao, Ruixiang Zhang, Zhen Li, Yuanzhe Chen, Yangqiu Song, and Huamin Qu. 2017. Understanding hidden memories of recurrent neural networks. *arXiv preprint arXiv:1710.10777* (2017).

[22] MLJar. 2018. MLJar. https://mljar.com/. Accessed: 2018-09-11.

[23] Randal S. Olson, Ryan J. Urbanowicz, Peter C. Andrews, Nicole A. Lavender, La Creis Kidd, and Jason H. Moore. 2016. *Applications of Evolutionary Computation: 19th European Conference, EvoApplications 2016, Porto, Portugal, March 30 – April 1, 2016, Proceedings, Part I*. Springer International Publishing, Chapter Automating Biomedical Data Science Through Tree-Based Pipeline Optimization, 123–137.

[24] Piotr Płoński. 2017. AutoML Comparison. https://medium.com/@MLJARofficial/automl-comparison-4b01229fae5e. Accessed: 2018-08-07.

[25] OpenML. 2014. arsenic-female-bladder Dataset. https://www.openml.org/d/949. Accessed: 2018-08-07.

[26] Laura Elena Raileanu and Kilian Stoffel. 2004. Theoretical comparison between the gini index and information gain criteria. *Annals of Mathematics and Artificial Intelligence* 41, 1 (2004), 77–93.

[27] Esteban Real, Sherry Moore, Andrew Selle, Saurabh Saxena, Yutaka Leon Suematsu, Jie Tan, Quoc V Le, and Alexey Kurakin. 2017. Large-Scale Evolution of Image Classifiers. In *International Conference on Machine Learning*. 2902–2911.

[28] Donghao Ren, Saleema Amershi, Bongshin Lee, Jina Suh, and Jason D Williams. 2017. Squares: Supporting interactive performance analysis for multiclass classifiers. *IEEE Transactions on Visualization and Computer Graphics* 23, 1 (2017), 61–70.

[29] Michael Revow. 1996. Ringnorm Dataset. http://www.cs.toronto.edu/~delve/data/ringnorm/desc.html. Accessed: 2018-09-07.

[30] SigOpt. 2018. SigOpt. https://sigopt.com/. Accessed: 2018-09-11.

[31] Jeffrey S Simonoff. 2012. *Smoothing methods in statistics*. Springer Science & Business Media.

[32] Hendrik Strobelt, Sebastian Gehrmann, Hanspeter Pfister, and Alexander M Rush. 2018. Lstmvis: A tool for visual analysis of hidden state dynamics in recurrent neural networks. *IEEE transactions on visualization and computer graphics* 24, 1 (2018), 667–676.

[33] Thomas Swearingen, Will Drevo, Bennett Cyphers, Alfredo Cuesta-Infante, Arun Ross, and Kalyan Veeramachaneni. 2017. ATM: A distributed, collaborative, scalable system for automated machine learning. In *2017 IEEE International Conference on Big Data, BigData 2017, Boston, MA, USA, December 11-14, 2017*. 151–162.

[34] Junpeng Wang, Liang Gou, Hao Yang, and Han-Wei Shen. 2018. GANViz: A Visual Analytics Approach to Understand the Adversarial Game. *IEEE Transactions on Visualization and Computer Graphics* 24, 6 (2018), 1905–1917.

[35] David H Wolpert. 1996. The lack of a priori distinctions between learning algorithms. *Neural Computation* 8, 7 (1996), 1341–1390.

[36] Tom Zahavy, Nir Ben-Zrihem, and Shie Mannor. 2016. Graying the black box: Understanding DQNs. In *International Conference on Machine Learning*. 1899–1908.

[37] Haipeng Zeng, Hammad Haleem, Xavier Plantaz, Nan Cao, and Huamin Qu. 2017. Cnncomparator: Comparative analytics of convolutional neural networks. *arXiv preprint arXiv:1710.05285* (2017).

[38] Jiawei Zhang, Yang Wang, Piero Molino, Lezhi Li, and David S Ebert. 2019. Manifold: A Model-Agnostic Framework for Interpretation and Diagnosis of Machine Learning Models. *IEEE Transactions on Visualization and Computer Graphics* 25, 1 (2019), 364–373.

[39] Barret Zoph and Quoc V Le. 2016. Neural architecture search with reinforcement learning. *arXiv preprint arXiv:1611.01578* (2016).