HyPockeTuner: Bringing Hyperparameter Optimization to Mobile Devices

Donghee Hong
Sungkyunkwan University
Suwon, Republic of Korea
dh.hong@skku.edu

Bongshin Lee
Yonsei University
Seoul, Republic of Korea
b.lee@yonsei.ac.kr

Jinwook Seo
Seoul National University
Seoul, Republic of Korea
jseo@snu.ac.kr

Jaemin Jo*
Sungkyunkwan University
Suwon, Republic of Korea
jmjo@skku.edu

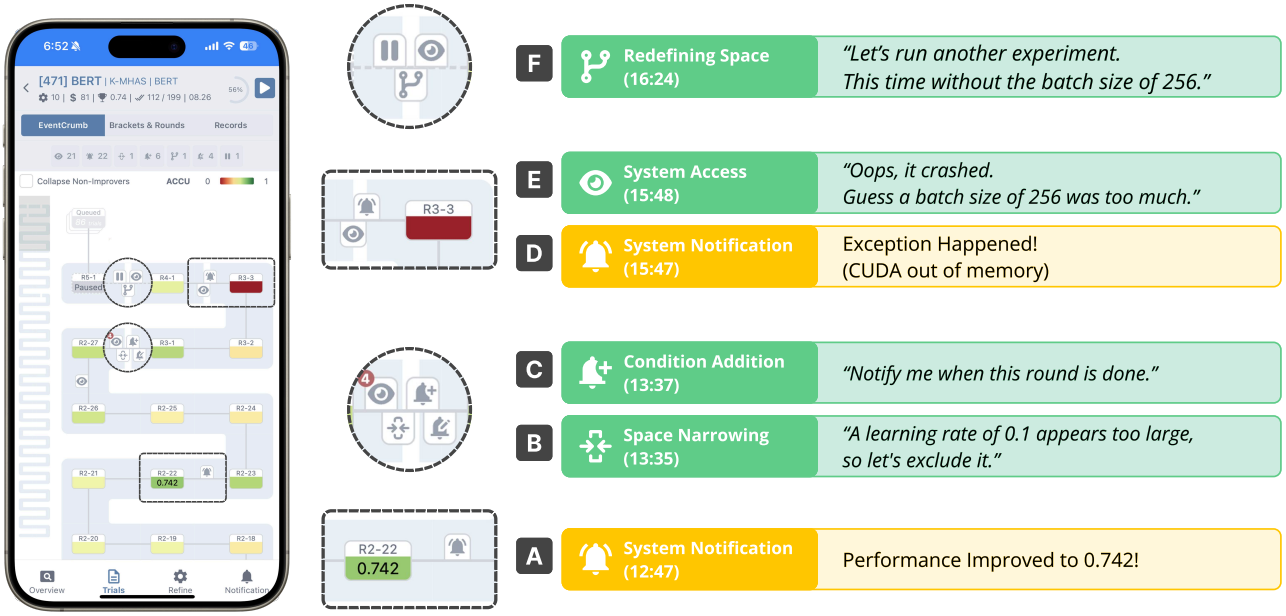

Figure 1: HyPockeTuner enables mobile hyperparameter optimization through a notification-driven workflow. (Left) The mobile interface presents an ongoing HPO experiment with the EventCrumb visualization, showing completed trials with performance metrics as nodes, bracket progression as links in shaded bands, and event bubbles indicating user interactions. (Right) A timeline illustrates a usage scenario: (A) When a record-high configuration was reached during a BOHB run, the system sent a push notification to the user. (B) The user performed a narrowing operation from their smartphone, excluding an unpromising hyperparameter value from the search space. (C) The user added a notification condition to capture when a round is completed. (D) A system notification was sent when a trial failed due to an out-of-memory exception. (E) Noticing the error, the user quickly accessed the system and discovered that the large batch size was the cause. (F) With this knowledge, the user launched a new experiment with a refined search space.

Abstract

Hyperparameter optimization (HPO) is a long-running process that can span hours or even days. While recent Human-in-the-Loop HPO systems enable monitoring and steering of the process, they are typically designed for desktop environments, which limits their effectiveness in managing prolonged experiments in practice. To address these limitations, we present HyPockeTuner, an interactive mobile system that enables users to monitor, steer, and reflect on HPO experiments anytime, anywhere from smartphones. Its mobile-tailored interface supports tracking experiment history and visualizing the relationship between user interventions and performance changes. HyPockeTuner also employs a notification workflow that alerts users to important events, reducing the burden of constant monitoring while enabling timely interventions. In a pilot study, we validated that users could readily identify critical events, such as performance improvements and intervention points, through our visualization. Furthermore, two five-day deployment studies with follow-up reflection sessions demonstrated that users could integrate experiment management into their daily routines and reflect on past decisions, generating insights for future improvement.

*Corresponding author.

CHI '26, Barcelona, Spain

ACM ISBN 979-8-4007-2278-3/26/04
https://doi.org/10.1145/3772318.3790977

CCS Concepts

• **Human-centered computing → Visualization systems and tools**; **Ubiquitous and mobile computing systems and tools**.

Keywords

Automated Machine Learning, Data Visualization, Smartphones, Hyperparameter Optimization, Mobile Data Visualization

ACM Reference Format:

Donghee Hong, Bongshin Lee, Jinwook Seo, and Jaemin Jo. 2026. HyPockeTuner: Bringing Hyperparameter Optimization to Mobile Devices. In *Proceedings of the 2026 CHI Conference on Human Factors in Computing Systems (CHI '26), April 13–17, 2026, Barcelona, Spain.* ACM, New York, NY, USA, 18 pages. https://doi.org/10.1145/3772318.3790977

1 Introduction

In machine learning (ML), hyperparameter optimization (HPO) is the problem of finding an optimal set of hyperparameters (e.g., a learning rate, the type of an optimizer) for training ML models, which are known to have a significant impact on model performance [10, 59]. To streamline such exploration in a high-dimensional hyperparameter space, a variety of automated HPO algorithms have been suggested (e.g., [26, 50]), which aim to efficiently explore the space without user intervention within a given resource and time budget. However, their autonomous and opaque behavior often prevents users from gaining insights into the optimization process and leveraging them for further improvements. As a remedy, Human-in-the-Loop HPO systems are actively researched [12, 35, 44, 55, 57, 62], enabling users to monitor the ongoing results and steer the optimization process.

In our interviews with ML practitioners, however, we found that existing Human-in-the-Loop HPO systems still face fundamental challenges in managing long-running experiments often spanning several hours to days, which become increasingly prevalent as ML models grow in scale. In such long-running experiments, users engage in a wide range of interactions, ranging from passive monitoring to active interventions, yet existing systems are primarily designed for desktop use and validated in short laboratory settings, failing to scale to the demands of flexible access across different contexts and locations. For instance, participants reported that continuous monitoring imposed cognitive burdens during focused work and physical constraints requiring desktop presence. Many expressed anxiety about leaving experiments unattended, uncertain whether failures would occur shortly after their departure. Consider a scenario that illustrates these challenges (Figure 1, right). A researcher launches an HPO experiment and leaves the office. While away, an out-of-memory exception occurs due to an excessively large batch size, but without mobile access or notifications, they remain unaware until returning the next day, wasting hours of potential computation time. Had they been notified promptly, they could have identified the cause, excluded the problematic hyperparameter value, and launched a redefined experiment from their smartphone, all without needing to return to their desktop.

To fill such a gap, we present HyPockeTuner, an interactive mobile HPO system that enables users to monitor, steer, and reflect on experiments anytime, anywhere from their smartphones. Guided by a literature review and user interviews, we identify the important types of progress and events that users should track during HPO experiments and design an event sequence visualization, EventCrumb, that presents this information effectively on smartphone screens. We also opt for a notification-driven workflow to allow users to subscribe to specific events of interest, reducing the need for constant monitoring. In a pilot study and two five-day deployment studies, we found that users could effectively access experiment information on mobile devices and integrate HPO management into their daily routines, gaining insights and identifying directions for future improvement.

The main contributions of this work are:

- HyPockeTuner, an interactive mobile HPO system with an event sequence visualization for tracking experiment history and understanding temporal progression.
- A notification-driven workflow with refinement interactions that reduces the cognitive and physical burdens of managing long-running HPO experiments, enabling seamless integration of experiment management into users' daily routines.
- Evaluation of HyPockeTuner through a pilot study with 12 participants validating mobile interface usability (Figure 2), and a five-day deployment study with two ML engineers demonstrating how mobile access and notifications change HPO management practices in real-world settings.

2 Related Work

2.1 Human-in-the-Loop Hyperparameter Optimization Systems

Despite the success of HPO algorithms, their autonomous nature raises concerns about transparency and controllability [55]. To address the concerns, several Human-in-the-Loop HPO systems have been proposed in academia, such as ATMSeer [55], HyperTendril [44], VisEvol [12], and an interactive dashboard with a paintable timeline [24]. ATMSeer [55], an interactive visualization system for model selection and hyperparameter tuning, stands out for its real-time visualization and control, allowing users to monitor the real-time progress of an HPO process and include a specific model in the middle of the process. In contrast, HyperTendril [44] and VisEvol [12] focus more on high-dimensional visual analytics on the explored hyperparameter configurations or the behavior of an optimization algorithm. In addition, Higuchi et al. [24] proposed a paintable timeline visualization technique that allows users to review tested hyperparameter values and interactively adjust search ranges. Users can drag rectangular selections directly on the timeline to define new search boundaries, and these painted regions become the refined hyperparameter space for subsequent optimization iterations.

Beyond academia, the growing adoption of HPO has led to many commercial and open-source systems, such as Amazon SageMaker [4], Google Cloud Vertex AI [20], Kubeflow [18], Microsoft Azure ML Platform [19], Microsoft Neural Network Intelligence (NNI) [41], Optuna [2], and Weights & Biases (W&B) [9]. These systems aim to support general Machine Learning Operations (MLOps) by managing and automating the development, deployment, and

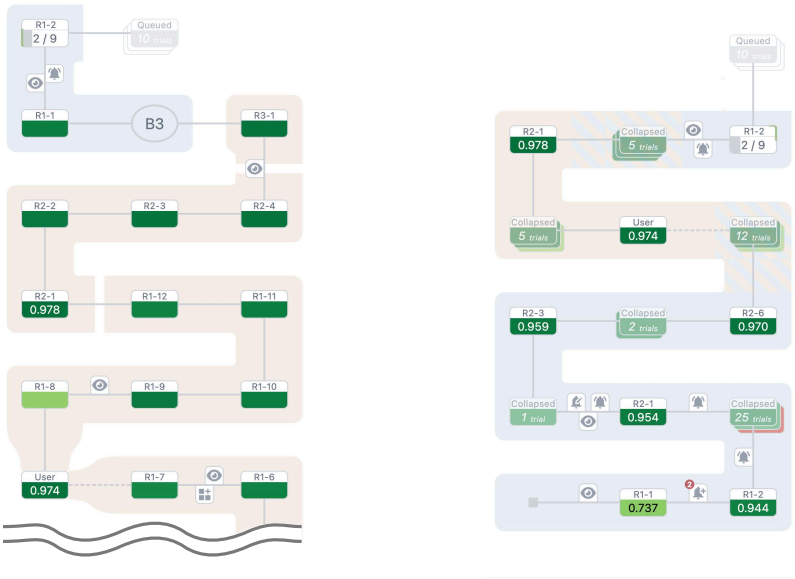

Figure 2: The EventCrumb visualization. (Left) The history of an HPO experiment visualized on a smartphone screen through EventCrumb (42 trials below the figure are omitted). (Right) EventCrumb supports collapsing the timeline, allowing the entire sequence to fit on the screen without scrolling.

maintenance of ML models. A subset of them also support notifications to facilitate monitoring tasks; for instance, Amazon SageMaker notifies users when an experiment is done through SNS or email, according to the rule defined in Amazon EventBridge. Another example is W&B, where users can write a code to specify a condition they want to monitor and then receive notifications via Slack or email when the condition is met.

However, existing systems have three major limitations in managing HPO experiments in practice. First, they lack effective visual support for understanding experiment history and user interactions. Commercial tools mainly rely on traditional visualizations such as line charts to depict performance trends or multicolumn tables to list tested configurations and their outcomes. While systems from academia, such as ATMSeer or HyperTendril, have introduced coordinated views or advanced visualizations (e.g., parallel coordinates), these remain insufficient; users still struggle to reconstruct the narrative of an experiment or to identify relationships between interventions (e.g., refining the search space) and outcomes (e.g., changes in the performance metric), insights that are essential for improving efficiency. Second, most systems are designed for desktop environments and provide little to no mobile support, constraining users to physical workspaces. For today's long-running experiments, this often forces users to lose their context when changing locations. Third, they provide limited control for HPO experiments. For example, it is challenging to modify the search space due to the lack of either algorithmic or interface support, forcing users to restart experiments from scratch using command-line tools. To tackle these challenges, we introduce HyPockeTuner, a system that enables mobile hyperparameter optimization, grounded in the practical concerns of long-running HPO experiments.

2.2 Hyperparameter Optimization Algorithms

Automated Machine Learning (AutoML) is a research area concerned with automating the development and training of ML models [23, 26]. HPO is one of the most fundamental tasks in AutoML, as every ML model has hyperparameters [26], which significantly affect the performance of the models [16]. Automated HPO algorithms aim to explore a multi-dimensional hyperparameter space within a given budget. Various HPO algorithms have been proposed [10, 50], from simple approaches, such as grid search and random search [8] to more sophisticated ones, such as evolution strategies [43, 60], Bayesian optimization [7, 30, 49, 51, 52], and multi-fidelity methods including Hyperband [33] and Asynchronous Successive Halving Algorithm (ASHA) [34]. Other approaches include Population-Based Training (PBT) [27], which jointly optimizes hyperparameters and model weights during training, and meta-learning approaches that leverage knowledge from previous optimization tasks [25].

In HPO, there is a general trade-off between the number of hyperparameter configurations to evaluate and the budget allocated in assessing each configuration. Suppose the budget (b) corresponds to the number of epochs users can run and is given as 16. One can evaluate a single random configuration by training a model for 16 epochs, which gives an accurate estimate of its performance, but the performance would be suboptimal as only one configuration is tested. In the other extreme case, one can evaluate 16 random configurations for one epoch to explore more configurations. However, the training process would be too short to evaluate each configuration sufficiently, and thus, the performance would be estimated less accurately, potentially overlooking good configurations.

Hyperband (HB) is designed to strike a balance in this trade-off based on successive halving [28]. HB organizes the optimization into multiple brackets, with each containing several rounds. After each round, only a fraction ($1/\eta$) of high-performing configurations advance to the next round. For consistency, we index both brackets and rounds starting from 1. The number of brackets is determined by the dividing factor (η) and the minimum and maximum budgets ($b_{\min}$ and $b_{\max}$), where the budget typically corresponds to the number of training epochs. For example, given $\eta = 3$, $b_{\min} = 1$, and $b_{\max} = 81$, HB produces four brackets. Bracket 1 starts with 81 random configurations evaluated at budget 1 in Round 1. HB selects the top 27 configurations for Round 2 with budget 3, then 9 for Round 3 with budget 9, continuing until one winner emerges in Round 5. Bracket 2 then begins with 27 configurations at a larger initial budget of 3, balancing broader exploration in early brackets with deeper evaluation in later ones. After all brackets are complete, the best-performing configuration across the entire run is selected.

Bayesian Optimization (BO) is used to approximate an unknown objective function given a finite set of data samples [49]. In our context, we use BO to identify the most promising configuration that should be evaluated next based on the previous evaluation results. Consisting of a surrogate model and an acquisition function, BO first randomly generates configurations since there are no tested configurations initially. Then, the configurations are evaluated by running a training process for each. Given the evaluation result, BO updates its surrogate model to approximate the performance of the configurations. Next, it recommends a configuration that maximizes the acquisition function according to the model, evaluates the recommended configuration, and refits the model. This is repeated until the predefined iteration or a stop condition is met.

BOHB [17] combines BO and HB. It is similar to HB in that it uses successive halving and follows the same schedule. The difference is that while HB randomly generates configurations in the first round of each bracket, in BOHB, this random sampling process is replaced with BO for guided search. Precisely speaking, the first bracket of BOHB is executed in the same manner as HB, but after that, it updates a surrogate model to approximate the evaluation results of the configurations tested in the first bracket. Then, the second bracket starts with the configurations sampled from BO, which would be more promising than random sampling.

We chose BOHB as the default HPO algorithm for our system for two reasons. First, its effectiveness across diverse ML tasks has been well established. BOHB has been widely tested and is available as a mature open-source implementation[1], serving as the foundation for many subsequent algorithms such as Distributed Evolutionary Hyperband (DEHB) [5] and MFES-HB [36]. Second, BOHB follows an iterative process that combines random sampling, Bayesian optimization, and successive halving in a progressive, bracketed structure, an approach that has become one of the most widely adopted in HPO. This choice not only offers interpretable checkpoints for user interactions but also provides opportunities for integrating other iterative HPO algorithms with our system.

[1]https://github.com/automl/HpBandSter

2.3 Event Sequence Visualization

Event sequence visualizations have been widely applied to analyze temporal patterns and relationships in discrete event data across various domains, including healthcare, computer security, and advertising [22]. Early approaches focused on detailed timelines for individual data items [31, 46, 56]. Later work extended to multiple sequences to improve scalability and support dataset-level overviews. For example, LifeFlow [58] pioneered tree-based aggregation for scalable overviews, and EventFlow [42] added support for interval events with graphical search. DecisionFlow [21] further scaled to thousands of event types through dynamic data structures, enabling exploratory analysis of large event datasets. More recent advances have focused on sophisticated tasks such as branching pattern extraction [38], cross-sequence comparison [61], and hierarchical aggregation [40].

In our work, we introduce EventCrumb, a mobile event sequence visualization tailored for BOHB experiments. Unlike prior systems designed for desktop environments, EventCrumb is optimized for smartphone screens, allowing users not only to review experiment history but also to control experiments from their smartphones anytime and anywhere. We further contribute a space-efficient encoding that distinguishes between algorithm-generated and user-driven events, as well as an aggregation scheme that accommodates hundreds of trials within the constraints of a mobile display.

3 Current Challenges in Managing HPO Experiments

To identify practical challenges, we conducted semi-structured interviews with eight participants (U1-8). All participants had experience with HPO processes across various domains; three participants were graduate students majoring in computer vision, natural language processing, and data science, respectively, and the remaining five were full-time engineers working in areas such as computer vision, finance, AI solutions, and marketing.

In the interviews, we asked the participants to describe their HPO workflows and the challenges they encountered, and to share their experiences with any existing HPO systems. Each interview lasted approximately 40 minutes, and we compensated each participant with 20,000 KRW (approximately 15 USD). From these interviews, we identified the following three critical challenges.

C1: Disconnected Experiment History and Context. Participants relied on a patchwork of existing tools to manage their HPO experiments, such as textual logs stored in files and visualization tools (e.g., W&B, TensorBoard). While these tools present the current status, they fall short in capturing the full experiment context, particularly the human decisions and modifications made during the process, because automated logging is separated from user interactions. U1 noted, "*After I modify the hyperparameter space, it is hard to recall which hyperparameters I modified. I had to remember the modifications separately.*" The absence of the context made it difficult to establish cause-and-effect relationships or leverage past insights to guide future decisions, as U5 explained, "*It's difficult to understand which of my changes improved the metric.*"

C2: Cognitive Burdens and Physical Constraints in Monitoring. Although HPO algorithms are designed to operate fully autonomously, participants were eager to remain involved in the

process. Their engagement ranged from passive monitoring, simply ensuring that experiments were running without errors, to more active interventions, such as terminating unpromising runs, modifying the search space when performance plateaued, or resolving unexpected failures. U6 described the issue with leaving experiments unattended: "*I launch experiments and leave them to luck when going home, not knowing if they failed 10 minutes after I left.*" However, these practices imposed both cognitive burdens and physical constraints. For example, U7 explained, "*Monitoring itself takes less than 30 seconds, but switching from two hours of focused work to check experiments completely breaks concentration.*" To cope with these burdens, participants devised various workarounds. One approach was to use smartphones for remote monitoring, reducing the need to remain in front of a desktop; U2 recalled, "*I attempted monitoring using my smartphone during lunch breaks but found W&B extremely difficult to view, because it is not mobile-friendly at all.*" Another workaround was creating custom notification systems. U8 implemented Python-based Slack webhooks, explaining, "*I made custom Slack notifications with triggers for threshold alerts. I had to implement it myself, and managing the notifications working correctly was another bothersome task.*"

C3: Limited Steerability of HPO Experiments. Participants often wanted to reflect on their intentions during an experiment but found little support for doing so. Existing tools typically impose binary choices: either wait or restart the experiment with modified settings. Although a few advanced systems, such as Optuna [2] and ATMSeer [55], support refinement interactions, they offer limited or no functionality for mobile devices. U8 noted, "*Looking at accuracy over time, if it does not reach my target, the only option I have is to terminate completely.*" As a result, participants had to watch the algorithms explore unpromising regions without the ability to intervene or redirect the search. U7 explained, "*Looking at the initial results, even though I see some hyperparameter values are obviously unpromising, I have to stop everything and start over.*" Such full resets are inefficient for HPO algorithms, as weights between parameter values must be relearned from scratch. Beyond inefficiency, the restart process itself was tedious. U2 described, "*Having to return to the terminal, stop execution with Ctrl+C, and modify hyperparameter values in the code is a cumbersome process.*"

4 The HyPockeTuner System

HyPockeTuner is designed to address the three challenges identified in Section 3. In this section, we outline the design goals that guided its development, introduce an event sequence visualization, EventCrumb, and describe HyPockeTuner's user interface.

4.1 Design Goals

DG1: Support Essential Tasks Anytime, Anywhere. Existing systems force users to switch between mobile monitoring and desktop control, fragmenting the HPO workflow (**C2**) and preventing timely interventions (**C3**). To enable users to control and interact with the HPO experiment from anywhere, at any time, we opt for smartphones as the main platform and design a user interface tailored for smartphone screens and touch-based interactions (Figure 5). Consequently, our design choices diverge from traditional desktop-based systems, where sophisticated, high-dimensional visualization components were often adopted [44]. We design HyPockeTuner for easy access and simplicity rather than the ability to conduct a complex visual analysis of the HPO experiment. To alleviate the overhead of switching desktop machines and smartphones, we also support users to perform essential tasks in HyPockeTuner, which had previously required desktop access, such as adjusting the hyperparameter space or testing a custom configuration.

DG2: Provide Visual Support for History and Context. We aim to bridge the gap between experiment history and user interventions by showing them on a unified timeline (**C1**). We chronologically integrate three types of information produced from the experiment: the progress of an HPO algorithm (bracket/round completions, trial executions), user interventions (narrowing/redefining search spaces, adding user trials and notification conditions), and their outcomes (performance metrics, triggered notifications). This unified timeline enables users to link their steering actions and subsequent performance changes. Unlike desktop systems that fragment this information across multiple views or dense tables, we design a serpentine timeline optimized for small screens that preserves the complete experimental history. By accessing this history on smartphones, users can identify which interventions improved performance, understand the impact of their decisions, and learn from past attempts.

DG3: Minimize the Burden of Constant Monitoring. Previous systems require users to constantly monitor the HPO experiment for significant changes, causing the burdens of constant attention (**C2**). We address this challenge with a notification-driven workflow and a mobile-first design that eliminates the need to remain in front of a desktop. In this workflow, users can flexibly adjust notification conditions without modifying code and receive alerts about important events, such as system failures. This approach enables users to offload continuous monitoring and intervene in the experiment only when necessary.

DG4: Leverage the Synergy of Automation and Steering. Automated HPO algorithms often prevent users from directly steering the process (**C3**), while the manual search can be biased and inefficient. In HyPockeTuner, we take a hybrid approach; while we use an HPO algorithm to automate the search process, users can still refine the process in the middle of computation or prioritize specific experiments or configurations, integrating their prior knowledge into the experiment. To realize such a balance between automated approaches and human steerability, we not only visualize the real-time progress of the HPO experiment but also improve an automated HPO algorithm, BOHB, to be progressive, allowing users to pause or resume an experiment when needed and to narrow the hyperparameter space as insights are gained during the process.

4.2 EventCrumb, An Event Sequence Visualization for BOHB Experiments

To provide visual support for tracking history and context (**DG2**) on smartphones (**DG1**), we introduce EventCrumb, an event sequence visualization for BOHB experiments (Figure 3). When designing EventCrumb, we made two design decisions to adapt to the limited space of smartphone screens. First, instead of a chronological scale, which corresponds to the actual time of events, we decided to use

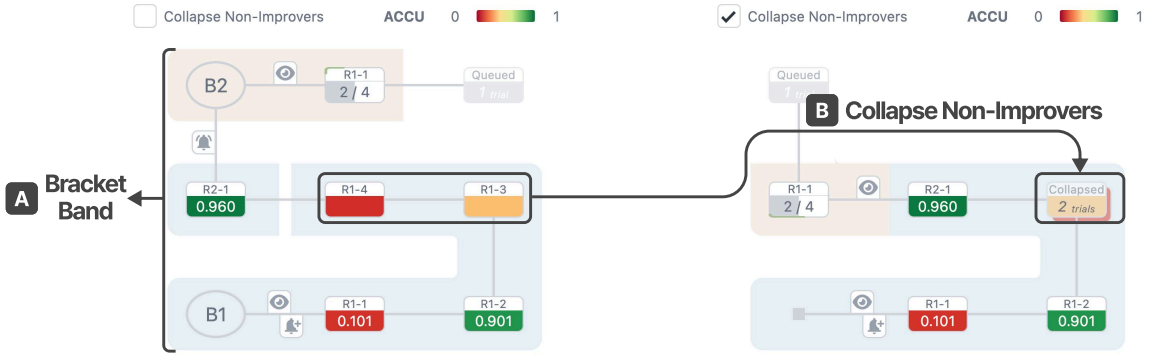

Figure 3: Visual encoding of EventCrumb to represent a BOHB experiment. (B) Non-improvers can be collapsed into a stacked representation, yielding a succinct overview of the experiment trajectory.

a sequential scale, in which the distances between events do not correspond to chronological distances [11]. This choice reflects the nature of many HyperBand-based HPO algorithms, including BOHB, where trial budgets increase exponentially; if the results were mapped directly to a linear timeline, early trials with small budgets would be compressed into a narrow region, while later trials with larger budgets would dominate the display, leading to suboptimal information density. This decision emphasizes *what happened* rather than *when exactly it happened*, while the precise timing is provided in a detailed view.

Second, while previous event sequence visualizations typically employ a straight (usually from left to right) timeline, we adopted a serpentine design in which time flows from the bottom-left corner, placing recent events at the top for immediate visibility. Unlike the straight line, which requires excessive scrolling on smartphones, the serpentine design is more space-efficient, displaying multiple events per row; we set the number of trial nodes per row to three. To help users interpret temporal order, we also provide bands and connecting links between the trial nodes.

4.2.1 Terminology. **An experiment** refers to a single run of the BOHB algorithm, initialized with the user-defined budgets (b_{min} and b_{max}), the hyperparameter search space, and a dividing factor (η). In our case, the budget directly corresponds to the number of epochs used to train a model under a given hyperparameter configuration. Users can queue multiple experiments in the Workspace view (Figure 5(A)), although only one experiment can utilize the computing resources (e.g., GPUs) at a time. BOHB organizes trials in a two-level hierarchy. Each experiment contains **brackets**, whose number is given by $s_{max} + 1$. A bracket i consists of $s_{max} - i + 2$ **rounds**, in which a pool of hyperparameter configurations is sampled and evaluated with progressively larger budgets (i.e., more epochs). After each round, only the top-performing configurations advance to the next round. We define a single execution of a configuration under a given budget as a **trial**. Among trials, we further identify a subset called **improvers**, which are those that achieve the best performance observed so far at a certain moment and are of particular interest to users.

4.2.2 Visual Encoding. Figure 3 illustrates an example of EventCrumb with two brackets and seven trials. Trials are represented as **trial nodes**, drawn as rounded rectangles whose colors encode the evaluation results (e.g., accuracy). To emphasize temporal order, we connect adjacent trial nodes with links that indicate the chronological sequence from bottom to top. Brackets are visualized as colored **bracket bands** enclosing the trial nodes (Figure 3(A)), and different rounds are separated by gaps. To make grouping more salient, we alternate the background color of brackets (e.g., blue and beige). Each bracket band begins with a **bracket node** (B1 / B2 in Figure 3), which displays the bracket ID, serving as a visual anchor for tracking algorithm progression.

Figure 4 shows different visual states of trial nodes. For completed trials, we color-encode the performance metric as the color of the node (Figure 4(A)). To support users with color-vision deficiencies, we also provide four extra accessible color palettes as alternatives to the default scheme. For improvers, we additionally display the exact performance metric in the node (Figure 4(A)-right), enabling users to access this important information without requiring additional detail-on-demand interactions. Another special type of trial is a **user trial**, which is not sampled by the BOHB algorithm but manually added by users during execution, with the highest priority to test a hypothesis. For user trials, we visually narrow the bracket band behind the trial node to indicate that the trial lies outside the regular evaluation schedule (Figure 4(C)).

We visualize events occurring between two trial nodes as **event bubbles** placed along the link that connects the corresponding nodes. Each event type is represented by an icon within the bubble, such as system access (), adding user trials (), narrowing the search space (), redefining the search space (), adding conditions (), editing conditions (), pausing experiments (), resuming experiments (), and push notifications (). Icons are arranged in relative temporal order along the link. When multiple events of the same type occur within a single link, they are aggregated into one bubble that displays a badge indicating the count, positioned according to the earliest occurrence.

4.2.3 User Interaction. In addition to the detail-on-demand interaction, which shows a detail pop-up when users tap a trial node or link, we designed three user interactions to enhance the accessibility and visibility of our visualization on smartphone screens. First, inspired by familiar code editors such as VS Code, we provide a **minimap**

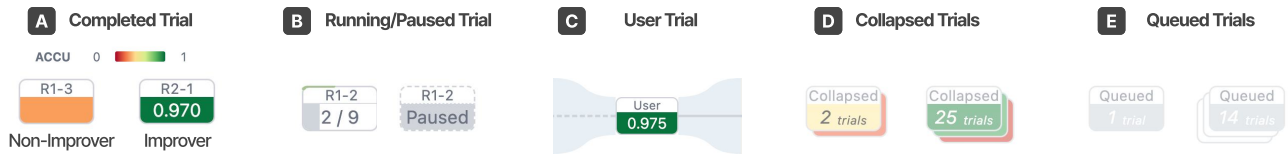

Figure 4: (A–C) Trial nodes represent the status and evaluation results of individual trials. (D) Non-improvers can be collapsed into a stack, with color encoding indicating the highest, median, and lowest performance metrics within the stack. (E) Trials waiting in the queue are also shown in a stacked form.

in the top-left corner, a compressed overview of EventCrumb along with a viewport indicator showing the current scroll position. This component is designed to support navigation on long timelines by enabling users to tap or drag within the minimap to quickly jump to a specific bracket or scroll location.

As experiments progress, the number of trial nodes can grow into the hundreds, making the layout too long to fit within smartphone screens. To address this scalability concern, we allow users to **collapse non-improvers** into a single stacked node (Figure 3(B)). A stacked node visually resembles three trial nodes, representing the highest, median, and lowest performance metrics within the group (Figure 4(D)). When collapsed, all event bubbles between non-improvers are hidden from view. The number of event bubbles can also become overwhelming, making it difficult for users to locate a specific event. To address this issue, we provide **event bubble shortcuts** displayed at the top of the visualization. These shortcuts summarize all event types that occurred during the experiment, along with their frequencies. Inspired by find functions in web browsers, users can tap an event icon to jump directly to the most recent bubble of that type, and then move forward or backward through other occurrences using the previous/next buttons.

4.3 System Overview

HyPockeTuner (Figure 5) consists of two primary views: the Workspace view and the Main view, which comprises four tabs: Overview, Trials, Refine, and Notifications.

Workspace View (Figure 5(A)). The Workspace view is designed to help the user grasp an overview of multiple BOHB experiments and create one if needed. Following the common layout of MLOps dashboards such as Microsoft NNI [41], the view lists the experiments from the newest to oldest, with each item showing the name, the number of hyperparameters being optimized, and the progress. The user can tap an experiment to load it in the Main view. Note that this experiment being viewed can differ from the one being executed at the moment; for example, the user can see the details of a completed experiment.

To create a new BOHB experiment, the user can tap the "New Experiment" button and open the Experiment dialog (Figure 5(A1)). Specifying the basic settings for the experiment, such as name, dataset, and model being used, and BOHB parameters (η, b_{min}, b_{max}), the user can configure the hyperparameter space. We support four types of hyperparameters: fixed (specific values), nominal (categorical choices with checkboxes), ordinal (discrete numerical levels with editable lists), and continuous (value ranges with min/max specification). After configuration, the user can add the experiment to a queue ("ADD") or launch it immediately, pausing any ongoing experiment ("LAUNCH IMMEDIATELY").

Overview Tab (Figure 5(B)). When the user taps an experiment in the Workspace view, they are taken to the Overview tab of the selected experiment. This tab, consisting of four sections, is designed to help the user review the experiment's configuration and status, as well as stop or resume the experiment. The Progress section displays the experiment's settings and current status, employing a progress-ring widget with a button to stop or resume the experiment. The Performance section visualizes the performance metric (accuracy by default) of completed trials over time. Improvers are highlighted with star glyphs, which are connected by lines to form a visual trajectory of optimal values. The Hyperparameter Effects section illustrates the influence of each hyperparameter on accuracy using diverging bar charts derived from SHAP values [39]. For example, a positive value indicates that the corresponding hyperparameter setting tends to increase accuracy by that amount. Finally, the GPU section presents line charts of GPU temperature and utilization over the past hour, enabling users to monitor potential hardware anomalies.

Trials Tab (Figure 5(C)). The Trials tab visualizes the experiment's trajectory from three perspectives: the default EventCrumb, Brackets & Rounds, and Records modes (Figure 5(C1)). The Brackets & Rounds mode presents the hierarchical relationship between brackets, rounds, and trials, aligning with the progression of the BOHB algorithm. Selecting a bracket expands a bump chart that depicts the algorithm's progression, where curved lines connect trials that advance from one round to the next, revealing survival patterns. The Records mode adopts a familiar grid-based layout, displaying hyperparameter configurations, color-coded performance metrics, and allocated budgets to facilitate detailed comparison. From any viewing mode, users can select a trial to inspect its details in a pop-up dialog. To support exploratory testing [2], the user may tweak the configuration of the selected trial, which is then added as a user trial and executed with top priority (Figure 5(C2)).

Refine Tab (Figure 5(D)). The Refine tab is designed to address the steerability challenge (**C3**) by enabling two types of refinement operations: **narrowing** the space continues the experiment with a subspace while preserving BOHB's internally assigned scores to each hyperparameter value, saving the time needed to relearn these weights (Figure 5(D2)); **redefining** the space launches an independent experiment with an adjusted space, allowing expansion of the hyperparameter space but requiring exploration from

Figure 5: The HyPockeTuner interface, consisting of the Workspace view (A), where users select an experiment to inspect from the list, the Overview tab (B), the Trials tab (C), the Refine tab (D), and the Notification tab (E). In the Trials tab, users can switch between three viewing modes: the EventCrumb mode, the Brackets & Rounds mode, which presents trials in the bracket–round hierarchy, and the Records mode, which provides a traditional tabular representation.

scratch (Figure 5(D3)). This balances the efficiency of preserving learned information with the flexibility of complete reconfiguration, providing the nuanced control that was previously unavailable.

Both refinements require the user to specify a new hyperparameter space, and we provide two ways of doing this: 1) **choosing base trials** and 2) **manually configuring a hyperparameter space**. In the first approach, the user can begin by choosing a set of trials to build a new hyperparameter space (Figure 5(D1)), making the space the union set of the hyperparameter configurations of the selected trials, and then adjusting it further. For the adjustment step, we additionally show the number of hyperparameter values that have been excluded so far from the initial experiment setting and will be excluded by the current refinement interaction to show how narrow the space becomes. Additionally, the user can modify the hyperparameter space without choosing base trials, configuring each hyperparameter manually.

Notifications Tab (Figure 5(E)). To minimize the need for constant monitoring (**C2**), we adapt the familiar concept of push notifications to the HPO context. In this notification-driven workflow, users receive smartphone alerts when specific events, dynamically configured through the interface, occur during an experiment. We design six types of triggering conditions: **Experiment** conditions alert when there is a change in an experiment's state (Started, Done, Paused, Resumed); **Progress** conditions capture when certain brackets or rounds are done, helping users engage at the moment when partial outcomes are ready; **Performance** conditions watch the model's performance and capture when accuracy reaches a user-defined target, or improves by more than a user-defined amount, Δ; **Timeout** conditions can be combined with Progress or Performance conditions to detect cases where the parent condition is not triggered within a user-defined time window; for example, combined with a performance condition, a user may specify: *"notify me when the performance does not improve after one hour."*; **Emergency** conditions capture urgent events such as high disk usage, high GPU temperatures, and low GPU utilization; and **User Trial** conditions alert when user trials are done.

The Notifications tab consists of two lists, allowing the user to manage received notifications and triggering conditions. The Notifications list displays received notifications in chronological order from newest to oldest, while the Conditions list shows triggering conditions that users can activate or deactivate through toggle switches. The Progress and Performance conditions can be added via dedicated dialogs that visualize the experiment's current status to help users set meaningful thresholds (Figure 5(E1)). Users can add Timeout conditions to detect stalled experiments.

4.4 Implementation

The HyPockeTuner system[2] consists of three layers: a client, a server, and workers. The client is written in JavaScript with React [54] and visx [1], and the push notifications are implemented using the standard Web Push Protocol [53], which is compatible with both Android and iOS. Built upon FastAPI [48] and PyTorch [45], the server provides APIs to manage experiments, user trials, notifications, and triggering conditions.

[2]Code: https://github.com/skku-idclab/HyPockeTuner
Demo: https://idclab.skku.edu/HyPockeTuner-demo/

We extend the original implementation of the BOHB algorithm [17] in two ways. First, we make it progressive, enabling server-side logic, such as checking if triggering conditions are met or pushing intermediate results to the client, to be executed between trials. Internally, the system manages a priority queue of trials whose configurations and order are determined by the BOHB algorithm, with explicit priority given to user-requested trials. Second, we enable dynamic narrowing of the search space during a BOHB run. Specifically, we mask out the probability mass assigned to excluded hyperparameter values or ranges, then rescale the remaining probabilities so that their sum equals one. Finally, to ensure external validity, we provide an API that supports the integration of existing model training code into our system; details are included in the supplementary material.

5 Evaluation

We evaluated HyPockeTuner from two perspectives, addressing the following questions:

(1) How quickly and accurately can users acquire different types of data facts using EventCrumb?
(2) How can users employ HyPockeTuner to manage long-running HPO experiments in practice?

To address the first question, we conducted a pilot study with 12 participants. Each participant was asked to answer multiple-choice questions (four options each) designed to test their ability to identify specific information supported by the visualization. To address the second question, we carried out two deployment studies where participants freely used our system to manage their own HPO experiments over five days. In this section, we report the findings from both studies and discuss them in relation to the challenges and design goals in section 6.

5.1 Exploratory Pilot Study

We initially considered conducting a comparative study against existing approaches, such as the visualizations used in desktop-based HPO dashboards (e.g., W&B, TensorBoard, and Microsoft NNI) or the common practice of inspecting experiment logs in spreadsheet software. However, our preliminary exploration revealed fundamental challenges in designing a valid and meaningful comparison. Desktop tools primarily support retrospective analysis, whereas EventCrumb is designed to highlight experiment progression and user interventions. As a result, tasks central to our system were not well supported by the visualizations in these tools. Moreover, existing desktop tools lack dedicated and effective mobile interfaces, which prevents a fair comparison on mobile devices. A comparative study with the mobile version of W&B further confirmed this limitation: participants struggled with basic navigation and were unable to complete the required tasks within the given time. Given these constraints, we chose to evaluate our system independently, focusing on assessing the effectiveness and usability of EventCrumb.

Study Design. Each participant was asked to use EventCrumb to answer eight questions (Table 1) about a completed HPO experiment. To evaluate the effectiveness of EventCrumb in understanding experiment history, participants interacted only with the EventCrumb tab, while all other views were disabled.

Table 1: Pilot study questions.

ID	Question
Q1	Identify the trial with the highest performance metric, including its ID, performance metric, and budget.
Q2	Find when the most recently completed trial finished and the performance metric it achieved.
Q3	Determine when the hyperparameter space was last modified and how many hyperparameters were changed.
Q4	Count how many trials have been completed since the last system access.
Q5	Count the total number of performance improvements throughout the experiment.
Q6	For the most recent "Target performance reached" notification, identify the user-specified target, the triggering trial's ID, and its performance metric.
Q7	For a user trial, identify its budget, performance metric, and whether it improved upon the previous best.
Q8	Find the longest consecutive sequence of trials without performance improvement.

Participants. We recruited participants from a university. To ensure that participants had sufficient background knowledge and practical experience in HPO, we administered a set of screening questions. Specifically, we asked whether they had (1) completed at least one semester-long university-level ML course, (2) trained and evaluated classification or regression models using libraries such as scikit-learn, TensorFlow, or PyTorch within the past year, (3) attempted to improve model performance by adjusting hyperparameters such as learning rate, batch size, or regularization within the past year, and (4) conducted long-running experiments involving model training or hyperparameter tuning lasting more than 24 hours. Based on these criteria, we recruited 12 participants (P1-12; 5 female and 7 male), consisting of 5 PhD students, 6 master's students, and 1 undergraduate student, aged 22-28 (M = 25.5, SD = 2.1). None of the participants had taken part in the needfinding study. Each participant received 20,000 KRW (approximately 15 USD) for a 60-minute session.

Dataset. To enhance ecological validity, we collected event logs from actual HPO experiments during a preliminary five-day study. Two ML experts used a prototype version of our system (without EventCrumb) to optimize deep learning models for their own problems: one focused on multi-label text classification with BERT, and the other on image segmentation with UNet. The first experiment generated a dataset with 2 brackets, 5 rounds, 22 trials, and 16 events while tuning 10 hyperparameters. The second produced a larger dataset comprising 2 brackets, 6 rounds, 55 trials, and 42 events while tuning 9 hyperparameters. We employed the first dataset as a tutorial to familiarize participants with the interface, and the second as the main test dataset in the experiment.

Task and Questions. Participants were asked to select the correct answer from four options for each of eight questions (Table 1). Four of these questions (Q1–Q4) were adapted from evaluations of desktop-based HPO systems in prior studies [44, 55], while the remaining four (Q5–Q8) were newly designed for the mobile HPO context. We designed the questions to focus on objective details of the experiment rather than relying on subjective interpretation to better isolate the effectiveness of our visualization.

From the pool of questions used in prior studies [44, 55], we excluded algorithm-specific ones (e.g., HyperTendril's fANOVA importance analysis or ATMSeer's algorithm distribution histograms), as they require larger displays and prior knowledge of the respective systems. We also excluded questions that depended on multi-dimensional visualizations (e.g., HyperTendril's parallel coordinates), which are unsuitable for smartphone screens. Finally, we discarded questions that required cross-experiment comparisons, since our interface displays only one experiment at a time. As a result, we defined four questions: model selection (Q1), monitoring the most recent trial (Q2), tracking a refinement interaction (Q3), and quantifying the progress (Q4).

As both studies assume continuous desktop monitoring, we designed four extra questions (Q5-Q8) that are relevant to the challenges we identified (**C1**-**C3**) and we want to address in mobile HPO: tracking cumulative performance improvements (Q5, **C1**), tracking; notification contexts (Q6, **C2**), judging the impact of user intervention (Q7, **C3**), and detecting performance plateaus (Q8, **C1**).

Procedure and Apparatus. Each one-hour session consisted of five phases. First, participants finished a consent form and completed a screening questionnaire (10 minutes). Next, we provided a tutorial of EventCrumb using the training dataset, during which participants practiced small exercises related to the main questions (20 minutes). After completing the tutorial, participants were encouraged to freely explore the interface and ask questions to ensure familiarity with the system (5 minutes). Third, participants answered the eight questions using the test dataset (10 minutes). Finally, participants completed a System Usability Scale (SUS) questionnaire and participated in a semi-structured interview about their experience (15 minutes).

We ran EventCrumb on an iPhone 15 Pro (2,556×1,179 resolution), and recorded the smartphone screen and interaction logs (e.g., scrolling, tapping nodes or links, using shortcuts, and collapsing the timeline) for subsequent analysis. In addition, we used a custom-built session-management desktop application to present questions and answer choices, and to record participants' response times and accuracy.

Results. Table 2 presents the accuracy, mean completion time, and interaction counts for each question. The overall accuracy across all questions was 96.9% (93/96 correct responses), demonstrating EventCrumb's effectiveness in conveying factual information about HPO experiments. Questions Q1–Q4, which were adapted from prior studies, achieved 97.9% accuracy (47/48), and Q5–Q8, considering HPO on mobile, reached 95.8% (46/48). For the three incorrect answers, we followed up with the participants during the debriefing interviews. P4 confused a "system access" event bubble with a user trial node in Q4, P12 overlooked detailed information in a detail

Table 2: Accuracy, completion time, and interaction counts for pilot study questions.

Question	Accuracy	Time (s) Mean (SD)	# of Interactions, Mean (SD)				
			Scrolling	Tapping nodes or links	Using shortcuts	Collapsing timeline	Total
Q1	100% (12/12)	14.7 (5.1)	0.2 (0.6)	1.1 (0.3)	0.0 (0.0)	0.0 (0.0)	1.3 (0.6)
Q2	100% (12/12)	18.2 (4.6)	1.0 (0.0)	1.2 (0.4)	1.0 (0.0)	0.9 (0.3)	4.1 (0.5)
Q3	100% (12/12)	34.5 (25.4)	3.5 (2.9)	2.0 (1.6)	2.7 (1.4)	0.0 (0.0)	8.2 (4.6)
Q4	91.7% (11/12)	16.7 (8.0)	3.8 (1.9)	0.2 (0.5)	2.4 (1.6)	0.3 (0.8)	6.8 (3.5)
Q5	100% (12/12)	15.7 (8.2)	2.2 (1.3)	0.1 (0.3)	1.8 (0.4)	1.2 (0.6)	5.2 (1.6)
Q6	91.7% (11/12)	76.5 (38.2)	8.3 (4.3)	4.5 (3.5)	4.8 (2.8)	1.0 (0.9)	18.6 (7.8)
Q7	91.7% (11/12)	32.0 (22.4)	3.6 (2.8)	2.3 (1.4)	1.8 (1.0)	0.3 (1.2)	8.0 (5.2)
Q8	100% (12/12)	9.0 (3.1)	1.2 (0.6)	0.0 (0.0)	1.3 (0.5)	1.0 (0.0)	3.6 (0.9)
Overall	**96.9% (93/96)**	**27.3 (14.4)**	**3.0 (3.2)**	**1.4 (2.0)**	**2.0 (1.8)**	**0.6 (0.7)**	**7.0 (6.2)**

pop-up that is visible after scrolling in Q6, and P11 forgot that the textual accuracy displayed in a trial node indicates that the node is an improver in Q7.

The mean task completion time across questions was 27.3 seconds with large variation (*Range* = 14.7-76.5 seconds), indicating differences in question difficulty. The mean number of interactions was 7.0 (*SD* = 6.2), with substantial variation (*Range* = 1.3–18.6) reflecting various task complexity. Simple lookup tasks required minimal interactions (Q1: M = 1.3, Q8: M = 3.6), while complex search tasks involving scanning and multi-level inspection demanded more interactions (Q6: M = 18.6, Q3: M = 8.2, Q7: M = 8.0).

The three questions with the longest completion time (Q3, Q6, Q7) required participants to locate a specific event bubble or user trial within the event sequence, which demanded visual search. For example, Q6, the most time-consuming question, asked participants to identify a "push notification" event bubble (🔔) of a particular notification type ("target performance reached"). Since the notification type was not displayed directly on the bubble to avoid overloading the visualization with too many icons, participants had to tap each bubble (🔔) and inspect the detail popup to determine its notification type. Such two-level search interactions may have increased response time. Q3, the second most time-consuming question, revealed differences in mean completion time depending on participants' search strategies. Analysis on interaction logs revealed that participants who used the event bubble shortcuts (M = 19.9s, n = 5) completed the task faster than those who relied on linear scanning (M = 54.9s, n = 2). Participants who combined both approaches showed intermediate performance (M=40.9s, n=5). In post-study interviews, P12, who used a combined approach, explained "*I was unfamiliar with the event bubble shortcuts at first, but once familiar, this would be quickly resolved.*"

Regarding perceived usability, EventCrumb achieved an SUS score of 82.3 (*SD* = 10.4, *Range* = 62.5–100), placing ours in the top quartile of evaluated systems according to a previous study [6]. As benefits of EventCrumb, ten out of 12 participants (83.3%) highlighted the ability to quickly identify improvements over the course of an experiment at a glance, enabled by color coding and the collapsing interaction. P11 appreciated: "*Collapse by Non-Improver is really good because it summarizes performance improvements at once and filters out unnecessary information.*"

Eight participants (66.7%) emphasized that the visualization helped them recall the change history alongside trial results (e.g., refining the search space), which mitigates the common problem of forgetting past experimental decisions. P2 noted: "*With W&B, I often forget how I changed settings throughout the experiment. Understanding past decisions may help prevent making similar mistakes.*" P11 added: "*W&B automatically logs results, but for modifications I make, such as narrowing the search space, I have to log them separately. Here, everything is captured and shown in one place.*" Beyond intervention tracking, participants contrasted the analytical purposes of timeline and tabular representations. P7 noted: "*Table-based tools show individual experiment results well, but struggle to convey how performance changes as you iteratively tune hyperparameter values. The timeline makes this temporal progression visible.*" P2 further distinguished their roles: "*W&B excels at post-hoc analysis with charts. For managing running experiments, a timeline showing results chronologically works better. They serve different purposes.*"

Seven participants (58.3%) reported that our visualization is optimized well for smartphone screens; P2 commented: "*The zigzag layout fits the mobile screen perfectly, which makes it easy to follow vertically.*" Three participants (25%) noted challenges when accessing desktop-oriented tools on mobile devices. P2 recalled: "*When viewing W&B on my phone, the interface breaks and I have to pinch and drag just to fit content on the screen.*" P7 described a common pattern of abandoning mobile access: "*Yesterday I tried checking W&B on my smartphone while lying down at home, but gave up because it was too frustrating. I just waited until the next morning to check on the desktop.*" P9 noted specific interaction difficulties: "*On W&B mobile, clicking is uncomfortable, and the table requires constant scrolling back and forth to see hyperparameter values.*"

Although EventCrumb was primarily designed for monitoring and tracking, eight participants (66.7%) recognized its usefulness in other contexts. P11 highlighted its educational value, noting: "*Visual representations help novices to understand how HPO algorithms work better than textual explanations.*" P12 emphasized its potential for collaboration: "*I can show colleagues exactly what I did, such as what I changed and how the performance changed after that, which is much clearer than describing it with W&B logs.*"

One limitation of EventCrumb was that too many icons overwhelmed several users, particularly those designed for different

notification types. P10 noted: "*The meaning of icons relevant to notifications was not immediately clear. I needed some time to be familiar with them.*" In response to this feedback, we improved our design by adding a help button that reveals a legend explaining all event types. Five participants also pointed out the absence of traditional visualizations for HPO analytics, as we deliberately narrowed the scope of the evaluation to EventCrumb rather than the entire HyPockeTuner system. P9 commented: "*W&B shows how performance changes epoch by epoch through line charts. EventCrumb is good for managing and monitoring HPO experiments, but I would need those charts as well to understand the experiment.*"

5.2 Deployment Study

To evaluate if and how HyPockeTuner can help users perform HPO experiments in practice, we conducted two deployment studies in collaboration with two ML experts (denoted as E1 and E2), each focusing on training a vision-language model and a multi-label text classification model, respectively. None of the participants had taken part in the needfinding study or the pilot study, nor had any relationship with the research team. In this section, we report the results of the deployment studies, focusing on the important observations we made. For completeness, we provide a chronological summary of each study in the supplementary material.

Study Design. In the deployment study, we aim to evaluate our system under a scenario where participants train a deep-learning model for five weekdays, from Monday to Friday. The participants chose the dataset, deep learning model (i.e., Python source code), and the hyperparameters of their interest. They used their own smartphones, and we modified the source code for training the model to conform to our system's API. We asked the participants to identify the best hyperparameter configuration that achieves the highest evaluation metric. Since the participants were employed at a company or a laboratory, they should perform the task with their regular work routines. We believe this is externally valid as HPO processes are often done in parallel with other tasks, such as model development. Each participant received 150,000 KRW (approximately 110 USD) for their five-day participation.

The BOHB algorithm assigns different budgets to configurations as the experiment proceeds, and we regard the budget as the number of epochs for which we train a model (e.g., a budget of 1: 1 epoch). Since at least one epoch is needed to evaluate a configuration, the system cannot report the result to the user before training for one epoch. We call the minimum time that the user should wait for the initial result a **training time quantum**.

While conducting a preliminary test to validate our experiment design, we learned that the training time quantum plays a critical role in the validity and results of the study. If it is too short, i.e., a few seconds, as in the evaluation of the ATMSeer system [55], the study becomes less externally valid since only simple ML models can be trained for one epoch within that timespan and provide the result almost in real-time. In contrast, if it is too long, the length of the study should increase correspondingly so that the user can test a sufficient number of configurations. In our case, the training time quantum was approximately 15 minutes for both participants; this means that it took about 15 minutes to evaluate a configuration assigned the minimum budget of 1 and about 20.25 hours for one assigned the maximum budget of 81.

Procedure. One day before the study (Sunday), we held a one-hour preparation session with each participant in our lab. During the first 40 minutes, we set up our client on their smartphones as an application and provided a tutorial in which they could access the full system. In the remaining 20 minutes, participants practiced by conducting an HPO experiment on a toy dataset, where they were asked to optimize eight hyperparameters for training a classification model on the MNIST dataset [14]. The participants could freely use our system from Monday to Friday. We had regular online debriefing sessions at 7 pm every day, where we conducted semi-structured interviews based on the daily interaction logs to clarify their findings and intentions. In the last debriefing session on Friday, we gathered their subjective feedback with an SUS score.

One week after the study, we conducted a retrospective session with each participant to evaluate the EventCrumb's effectiveness in supporting experimental recall and future planning, using a scenario in which they designed a new experiment based on their prior experience. Each one-hour semi-structured interview followed a two-phase protocol. First, we asked participants to (1) recall and describe the decisions they had made during the experiments and (2) explain what they would do differently to achieve better performance more quickly if they were to run a similar experiment. Second, we showed them the EventCrumb visualization and asked the same questions again to assess whether our visualization improved their ability to recall past decisions and plan future experiments.

5.2.1 Training a Vision-Language Model for Image-Text Retrieval. E1 was an ML researcher developing a CLIP model [47] for image-text retrieval tasks. We used the MS-COCO dataset [37], consisting of 118,287 training images with 591,753 captions and 5,000 validation images with 25,014 captions. The model architecture employed ResNet-50 as the vision encoder and a Transformer-based text encoder with 1,024-dimensional embeddings. E1 explored a 9-D hyperparameter space: 4 Continuous, 3 Ordinal, and 2 Nominal hyperparameters. We used the Image-Text Retrieval Score as the performance metric, calculated as the average of recalls (R@1, R@5, and R@10) for both image-to-text and text-to-image retrieval. Over five days, E1 completed 115 trials across 4 experiments, with 88 logged events: 60 system accesses, 3 narrowing operations, 1 redefinition of the hyperparameter space, 1 addition of a user trial, and 9 additions and 2 edits of notification conditions.

The system delivered 20 notifications: 12 system-generated alerts and 8 from user-defined conditions. E1 viewed 19 notifications; only one remained unchecked. We measured *re-entry latency* as the time elapsed from when a notification is delivered to when the user opens the application to view it. Re-entry latencies for viewed notifications were low (median = 0 minutes, max = 48.7 minutes), indicating that participants typically returned to the system promptly after receiving a notification. The mean time-to-first-improver after the initial trial was 180.56 minutes (SD = 54.96) across E1's experiments.

Perceived Benefits of Mobile HPO Management. E1 used HyPockeTuner 25 times over five days, 14 times from his laboratory and 11 times from home, where multiple accesses made within a short interval (30 seconds) were counted as a single use. He found our system to be accessible, noting: "*Previously, accessing the experiment*

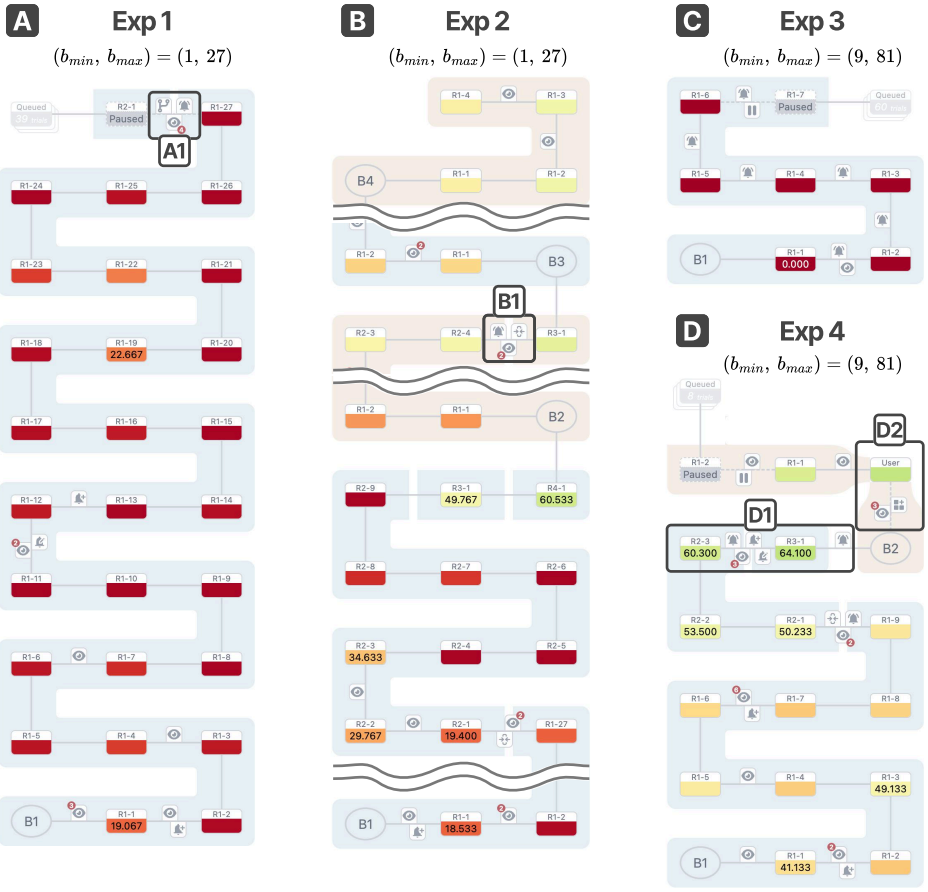

Figure 6: HPO experiments by E1, who optimized hyperparameters for training a vision-language model for image–text retrieval. (A1) After a notification indicating completion of the first round, he accessed the system. Noticing the poor performance of early trials, he initiated a new experiment (B) with a refined hyperparameter space. (B1) After being notified of the completion of another round in the second experiment, he further narrowed the hyperparameter space. (D1) In the later phase of the study, he began using Performance Improved conditions to receive notifications whenever the model achieved a new highest score. (D2) Before going to bed, he added a user trial with a large budget (100) to train the model for more epochs overnight.

at home was tedious since I had to set up connections to the status, but HyPockeTuner was much more accessible. I could conveniently check the server before going to bed or right after waking up." His interactions at home included setting a one-hour timeout before taking a shower (Monday, 21:17), performing a narrowing operation before going to bed (Tuesday, 00:34), and running a user trial with a large budget (100) before going to bed (Thursday, 23:08). He further explained: "*Since I cannot check the progress while sleeping, I deliberately request tasks that take a long time so that I can review the results when I wake up, making my time more productive.*"

Refinement Operations Improved Performance. E1 performed four refinement operations during the experiment. After identifying a large number of poorly performing trials (Monday, 21:17), he redefined the hyperparameter space by selecting the top five trials as base trials (Figure 6(A1)). The following day, he narrowed the space twice (00:34, 15:00) by excluding less promising hyperparameter values identified through the Hyperparameter Effects section of the Overview tab (Figure 6(B)). With the space narrowed three times, he achieved a maximum score of 60.5. He also observed that a batch size of 16 frequently appeared among the high-performing trials in EventCrumb. Finally, he launched a new experiment with the same hyperparameter space but with increased budgets and explicitly included the batch size of 16, which resulted in the highest score of 64.1 at 81 epochs. He reflected: "*Without this system, I would have stopped at 60.5, thinking it sufficient. I would not have realized that the score could reach 64.1 simply by adding a single value (16).*"

On-the-fly Hypothesis Testing. After achieving the highest score of 64.1 on Wednesday, E1 became curious whether the score would improve with larger budgets. To test this hypothesis, he added a user trial of the same configuration but with an increased budget (100) before going to bed (Figure 6(D2)). The next morning, he observed that the score had dropped to 63.5, which he interpreted as a sign of overfitting, concluding that further increasing the budget would not yield improvements. He reflected: "*By using the user trial, I could easily test my hypothesis that the model would overfit when trained for more epochs overnight.*"

Transition of Notification Use. Throughout the experiments, E1's notification usage had shifted. Initially, he relied on Progress conditions, such as alerts for bracket or round completion (Figure 6(A1, B1)), to refine the hyperparameter space when these conditions occurred. He then shifted to Performance Reach conditions to be notified when a new experiment achieved a higher score than the previous one. Finally, he adopted Performance Improved conditions to receive notifications each time the model recorded a new highest score (Figure 6(D1)).

Reflection and Learning from Past Experiments. In the retrospective session, without EventCrumb, E1 recalled improving the score from 60.5 to 64.1 by adjusting the hyperparameter space and budgets, but struggled to remember specific details. With EventCrumb, however, he was able to reconstruct the narrative of the experiment, for example, that he initially excluded the batch size of 16 when narrowing down the space, but later added it back, which contributed to achieving the highest score. He noted: "*I can see when I received notifications and when I narrowed the hyperparameter space. All my interactions are visible in EventCrumb. Previously, I had to use note-taking apps, which were cumbersome.*"

Looking back, E1 also identified directions for improvement. He regretted setting the minimum budget to one: "*Looking at EventCrumb, I can see early trials with a budget of 1 achieved similar scores, which makes them hard to distinguish. I would have set larger minimum budgets, such as 3.*" He further suggested that it would be promising to test the configuration with the highest score for fewer epochs than 100 (i.e., before overfitting occurred), as there might exist an earlier maximum point. Finally, E1 emphasized the benefit of our system for archiving: "*Later, when conducting similar experiments, I can reference this, since we usually don't document hyperparameter settings or experiment details.*"

5.2.2 *Training a Multi-Label Text Classification Model.*

E2 was a researcher working on NLP models for multi-label text classification of online news comments. We used the K-MHaS dataset [32], a multi-label corpus for hate speech detection, consisting of 109K Korean utterances annotated with nine hate speech labels. We randomly sampled 20% as test data and 80% as training data. The model employed BERT base [15] with 110 million parameters. E2 explored a 10-dimensional hyperparameter space: four Nominal, four Ordinal, and two Continuous hyperparameters. We used classification accuracy as the performance metric. Over five days, E2 completed 163 trials across 4 experiments, with 119 logged events, including 61 system accesses, 1 narrowing operation, 3 redefinitions of the hyperparameter space, 1 user trial added, 13 notification conditions added, and 2 notification conditions edited.

The system delivered 47 notifications: 24 system-generated alerts and 23 from user-defined conditions. E2 accessed the application after 34 notifications, with a median re-entry latency of 6.2 minutes (max = 165.8 minutes). The mean time-to-first-improver was 97.88 minutes (SD = 107.62 minutes).

Access Across Different Locations and Contexts. E2 accessed the system 10–20 times per day, mostly for monitoring tasks in various locations and contexts, such as during breaks, restroom visits, and meal times. Beyond simple monitoring, he launched experiments from restaurants during lunch twice (Tuesday 13:53, Thursday 12:23). He also used the system while on the move, such as during his commute (Tuesday 14:02, 14:19, 14:41). In the interview, E2 emphasized that mobile access was convenient: "*I used it this much because I could make changes on mobile rather than just watching. If it were a desktop app, I wouldn't have used it as often. I would rather use SSH to connect to the server, which I'm familiar with. Either way, I would still be tied to my desktop.*"

Benefits of an Integrated Timeline. E2 found EventCrumb useful because it visualized both experiment progression and user interventions on a single timeline. For example, he highlighted the system access event bubble (👁) as particularly helpful, since it enabled the quick identification of trials completed after the last visit that he had not yet examined. He also appreciated being able to see when and where notifications occurred after his previous access: "*I favored the icon showing the last visit because it helped me focus once more on the trials that came after it.*"

Early Detection of Failures. After launching the experiment on Monday, E2 received Exception Happened notifications that evening, indicating CUDA out-of-memory errors (Figure 7(A1)). With HyPockeTuner, he discovered that the batch size of 256 was too large, causing the failure. E2 emphasized: "*These real-time notifications are really important. If I had just left the experiment unattended, I wouldn't have known about the errors until coming back the next day.*"

Iterative Refinement of Experiment Settings. E2 conducted four experiments over the five days, each configured based on insights from the previous run. First, he excluded the batch size of 256 when launching the second experiment on Tuesday (Figure 7(A2)). He also refined the minimum and maximum budgets. The initial range was (1, 81), which achieved a maximum accuracy of 0.74. He then adjusted the range to (1, 9) to quickly check whether OOM errors persisted, reaching a maximum accuracy of 0.72 (Figure 7(B)). Concluding that 1–3 epochs were insufficient, he increased the budgets to (3, 27), which achieved a maximum accuracy of 0.75 (Figure 7(C1)). Finally, he tested the range (27, 81) to investigate the effect of extended training epochs, ultimately achieving the highest accuracy of 0.77 (Figure 7(D1)). E2 emphasized the value of iterative refinement: "*With small budgets, I only got poor results, but refinement operations let me cut off unsuccessful experiments early and focus on testing more promising ones.*"

Reflection and Learning from Past Experiments. In the retrospective session, with EventCrumb, E2 was able to recall details of the experiment; for example, that a batch size of 128 occasionally caused OOM errors, which he could not remember without visual support. He also reflected on his decision patterns: "*I can clearly see in EventCrumb how I made changes to the experiment after notifications, such as narrowing the space.*"

E2 identified opportunities for improvement, noting that starting with a minimum budget of 3 instead of 1 would have been more efficient, since one epoch did not result in meaningful model convergence. He further emphasized that distinguishing between multiple, similar experiments and recalling the rationales for interventions would be nearly impossible without the temporal context provided by EventCrumb, highlighting its value not only for real-time monitoring but also as a learning instrument for improving future optimization strategies.

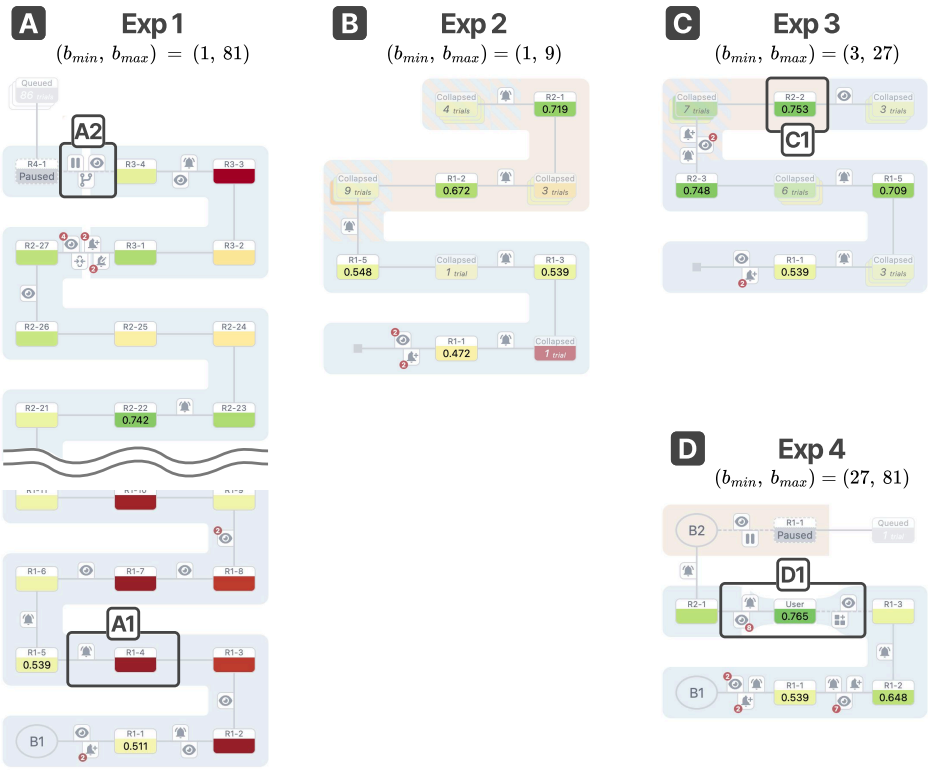

Figure 7: HPO experiments conducted by E2, who optimized hyperparameters for training a multi-label text classification model. (A1) He received an out-of-memory exception notification in the fourth trial. (A2) Concluding that a batch size of 256 caused the error, he launched Experiment B with smaller budgets (1, 9) and with that value excluded; this run proceeded without errors. (C1) In the third experiment, with increased budgets (3, 27), a new record was achieved, and he was notified. (D1) Noticing the potential of larger budgets, he added a user trial based on the previous best configuration but with more budget, which yielded the highest accuracy.

6 Discussion and Future Work

Human-in-the-Loop HPO Provides Benefits over Full Automation. Surprisingly, most of the users we met during the study (U1–8, P1–12, and E1–2) already had prior experience managing HPO experiments, with interventions ranging from passive actions, such as simple monitoring, to more active ones, such as stopping the experiment and refining the hyperparameter space. Many had also developed their own ad hoc solutions to streamline the process, for example, by using remote desktop applications or integrating custom notification code into the training pipeline. This observation challenges a fundamental assumption of previous HPO algorithms that they operate without any human intervention.

Throughout our study, we observed the benefits of Human-in-the-Loop HPO compared to fully automated optimization. First, engaging humans in the HPO process enables the identification of improvement opportunities that automation alone would overlook, leading to performance gains. For example, in the first deployment study, E1 achieved the maximum score (64.1) by increasing the budget and introducing a hyperparameter value (batch size = 16) based on his own observations. He noted that without recognizing this promising hyperparameter, he would have been satisfied with the suboptimal result of 60.5 that was initially found by the algorithm.

Second, Human-in-the-Loop HPO allows users to flexibly plan and conduct experiments. Furthermore, when properly supported, for example, through mobile access or an event-driven workflow, this integration can lead to more productive use of both human attention and computational resources. For instance, E1 set a one-hour timeout notification before taking a shower so that he would not forget to check the results afterward. He also deliberately launched a long-running task before going to bed, ensuring that the experiment would not finish during his sleep and leave the computation resources idle. These behaviors demonstrate how Human-in-the-Loop HPO enables users to flexibly align experimentation with their contexts while making more effective use of both their attention and available resources.

Third, involving humans in the HPO process can reduce the anxiety that users may experience. During the deployment studies, we observed several types of threats to HPO. For example, assigning too little budget can be ineffective, trials fail due to unexpected exceptions (e.g., OOM errors), and the algorithm may waste time exploring unpromising configurations. Such unpredictable

and uncontrollable factors often create anxiety, prompting users to continuously monitor results, even though HPO algorithms are intended to operate without human attention. For this reason, actively engaging users in the process can help alleviate such discomfort.

These observed benefits indicate that engaging humans in HPO can be a practical and robust option in real-world use. However, the dominant paradigm in both past and current HPO research continues to focus on full automation. As a result, enabling human intervention is challenging to realize due to the lack of algorithmic support. In our case, for example, the original BOHB implementation [17] was monolithic, requiring us to modify it into a progressive version so that intermediate results became visible and users could intervene during the process. Our work highlights a need for future work on Human-in-the-Loop HPO algorithms.

EventCrumb Bridged the Gap between Experiment History and User Context through Unified Visualization. Our visualization brings together three types of events, experiment progress (brackets, rounds, and trials), user interventions, and their outcomes (performance metrics and triggered notifications), on a unified, collapsible timeline, whereas previously these pieces of information were presented in a disconnected form. Beyond supporting accurate and efficient task performance, as shown in our pilot study, our approach enabled more advanced analysis and reasoning.

First, our visualization facilitated pattern discovery. In particular, when non-improvers were aggregated, users could readily identify key phases in experiment history, such as periods of improvement and plateau. Moreover, aligning these patterns with user interventions allowed them to reason about cause-and-effect relationships, for example, linking refinements to subsequent performance gains. Second, the visualization supported linking experiment progress with user context. For example, participants used the system access event bubble (👁) as a temporal anchor, helping them quickly identify trials completed since their previous visit and the changes that required inspection. Finally, we observed the broader opportunities of our visualization beyond HPO. Participants acknowledge the usefulness of EventCrumb as a teaching tool to illustrate how HPO algorithms work, and also highlighted its potential for team communication by visually conveying decisions and their performance impacts, thereby serving as an effective communication medium.

HyPockeTuner Reduced the Burdens for Experiment Management while Accommodating Individual Daily Routines. Participants reported that our system reduced several types of burdens present in previous experiment management practices. The first was a physical burden, as they needed to remain at the desktop to make changes, an issue we addressed by enabling mobile access (**DG1**). The second was a cognitive burden for monitoring, since participants occasionally had to divert attention to the experiment, disrupting their workflow and even causing anxiety; we mitigated this through a notification-driven workflow (**DG3**). The third was another cognitive burden, namely the difficulty of tracking refinement operations, which was addressed through our visualization (**DG2**) and supporting refinement operations (**DG4**).

Lowering these burdens can encourage more active management and intervention of HPO experiments, ultimately allowing experiment management to be smoothly incorporated into daily routines. Participants frequently accessed the system throughout the day from different locations—such as the lab or home, and in various contexts, including breaks, meals, and commutes. This flexibility also allowed them to adapt their interactions to the situation, such as setting a one-hour timeout before taking a shower or deliberately leaving long-running tasks before going to bed. These practices demonstrate that our system enables a tight integration of experiment management with daily routines.

Interestingly, these interaction patterns parallel user behaviors observed in idle games, where gameplay runs autonomously with minimal player input, and players periodically return to allocate accumulated resources [3, 13]. Similarly, our participants leveraged naturally occurring idle periods, such as sleep or breaks, to check progress and issue interventions, echoing prior observations that players of ambient or background games strategically schedule interactions around daily routines [29]. This parallel suggests that design strategies from idle games may offer useful insights for HPO.

Generalizability of EventCrumb to Other HPO Algorithms. Although EventCrumb was developed for the bracketed structure of BOHB, its design principles for showing temporal trial progression and user interventions are applicable to a wide range of HPO algorithms. Across most HPO algorithms, three elements remain consistent: trials proceed over time, each trial yields measurable performance, and users make decisions such as stopping, adjusting, or adding trials. These shared properties provide room for adapting EventCrumb to alternative HPO algorithms.

For example, multi-fidelity methods such as Hyperband [33], ASHA [34], and DEHB [5] involve staged evaluations and can be adapted to EventCrumb. Bayesian optimization [49], evolutionary strategies [60], and grid or random search [8] do not define explicit rounds, but their trials can still be shown in temporal order with performance and intervention points visualized. In these cases, users may define custom trial groupings to create structures; for example, users may group every ten trials in a random search or each generation in an evolutionary method as a bracket. In contrast, population-based training [27] or meta-learning approaches [25] do not produce sequential, independent trials and would require substantial modifications to accommodate their fundamentally different trial paradigm.

Limitations and Future Work. While HyPockeTuner demonstrates the feasibility of mobile HPO management, several limitations warrant discussion and future work. Our deployment studies involved participants with specific deep learning tasks and controlled experimental settings. Broader deployment across diverse ML domains, team sizes, and organizational contexts would strengthen the generalizability of our findings.

The training time quantum significantly influences our approach's practicality. In our studies, both participants experienced approximately 15-minute intervals for minimum budget trials, extending to 6.75 hours for maximum budgets. This timeframe aligned well with the observed monitoring frequency of once per 77 minutes on average. However, scenarios with substantially longer training cycles, such as large language model pre-training, where single epochs may require days, would alter user interaction patterns. While our notification system partially addresses this, the effectiveness of mobile interventions could diminish when feedback loops extend

beyond daily cycles. Therefore, an interesting direction for future work is to extend our approach to support longer time quanta.

Our system currently provides only a mobile interface. While this choice is a good first step for exploring the potential of mobile devices in isolation, the limited screen space and reliance on touch interactions may restrict the scope of analysis or involvement. Future work could explore a hybrid approach where a mobile interface complements a desktop interface, leveraging the strengths of both. Furthermore, extending the current EventCrumb design to accommodate other iterative HPO algorithms remains an important direction for future work. Finally, as noted by two participants in the pilot study, our tools also hold potential for use in educational settings or supporting communication about optimization workflows, beyond their primary role in experiment management.

7 Conclusion

Despite advances in Human-in-the-Loop HPO systems, existing approaches remain largely confined to desktop environments, limiting their effectiveness for managing the long-running experiments that are increasingly common in practice. To address this gap, we introduced HyPockeTuner, a mobile HPO system featuring EventCrumb, that enables users to monitor, steer, and reflect on experiments through notification-driven interactions and timely interventions from smartphones. Through a pilot study and two five-day deployment studies, we found that mobile access and notifications enabled users to effectively manage HPO experiments while integrating the process into their daily routines. These findings demonstrate that mobile HPO can serve as a practical and effective option for managing long-running HPO experiments, opening promising directions for future Human-in-the-Loop HPO systems.

Acknowledgments

This work was supported by the National Research Foundation of Korea (NRF) grant funded by the Korea government (RS-2025-24873100, RS-2023-NR077081) and by the Institute of Information and Communications Technology Planning and Evaluation (IITP) grant funded by the Korea government (MSIT), Artificial Intelligence Graduate School Program, Sungkyunkwan University (RS-2019-II190421), Yonsei University (RS-2020-II201361), and Seoul National University (RS-2021-II211343).

References

[1] airbnb. 2024. visx - a collection of expressive, low-level visualization primitives for React. https://airbnb.io/visx/. Last Accessed 1 April 2024.

[2] Takuya Akiba, Shotaro Sano, Toshihiko Yanase, Takeru Ohta, and Masanori Koyama. 2019. Optuna: A Next-generation Hyperparameter Optimization Framework. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining* (Anchorage, AK, USA) *(KDD '19)*. Association for Computing Machinery, New York, NY, USA, 2623–2631. doi:10.1145/3292500.3330701

[3] Sultan A. Alharthi, Olaa Alsaedi, Phoebe O. Toups Dugas, Theresa Jean Tanenbaum, and Jessica Hammer. 2018. Playing to Wait: A Taxonomy of Idle Games. In *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems* (Montreal QC, Canada) *(CHI '18)*. Association for Computing Machinery, New York, NY, USA, 1–15. doi:10.1145/3173574.3174195

[4] Amazon. 2024. Amazon SageMaker - Build, train, and deploy machine learning (ML) models for any use case with fully managed infrastructure, tools, and workflows. https://aws.amazon.com/sagemaker/?nc1=h_ls. Last Accessed 1 April 2024.

[5] N. Awad, N. Mallik, and F. Hutter. 2021. DEHB: Evolutionary Hyberband for Scalable, Robust and Efficient Hyperparameter Optimization. In *Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence, IJCAI-21*, Z. Zhou (Ed.). ijcai.org, 2147–2153.

[6] Aaron Bangor, Philip Kortum, and James Miller. 2009. Determining what individual SUS scores mean: Adding an adjective rating scale. *Journal of usability studies* 4, 3 (2009), 114–123.

[7] James Bergstra, Rémi Bardenet, Yoshua Bengio, and Balázs Kégl. 2011. Algorithms for hyper-parameter optimization. In *Proceedings of the 25th International Conference on Neural Information Processing Systems* (Granada, Spain) *(NIPS'11)*. Curran Associates Inc., Red Hook, NY, USA, 2546–2554.

[8] James Bergstra and Yoshua Bengio. 2012. Random search for hyper-parameter optimization. *J. Mach. Learn. Res.* 13, null (Feb. 2012), 281–305.

[9] Lukas Biewald. 2020. Experiment Tracking with Weights and Biases. https://www.wandb.com/ Software available from wandb.com.

[10] Bernd Bischl, Martin Binder, Michel Lang, Tobias Pielok, Jakob Richter, Stefan Coors, Janek Thomas, Theresa Ullmann, Marc Becker, Anne-Laure Boulesteix, et al. 2023. Hyperparameter optimization: Foundations, algorithms, best practices, and open challenges. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery* 13, 2 (2023), e1484.

[11] Matthew Brehmer, Bongshin Lee, Benjamin Bach, Nathalie Henry Riche, and Tamara Munzner. 2016. Timelines revisited: A design space and considerations for expressive storytelling. *IEEE transactions on visualization and computer graphics* 23, 9 (2016), 2151–2164.

[12] A. Chatzimparmpas, R. M. Martins, K. Kucher, and A. Kerren. 2021. VisEvol: Visual Analytics to Support Hyperparameter Search through Evolutionary Optimization. *Computer Graphics Forum* 40, 3 (2021), 201–214. doi:10.1111/cgf.14300 arXiv:https://onlinelibrary.wiley.com/doi/pdf/10.1111/cgf.14300

[13] Joe Cutting, David Gundry, and Paul Cairns. 2019. Busy doing nothing? What do players do in idle games? *International Journal of Human-Computer Studies* 122 (2019), 133–144. doi:10.1016/j.ijhcs.2018.09.006

[14] Li Deng. 2012. The mnist database of handwritten digit images for machine learning research. *IEEE Signal Processing Magazine* 29, 6 (2012), 141–142.

[15] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, Jill Burstein, Christy Doran, and Thamar Solorio (Eds.). Association for Computational Linguistics, Minneapolis, Minnesota, 4171–4186. doi:10.18653/v1/N19-1423

[16] Xianping Du, Hongyi Xu, and Feng Zhu. 2021. Understanding the effect of hyperparameter optimization on machine learning models for structure design problems. *Computer-Aided Design* 135 (2021), 103013.

[17] Stefan Falkner, Aaron Klein, and Frank Hutter. 2018. BOHB: Robust and Efficient Hyperparameter Optimization at Scale. In *Proceedings of the 35th International Conference on Machine Learning (Proceedings of Machine Learning Research, Vol. 80)*, Jennifer Dy and Andreas Krause (Eds.). PMLR, 1437–1446. https://proceedings.mlr.press/v80/falkner18a.html

[18] Johnu George, Ce Gao, Richard Liu, Hou Gang Liu, Yuan Tang, Ramdoot Pydipaty, and Amit Kumar Saha. 2020. A Scalable and Cloud-Native Hyperparameter Tuning System. arXiv:2006.02085 [cs.DC]

[19] Google. 2024. Azure Machine Learning - Use an enterprise-grade AI service for the end-to-end machine learning lifecycle. https://azure.microsoft.com/en-us/products/machine-learning. Last Accessed 1 April 2024.

[20] Google. 2024. Vertex AI - Innovate faster with enterprise-ready AI, enhanced by Gemini models. https://cloud.google.com/vertex-ai?hl=en. Last Accessed 1 April 2024.

[21] David Gotz and Harry Stavropoulos. 2014. Decisionflow: Visual analytics for high-dimensional temporal event sequence data. *IEEE transactions on visualization and computer graphics* 20, 12 (2014), 1783–1792.

[22] Yi Guo, Shunan Guo, Zhuochen Jin, Smiti Kaul, David Gotz, and Nan Cao. 2021. Survey on visual analysis of event sequence data. *IEEE Transactions on Visualization and Computer Graphics* 28, 12 (2021), 5091–5112.

[23] Xin He, Kaiyong Zhao, and Xiaowen Chu. 2021. AutoML: A survey of the state-of-the-art. *Knowledge-based systems* 212 (2021), 106622.

[24] Keita Higuchi, Shotaro Sano, and Takeo Igarashi. 2021. Interactive Hyperparameter Optimization with Paintable Timelines. In *Proceedings of the 2021 ACM Designing Interactive Systems Conference* (Virtual Event, USA) *(DIS '21)*. Association for Computing Machinery, New York, NY, USA, 1518–1528. doi:10.1145/3461778.3462077

[25] Timothy Hospedales, Antreas Antoniou, Paul Micaelli, and Amos Storkey. 2021. Meta-learning in neural networks: A survey. *IEEE transactions on pattern analysis and machine intelligence* 44, 9 (2021), 5149–5169.

[26] Frank Hutter, Lars Kotthoff, and Joaquin Vanschoren. 2019. *Automated Machine Learning: Methods, Systems, Challenges* (1st ed.). Springer Publishing Company, Incorporated.

[27] Max Jaderberg, Valentin Dalibard, Simon Osindero, Wojciech M Czarnecki, Jeff Donahue, Ali Razavi, Oriol Vinyals, Tim Green, Iain Dunning, Karen Simonyan, et al. 2017. Population based training of neural networks. *arXiv preprint arXiv:1711.09846* (2017).

[28] Kevin Jamieson and Ameet Talwalkar. 2016. Non-stochastic Best Arm Identification and Hyperparameter Optimization. In *Proceedings of the 19th International Conference on Artificial Intelligence and Statistics (Proceedings of Machine Learning Research, Vol. 51)*, Arthur Gretton and Christian C. Robert (Eds.). PMLR, Cadiz, Spain, 240–248. https://proceedings.mlr.press/v51/jamieson16.html

[29] Brendan Keogh and Ingrid Richardson. 2018. Waiting to play: The labour of background games. *European Journal of Cultural Studies* 21, 1 (2018), 13–25.

[30] Aaron Klein, Stefan Falkner, Jost Tobias Springenberg, and Frank Hutter. 2016. Learning curve prediction with Bayesian neural networks. In *International Conference on Learning Representations*.

[31] Robert Kosara and Silvia Miksch. 2001. Metaphors of movement: a visualization and user interface for time-oriented, skeletal plans. *Artificial intelligence in medicine* 22, 2 (2001), 111–131.

[32] Jean Lee, Taejun Lim, Heejun Lee, Bogeun Jo, Yangsok Kim, Heegeun Yoon, and Soyeon Caren Han. 2022. K-MHaS: A Multi-label Hate Speech Detection Dataset in Korean Online News Comment. In *Proceedings of the 29th International Conference on Computational Linguistics*, Nicoletta Calzolari, Chu-Ren Huang, Hansaem Kim, James Pustejovsky, Leo Wanner, Key-Sun Choi, Pum-Mo Ryu, Hsin-Hsi Chen, Lucia Donatelli, Heng Ji, Sadao Kurohashi, Patrizia Paggio, Nianwen Xue, Seokhwan Kim, Younggyun Hahm, Zhong He, Tony Kyungil Lee, Enrico Santus, Francis Bond, and Seung-Hoon Na (Eds.). International Committee on Computational Linguistics, Gyeongju, Republic of Korea, 3530–3538. https://aclanthology.org/2022.coling-1.311

[33] Lisha Li, Kevin Jamieson, Giulia DeSalvo, Afshin Rostamizadeh, and Ameet Talwalkar. 2018. Hyperband: A novel bandit-based approach to hyperparameter optimization. *Journal of Machine Learning Research* 18, 185 (2018), 1–52.

[34] Liam Li, Kevin Jamieson, Afshin Rostamizadeh, Ekaterina Gonina, Jonathan Ben-tzur, Moritz Hardt, Benjamin Recht, and Ameet Talwalkar. 2020. A System for Massively Parallel Hyperparameter Tuning. In *Proceedings of Machine Learning and Systems*, I. Dhillon, D. Papailiopoulos, and V. Sze (Eds.), Vol. 2. 230–246. https://proceedings.mlsys.org/paper_files/paper/2020/file/a06f20b349c6cf09a6b171c71b88bbfc-Paper.pdf

[35] Tianyi Li, Gregorio Convertino, Wenbo Wang, Haley Most, Tristan Zajonc, and Yi-Hsun Tsai. 2018. HyperTuner: Visual Analytics for Hyperparameter Tuning by Professionals . In *2018 IEEE Workshop on Machine Learning from User Interaction for Visualization and Analytics (MLUI)*. IEEE Computer Society, Los Alamitos, CA, USA, 1–11. doi:10.1109/MLUI52768.2018.10075647

[36] Yang Li, Yu Shen, Jiawei Jiang, Jinyang Gao, Ce Zhang, and Bin Cui. 2020. MFES-HB: Efficient Hyperband with Multi-Fidelity Quality Measurements. In *AAAI Conference on Artificial Intelligence*. https://api.semanticscholar.org/CorpusID:227342118

[37] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C. Lawrence Zitnick. 2014. Microsoft COCO: Common Objects in Context. In *Computer Vision – ECCV 2014*, David Fleet, Tomas Pajdla, Bernt Schiele, and Tinne Tuytelaars (Eds.). Springer International Publishing, Cham, 740–755.

[38] Zhicheng Liu, Bernard Kerr, Mira Dontcheva, Justin Grover, Matthew Hoffman, and Alan Wilson. 2017. CoreFlow: Extracting and Visualizing Branching Patterns from Event Sequences. *Computer Graphics Forum* 36, 3 (2017), 527–538. doi:10.1111/cgf.13208 arXiv:https://onlinelibrary.wiley.com/doi/pdf/10.1111/cgf.13208

[39] Scott M. Lundberg and Su-In Lee. 2017. A unified approach to interpreting model predictions. In *Proceedings of the 31st International Conference on Neural Information Processing Systems* (Long Beach, California, USA) *(NIPS'17)*. Curran Associates Inc., Red Hook, NY, USA, 4768–4777.

[40] Jessica Magallanes, Tony Stone, Paul D Morris, Suzanne Mason, Steven Wood, and Maria-Cruz Villa-Uriol. 2021. Sequen-c: A multilevel overview of temporal event sequences. *IEEE Transactions on Visualization and Computer Graphics* 28, 1 (2021), 901–911.

[41] Microsoft. 2021. *Neural Network Intelligence*. https://github.com/microsoft/nni

[42] Megan Monroe, Rongjian Lan, Hanseung Lee, Catherine Plaisant, and Ben Shneiderman. 2013. Temporal event sequence simplification. *IEEE transactions on visualization and computer graphics* 19, 12 (2013), 2227–2236.

[43] Randal S Olson, Nathan Bartley, Ryan J Urbanowicz, and Jason H Moore. 2016. Evaluation of a tree-based pipeline optimization tool for automating data science. In *Proceedings of the genetic and evolutionary computation conference 2016*. 485–492.

[44] Heungseok Park, Yoonsoo Nam, Ji-Hoon Kim, and Jaegul Choo. 2020. HyperTendril: Visual analytics for user-driven hyperparameter optimization of deep neural networks. *IEEE Transactions on Visualization and Computer Graphics* 27, 2 (2020), 1407–1416.

[45] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. 2019. PyTorch: An Imperative Style, High-Performance Deep Learning Library. In *Advances in Neural Information Processing Systems 32*, H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché Buc, E. Fox, and R. Garnett (Eds.). Curran Associates, Inc., 8024–8035. http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf

[46] Catherine Plaisant, Brett Milash, Anne Rose, Seth Widoff, and Ben Shneiderman. 1996. LifeLines: visualizing personal histories. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (Vancouver, British Columbia, Canada) *(CHI '96)*. Association for Computing Machinery, New York, NY, USA, 221–227. doi:10.1145/238386.238493

[47] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, Gretchen Krueger, and Ilya Sutskever. 2021. Learning Transferable Visual Models From Natural Language Supervision. In *Proceedings of the 38th International Conference on Machine Learning (Proceedings of Machine Learning Research, Vol. 139)*, Marina Meila and Tong Zhang (Eds.). PMLR, 8748–8763. https://proceedings.mlr.press/v139/radford21a.html

[48] Sebastián Ramírez. 2018. *FastAPI*. https://github.com/tiangolo/fastapi

[49] Bobak Shahriari, Kevin Swersky, Ziyu Wang, Ryan P Adams, and Nando De Freitas. 2015. Taking the human out of the loop: A review of Bayesian optimization. *Proc. IEEE* 104, 1 (2015), 148–175.

[50] Radwa El Shawi, Mohamed Maher, and Sherif Sakr. 2019. Automated Machine Learning: State-of-The-Art and Open Challenges. *ArXiv* abs/1906.02287 (2019). https://api.semanticscholar.org/CorpusID:174802418

[51] Jasper Snoek, Hugo Larochelle, and Ryan P. Adams. 2012. Practical Bayesian optimization of machine learning algorithms. In *Proceedings of the 26th International Conference on Neural Information Processing Systems - Volume 2* (Lake Tahoe, Nevada) *(NIPS'12)*. Curran Associates Inc., Red Hook, NY, USA, 2951–2959.

[52] Jasper Snoek, Kevin Swersky, Rich Zemel, and Ryan Adams. 2014. Input Warping for Bayesian Optimization of Non-Stationary Functions. In *Proceedings of the 31st International Conference on Machine Learning (Proceedings of Machine Learning Research, Vol. 32)*, Eric P. Xing and Tony Jebara (Eds.). PMLR, Bejing, China, 1674–1682. https://proceedings.mlr.press/v32/snoek14.html

[53] Martin Thomson, Elio Damaggio, and Brian Raymor. 2016. *Generic Event Delivery Using HTTP Push*. Internet-Draft draft-ietf-webpush-protocol-12. Internet Engineering Task Force. https://datatracker.ietf.org/doc/draft-ietf-webpush-protocol/12/ Work in Progress.

[54] Jordan Walke. 2024. React - The library for web and native user interfaces. https://react.dev/. Last Accessed 1 April 2024.

[55] 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 *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems* (Glasgow, Scotland Uk) *(CHI '19)*. Association for Computing Machinery, New York, NY, USA, 1–12. doi:10.1145/3290605.3300911

[56] Taowei David Wang, Catherine Plaisant, Alexander J. Quinn, Roman Stanchak, Shawn Murphy, and Ben Shneiderman. 2008. Aligning temporal data by sentinel events: discovering patterns in electronic health records. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (Florence, Italy) *(CHI '08)*. Association for Computing Machinery, New York, NY, USA, 457–466. doi:10.1145/1357054.1357129

[57] Daniel Karl I. Weidele, Justin D. Weisz, Erick Oduor, Michael Muller, Josh Andres, Alexander Gray, and Dakuo Wang. 2020. AutoAIViz: opening the blackbox of automated artificial intelligence with conditional parallel coordinates. In *Proceedings of the 25th International Conference on Intelligent User Interfaces* (Cagliari, Italy) *(IUI '20)*. Association for Computing Machinery, New York, NY, USA, 308–312. doi:10.1145/3377325.3377538

[58] Krist Wongsuphasawat, John Alexis Guerra Gómez, Catherine Plaisant, Taowei David Wang, Meirav Taieb-Maimon, and Ben Shneiderman. 2011. LifeFlow: visualizing an overview of event sequences. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (Vancouver, BC, Canada) *(CHI '11)*. Association for Computing Machinery, New York, NY, USA, 1747–1756. doi:10.1145/1978942.1979196

[59] Li Yang and Abdallah Shami. 2020. On hyperparameter optimization of machine learning algorithms: Theory and practice. *Neurocomputing* 415 (2020), 295–316.

[60] Steven R. Young, Derek C. Rose, Thomas P. Karnowski, Seung-Hwan Lim, and Robert M. Patton. 2015. Optimizing deep learning hyper-parameters through an evolutionary algorithm. In *Proceedings of the Workshop on Machine Learning in High-Performance Computing Environments* (Austin, Texas) *(MLHPC '15)*. Association for Computing Machinery, New York, NY, USA, Article 4, 5 pages. doi:10.1145/2834892.2834896

[61] Jian Zhao, Zhicheng Liu, Mira Dontcheva, Aaron Hertzmann, and Alan Wilson. 2015. MatrixWave: Visual Comparison of Event Sequence Data. In *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems* (Seoul, Republic of Korea) *(CHI '15)*. Association for Computing Machinery, New York, NY, USA, 259–268. doi:10.1145/2702123.2702419

[62] Marc-André Zöller, Waldemar Titov, Thomas Schlegel, and Marco F Huber. 2023. XAutoML: A Visual Analytics Tool for Understanding and Validating Automated Machine Learning. *ACM Transactions on Interactive Intelligent Systems* 13, 4 (2023), 1–39.