Continual Human-in-the-Loop Optimization

Yi-Chi Liao
Department of Computer Science
ETH Zurich, Switzerland
yichi.liao@inf.ethz.ch

Paul Streli
Department of Computer Science
ETH Zurich, Switzerland
paul.streli@inf.ethz.ch

Zhipeng Li
Department of Computer Science
ETH Zurich, Switzerland
zhipeng.li@inf.ethz.ch

Christoph Gebhardt
Department of Computer Science
ETH Zurich, Switzerland
christoph.gebhardt@inf.ethz.ch

Christian Holz
Department of Computer Science
ETH Zurich, Switzerland
christian.holz@inf.ethz.ch

ABSTRACT

Optimal input settings vary across users due to differences in motor abilities and personal preferences, which are typically addressed by manual tuning or calibration. Although human-in-the-loop optimization has the potential to identify optimal settings during use, it is rarely applied due to its long optimization process. A more efficient approach would continually leverage data from previous users to accelerate optimization, exploiting shared traits while adapting to individual characteristics. We introduce the concept of Continual Human-in-the-Loop Optimization and a Bayesian optimization-based method that leverages a Bayesian-neural-network surrogate model to capture population-level characteristics while adapting to new users. We propose a generative replay strategy to mitigate catastrophic forgetting. We demonstrate our method by optimizing virtual reality keyboard parameters for text entry using direct touch, showing reduced adaptation times with a growing user base. Our method opens the door for next-generation personalized input systems that improve with accumulated experience.

CCS CONCEPTS

• **Computing methodologies → Machine learning**; • **Human-centered computing → Interaction techniques**.

KEYWORDS

Continual learning, lifelong learning, continual optimization, Bayesian optimization, human-in-the-loop optimization, meta-learning, mid-air keyboard, typing, virtual reality.

1 INTRODUCTION

Due to the diverse motor abilities, preferences, and behaviors among users, optimal settings for input interactions in virtual reality (VR) and augmented reality (AR) vary significantly between individuals [67, 74, 93]. Today's interactive systems either commonly overlook this variability, or rely on manual user adjustment and explicit calibration procedures that can result in inefficient interactions or increased setup time. Human-in-the-loop optimization (HiLO) presents an alternative approach that optimizes interactions based on the user's past performance with specific design parameters.

HiLO has been shown to be effective across a wide range of applications, including target selection [9, 45], text input [74], visual design [37, 38], wearable devices [32], game development [31], and animation rendering [7]. While computational optimizers help avoid exhaustive testing of every design option [73] and aim to identify optimal solutions with minimal user trials—thereby reducing the time users spend with suboptimal settings [32] — a significant barrier to HiLO's broader adoption remains its relatively low sample efficiency. Without informed priors, HiLO often relies on initial random searches to explore the problem space, still requiring numerous trials to converge to optimal solutions. Moreover, each user must start the optimization process from scratch [3]. For instance, Chan et al. [9] found that optimizing the transfer function for 3D selection can take 60 to 90 minutes per user.

While individual users may have distinct preferences and performance with different input settings, shared traits across the user population could be leveraged to improve the optimizer efficiency across users [44]. Ideally, optimization would become more sample-efficient as data from prior users accumulates, allowing subsequent users to benefit from earlier optimization experiences while maintaining sufficient flexibility to ensure the discovery of optimal solutions tailored to individual needs.

Thus, in this paper, we investigate the question: *Can an optimizer continually learn from prior user experiences to improve its efficiency over time?* Despite the potential, current computational methods do not support a continually improving optimization for HiLO. Moreover, the problem itself and the corresponding challenges have not been thoroughly formulated in the existing literature.

One related concept is meta-Bayesian optimization [13, 83, 88, 89], which combines meta-learning and Bayesian optimization. Liao et al. [44] demonstrated the use of meta-Bayesian optimization for online HiLO, where a batch of "prior users" needs to go through a full optimization process from scratch, enabling the optimizer to perform more efficiently with subsequent "end-users." However, this approach suffers from a key limitation: its computation time during deployment increases with the number of prior users, leading to scalability issues. Beyond a certain point, it becomes impractical,

as users would experience significant delays during adaptation. Additionally, this method assumes that "prior users" can dedicate sufficient time to undergo a thorough optimization, which may not be feasible in practice. Therefore, these methods are not suitable for direct transformation into a continual learning framework.

Another closely related concept is continual learning [2, 10, 85], where a model improves its predictive capabilities by accumulating knowledge across different tasks (in the context of HiLO, a task is optimizing for a specific user). A few recent works have emerged that address continual learning within the context of optimization [28, 70], but their target problems are limited to linear (1D) discrete bandit problems, where typical HiLO tackles interactions with continuous and multiple-dimensional parameter spaces. The unique challenges and the corresponding methods of continual learning for HiLO remain unaddressed and unexplored.

To fill in the research gap, we formulate the problem and concept of *Continual Human-in-the-Loop Optimization (CHiLO)* — a computational optimizer capable of continuously improving its efficiency and performance by leveraging accumulated experience from previous users (illustrated in Figure 1). We further identify the key technical challenges that are aligned with those in continual learning [85], including *scalability*, *catastrophic forgetting*, the *stability-plasticity dilemma*, and *model instability due to uneven data distribution* of observations. These challenges guide the design principles for building CHiLO methods. Finally, we propose our novel method, the *Population-Informed Continual Bayesian Optimization (ConBO)*. ConBO's core is a Bayesian Neural Network (BNN) [49] trained on data synthesized from individual models of previous users, each representing a unique set of user characteristics. This method facilitates more stable and robust continual optimization by progressively integrating population-level user experiences with each new user.

We validate the efficacy and generalizability of ConBO using a series of standard benchmarking optimization functions, demonstrating that ConBO enhances optimization efficiency as more user data accumulates. Finally, we apply ConBO to optimize mid-air keyboard configurations for text entry using direct touch in VR. Our evaluations show a significant improvement in both user performance and convergence time as the number of users increases, compared to optimizing from scratch for each individual user.

In summary, we contribute

(1) The concept of *Continual Human-in-the-Loop Optimization* (CHiLO). We formally define the problem, identify potential challenges, and provide design principles for future methods.
(2) Population-informed Continual Bayesian Optimization (ConBO), a first approach to CHiLO. ConBO trains a BNN-based optimization with data synthesized from prior users' models.
(3) A demonstration of ConBO's efficacy in a real-world HiLO problem. Here, we showcase significant performance improvements.

2 RELATED WORK

2.1 Human-in-the-Loop Bayesian Optimization

System settings enable the configuration of an interactive system's parameters to optimize user interaction. These parameters influence user performance on a given task, which can be measured

through user evaluations and described using appropriate metrics [34, 40, 46]. Optimizing interaction for a user is thus equivalent to determining the parameters that maximize the performance metrics of interest. Formally, we seek the optimal parameters $x^* = \text{argmax}_{x \in \mathcal{X}} f(x)$, where x is a design vector from the space of all possible parameters $\mathcal{X}$, dependent on the system's characteristics, and f represents the relationship between system settings and user performance.

Since it remains difficult to accurately model human behavior, f is generally unknown for most tasks, making optimization reliant on empirical evaluations of $f(x')$ for a given x'. Given the cost of real-world evaluations, minimizing the number of evaluations to find the global optimum is crucial. Human-in-the-loop optimization (HiLO) addresses this by using computational optimizers to select and evaluate promising design candidates [32, 44, 46, 90]. A common approach in HiLO is Bayesian optimization [17, 53, 73], which guides the next configuration x to evaluate without assuming a specific functional form of f. It employs two core components: a surrogate model, typically a Gaussian Process (GP), to extrapolate a belief about unobserved points based on prior evaluations, and an acquisition function to determine the usefulness of evaluating a given point next. Bayesian optimization has been successfully applied in various human-involved applications, such as tuning exoskeleton parameters [32, 33, 82], wearable devices [8, 21, 45], haptic interfaces [8, 45], input techniques [47], and design tools [7, 39, 52].

However, a key limitation of Bayesian optimization is that, when initializing the surrogate model without prior knowledge, it requires multiple exploratory trials to sufficiently sample the search space [86]. During this exploration phase, the algorithm may evaluate severely suboptimal system configurations, which impedes the user experience. This inefficiency has motivated methods that enhance Bayesian optimization by leveraging prior data through transfer learning and meta-learning.

2.2 Transfer and Meta-Learning for Human-in-the-Loop Optimization

Transfer learning involves pre-training a model with a task to enhance efficiency when deployed on an unseen but relevant task [87]. Meta-learning is a related concept, aiming to enable models to adapt quickly to new tasks by leveraging accumulated experience across multiple similar tasks [36, 72, 81, 84]. While these concepts have been extensively explored in deep learning, especially in applications like few-shot recognition and reinforcement learning [5, 11, 15, 55, 66, 69], their application to optimization problems has only recently emerged [3].

Bayesian optimization, with its data-driven surrogate model (typically GP), holds potential for extensions into meta-learning and transfer learning. Previous efforts have explored constructing a GP model that accumulates observations across tasks [4, 6, 80, 91]. However, the primary challenge with this approach is scalability, as the GP fitting grows cubically with the number of observations; i.e., $O(n^3)$ where n represents the number of observations across all tasks. Some methods have replaced components of Bayesian optimization with deep neural networks (e.g., deep kernels or deep acquisition functions) to more effectively leverage past experiences [24, 83, 88]. Another approach, the weighted-sum method,
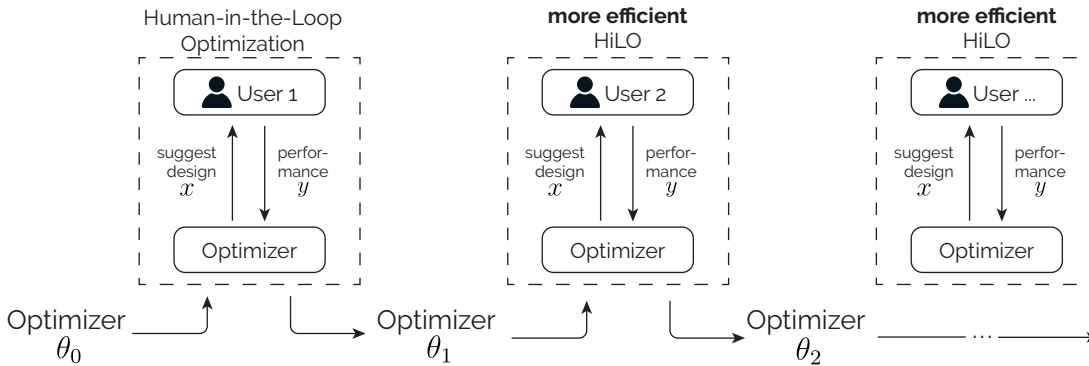
Figure 1: Illustration of *Continual Human-in-the-Loop Optimization (CHiLO)*: The optimizer, parameterized by θ, continuously evolves by accumulating experience from optimizations for different users. This enables more efficient adaptation to new users over time.

constructs individual GP models for each task or user and aggregates their acquisition functions based on weighted contributions. This method offers better scalability [42, 68, 89]. For instance, a recent work, TAF$^+$, is applied in wrist-based input techniques [44]. However, the online adaptation time of this approach increases linearly with the number of prior models (users), making it impractical for large-scale continual learning [83].

Overall, current transfer and meta-Bayesian optimization methods face significant limitations that prevent their direct application to our intended goal (CHiLO), where an optimizer has to handle a sequence of users and keep improving its performance and efficiency over time. For example, the single GP model approach is limited by computational scalability and struggles to encode or balance users with highly diverse characteristics. While some works have explored using BNNs as surrogate models [78], these approaches are typically meant for one single problem requiring large data points to tackle, not for handling sequential tasks (users) with similar but non-identical structures. Continual training of neural networks on such tasks often leads to issues with model instability and catastrophic forgetting [2, 85]. Furthermore, weighting-based methods increase adaptation time linearly to the previously seen tasks, which limits their practical use in continual learning scenarios. Lastly, approaches like Liao et al. [44] assume the availability of a large batch of "prior users" who can contribute to a long and thorough optimization process, which may not be feasible in practice. Therefore, there is a strong need for novel methods that extend Bayesian optimization into the realm of continual learning.

2.3 Continual Learning and its Application to Optimization

Inspired by the human ability to learn and improve through accumulating experiences across various tasks, continual learning (also known as life-long learning) is a machine learning paradigm where models are designed to learn and adapt continuously and incrementally from new data across different tasks [2, 85]. However, achieving this goal presents several significant challenges, including catastrophic forgetting, scalability, and maintaining model stability.

To address these challenges, various approaches have been developed. Regularization-based methods penalize substantial changes to critical parameters, enabling models to retain knowledge from previous tasks while learning new ones [35, 43]. Replay-based methods ensure that models revisit past data or use synthetically generated data from previous tasks to prevent the forgetting of previously acquired information [61, 75]. Parameter-isolation methods mitigate task interference by allocating different parts of the model's parameters to different tasks [12, 64]. Additionally, meta-learning-based methods focus on continually training meta-models that can quickly adapt to new tasks by leveraging the knowledge acquired from previous tasks [25, 62]. Continual learning has demonstrated successful applications in domains such as computer vision [61, 92], robotics [41, 54], natural language processing [22, 79], and reinforcement learning [63, 71].

Recent research has attempted to achieve continual learning in optimization tasks. One worth mentioning example is Schur et al. [70], which introduces a method called *Lifelong Bandit Optimization (LiBO)*. LiBO uses meta-learning to continually refine a meta-kernel function of a Gaussian Process, enabling the optimizer to accumulate experience from previous bandit problems. This method is based on meta-learning for kernels [23, 58]. However, LiBO primarily addresses linear discrete bandit problems, which are significantly simpler than multi-dimensional, continuous Human-in-the-Loop Optimization (HiLO) tasks. Moreover, a key component of LiBO is its phase of "forced exploration," making it unsuitable for our goal of minimizing random exploration for new users. While the lifelong bandit optimization problem is introduced in this work, its formulation and challenges are heavily focused on learning a kernel function. Therefore, we argue for the need to introduce the problem and challenges of Continual HiLO and to develop computational methods that extend beyond bandit problems, eliminating the requirement for forced exploration.

2.4 Input Personalization for AR/VR Interactions

Due to the high variance in users' motor capabilities [93], it is valuable to adapt input parameters to individual users. Traditionally, this personalization relies on explicit calibration processes, requiring users to perform specific tasks for the system to determine appropriate configurations before actual interaction begins. For example, users of wearable devices often need to perform predefined motions to allow the system to calibrate sensor ranges and adjust parameters accordingly [18, 19]. Similarly, gaze interactions typically involve calibration tasks where users are asked to track a series of targets [59].

More recently, input personalization has been framed as a HiLO problem. Bayesian optimization, for instance, has been applied to identify personalized transfer functions for target selection [9] and to adapt keyboard dimensions for gestural typing [74]. However, these methods treat optimization as a more principled calibration process that is completed before the actual interaction. Chan et al. [9], for example, requires users to undergo a 60-90 minute optimization session, while Shen et al. [74] utilizes a two-stage process where users first optimize and then manually select the final design. Therefore, these procedures are more suitable to be a pre-interaction step rather than a real-time adaptation.

We want to emphasize that a key difference between these works and our paper is that we are not merely applying Bayesian optimization to a specific task and investigating its performance; rather, we aim to develop a framework, CHiLO, that allows the optimizer to continually improve over time and across users. This enables faster, more efficient online adaptation as the system accumulates knowledge from previous users. A related work by Liao et al. [44] employs meta-Bayesian optimization to enhance online optimization efficiency. However, as discussed earlier, their weighting-based approach is not scalable for continual learning due to challenges like computational complexity and the assumption of having a group of prior users available for extensive optimization procedures.

3 CONTINUAL HUMAN-IN-THE-LOOP OPTIMIZATION

In this section, we formally define the concept of Continual Human-in-the-Loop Optimization (CHiLO), which serves as the foundation for the design principles underlying our method.

The goal of CHiLO is to optimize the parameter settings of an interactive system for a sequence of users, assuming that optimal parameters to maximize the performance metric of interest vary between individuals. As each user arrives and undergoes the human-in-the-loop optimization, there is access to observations from prior users to progressively improve the optimizer. The objective is to find the optimal solution for each user while minimizing the number of suboptimal trials.

3.1 Problem Statement

We aim to find the optimal parameters $x_u^* = \arg\max_{x \in \mathcal{X}} f_u(x)$ for each user $u \in \mathcal{U} = \{1, \cdots, N\}$, where $\mathcal{X}$ represents the continuous multi-dimensional input space, and $\mathcal{U}$ is the set of N users. The function $f_u : \mathcal{X} \rightarrow \mathbb{R}$ describes the user-dependent relationship between the input settings x and the corresponding user performance $f_u(x)$ according to a chosen performance metric. Since f_u is unknown and cannot be explicitly modeled, we treat it as a black-box function during optimization. For each user u, evaluations at a given input setting x yield an observed performance $y = f_u(x) + \epsilon$, where $\epsilon \sim \mathcal{N}(0, \sigma^2)$ represents independent and identically distributed (i.i.d.) Gaussian noise, capturing the stochastic variability in user behavior.

During optimization, the optimizer, parameterized by θ, selects a design vector x_u^t at each iteration t and observes the corresponding output y_u^t, based on the user's history of prior observations $\mathcal{D}_u^{t-1} = \{[x_u^0, y_u^0], [x_u^1, y_u^1], \ldots, [x_u^{t-1}, y_u^{t-1}]\}$. Additionally, in CHiLO, the optimizer also leverages observations from prior users $\{\mathcal{D}_i^T\}_{i=1}^{u-1}$, where $\mathcal{D}_i^T$ represents the history over T steps of optimization for previous users i.

Thus, the design vector at iteration t is selected as

$$x_u^t = g\left(\mathcal{D}_u^{t-1}, \{\mathcal{D}_i^T\}_{i=1}^{u-1} | \theta\right), \tag{1}$$

where g is the function guiding the selection process based on the current user's history and the data from prior users, with θ parameterizing the optimization strategy.

We aim to propose a method with an optimal selection function g and corresponding parameters θ such that, for each incoming user, the optimizer generates a sequence of solutions $\{x_u^t\}_{t=1}^T$ that minimize cumulative regret. The cumulative regret $R(N, T)$ for N users over T optimization iterations is defined as

$$R(N, T) = \sum_{u=1}^{N} \sum_{t=1}^{T} \left[f_u(x_u^*) - f_u(x_u^t)\right], \tag{2}$$

where $f_u(x_u^*)$ represents the optimal performance for user u, and $f_u(x_u^t)$ is the performance at iteration t.

The regret quantifies the cumulative difference between the optimal performance $f_u(x_u^*)$ and the performance at each iteration. Minimizing the regret requires the optimizer to efficiently estimate the user's expected performance and progressively approach the optimal solution by leveraging both the user's own observations and the knowledge gathered from previous users.

3.2 Challenges and Design Principles of CHiLO

CHiLO faces several challenges akin to those encountered in continual learning. Below, we outline the most significant challenges and propose corresponding design principles to address them.

The first challenge is **scalability across a large number of users**. For CHiLO to be effective, the optimizer must accumulate and leverage user experiences over time. As the amount of data grows, the optimizer must scale efficiently, both in encoding increasingly large datasets and maintaining computational efficiency when applied to new users. Many traditional optimization methods struggle with this challenge. For example, Bayesian optimization uses surrogate models like Gaussian Processes (GPs) to fit observed data. However, due to their high computational complexity, GPs often fail to scale well with large datasets.

The second challenge is **catastrophic forgetting**. Since CHiLO involves the sequential optimization of different users with unique characteristics, the optimizer may become overly specialized to the most recent users, leading to the loss of knowledge gained from

prior users. This phenomenon, known as catastrophic forgetting, can significantly hinder consistent performance across diverse user populations. For instance, in the case of a mid-air keyboard, some users may perform better on a smaller keyboard, while others prefer a larger one. If the optimizer adapts too strongly to one user group after consecutive optimizations, it risks forgetting how to optimize for previously encountered user types, leading to poor performance when those users reappear.

The third challenge is the **stability-plasticity dilemma**, which closely relates to the **exploration-exploitation dilemma**. This challenge involves balancing the use of prior knowledge (exploitation) with the need to adapt to new tasks (exploration). In CHiLO, new users may have characteristics that differ significantly from previous users. If the optimizer relies too heavily on prior data, it may fail to discover the global optimum for users with novel characteristics. On the other hand, focusing solely on new user data can lead to inefficient exploration, overfitting, or over-specialization, ultimately reducing the optimizer's ability to generalize across a diverse user population.

The fourth challenge is **model instability due to uneven distribution of observations**. During optimization, different users may explore different regions of the design parameter space. If certain regions are evaluated more frequently, the optimizer will accumulate more experience in those areas, leading to better predictions. However, this can result in neglecting other regions, causing the model to become unstable and unreliable when predicting in underexplored areas. This issue is especially problematic in large design spaces, where certain regions may be overlooked due to perceived lower importance or higher evaluation costs.

These challenges are interrelated. For example, poor scalability can lead to model instability as the optimizer struggles to handle large datasets, and catastrophic forgetting might occur when the optimizer focuses too narrowly on recent tasks, leading to an imbalanced distribution of observations.

In response to these challenges, we propose the following design principles for general CHiLO methods:

(1) **Efficient Scalability**: Methods for CHiLO must efficiently scale as data and the number of tasks increases. This requires the ability to manage large datasets and incorporate new data without significantly increasing computational complexity.

(2) **Knowledge Retention**: Optimizers must be capable of retaining knowledge from previous tasks while adapting to new ones, thus preventing catastrophic forgetting.

(3) **Mechanism for Balancing Stability and Plasticity**: The methods should include mechanisms that allow the optimizer to adapt to new users without disproportionately skewing the model towards highly specific user characteristics.

(4) **Robustness Across the Parameter Space**: The methods should maintain robustness and stability across the entire design parameter space, even when certain regions are underexplored or overexplored.

4 POPULATION-INFORMED CONTINUAL BAYESIAN OPTIMIZATION (CONBO)

In line with the design principles outlined above, we propose our *Population-Informed Continual Bayesian Optimization (ConBO)*, a

novel approach that leverages population-level user characteristics to enable more stable and robust continual optimization. Our method is built on the foundation of Bayesian optimization (BO), which is not only a widely used method for HiLO problems [73], but also a powerful tool for constructing surrogate models from observed data during the optimization process. This data-driven nature makes it well-suited for continual learning across users, as it allows for learning from data beyond just one user. In BO, the surrogate model, typically a Gaussian Process, estimates the mean $\mu(x)$ and variance $\sigma^2(x)$ of a Gaussian distribution over the output of the objective function for any given design decision x, $q(f(x)|x, \mathcal{D}) = \mathcal{N}(\mu(x), \sigma^2(x))$. The acquisition function $a(x)$ uses these predictions to compute acquisition values, which indicate the "worth" of each design x. The design with the highest acquisition value $x' = \arg\max_{x \in \mathcal{X}} a(x)$ is selected for user evaluation, resulting in a corresponding performance measure y'. This pair (x', y') is then used to update the surrogate model, refining its predictions in subsequent iterations. As more data points (x, y) are collected, the surrogate model becomes increasingly accurate, leading to better estimates of the true function $f(x)$, and thus guiding the optimization process more effectively.

Following the BO framework, ConBO relies on a surrogate model and employs a standard acquisition function. However, in ConBO, instead of being fitted solely to a specific user's data, the surrogate model encapsulates the accumulated knowledge from all prior users, resulting in a *meta-surrogate model* or *population model*. This population model facilitates more efficient optimization for new users by leveraging insights from previous user interactions, allowing for informed and effective decision-making.

Here, we outline the key components and workflow of ConBO.

4.1 Working Principles of ConBO

To enable **scalability**, we employ a *BNN* as the population model instead of the typical GP. The BNN is better suited for large-scale problems, as it can predict both the mean and variance of user performance for any design while supporting BO [77, 78]. Unlike GPs, which suffer from cubic computational complexity as data grows, BNNs can handle large datasets and can be explicitly optimized to provide predictions with varying uncertainty.

To ensure **knowledge retention** across users, we implement a *memory-replay mechanism*, where ConBO retrains the population model after each user by incorporating all accumulated data. Without this, the population model risks forgetting previous user data by adapting only to the current user's data, leading to instability.

Addressing the **stability-plasticity dilemma**, ConBO uses a mixture-model approach [13, 89]. The BNN-based population model and a GP model, trained on the current user's data, work together. Initially, the population model guides the optimization based on population-level knowledge, and as iterations progress, the GP model, tailored to the current user, takes on greater weight, allowing personalized optimization without losing the ability to generalize across users.

Ensuring **robustness across the entire design space** is essential to prevent the population model from overfitting to regions with more observations. This is achieved through a generative memory-replay mechanism, where grid-based predictions from previous

Meta-training using previous models

Adapting to a new user

previous model GP_1 $[\mu_{1,s}, \sigma_{1,s}^2]_{s\in\mathcal{S}}$

previous model GP_2 $[\mu_{2,s}, \sigma_{2,s}^2]_{s\in\mathcal{S}}$

previous model GP_3 $[\mu_{3,s}, \sigma_{3,s}^2]_{s\in\mathcal{S}}$

previous model GP_4 $[\mu_{4,s}, \sigma_{4,s}^2]_{s\in\mathcal{S}}$

variance filter $[\bar{\mu}_s, \bar{\sigma}_s^2]_{s\in\mathcal{S}}$

population model (BNN)

acquisition values a_m

next design x

current user u

performance y

user-specific model GP_u
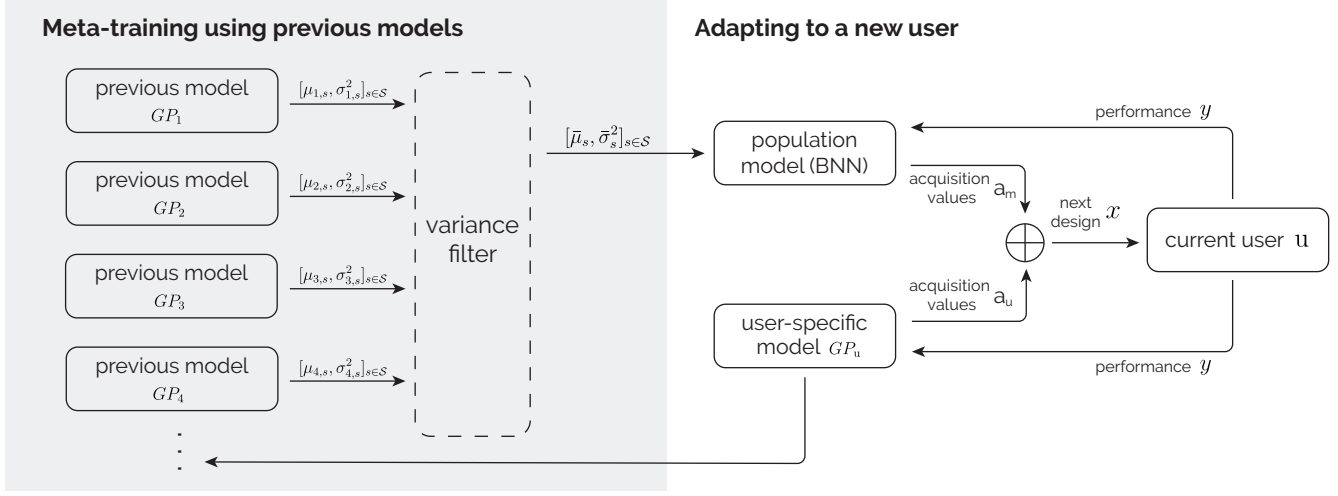
acquisition values a_u

performance y

Figure 2: Illustration of the key elements and workflow of ConBO: ConBO utilizes a population model (BNN) to continually learn the population-level user characteristics. It is further trained by previous user models ($GP_{1,2,3,4...}$). We sample points across the design space and query the predicted means and variances from all GPs (*meta-training*). ConBO filters out unreliable predictions, and trains the population model with the rest reliable predictions (*variance filter*). When deploying on a new user u, the population model and a user-specific model (GP_u) jointly guide the optimization process (*adapting to new users*). When an adaptation is complete, this user's model will be stored as a previous model.

users' models are selectively retained based on uncertainty levels. This prevents overfitting or underfitting to specific regions in the search space.

4.2 Key Components of ConBO

ConBO consists of these key components (see Figure 2): the population model, the current user-specific model (user-specific model), and a set of previously gathered models (previous models).

Population model: The population model is a BNN designed to capture and generalize population-level user characteristics, while adapting to individual users during deployment. Unlike a typical surrogate model that is trained on data from a single user, the population model in ConBO is trained on data synthesized from all prior users. Specifically, it receives grid-based predictions from previous models (described below), which include both the predicted mean and variance of performance over the design space. These predictions are used to retrain the population model, allowing it to generalize across users. The population model uses a multi-layer perceptron with dropout active during inference to capture epistemic uncertainty. During inference, multiple samples are drawn to estimate uncertainty. The model has two linear outputs: one for the mean and another for the log-variance, which accounts for aleatoric uncertainty. The total predictive variance is approximated as the sum of the mean output variances and the variance of predicted means, while the final mean estimate is the mean of predicted means [30].

During adaptation for a new user, the population model takes the design parameter settings as input, and estimates the average population mean and variances at given design points. This allows the population model to initially guide the optimization process by identifying the most promising design candidates for user

evaluation based on the acquisition function values (e.g., Expected Improvement) at the selected design points. Lastly, the population model adapts to the new user during online optimization with the observed user performance, allowing for more tailored suggestions.

Current user-specific model (user-specific model): For each new user currently undergoing the optimization process, ConBO constructs a GP model specifically fitted to the observations collected from that user. We refer to this model as the *current* user-specific model (i.e., user-specific model), to differentiate from *previous* user-specific models, which will be introduced in the next paragraph. While both the population model and the user-specific model adapt to the ongoing user's data, the user-specific model does not incorporate data from prior users. This allows it to generate acquisition values highly focused on the current user's characteristics. The acquisition values from the user-specific model are combined with those from the population model through a weighted-sum approach. Initially, the population model has a stronger influence due to its population-level knowledge. As more iterations occur and the user-specific model gains more data, the user-specific model's influence grows, allowing for a more personalized optimization process tailored to the current user. The use of mixed models to generate acquisition values is an established approach [13, 44, 89]. Additionally, we demonstrate the performance improvement of incorporating the user-specific model alongside the population model during adaptation, rather than relying solely on the population model, in our simulated experiments (see Appendix A).

Previous user-specific models (previous models): After each user completes the optimization process, their data is used to construct a GP model, which is stored in a library of previous surrogate models. These previous models generate grid-based predictions (mean and variance) across the entire design space, which are then

used to retrain the population model. By filtering out predictions with high variance (i.e., uncertainty), ConBO ensures that only reliable information is used for population model training.

4.3 Workflow of ConBO

ConBO operates in a sequential manner, optimizing for one new user at a time. Figure 2 gives an overview. Following the same structure, we summarize the workflow into two primary phases, **Adaptation to a New User** and **Meta-Training**, below.

1a. **Adaptation to a new user – proposing a design candidate:** For each new user, the population model (BNN) and the user-specific model (GP) jointly generate acquisition values over the design space. Initially, the population model is prioritized, leveraging population-level knowledge from previous users. As the user-specific model collects more data from the new user, its influence progressively increases. This dynamic weighting enables efficient adaptation to new users without drastically altering the population model. After the optimization process, the user-specific model is stored as part of the previous models. To manage the dynamic weighting, we follow prior work [44] to introduce two hyperparameters: α_1 and α_2 to govern the influence of both models during adaptation. For each surrogate model k, we compute its acquisition function a_k as the Expected Improvement (EI),

$$a_k(\boldsymbol{x}) = \mathrm{EI}_k(\boldsymbol{x}) = \int_{-\infty}^{\infty} \max(0, f(\boldsymbol{x}) - f(\boldsymbol{x}^+)) q_k(f(\boldsymbol{x}) \mid \boldsymbol{x}, \mathcal{D}) \, df(\boldsymbol{x}), \tag{3}$$

where $\boldsymbol{x}^+$ is the design vector corresponding to the best observed value $f(\boldsymbol{x}^+)$, and $q_k(f(\boldsymbol{x}) \mid \boldsymbol{x}, \mathcal{D})$ is the predictive distribution of surrogate model k, representing the current belief about user performance at $\boldsymbol{x}$.

The weight of the acquisition values generated by the population model at iteration t, denoted as $w_{m,t}$, is calculated as:

$$w_{m,t} = \begin{cases} 1, & \text{if } t \leq \alpha_1 \\ 1 - (t - \alpha_1)\alpha_2, & \text{if } \alpha_1 < t \leq \alpha_1 + \frac{1}{\alpha_2} \\ 0, & \text{otherwise.} \end{cases} \tag{4}$$

The weight for the acquisition values from the user-specific model, $w_{u,t}$, is the complement of $w_{m,t}$, computed as $w_{u,t} = 1 - w_{m,t}$. The final acquisition function for a given design candidate $\boldsymbol{x}$ at iteration t is a weighted sum of the EI values from the population model and the user-specific model, which can be denoted as:

$$a_t(\boldsymbol{x}) = w_{m,t} a_{m,t}(\boldsymbol{x}) + (1 - w_{m,t}) a_{u,t}(\boldsymbol{x}), \tag{5}$$

where $a_{m,t}$ is the acquisition value generated by the population model, and $a_{u,t}(\boldsymbol{x})$ denotes the acquisition value derived from the user-specific model. The design candidate with the highest acquisition value ($\boldsymbol{x}^*$) will then be selected to be evaluated by the user in this iteration; i.e., $\boldsymbol{x}_{\text{next}} = \mathrm{argmax}_{\boldsymbol{x} \in \mathcal{X}} a_t(\boldsymbol{x})$.

Note that when ConBO is first deployed, it lacks sufficient knowledge from prior users, requiring early participants to undergo a certain amount of random exploration. However, as the population model evolves and gathers more information, the need for random exploration diminishes. For later users, random exploration may no longer be necessary. In such cases, the user-specific model is initialized with a zero-mean function (i.e., all x values lead to

the same mean and variance predictions), and the optimization is fully governed by the population model until observations from the current user are collected (i.e., the iteration t exceeds α_1), at which point the GP model is updated. These users benefit from the accumulated knowledge of earlier participants, relying initially on the population model-guided optimization.

Two additional hyperparameters control the number of random explorations required for each new user: r_0 (the initial number of random explorations) and d_r (the decay rate of random exploration). The number of random explorations for the u-th participant (count from 1) is denoted as r_u, which is calculated as

$$r_u = \max(0, r_0 - (u - 1)d_r). \tag{6}$$

1b. **Adaptation to a new user – model adaptation:** During online adaptation, the population model keeps a record of all the observed design instances and their corresponding user performances (x, y). We use the Gaussian negative log-likelihood loss [56] for a probabilistic update of the predicted mean and variance at a design point x, based on the observed y from the user evaluation, after *each new observation*. This enables the population model to adapt to the new user's performance characteristics. Simultaneously, the user-specific model is updated with the same observed data (x, y), capturing the current user-specific uncertainty distribution over the design space. In later iterations, the user-specific model will primarily lead the acquisition function as its weights $(a_{u,t}(x))$ increases. Notably, the population model adaptation is *essential* for ConBO, particularly during the initial optimization iterations ($t \leq \alpha_1$), where the acquisition values are entirely based on the population model. Without updating the population model, it would repeatedly propose the same design $\boldsymbol{x}^*$ (as defined in Equation 4), further limiting the user-specific model's opportunity to learn about the user's preferences due to a lack of diverse observations in the design space. Once the adaptation process for a user is complete, the user-specific model is stored as a prior-user model. Then, it will contribute to future population model training.

2a. **Meta-training – mean and variance generation:** The population model is pre-trained using a generative replay mechanism. Sample points are generated across the design space, following both a predefined grid and randomly selected points. The prior-user models predict the mean and variance at these points, which form the dataset for training the population model.

2b. **Meta-training – data synthesis and variance filtering:** Predictions with variance exceeding a threshold λ are considered unreliable due to high uncertainty, often caused by different users exploring distinct regions of the design space. Unexplored areas naturally exhibit high variance, indicating this user model's lack of information in those design regions. To ensure the quality of the training data, predictions from unexplored regions with high uncertainty are filtered out. If no query from a specific prior-user model yields a variance below the threshold, it suggests that this particular user's performance is highly noisy and unstable across the entire design space or simply lacks enough observations. In such cases, predictions from this prior-user model are excluded from population model training. In our later user study, we set the threshold as half of the objective value range (i.e., $\lambda = 5$ for a normalized range of $[-5, 5]$). Importantly, we did not observe any

scenario where a prior-user model's predictions were completely excluded at any point in our user study, indicating the threshold was reasonable.

2c. **Meta-training – continual update:** The population model for a user u is retrained using the filtered dataset. The training loss consists of two terms, one for the predicted mean and one for the predicted variance. The loss function is defined as:

$$\mathcal{L}_u = \frac{1}{|\mathcal{S}|} \sum_{s \in \mathcal{S}} \left[\left(\hat{\mu}_s - \frac{1}{|\mathcal{U}_u^s|} \sum_{i \in \mathcal{U}_u^s} \mu_{i,s} \right)^2 + \left(\hat{\sigma}_s^2 - \frac{1}{|\mathcal{U}_u^s|} \sum_{i \in \mathcal{U}_u^s} \sigma_{i,s}^2 \right)^2 \right] \tag{7}$$

where $\mathcal{S}$ is the set of samples used for pretraining the BNN population model, $\mathcal{U}_u^s$ is the set of prior users for user u with sufficient confidence about sample s, $\hat{\mu}_s$ and $\hat{\sigma}_s^2$ are the predicted mean and variance for sample s from the BNN, and $\mu_{i,s}$ and $\sigma_{i,s}^2$ are the target mean and variance according to the user-specific model (GP) of prior user i for sample s. When encountering a new user, this process repeats, initiating a new cycle in the workflow. The detailed of the workflow is presented in Algorithm 1.

Summary: ConBO leverages a BNN as a population model to capture population-level knowledge. While the population model adapts to the new user with the online observations, its acquisition values are combined with insights from a user-specific GP to further provide user-specific suggestions. ConBO adapts to new users while minimizing the need for random exploration. Memory replay and generative data synthesis prevent catastrophic forgetting, ensuring scalability, knowledge retention, and robustness, all in alignment with the core design principles of continual optimization.

4.4 Simulated Tests with Benchmark Functions

We evaluated ConBO's effectiveness and generalizability in simulated tests using standard benchmark functions (e.g., Branin and McCormick). These simulated tests, detailed in Appendix A, explored various design choices for achieving CHiLO, including a range of both GP-based and BNN-based implementations. Additionally, the tests included adaptations of existing methods to enable CHiLO, variations in replay mechanisms, the inclusion or exclusion of the user-specific model, and the use or omission of a variance filter. A complete list of the compared alternatives is provided in subsection A.1. The results demonstrate the advantages of ConBO, such as using a BNN as the population model, incorporating the user-specific model, our memory-replay mechanism, and a variance filter. ConBO outperforms alternatives in adaptation accuracy while maintaining consistent online computational costs.

Our simulated tests highlight that GP-based methods exhibit increasing computational time as the number of observations or users grows. Methods that aggregate predictions across multiple GPs, such as TAF [44], experience linearly increasing computation times with the number of users; that is, $\mathbf{O}(m)$, where m indicates the number of previous models. Our experiments showed that TAF's computation time surpasses that of ConBO from the 4th user. Single-GP methods, on the other hand, suffer from cubically increasing computation time with the total number of observations; that is, $\mathbf{O}(n^3)$, where n indicates the number of total observations. For example, our synthetic results demonstrate that the computation time

of single-GP methods exceeds ConBO's after approximately 350 evaluations – a number easily surpassed when adapting across multiple users. Detailed illustrations of computation costs are provided in Figure 4d. These findings emphasize ConBO's effectiveness in achieving continual human-in-the-loop optimization with superior computational scalability.

Finally, in subsection A.8, we showcase that ConBO's memory-replay mechanism effectively retains the experience of previously seen tasks, leading to better performance when re-optimizing the seen synthetic users. In contrast, without any memory-replay mechanism, the population model struggles to perform well when re-optimizing for older tasks.

5 USER STUDY: MID-AIR KEYBOARD PERSONALIZATION

We selected text input on a mid-air keyboard in VR, using direct touch with index fingers, as a representative task to showcase the benefits of ConBO with real-world users. Text entry is a fundamental input task with well-established performance metrics, and VR offers a unique opportunity for interface personalization. Optimal keyboard size and placement varies between participants due to differences in prior experience, body size, motor abilities, and age [74]. Yet, virtual keyboards are typically displayed in a predetermined, standard size. If customization is available, it often relies on manual adjustments by the user, which, in cases of unfamiliarity with the modality, can lead to suboptimal configurations. Thus, ConBO offers a unique opportunity for automatic keyboard personalization and resulting performance improvements.

5.1 Optimization Task

Our objective is to maximize typing performance in terms of net words per minute (net WPM) [20, 65], defined as:

$$\text{Net WPM} = \frac{C/5 - E}{T/60}, \tag{8}$$

where E is the number of character errors between the target and the typed phrase, according to the Levenshtein distance [1, 50], C is the number of typed characters, and T is the text entry duration in seconds. Based on a pilot study with three participants, we determined that a reasonable range for typing performance is between 5 and 20 net WPM. This range was then linearly normalized to an objective value of $[-5, 5]$ during the optimization process.

For keyboard configuration, we optimized both the placement (in terms of distance along the user's sagittal axis) and the scaling of the keyboard size (keeping proportions constant). The ranges for these parameters, as shown in Table 1, were determined based on a pilot test with three participants. These parameters were normalized to a $[0, 1]$ range during the optimization process.

5.2 Experiment Setup

We compared our method, ConBO to standard Bayesian optimization (Standard BO) and manual user adjustment (Manual) as baseline adaptation conditions. A within-subject study was conducted, with conditions counterbalanced using a Latin-square design. Each condition was evaluated with 10 phrases, and an optimization iteration was performed after the completion of each phrase. Typing

---

Algorithm 1 ConBO: Population-Informed Continual Bayesian Optimization

---

Require: Users $\mathcal{U}$, design space $\mathcal{X}$, population model q_m (Bayesian Neural Network), user-specific models q_u (Gaussian Process, including both the previous and the current user-specific models) , number of iterations T, variance threshold λ, acquisition function parameters α_1 and α_2, random sampling parameters r_0 and d_r

Ensure: Optimized design choice for each user $u \in \mathcal{U}$

1: Initialize population model q_m
2: Initialize user-specific model q_u for each $u \in \mathcal{U}$
3: **for** each user $u \in \mathcal{U}$ **do**
4:     Determine the number of random samples r_u based on Equation 6
5:     Sample r_u initial design points from the design space $\mathcal{X}$, these sample design points are denoted as $\{x_u^0, \ldots, x_u^{r_u}\}$
6:     Perform user evaluation f_u at $\{x_u^0, \ldots, x_u^{r_u}\}$ to obtain $\{y_u^0, \ldots, y_u^{r_u}\}$
7:     **If** $r_u > 0$, train user-specific model q_u with $\{[x_u^0, y_u^0], \ldots, [x_u^{r_u}, y_u^{r_u}]\}$; **Else**, initialize the user-specific model q_u with a standard zero-mean function.
8:     **for** $t = r_u + 1$ to T **do**
9:         Sample N_{aq} points $\tilde{x}$ over the design space $\mathcal{X}$ to form set $\tilde{\mathcal{X}}$
10:         **for** each point $\tilde{x} \in \tilde{\mathcal{X}}$ **do**
11:             **If** $t \leq \alpha_1 + \frac{1}{\alpha_2}$, compute acquisition values $a_{m,t}(\tilde{x}_t)$ using the population model q_m; **Else**, set $a_{m,t} = 0$
12:             **If** $t > \alpha_1$, compute acquisition values $a_{u,t}(\tilde{x}_t)$ using the user-specific model q_u; **Else**, set $a_{u,t} = 0$
13:             Compute the weight $w_{m,t}$ via Equation 4
14:             Compute $a_t(\tilde{x}_t) = w_{m,t} a_{m,t}(\tilde{x}) + (1 - w_{m,t}) a_{u,t}(\tilde{x})$
15:         **end for**
16:         Select next design point $x_t = \arg\max_{\tilde{x} \in \tilde{\mathcal{X}}} a_t(\tilde{x})$
17:         Perform user evaluation for $f_u(x_t)$ to obtain y_t
18:         Update the current user-specific model q_u with (x_t, y_t)
19:         Update the population model q_m with (x_t, y_t) using Gaussian negative log-likelihood loss
20:     **end for**
21:     Store q_u in memory
22:     Sample points x_s across $\mathcal{X}$ to form set $\mathcal{S}$
23:     **for** each $s \in \mathcal{S}$ **do**
24:         **for** each $\tilde{u} \in \{\tilde{u} \in \mathcal{U} \mid \tilde{u} \leq u\}$ **do**
25:             Estimate mean $\mu_{\tilde{u},s}$ and variance $\sigma^2_{\tilde{u},s}$ according to $q_{\tilde{u}}(f(x_s)|x_s, \mathcal{D}_{\tilde{u}})$
26:         **end for**
27:         Apply variance filtering to obtain set of samples with sufficient confidence: $\mathcal{U}_{\tilde{u}}^s = \{\tilde{u} \in \mathcal{U} \mid \tilde{u} \leq u \wedge \sigma^2_{\tilde{u},s} < \lambda\}$
28:     **end for**
29:     Re-train population model q_m with filtered dataset $\mathcal{D}_{\text{filt},u+1} = \{(\mu_{\tilde{u},s}, \sigma^2_{\tilde{u},s}) \mid \tilde{u} \in \mathcal{U}_{\tilde{u}}^s\}_{\tilde{u} \leq u}^{s \in \mathcal{S}}$ via Equation 7
30: **end for**
31: **return** Optimized design points for all users $u \in \mathcal{U}$

Table 1: Adjustable parameters for the keyboard optimization.

Design Parameter		Range
x_0:	distance from the user along the sagittal axis	$[25, 65]$ cm
x_1:	scaling of the virtual keyboard (width $\times$ height)	$[39 \times 10.5, 90 \times 31.5]$ cm

performance, measured in net words per minute (net WPM), was recorded after each phrase as the performance metric (objective function) for optimization.

5.2.1 Participants, Task, and Procedure. We recruited a total of 12 participants (2 females). Their average age is 27.75 years, ranging from 24 to 38. The participants' heights range from 160 to 195 cm. For further details regarding their eyesight, heights, and arm lengths, please refer to Appendix subsection B.1. In each iteration, participants were asked to transcribe a target phrase using a mid-air keyboard, optimized for size and distance along the sagittal axis.

Seated in a stationary chair, participants were instructed to type the phrase by using their index finger to poke each character individually on the mid-air keyboard. They were instructed to type "*as quickly and as accurately as possible*" with no option for corrections.

Each participant completed 30 iterations (phrases) throughout the study, corresponding to 10 sentences per adaptation procedure. The target phrases were randomly selected from a set of 184 phrases, drawn from the MacKenzie & Soukoreff phrase set [51] and constrained to a length of 28 to 32 characters.

For ConBO and Standard BO the keyboard size and distance were dynamically adjusted by the respective optimization method.

In the manual adjustment procedure (Manual), participants had the opportunity to instruct the experimenter to adjust the keyboard's distance to their preference before starting. They could test the keyboard settings by typing and making as many adjustments as needed until they were satisfied. Once a configuration was chosen, it remained fixed for the entire duration of that procedure's 10 iterations. Before the experiment began, participants completed a training session in which they typed 10 phrases, with keyboard sizes and distances randomly varying after each iteration. This allowed participants to familiarize themselves with mid-air typing in VR and the study interface.

5.2.2 Interface and Apparatus. The study interface was implemented in Unity 2022.3.40 and run on a Windows 10 desktop computer. The user interface consisted of three main elements: a virtual canvas displaying the target sentence, a text field for the user's input, and a mid-air keyboard. Participants wore a Meta Oculus Pro headset, with the Meta virtual keyboard[1] used for text input. The keyboard was positioned directly in front of the user, angled at 30° below the horizon. Its distance from the user and size was dynamically adjusted by the respective optimization methods, with the scaling factor applied uniformly across all dimensions of the keyboard. A virtual canvas displaying the target sentence was placed 20° above the horizon, directly in front of the user's viewpoint. For hand-tracking, we employed the Meta hand-tracking SDK[2] integrated into Unity.

5.3 Parameter Settings of Adaptive Keyboard Optimizers

5.3.1 ConBO configuration. In the following section, we will detail the implementation of ConBO's components: (1) the population model (BNN), (2) the current and previous user-specific models (GP), (3) the settings used for adapting to new users, and (4) the settings applied during population model training.

1. population model Configuration: Our *population model* is a Bayesian Neural Network consisting of three fully connected layers, each with 100 nodes in the hidden layers. The input dimension is 2, corresponding to the two design parameters (as described in Table 1), while the output represents the prediction of the objective function (net WPM) and the estimated aleatoric variance. Dropout layers, with a dropout rate of 0.1, are applied after the last hidden layer to introduce stochasticity and provide regularization. The Rectified Linear Unit (ReLU) is used as the activation function. From a 2-people pilot test, we set the training epoch as such: During population model training (Step 4 in the workflow), BNN is trained with 800 epochs. During online adaptation (Step 1 in the workflow), 20 epochs are run for each new observation (iteration).

2. Current and Previous User-specific Models Configuration: For the current and prior *user-specific models* in ConBO, we implemented Gaussian Process (GP) models, following the approaches by Liao et al. [44] and Chan et al. [9]. The GPs are single-task and use a Matérn 5/2 kernel, which models homoscedastic noise levels in the observed data[3]. This kernel offers a flexible approach for modeling smooth functions while accounting for varying length scales in the input space.

3. Adaptation to Unseen Users: When deploying ConBO for new users, we gradually reduce the number of random explorations as more participants are added. The initial number of random explorations, r_0, is set to 6, with a decay rate d_r of 2. Thus, the first participant experiences 6 random explorations followed by 4 optimization iterations. The second participant experiences 4 random explorations and 6 optimization iterations, and so on. Starting from the fourth participant, no random explorations are conducted; all suggested keyboard settings are generated based entirely on the combined acquisition function, following Expected Improvement. We merge the EI values from both the population model and the current user-specific model using the weighting factor from Equation 4. Specifically, when computing the acquisition values, we sampled a 40×40 grid evenly across the 2-dimensional design space (i.e., $N_{aq} = 40 \times 40$ in Algorithm 1). This fine resolution ensures better accuracy in identifying the optimal acquisition value, as this directly impacts the design point selected for optimization during interactions with unseen users. The hyperparameters $\alpha_1 = 5$ and $\alpha_2 = 0.2$ ensure that the influence of the current user's model begins to increase after the fifth iteration and then grows linearly.

4. Population model (BNN) Training: To train the population model, we first generate a grid of 20×20 points, evenly spaced across the entire design space, followed by 100 randomly sampled points. Note that BNNs generally produce smooth predictions, enabling effective interpolation between sampled points. The 20×20 grid balances computational efficiency with capturing general trends across the design space. The additional 100 randomly sampled points further diversify the training data, reducing the need for a finer grid. For each point, we query the predicted means and variances from all previous user-specific models. We set a variance threshold of $\lambda = 5$. Since the normalized objective range is 10, any variance exceeding 5 (half the objective range) is considered unreliable and is excluded from further consideration. This approach ensures that the population model's predictions remain robust and reliable for guiding future optimizations.

5.3.2 Standard BO configuration. For Standard BO we used a single-task GP with Matern 5/2 kernel. We configured the optimization with 10 restarts for the acquisition function, 1,024 restart candidates for the acquisition function optimization, and 512 Monte Carlo samples to approximate the acquisition function.

Based on insights from a pilot test with two participants, we determined that the first 6 iterations of the 10-iteration procedure would be random searches, followed by 4 iterations of optimization. As in ConBO, we used Expected Improvement as the acquisition function to guide the optimization process.

5.4 Results

To evaluate optimization performance, we analyze the maximum typing performance (net WPM) achieved up to each iteration for each user between conditions. The typing speed results are shown in Figure 3a. We conducted a two-way repeated measures ANOVA to examine the effects of the *adaptation procedures* and *iteration* on performance (Net WPM, Condition × Iteration).

---

[1]https://developer.oculus.com/blog/virtual-keyboard-meta-quest-developers/
[2]https://developer.oculus.com/documentation/unity/unity-handtracking-overview/
[3]For more details, see https://botorch.org/docs/models.

The analysis revealed no significant interaction effect between condition and iteration, $F(18, 198) = 0.57$, $p = 0.561$, indicating that the influence of condition on performance did not significantly change across iterations. However, there was a significant main effect of condition, $F(2, 22) = 3.94$, $p = 0.039$, indicating performance differences across conditions. Additionally, we found a highly significant main effect of iteration, $F(9, 99) = 1487.13$, $p < 0.001$, suggesting that performance significantly improved throughout the iterations. To further explore the significant main effect of the procedures, pairwise comparisons with *Bonferroni correction* were performed. A significant difference was found between ConBO and Standard BO $t(11) = 2.97$, $p = 0.038$, indicating that ConBO outperforms Standard BO with a moderate effect size (Hedges' g = 0.733). No significant differences were observed between ConBO and Manual ($t(11) = 1.04$, $p = 0.965$) or between Standard BO and Manual ($t(11) = -1.53$, $p = 0.460$). Building on the overall performance differences between procedures, we conducted one-way repeated measures ANOVA for each iteration. Significant differences were found at iterations 2, 3, 4, and 5, where ConBO significantly outperformed Standard BO.

These results demonstrate that ConBO outperforms Standard BO in terms of maximum typing performance. Alternatively, we can assess the differences in regret throughout the optimization process (Equation 2) across iterations and participants. Regret quantifies the difference between the achieved performance and the user's optimal achievable performance. For each user, we use the highest observed performance across all procedures and iterations as a proxy for their optimal performance. Essentially, regret reflects the efficiency of the adaptation method, capturing the extent to which users spent in suboptimal configurations.

We visualize the regret for all conditions at each iteration in Figure 3b. A two-way repeated measures ANOVA was conducted to examine the effects of *adaptation procedures* and *iteration* on regret values. We did not find a significant interaction between condition and iteration (CONDITION × ITERATION), $F(18, 198) = 0.91$, $p = 0.561$. However, there was a significant main effect of condition, $F(2, 22) = 3.76$, $p = 0.039$, indicating that regret differed across conditions. Additionally, there was a highly significant main effect of iteration, $F(9, 99) = 46.12$, $p < 0.001$, suggesting that regret significantly changed over time. Pairwise comparisons using *Bonferroni correction* revealed significant differences between ConBO and Standard BO, $t(11) = 2.97$, $p = 0.038$, with a moderate effect size (Hedges' g = 0.995). No significant difference was found between Standard BO and Manual, $t(11) = 1.53$, $p = 0.460$, or between ConBO and Manual, $t(11) = -1.04$, $p = 0.965$.

We further visualize the overall regret values for each condition in Figure 3c, highlighting the differences across procedures. An additional repeated measures ANOVA was conducted and showed a significant difference between conditions with $F(2, 22) = 3.76$, $p = 0.039$. Post-hoc pairwise comparison with Bonferroni correction showed that a significant difference was found between ConBO and Standard BO ($t(11) = 2.97$, $p = 0.038$). To summarize all the users and all iterations (as per Equation 2), ConBO, Standard BO, and Manual have total regrets of 342.79, 634.41, and 449.06 net WPMs, respectively. Thus, based on both net WPM and regret, we conclude that ConBO outperforms Standard BO.

We analyze whether ConBO enables progressively more efficient adaptation as more users are introduced. To investigate this, we divided the users into four groups based on the order in which they participated (Figure 3d): Group 1 consists of participants 1 to 3, Group 2 includes participants 4 to 6, Group 3 includes participants 7 to 9, and Group 4 comprises participants 10 to 12. We examine the observed regrets within each group using a one-way ANOVA. The results showed a significant difference across the four groups, $F(3, 40) = 4.75$, $p < 0.004$. To further explore the differences between groups, we conducted post-hoc Tukey's HSD tests. There was a significant difference between Group 1 and Group 3, with Group 1 showing significantly higher regret than Group 3 (mean difference = −2.17, $p = 0.006$, 95% CI [−3.85, −0.48]). Similarly, a significant difference was found between Group 1 and Group 4, with Group 1 displaying significantly higher regret than Group 4 (mean difference = −2.03, $p = 0.011$, 95% CI [−3.72, −0.34]). As comparison baselines, we also plot the regret values when using Standard BO and Manual grouped every 3 participants in Figure 3d and Figure 3e.

Finally, Appendix B presents additional analyses. Appendix subsection B.1 provides detailed information about the participants, while Appendix subsection B.2 highlights the high diversity of optimal keyboard settings across users generated from various conditions, demonstrating ConBO's effectiveness in producing customized settings. Appendix subsection B.3 analyzes the progressive evolution of the population model as it adapts across users.

5.5 Findings and Discussion

We found that ConBO enables faster typing speeds with lower regret values compared to Standard BO proving it to be a more efficient keyboard adaptation method that benefits from the continually adapting population model. In the first five iterations, Standard BO performs worse, as expected, due to the need for random searches to explore the problem space. ConBO improves more rapidly over these iterations, demonstrating its ability to adapt to individuals and extend optimization beyond the population model's initial suggestions.

Manual offers a seeming advantage by allowing participants to freely adjust keyboard settings before starting. On average, participants made 4.92 adjustments ($std = 1.85$) before confirming their preferred setup. Despite this, Manual did not outperform the optimization-based adaptation methods at any point, suggesting that user-driven adjustments based on preferences or intuition do not lead to better results. This highlights the value of computational approaches over manual calibration for optimizing input devices.

Further analyzing regret values across user groups using ConBO, the first group (users 1–3), which was initially subjected to random explorations, had the highest cumulative regret. On the other hand, later groups (users 7–9 and 10–12) outperformed the first, showing that ConBO benefits from prior user performance observations over time, leading to more accurate and efficient keyboard adaptations.

6 DISCUSSION

In this work, we introduce CHiLO, a novel concept for boosting optimization efficiency by continually accumulating information across users. Based on our summarized principles, we further propose
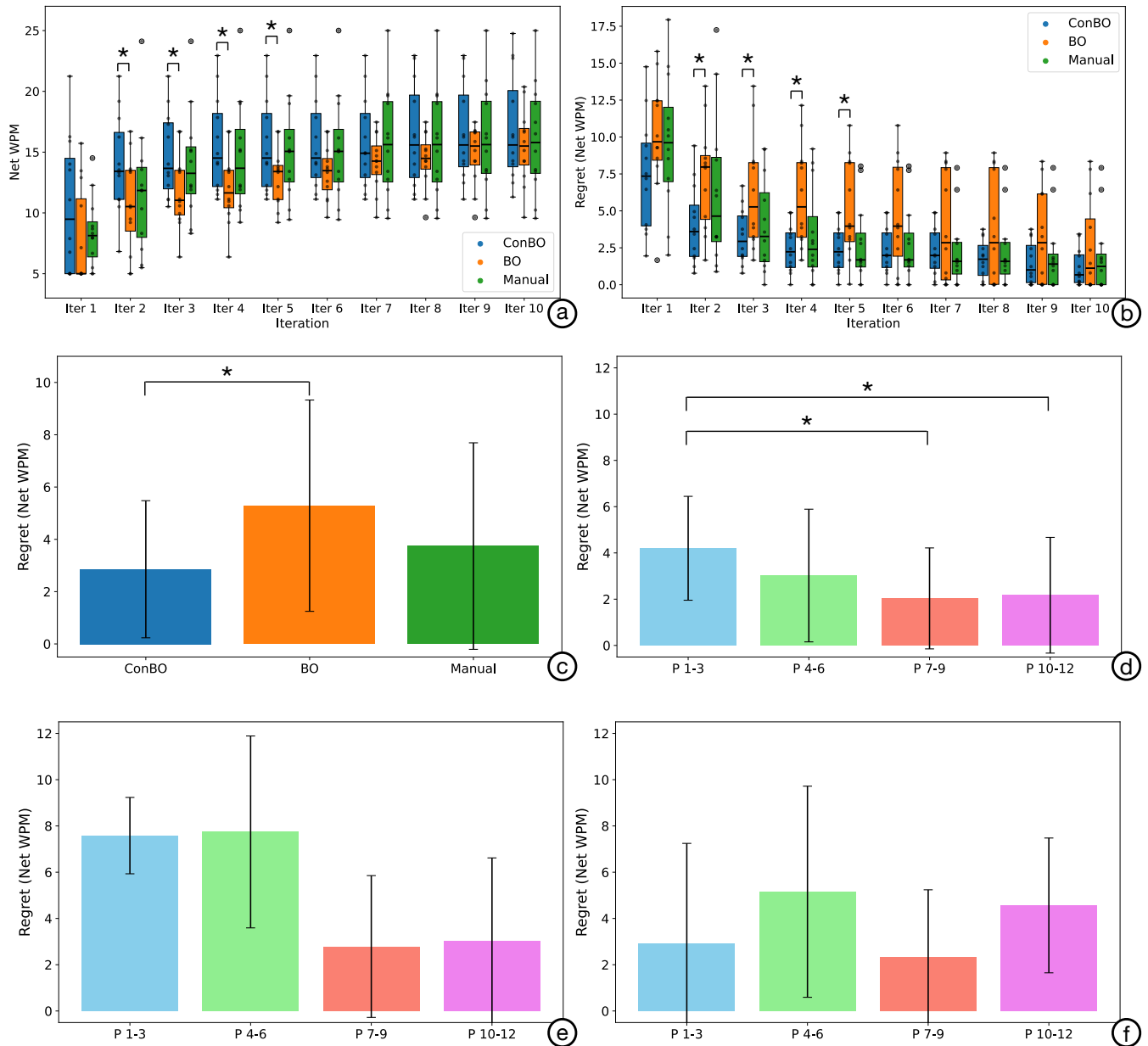
Figure 3: Results of our user study. (a) The net WPM at each iteration for three adaptation procedures; the ∗ sign indicates a significant difference found between the ConBO and Standard BO at that iteration. The scattered dots visualize each data point from individual participants. (b) The regret across all the iterations for all three procedures; the ∗ sign indicates a significant difference between the ConBO and Standard BO. The scattered dots visualize each data point from individual participants. (c) The average regret for all conditions; a significant difference found ConBO and Standard BO. (d) The mean regret values for each user group using ConBO; the ∗ sign indicates a significant difference between the groups. (e) The mean regret values for each user group using Standard BO. (f) The mean regret values for each user group using Manual.

ConBO, a Bayesian optimization-based method that continually updates a population model to capture population-level characteristics. To further enhance ConBO 's robustness, we incorporate a

memory-replay mechanism, which utilizes the previous users' models to train the population model. We evaluated ConBO through a user study focusing on a representative AR/VR input problem: mid-air typing on a virtual keyboard.

The results showed that ConBO significantly outperforms two representative baselines — standard BO and manual adjustment — demonstrating its ability to effectively accumulate user experience and adapt more efficiently across different users. ConBO demonstrated a clear trend of learning even within 10 users: the earlier users exhibited higher regret than later ones, indicating that ConBO continually improves over time. Also, when using ConBO, only the first three users encountered randomly searched keyboard configurations. From the fourth user onward, all the generated keyboards are guided by the population model. This is a clear distinction to the previous meta-Bayesian optimization methods which require a fixed set of prior users that is significantly larger to go through a full optimization procedure.

In summary, ConBO shows positive results in gradually adapting more efficiently with an increasing number of users. As Conbo is built upon BO, it does not impose strong assumptions or requirements for specific problems. We hope our results motivate researchers and practitioners to deploy ConBO for their applications. Despite these promising results, several open questions and limitations remain. There is potential for further optimizing the model's scalability, refining its memory management, and exploring broader applications of CHiLO beyond the AR/VR contexts.

6.1 Limitations and Future Work

We believe our work makes a novel contribution by introducing CHiLO for dynamic user interface and interaction personalization, which we hope will inspire future research and development in the following areas:

Addressing Individual Diversity and User Sequence Challenges. ConBO leverages a memory-replay mechanism to ensure that the population model is trained on data from all prior users, minimizing the risk of catastrophic forgetting. However, a potential limitation is that the population model may primarily capture the characteristics of the "general users," potentially leading to suboptimal performance for extreme users whose optimal settings significantly differ from the majority. This challenge is potentially severe in high-dimensional search spaces, where outliers could deviate even further from the average users. Future research should explore the use of more advanced models able to capture the diverse characteristics of different user groups. Additionally, the order in which users are encountered can influence the performance of ConBO. For instance, if the initial users represent average user behavior, the population model is more likely to transfer well to the next users. Conversely, if the first few users are outliers, the population model requires more iterations to adapt to the broader user population. This sequence effect is a known open challenge in continual learning [29, 48]. The extent of its impact depends on the nature of target interactions. Future research should empirically test the effects of user order in both synthetic tasks and real-world scenarios.

Re-adaptation to Changing User Characteristics. Our current implementation assumes that user characteristics and optimal performance remain constant. However, in practice, users' abilities may change due to factors like learning or fatigue, potentially requiring re-adaptation. ConBO could handle these fluctuations by treating significant performance changes as signals to restart the adaptation process. For example, if a user's performance deviates significantly from the expected trajectory, the system could treat them as a new user and begin a new optimization cycle. Future work should focus on real-time detection of these changes and developing effective re-adaptation strategies.

Development of Alternative CHiLO Methods. We see potential for further refinement of our proposed algorithm, as well as alternative implementations that may offer different trade-offs, making them more suitable for specific applications. For instance, our current approach requires access to all prior user models, which may not be feasible for memory-constrained devices or environments without cloud access. In such cases, a more size-efficient memory replay strategy could be necessary. A more refined approach could involve computing the importance of previous models and selectively using only the most relevant ones for population model training.

Also, our acquisition function is based on EI, a common option for BO. EI inherently balances exploration and exploitation, making it well-suited for general tasks. Future work could explore alternative acquisition functions, such as Probability of Improvement and Upper Confidence Bound, and empirically test their performances. Also, instead of the greedy approach of selecting the design point with the highest acquisition value, one could investigate converting acquisition values into log-probabilities and sampling from them, further promoting exploration. Our ConBO evaluates acquisition values by sampling over a fixed grid. Future work could explore more advanced optimization techniques, such as the DIRECT method [26] or gradient-based optimization with multiple restarts [76]. These methods should improve the accuracy of finding the optimal acquisition value.

Additionally, while we update the population model after each user, exploring different update frequencies could help balance computational costs and model accuracy. Beyond using a BNN as the meta-surrogate, prior work has investigated meta-kernel approaches for continual learning in bandit optimization problems. Extending these kernel-based methods to broader optimization tasks presents another promising avenue for future research.

Continual Optimization across Applications and Potential Parallelization. We introduce ConBO as a tool for improving adaptation by leveraging previous users' interactions with the same optimization task. However, we see the potential for ConBO to transfer knowledge not only across users but also across different applications, as many interactive systems share similar environments and parameter search spaces. While our study focused on mid-air typing, future work should explore ConBO's generalizability across various input methods, such as touch typing, gesture-based input, and gaze-based input. This would test ConBO's ability to learn shared characteristics across different interaction types and validate its broader applicability. Advanced Bayesian Neural Network architectures, such as multi-task BNNs, could further enhance ConBO's capacity to encode both task-specific and shared information, enabling it to generalize effectively across users and applications.

Our population model is trained on all previously collected GPs, opening the possibility of parallel computing. The population model can be periodically updated on a centralized server while being duplicated and deployed to local devices. These local copies can be optimized for specific users or interactions in parallel. The updated

GP models from local devices are then sent back to the server to jointly refine the population model, allowing for scalable and distributed optimization across multiple users and applications.

Expanding CHiLO for Complex Scenarios. Our ConBO merely takes an initial step toward CHiLO, focusing on relatively simple single-objective problems. However, real-world HiLO scenarios are often more complex and may involve multiple objective functions, such as balancing accuracy and efficiency (e.g., [9]) or trading off between recognition and information transfer (e.g., [45]). Addressing these challenges requires identifying the Pareto frontier within the objective space, a principled and widely used approach in multi-objective optimization tasks. Future work should explore extending CHiLO to handle multi-objective tasks by adapting the BNN to predict multiple objectives simultaneously and also updating the acquisition function to Expected Hypervolume Increase (EHVI), which is well-suited for identifying the multi-objective trade-offs.

Another common real-world scenario is where the objective functions cannot be explicitly measured, such as optimizing for user preferences. Preferential Bayesian Optimization (PBO) addresses this issue by eliciting user preferences over a set of choices [37, 38]. Extending CHiLO to preferential tasks is a direction worth exploring: A preferential population model could be developed using a framework similar to ConBO, but additional research is needed to investigate methods for suggesting multiple design choices that leverage both the population model and the current user-specific model. Furthermore, empirical studies are essential to assess the practical effectiveness and scalability of CHiLO in PBO scenarios, particularly in capturing and adapting to diverse user preferences.

Enhancing Real-World Adaptation Efficiency. Lastly, while ConBO improves the overall efficiency of keyboard adaptation, two areas remain for improvement: (1) the first few users need to undergo random explorations, and (2) each user is required to type 28–32 characters for every keyboard update. Future HCI research should investigate alternative approaches to reduce such user efforts. A potential direction is incorporating established user models [16, 57] to provide an initial estimate of general population performance. These models could replace or reduce the need for random exploration, enhancing the experience for the initial users. Additionally, requiring 28–32 characters per optimization iteration for accurate performance estimation may not always be necessary. In some cases, a rough estimate (e.g., 10–20 characters) could suffice. Future research could investigate multi-fidelity Bayesian Optimization [27, 60], which accommodates evaluations of varying accuracy levels. Low-fidelity evaluations could quickly discard suboptimal designs, while high-fidelity evaluations could refine promising candidates. Such advancements would make ConBO more efficient and user-friendly, extending its applicability beyond keyboard interaction and text input.

7 CONCLUSION

In this work, we address a novel challenge in Human in-the-loop Optimization: How can an optimizer continually accumulate experience and improve over time? The problem of continual learning for optimization, along with its technical challenges, had not been formally established in HCI, and through this work, we aim to bridge

this gap by introducing CHiLO. We provide a formal mathematical framework and outline key design principles for its implementation.

Building on these principles, we propose ConBO, a Bayesian Neural Network-based approach that captures population-level user characteristics through a population model and incorporates a generative memory-replay mechanism using stored models from previous users. Our evaluation of ConBO on mid-air text input in VR demonstrated significant improvement over standard Bayesian optimization in terms of adaptation efficiency and overall performance, while offering performance comparable to manual adjustments without requiring explicit user feedback.

The results further show a clear trend of improvement over time, with later user groups benefiting from the accumulated knowledge of the system. This illustrates ConBO's ability to progressively adapt more efficiently as it learns from prior users. We hope these positive results encourage further research into applying ConBO to a wider range of applications, extending beyond HCI.

We believe that ConBO can be a key enabler for truly adaptive and personalized interaction in the future, with systems improving across users, devices, and tasks—ultimately evolving to enable efficient and frictionless interaction for a wide range of applications.

8 OPEN SCIENCE

ConBO is available on our project page at *https://siplab.org/projects/Continual_Human-in-the-Loop_Optimization*. We hope that our implementation encourages the development of more advanced methods and can help expand the application scope of CHiLO.

REFERENCES

[1] Alexandr Andoni, Robert Krauthgamer, and Krzysztof Onak. 2010. Polylogarithmic approximation for edit distance and the asymmetric query complexity. In *2010 IEEE 51st Annual Symposium on Foundations of Computer Science*. IEEE, 377–386.

[2] Abhijeet Awasthi and Sunita Sarawagi. 2019. Continual learning with neural networks: A review. In *Proceedings of the ACM India Joint International Conference on Data Science and Management of Data*. 362–365.

[3] Tianyi Bai, Yang Li, Yu Shen, Xinyi Zhang, Wentao Zhang, and Bin Cui. 2023. Transfer Learning for Bayesian Optimization: A Survey. *arXiv preprint arXiv:2302.05927* (2023).

[4] Rémi Bardenet, Mátyás Brendel, Balázs Kégl, and Michèle Sebag. 2013. Collaborative Hyperparameter Tuning. In *Proceedings of the 30th International Conference on International Conference on Machine Learning - Volume 28* (Atlanta, GA, USA) *(ICML'13)*. JMLR.org, II–199–II–207.

[5] Yoshua Bengio, Samy Bengio, and Jocelyn Cloutier. 1990. *Learning a synaptic learning rule*. Citeseer.

[6] Edwin V Bonilla, Kian Chai, and Christopher Williams. 2007. Multi-task Gaussian process prediction. *Advances in neural information processing systems* 20 (2007).

[7] Eric Brochu, Tyson Brochu, and Nando de Freitas. 2010. A Bayesian Interactive Optimization Approach to Procedural Animation Design. In *Proceedings of the 2010 ACM SIGGRAPH/Eurographics Symposium on Computer Animation* (Madrid, Spain) *(SCA '10)*. Eurographics Association, Goslar, DEU, 103–112.

[8] Bilal Catkin and Volkan Patoglu. 2023. Preference-based human-in-the-loop optimization for perceived realism of haptic rendering. *IEEE Transactions on*

Haptics 16, 4 (2023), 470–476.

[9] Liwei Chan, Yi-Chi Liao, George B Mo, John J Dudley, Chun-Lien Cheng, Per Ola Kristensson, and Antti Oulasvirta. 2022. Investigating Positive and Negative Qualities of Human-in-the-Loop Optimization for Designing Interaction Techniques. In *Proceedings of the 2022 CHI Conference on Human Factors in Computing Systems* (New Orleans, LA, USA) *(CHI '22)*. Association for Computing Machinery, New York, NY, USA, Article 112, 14 pages. https://doi.org/10.1145/3491102.3501850

[10] Andrea Cossu, Antonio Carta, Vincenzo Lomonaco, and Davide Bacciu. 2021. Continual learning for recurrent neural networks: An empirical evaluation. *Neural Networks* 143 (2021), 607–627. https://doi.org/10.1016/j.neunet.2021.07.021

[11] Yan Duan, John Schulman, Xi Chen, Peter L Bartlett, Ilya Sutskever, and Pieter Abbeel. 2016. Rl2: Fast reinforcement learning via slow reinforcement learning. *arXiv preprint arXiv:1611.02779* (2016).

[12] Chrisantha Fernando, Dylan Banarse, Charles Blundell, Yori Zwols, David Ha, Andrei A Rusu, Alexander Pritzel, and Daan Wierstra. 2017. Pathnet: Evolution channels gradient descent in super neural networks. *arXiv preprint arXiv:1701.08734* (2017).

[13] Matthias Feurer, Benjamin Letham, and Eytan Bakshy. 2018. Scalable meta-learning for bayesian optimization using ranking-weighted gaussian process ensembles. In *AutoML Workshop at ICML*, Vol. 7.

[14] Matthias Feurer, Benjamin Letham, Frank Hutter, and Eytan Bakshy. 2018. Practical transfer learning for Bayesian optimization. *arXiv preprint arXiv:1802.02219* (2018).

[15] Chelsea Finn, Pieter Abbeel, and Sergey Levine. 2017. Model-agnostic meta-learning for fast adaptation of deep networks. In *International conference on machine learning*. PMLR, 1126–1135.

[16] Gerhard Fischer. 2001. User modeling in human–computer interaction. *User modeling and user-adapted interaction* 11 (2001), 65–86.

[17] Peter I Frazier. 2018. A tutorial on Bayesian optimization. *arXiv preprint arXiv:1807.02811* (2018).

[18] Jun Gong, Zheer Xu, Qifan Guo, Teddy Seyed, Xiang 'Anthony' Chen, Xiaojun Bi, and Xing-Dong Yang. 2018. WrisText: One-Handed Text Entry on Smartwatch Using Wrist Gestures. 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, 1–14. https://doi.org/10.1145/3173574.3173755

[19] Jun Gong, Xing-Dong Yang, and Pourang Irani. 2016. WristWhirl: One-Handed Continuous Smartwatch Input Using Wrist Gestures. In *Proceedings of the 29th Annual Symposium on User Interface Software and Technology* (Tokyo, Japan) *(UIST '16)*. Association for Computing Machinery, New York, NY, USA, 861–872. https://doi.org/10.1145/2984511.2984563

[20] Patrick Grady, Jeremy A Collins, Chengcheng Tang, Christopher D Twigg, Kunal Aneja, James Hays, and Charles C Kemp. 2024. PressureVision++: Estimating Fingertip Pressure from Diverse RGB Images. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*. 8698–8708.

[21] Junqi Guo, Lan Yang, Rongfang Bie, Jiguo Yu, Yuan Gao, Yan Shen, and Anton Kos. 2019. An XGBoost-based physical fitness evaluation model using advanced feature selection and Bayesian hyper-parameter optimization for wearable running monitoring. *Computer Networks* 151 (2019), 166–180.

[22] Suchin Gururangan, Ana Marasović, Swabha Swayamdipta, Kyle Lo, Iz Beltagy, Doug Downey, and Noah A Smith. 2020. Don't stop pretraining: Adapt language models to domains and tasks. *arXiv preprint arXiv:2004.10964* (2020).

[23] James Harrison, Apoorva Sharma, and Marco Pavone. 2020. Meta-learning priors for efficient online bayesian regression. In *Algorithmic Foundations of Robotics XIII: Proceedings of the 13th Workshop on the Algorithmic Foundations of Robotics 13*. Springer, 318–337.

[24] Bing-Jing Hsieh, Ping-Chun Hsieh, and Xi Liu. 2021. Reinforced few-shot acquisition function learning for bayesian optimization. *Advances in Neural Information Processing Systems* 34 (2021), 7718–7731.

[25] Khurram Javed and Martha White. 2019. Meta-learning representations for continual learning. *Advances in neural information processing systems* 32 (2019).

[26] Donald R Jones, Cary D Perttunen, and Bruce E Stuckman. 1993. Lipschitzian optimization without the Lipschitz constant. *Journal of optimization Theory and Applications* 79 (1993), 157–181.

[27] Kirthevasan Kandasamy, Gautam Dasarathy, Jeff Schneider, and Barnabás Póczos. 2017. Multi-fidelity bayesian optimisation with continuous approximations. In *International conference on machine learning*. PMLR, 1799–1808.

[28] Parnian Kassraie, Jonas Rothfuss, and Andreas Krause. 2022. Meta-learning hypothesis spaces for sequential decision-making. In *International Conference on Machine Learning*. PMLR, 10802–10824.

[29] Zixuan Ke, Bing Liu, and Xingchang Huang. 2020. Continual learning of a mixed sequence of similar and dissimilar tasks. *Advances in neural information processing systems* 33 (2020), 18493–18504.

[30] Alex Kendall and Yarin Gal. 2017. What uncertainties do we need in bayesian deep learning for computer vision? *Advances in neural information processing systems* 30 (2017).

[31] Mohammad M. Khajah, Brett D. Roads, Robert V. Lindsey, Yun-En Liu, and Michael C. Mozer. 2016. Designing Engaging Games Using Bayesian Optimization.

In *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems* (San Jose, California, USA) *(CHI '16)*. Association for Computing Machinery, New York, NY, USA, 5571–5582. https://doi.org/10.1145/2858036.2858253

[32] Myunghee Kim, Ye Ding, Philippe Malcolm, Jozefien Speeckaert, Christoper J. Siviy, Conor J. Walsh, and Scott Kuindersma. 2017. Human-in-the-loop Bayesian optimization of wearable device parameters. *PLOS ONE* 12, 9 (09 2017), 1–15. https://doi.org/10.1371/journal.pone.0184054

[33] Myunghee Kim, Charles Liu, Jinsoo Kim, Sangjun Lee, Adham Meguid, Conor J Walsh, and Scott Kuindersma. 2019. Bayesian optimization of soft exosuits using a metabolic estimator stopping process. In *2019 international conference on robotics and automation (icra)*. IEEE, 9173–9179.

[34] Sunjun Kim, Byungjoo Lee, Thomas van Gemert, and Antti Oulasvirta. 2020. Optimal Sensor Position for a Computer Mouse. In *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems* (Honolulu, HI, USA) *(CHI '20)*. Association for Computing Machinery, New York, NY, USA, 1–13. https://doi.org/10.1145/3313831.3376735

[35] James Kirkpatrick, Razvan Pascanu, Neil Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska, et al. 2017. Overcoming catastrophic forgetting in neural networks. *Proceedings of the national academy of sciences* 114, 13 (2017), 3521–3526.

[36] Alice Y Kolb and David A Kolb. 2009. The learning way: Meta-cognitive aspects of experiential learning. *Simulation & gaming* 40, 3 (2009), 297–327.

[37] Yuki Koyama and Masataka Goto. 2022. BO as Assistant: Using Bayesian Optimization for Asynchronously Generating Design Suggestions. In *Proceedings of the 35th Annual ACM Symposium on User Interface Software and Technology* (Bend, OR, USA) *(UIST '22)*. Association for Computing Machinery, New York, NY, USA, Article 77, 14 pages. https://doi.org/10.1145/3526113.3545664

[38] Yuki Koyama, Issei Sato, and Masataka Goto. 2020. Sequential Gallery for Interactive Visual Design Optimization. *ACM Trans. Graph.* 39, 4, Article 88 (aug 2020), 12 pages. https://doi.org/10.1145/3386569.3392444

[39] Yuki Koyama, Issei Sato, Daisuke Sakamoto, and Takeo Igarashi. 2017. Sequential line search for efficient visual design optimization by crowds. *ACM Transactions on Graphics (TOG)* 36, 4 (2017), 1–11.

[40] Byungjoo Lee, Mathieu Nancel, Sunjun Kim, and Antti Oulasvirta. 2020. Auto-Gain: Gain Function Adaptation with Submovement Efficiency Optimization. In *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems* (Honolulu, HI, USA) *(CHI '20)*. Association for Computing Machinery, New York, NY, USA, 1–12. https://doi.org/10.1145/3313831.3376244

[41] Timothée Lesort, Vincenzo Lomonaco, Andrei Stoian, Davide Maltoni, David Filliat, and Natalia Díaz-Rodríguez. 2020. Continual learning for robotics: Definition, framework, learning strategies, opportunities and challenges. *Information fusion* 58 (2020), 52–68.

[42] Yang Li, Yu Shen, Huaijun Jiang, Wentao Zhang, Zhi Yang, Ce Zhang, and Bin Cui. 2022. TransBO: Hyperparameter Optimization via Two-Phase Transfer Learning. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*. 956–966.

[43] Zhizhong Li and Derek Hoiem. 2017. Learning without forgetting. *IEEE transactions on pattern analysis and machine intelligence* 40, 12 (2017), 2935–2947.

[44] Yi-Chi Liao, Ruta Desai, Alec M Pierce, Krista E. Taylor, Hrvoje Benko, Tanya R. Jonker, and Aakar Gupta. 2024. A Meta-Bayesian Approach for Rapid Online Parametric Optimization for Wrist-based Interactions. In *Proceedings of the CHI Conference on Human Factors in Computing Systems* (Honolulu, HI, USA) *(CHI '24)*. Association for Computing Machinery, New York, NY, USA, Article 410, 38 pages. https://doi.org/10.1145/3613904.3642071

[45] Yi-Chi Liao, John J. Dudley, George B. Mo, Chun-Lien Cheng, Liwei Chan, Antti Oulasvirta, and Per Ola Kristensson. 2023. Interaction Design With Multi-objective Bayesian Optimization. *IEEE Pervasive Computing* (2023), 1–10. https://doi.org/10.1109/MPRV.2022.3230597

[46] Yi-Chi Liao, Sunjun Kim, Byungjoo Lee, and Antti Oulasvirta. 2020. Button Simulation and Design via FDVV Models. In *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems* (Honolulu, HI, USA) *(CHI '20)*. Association for Computing Machinery, New York, NY, USA, 1–14. https://doi.org/10.1145/3313831.3376262

[47] Yi-Chi Liao, George B Mo, John J Dudley, Chun-Lien Cheng, Liwei Chan, Per Ola Kristensson, and Antti Oulasvirta. 2024. Practical approaches to group-level multi-objective Bayesian optimization in interaction technique design. *Collective Intelligence* 3, 1 (2024), 26339137241241313. https://doi.org/10.1177/26339137241241313 arXiv:https://doi.org/10.1177/26339137241241313

[48] David Lopez-Paz and Marc'Aurelio Ranzato. 2017. Gradient episodic memory for continual learning. *Advances in neural information processing systems* 30 (2017).

[49] David JC MacKay. 1995. Bayesian neural networks and density networks. *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment* 354, 1 (1995), 73–80.

[50] I. Scott MacKenzie. 2015. A Note on Calculating Text Entry Speed. https://www.yorku.ca/mack/RN-TextEntrySpeed.html#:~:text=Words%20Per%20Minute&text=The%20transformation%20requires%20multiplying%20the,using%20actual%20words%20per%20minute.

[51] I Scott MacKenzie and R William Soukoreff. 2003. Phrase sets for evaluating text entry techniques. In *CHI'03 extended abstracts on Human factors in computing systems*. 754–755.

[52] George Mo, John Dudley, Liwei Chan, Yi-Chi Liao, Antti Oulasvirta, and Per Ola Kristensson. 2024. Cooperative Multi-Objective Bayesian Design Optimization. *ACM Trans. Interact. Intell. Syst.* 14, 2, Article 13 (jun 2024), 28 pages. https://doi.org/10.1145/3657643

[53] J. Mockus. 1975. On the Bayes Methods for Seeking the Extremal Point. *IFAC Proceedings Volumes* 8, 1, Part 1 (1975), 428–431. https://doi.org/10.1016/S1474-6670(17)67769-3 6th IFAC World Congress (IFAC 1975) - Part 1: Theory, Boston/Cambridge, MA, USA, August 24-30, 1975.

[54] Anusha Nagabandi, Chelsea Finn, and Sergey Levine. 2018. Deep online learning via meta-learning: Continual adaptation for model-based rl. *arXiv preprint arXiv:1812.07671* (2018).

[55] Alex Nichol and John Schulman. 2018. Reptile: a scalable metalearning algorithm. *arXiv preprint arXiv:1803.02999* 2, 3 (2018), 4.

[56] David A Nix and Andreas S Weigend. 1994. Estimating the mean and variance of the target probability distribution. In *Proceedings of 1994 ieee international conference on neural networks (ICNN'94)*, Vol. 1. IEEE, 55–60.

[57] Antti Oulasvirta. 2019. It's time to rediscover HCI models. *Interactions* 26, 4 (2019), 52–56.

[58] Valerio Perrone, Rodolphe Jenatton, Matthias W Seeger, and Cédric Archambeau. 2018. Scalable hyperparameter transfer learning. *Advances in neural information processing systems* 31 (2018).

[59] Ken Pfeuffer, Melodie Vidal, Jayson Turner, Andreas Bulling, and Hans Gellersen. 2013. Pursuit calibration: Making gaze calibration less tedious and more flexible. In *Proceedings of the 26th annual ACM symposium on User interface software and technology*. 261–270.

[60] Matthias Poloczek, Jialei Wang, and Peter Frazier. 2017. Multi-information source optimization. *Advances in neural information processing systems* 30 (2017).

[61] Sylvestre-Alvise Rebuffi, Alexander Kolesnikov, Georg Sperl, and Christoph H Lampert. 2017. icarl: Incremental classifier and representation learning. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*. 2001–2010.

[62] Matthew Riemer, Ignacio Cases, Robert Ajemian, Miao Liu, Irina Rish, Yuhai Tu, and Gerald Tesauro. 2018. Learning to learn without forgetting by maximizing transfer and minimizing interference. *arXiv preprint arXiv:1810.11910* (2018).

[63] David Rolnick, Arun Ahuja, Jonathan Schwarz, Timothy Lillicrap, and Gregory Wayne. 2019. Experience replay for continual learning. *Advances in neural information processing systems* 32 (2019).

[64] Andrei A Rusu, Neil C Rabinowitz, Guillaume Desjardins, Hubert Soyer, James Kirkpatrick, Koray Kavukcuoglu, Razvan Pascanu, and Raia Hadsell. 2016. Progressive neural networks. *arXiv preprint arXiv:1606.04671* (2016).

[65] Timothy A Salthouse. 1984. Effects of age and skill in typing. *Journal of Experimental Psychology: General* 113, 3 (1984), 345.

[66] Adam Santoro, Sergey Bartunov, Matthew Botvinick, Daan Wierstra, and Timothy Lillicrap. 2016. Meta-Learning with Memory-Augmented Neural Networks. In *Proceedings of The 33rd International Conference on Machine Learning (Proceedings of Machine Learning Research, Vol. 48)*, Maria Florina Balcan and Kilian Q. Weinberger (Eds.). PMLR, New York, New York, USA, 1842–1850. https://proceedings.mlr.press/v48/santoro16.html

[67] Marius Schenkluhn, Christian Peukert, Anke Greif-Winzrieth, and Christof Weinhardt. 2023. Does One Keyboard Fit All? Comparison and Evaluation of Device-Free Augmented Reality Keyboard Designs. In *Proceedings of the 29th ACM Symposium on Virtual Reality Software and Technology* (Christchurch, New Zealand) *(VRST '23)*. Association for Computing Machinery, New York, NY, USA, Article 5, 11 pages. https://doi.org/10.1145/3611659.3615692

[68] Nicolas Schilling, Martin Wistuba, and Lars Schmidt-Thieme. 2016. Scalable Hyperparameter Optimization with Products of Gaussian Process Experts. In *Machine Learning and Knowledge Discovery in Databases*, Paolo Frasconi, Niels Landwehr, Giuseppe Manco, and Jilles Vreeken (Eds.). Springer International Publishing, Cham, 33–48.

[69] Jürgen Schmidhuber. 1992. Learning to Control Fast-Weight Memories: An Alternative to Dynamic Recurrent Networks. *Neural Computation* 4, 1 (1992), 131–139. https://doi.org/10.1162/neco.1992.4.1.131

[70] Felix Schur, Parnian Kassraie, Jonas Rothfuss, and Andreas Krause. 2023. Lifelong bandit optimization: no prior and no regret. In *Uncertainty in Artificial Intelligence*. PMLR, 1847–1857.

[71] Jonathan Schwarz, Wojciech Czarnecki, Jelena Luketina, Agnieszka Grabska-Barwinska, Yee Whye Teh, Razvan Pascanu, and Raia Hadsell. 2018. Progress & compress: A scalable framework for continual learning. In *International conference on machine learning*. PMLR, 4528–4537.

[72] Nicolas Schweighofer and Kenji Doya. 2003. Meta-learning in Reinforcement Learning. *Neural Networks* 16, 1 (2003), 5–9. https://doi.org/10.1016/S0893-6080(02)00228-9

[73] Bobak Shahriari, Kevin Swersky, Ziyu Wang, Ryan P. Adams, and Nando de Freitas. 2016. Taking the Human Out of the Loop: A Review of Bayesian Optimization. *Proc. IEEE* 104, 1 (2016), 148–175. https://doi.org/10.1109/JPROC.2015.2494218

[74] Junxiao Shen, Jinghui Hu, John J. Dudley, and Per Ola Kristensson. 2022. Personalization of a Mid-Air Gesture Keyboard using Multi-Objective Bayesian Optimization. In *2022 IEEE International Symposium on Mixed and Augmented Reality (ISMAR)*. 702–710. https://doi.org/10.1109/ISMAR55827.2022.00088

[75] Hanul Shin, Jung Kwon Lee, Jaehong Kim, and Jiwon Kim. 2017. Continual learning with deep generative replay. *Advances in neural information processing systems* 30 (2017).

[76] Jasper Snoek, Hugo Larochelle, and Ryan P Adams. 2012. Practical bayesian optimization of machine learning algorithms. *Advances in neural information processing systems* 25 (2012).

[77] Jasper Snoek, Oren Rippel, Kevin Swersky, Ryan Kiros, Nadathur Satish, Narayanan Sundaram, Mostofa Patwary, Mr Prabhat, and Ryan Adams. 2015. Scalable bayesian optimization using deep neural networks. In *International conference on machine learning*. PMLR, 2171–2180.

[78] Jost Tobias Springenberg, Aaron Klein, Stefan Falkner, and Frank Hutter. 2016. Bayesian optimization with robust Bayesian neural networks. *Advances in neural information processing systems* 29 (2016).

[79] Fan-Keng Sun, Cheng-Hao Ho, and Hung-Yi Lee. 2019. Lamol: Language modeling for lifelong language learning. *arXiv preprint arXiv:1909.03329* (2019).

[80] Kevin Swersky, Jasper Snoek, and Ryan P Adams. 2013. Multi-Task Bayesian Optimization. In *Advances in Neural Information Processing Systems*, C.J. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K.Q. Weinberger (Eds.), Vol. 26. Curran Associates, Inc. https://proceedings.neurips.cc/paper/2013/file/f33ba15effa5c10e873bf3842afb46a6-Paper.pdf

[81] Sebastian Thrun and Lorien Pratt (Eds.). 1998. *Learning to Learn*. Kluwer Academic Publishers, USA.

[82] Maegan Tucker, Myra Cheng, Ellen Novoseller, Richard Cheng, Yisong Yue, Joel W Burdick, and Aaron D Ames. 2020. Human preference-based learning for high-dimensional optimization of exoskeleton walking gaits. In *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 3423–3430.

[83] Michael Volpp, Lukas P Fröhlich, Kirsten Fischer, Andreas Doerr, Stefan Falkner, Frank Hutter, and Christian Daniel. 2019. Meta-learning acquisition functions for transfer learning in bayesian optimization. *arXiv preprint arXiv:1904.02642* (2019).

[84] Jane X Wang. 2021. Meta-learning in natural and artificial intelligence. *Current Opinion in Behavioral Sciences* 38 (2021), 90–95. https://doi.org/10.1016/j.cobeha.2021.01.002 Computational cognitive neuroscience.

[85] Liyuan Wang, Xingxing Zhang, Hang Su, and Jun Zhu. 2024. A Comprehensive Survey of Continual Learning: Theory, Method and Application. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 46, 8 (2024), 5362–5383. https://doi.org/10.1109/TPAMI.2024.3367329

[86] Xilu Wang, Yaochu Jin, Sebastian Schmitt, and Markus Olhofer. 2022. Recent Advances in Bayesian Optimization. *arXiv preprint arXiv:2206.03301* (2022).

[87] Karl Weiss, Taghi M Khoshgoftaar, and DingDing Wang. 2016. A survey of transfer learning. *Journal of Big data* 3 (2016), 1–40.

[88] Martin Wistuba and Josif Grabocka. 2021. Few-shot Bayesian optimization with deep kernel surrogates. *arXiv preprint arXiv:2101.07667* (2021).

[89] Martin Wistuba, Nicolas Schilling, and Lars Schmidt-Thieme. 2018. Scalable gaussian process-based transfer surrogates for hyperparameter optimization. *Machine Learning* 107, 1 (2018), 43–78.

[90] Kenta Yamamoto, Yuki Koyama, and Yoichi Ochiai. 2022. Photographic Lighting Design with Photographer-in-the-Loop Bayesian Optimization. In *Proceedings of the 35th Annual ACM Symposium on User Interface Software and Technology* (Bend, OR, USA) *(UIST '22)*. Association for Computing Machinery, New York, NY, USA, Article 92, 11 pages. https://doi.org/10.1145/3526113.3545690

[91] Dani Yogatama and Gideon Mann. 2014. Efficient transfer learning method for automatic hyperparameter tuning. In *Artificial intelligence and statistics*. PMLR, 1077–1085.

[92] Jaehong Yoon, Eunho Yang, Jeongtae Lee, and Sung Ju Hwang. 2017. Lifelong learning with dynamically expandable networks. *arXiv preprint arXiv:1708.01547* (2017).

[93] Difeng Yu, Mantas Cibulskis, Erik Skjoldan Mortensen, Mark Schram Christensen, and Joanna Bergström. 2024. Metrics of Motor Learning for Analyzing Movement Mapping in Virtual Reality. In *Proceedings of the CHI Conference on Human Factors in Computing Systems* (Honolulu, HI, USA) *(CHI '24)*. Association for Computing Machinery, New York, NY, USA, Article 724, 18 pages. https://doi.org/10.1145/3613904.3642354

A SIMULATED EXPERIMENTS USING BENCHMARK FUNCTIONS

The core idea of this paper is to develop a novel approach for training a population model to enable CHiLO. There are multiple potential ways to implement and train such a population model. In this section, we compare the optimization performance and computational efficiency of our ConBO approach (a BNN-based population model trained using predictions from previous models) against alternative approaches. Specifically, our experiments aim to achieve the following goals:

- **Goal 1: population model performance comparison.** We compare different methods for constructing the population model and use them to sequentially optimize 15 user functions. Our goal is to demonstrate that ConBO's GP-based generative memory replay approach delivers superior optimization performance across users.
- **Goal 2: ConBO with and without the user-specific model.** After confirming that ConBO's population model performs as well or better than alternative methods, we examine the benefits of incorporating the user-specific model to compute acquisition values, showing potential improvements in performance, especially in later iterations.
- **Goal 3: Computational efficiency.** We evaluate the computational cost of ConBO during deployment and compare it to other alternative implementations. The goal is to highlight the lightweight computational demands of ConBO, making it practical for real-world applications.
- **Goal 4: The effect of memory replay on forgetting .** Lastly, we analyze the effect of our memory-replay method on adaptation performance when revisiting previously encountered users.

In total, we run three simulated experiments. The first two experiments address the first three goals, while the last test focuses on the effect of memory replay. Our results show that ConBO offers optimization performance comparable to or better than alternative approaches, with additional performance gains when the user-specific model is integrated. In terms of computational efficiency, we find that BNN-based population models like ConBO maintain relatively low computation durations. Meanwhile, other GP-based population models experience increasing computation times as data grows, making them less suitable for direct application in CHiLO. Moreover, we found that our memory replay effectively improves the optimization performance when encountering previously seen tasks, indicating an enhancement to reduce model forgetting. This indicates that our optimizer continuously improves over time, achieving greater efficiency for users with traits it has already seen in past optimizations.

For the following simulations, we use common benchmark functions to generate a set of user functions. We begin by selecting a base function, such as the Branin function, to represent the population. By applying shifts and scaling transformations, we create a series of similar but distinct sub-functions, each representing an independent user. This allows us to simulate a scenario where the optimizer encounters different user-specific functions sequentially. This approach is commonly used in simulation studies [14, 44].

The following experiments were conducted on a Windows 10 system equipped with a 12th Gen Intel(R) Core i7 CPU and an NVIDIA GeForce RTX 3070 GPU.

A.1 Different Approaches for Achieving CHiLO

Here, we exhaustively outline different approaches for implementing and training a population model for CHiLO. To align with the broader BO framework, a population model must be capable of generating predicted means and variances for a given design candidate. These values are required for computing most acquisition functions in BO. Two computational models that fulfill these requirements are GP and BNN, both of which can be trained on observations and produce the necessary predictions of means and variances.

First, we introduce the approaches using BNN as the core of the population model:

1. ConBO. This is our proposed method, using a BNN as the population model. This BNN is continually retrained using the predicted means and variances generated from previous models, with each model representing a different user. Key strengths of ConBO include its well-distributed training data, which ensures the population model does not suffer from overfitting or underfitting in specific areas of the parameter space. Additionally, the BNN's predicted variance is carefully regulated: we filter out unreliable predictions based on variance thresholds, and the population model is trained to minimize predicted variance levels (see Equation 7). This keeps the variance within a reasonable range, improving the model's reliability. Furthermore, ConBO incorporates a GP model, representing the current user, to generate acquisition values, enabling more effective personalized adaptation and optimization. For a more detailed explanation of ConBO, please refer to section 4.

2. ConBO without GP. To assess the performance of ConBO's population model independently, we introduce a variation that excludes the influence of the current user's GP model, referred to as "*ConBO without GP.*" This allows us to evaluate the population model's performance on its own.

3. ConBO without Variance Filter. In the "*ConBO without Filter*" approach, we omit the variance filter. We assess the population model's performance without any constraints on the aggregated variances derived from prior user models.

4. BNN without Replay. A simpler approach to using BNN as the population model is to continuously update it without incorporating any memory replay. This method resembles previous works in BO where large sample sizes necessitate the use of BNNs as surrogate models, allowing them to handle the growing computation time more efficiently than GPs (e.g., [78]). We call this approach *BNN without Replay*. To assess the performance of the population model independently, this approach also does not incorporate the current user's GP model. While straightforward to implement, this approach presents several challenges such as catastrophic forgetting and model instability due to the uneven distribution of observations (as mentioned in subsection 3.2). Furthermore, there is a potential for unstable variance. Without regulation, the BNN's predicted variance can become unstable (extremely high or low) as more data accumulates, complicating the computation of acquisition values.

5. BNN with Direct Replay. Building on *BNN without Replay*, a more advanced variation involves using memory replay. In this

variation, the population model is updated with all observations from previous users after each user is completed. While this method effectively reduces forgetting, it faces potential challenges including unstable variances across the parameter space and the risk of overfitting in some regions while underfitting in others. We refer to this variation as "*BNN with Direct Replay.*"

Below, we introduce the other approaches which leverage GP as the population model:

6. Single-GP. Rather than using a BNN, one can employ a GP directly as the population model, capturing all observations from all users encountered. However, a major limitation of this approach is the cubic increase in computation time as more data is added. Beyond a certain point, updating the population model becomes prohibitively expensive and impractical for real-time applications.

7. Transfer Acquisition Function (TAF). To address the issue of cubic computation growth, previous works proposed a weighted-sum approach called Transfer Acquisition Function (TAF) [44, 89], designed for meta-learning in BO. This approach stores each user's data as a separate GP model. When a new user is encountered, all previously stored models generate acquisition values, which are then combined based on weights to guide the optimization for the current user. Typically, TAF has been applied in meta-learning scenarios where a fixed set of users, referred to as "prior users," undergo a complete optimization process to build a library of "prior models." These models then aid in optimizing new users without adding further prior models. This method can be extended to continual learning (CHiLO) by removing the distinction between "prior users" and "new users" In this setting, each new user's data is stored as an additional prior model, continuously growing the model library. Although this approach avoids the cubic increase in computation time, its computational cost still grows linearly with the number of models, which can become a bottleneck as the user base expands.

8. Standard BO (BO). Finally, we keep standard BO as a baseline, which utilizes the GP surrogate model to optimize for each current user. There is no transferring knowledge between users.

A.2 Base Function for Simulations

In our simulations, we utilize two common base functions: Branin[4] and McCormick[5], each with two design parameters and one objective function. They serve as the foundation for generating a group of similar yet distinct user functions.

We linearly normalize the input ranges of the Branin function ($x_1 \in [-5, 10], x_2 \in [0, 15]$) to match the McCormick function ($x_1, x_2 \in [0, 1]$), ensuring both have the same parameter range. While the output ranges of Branin and McCormick differ (Branin: approximately $[0.398, 300]$, McCormick: approximately $[-1.9133, 20]$), both functions were originally designed for minimization tasks. We flip the functions (multiplying by -1) to convert them into maximization problems.

[4]https://www.sfu.ca/~ssurjano/branin.html
[5]https://www.sfu.ca/~ssurjano/mccorm.html

A.3 Generating a Group of Synthetic Users

To simulate a diverse group of users with shared characteristics but varying behaviors, we create a set of sub-functions by shifting and scaling the base functions. We will refer to these sub-functions as user functions.

Each user function is generated by shifting the input values before passing them to the base function. The magnitude of these shifts is sampled from a uniform distribution within a specified range, adding diversity among the users. Mathematically, the shift is represented as $x'_n = x_n + \delta_n$, where x_n is the original input, x'_n is the shifted input, $n \in 1, 2$ represents the parameters, and the shift amount $\delta_n \sim U(-\frac{shift\_range}{2}, \frac{shift\_range}{2})$. This variation simulates different user responses to the same design. We also scale the output of the base function by a scalar factor, which is also sampled from a uniform distribution. This factor, denoted as $S \sim U(1 - \frac{scale\_range}{2}, 1 + \frac{scale\_range}{2})$, introduces further diversity by simulating users with different performance levels.

We first introduce the simulation experiment on Branin (A.4, A.5), followed by McCormick (A.6, A.7).

A.4 Branin (Test 1): Task and Settings

We generate 15 user functions, with each optimization method running for 30 iterations per user function. For each user function, we perform a grid search to determine its optimal performance. Based on this, we compute the regret at each iteration. Additionally, we record the computation time at every iteration across all user functions and methods. For this simulation, we set $shift\_range = 0.3$ and $scale\_range = 0.2$. Here, we provide the detailed settings for each approach:

1. ConBO. The parameter settings for ConBO are aligned with those used in our user study.

- The *population model* is a BNN consisting of three fully connected layers, each with 100 nodes using ReLU as the activation function. A dropout layer (dropout rate of 0.1) is applied after the last hidden layer. During population model training (Step 4 in the workflow), the BNN is trained for 1200 epochs. For adaptation (Step 1), we run 15 epochs for each new observation.
- The *user-specific models* (current and prior) are implemented as Gaussian Processes (GPs) with a Matérn 5/2 kernel, accounting for homoscedastic noise.
- The initial number of random exploration points, r_0, is set to 16, with a decay rate d_r of 5. We merge the EI values from both the population model and the current user-specific model using a weighting factor from Eq. 4, where $\alpha_1 = 16$ and $\alpha_2 = 0.1$. This ensures that the influence of the current user's model increases after the fifth iteration, growing linearly.
- For population model training, we first generate a grid of 30×30 points evenly spaced across the design space, followed by 50 randomly sampled points. The predicted means and variances from all previous user-specific models are then queried. A variance threshold of $\lambda = 50$ is applied.

2. ConBO without GP. This variant's setting is similar to ConBO, except for setting $\alpha_1 = 30$, meaning that all EI values are derived from the population model, as no adaptation from the user-specific model is applied.

3. ConBO without Variance Filter. This variant is similar to ConBO, with the key difference being the disabled variance filter. Consequently, all mean and variance estimates across the design space from every previous model are incorporated into the training of the population model. This includes estimates from a user's model in regions where it has encountered few or no observations—an inclusion that the variance filter would otherwise exclude.

4. BNN without Replay. All other settings are identical to those in ConBO, except that there is no memory replay to retrain the population model. Instead, the population model keeps updating based on new observations.

5. BNN with Direct Replay. Similar to the *BNN without Replay*, but here, after each user's optimization is complete, the population model is retrained on all past observations.

6. Single GP. This method uses a single-task GP with a Matérn 5/2 kernel. The optimization process involves 10 restarts for the acquisition function, with 1,024 candidate points and 512 Monte Carlo samples. Of the 30 total iterations, the first 16 are random searches, followed by 14 optimization iterations guided by the Expected Improvement acquisition function. We retain the GP as a population model and update it with new observations over time. The number of random search decay across users, with an initial r_0 of 16 and a decay rate d_r of 5.

7. Transfer Acquisition Function (TAF). The TAF setting mostly follows prior work [44], where observations from each user function are stored in independent GP models. The number of random searches decays across users, with an initial r_0 of 16 and a decay rate d_r of 5. We further follow the original paper's settings, with weights decaying using $d_1 = 0$ and $d_2 = 0.05$, which were identified through pilot tests to enhance performance.

8. Standard BO (BO). This approach basically follows *Single GP* but the GP model restarts for each new user.

A.5 Branin (Test 1): Results

Figure 4 shows the results of our simulation on the Branin function.

Goal 1: population model performance comparison. To analyze the performance of our approach with respect to our first goal, we compare *ConBO without GP* against other approaches (Figure 4a and Figure 4b). Due to the large number of baselines, we separate the approaches into BNN-based (Figure 4a) and GP-based (Figure 4b) groups. Examining Figure 4a, we found that *ConBO without GP* surpasses other BNN-based variations. Although these methods show comparable performance in the early iterations, they fail to adapt effectively to specific functions over time, resulting in poorer performance in later iterations. Further analysis (Figure 5) shows a key issue with the direct replay and no-replay approaches: the predicted variances in the BNN models are unregulated. As more data is introduced, these methods exhibit highly unstable variance estimates, with values becoming excessively high. Moreover, since the training data is unevenly distributed across the parameter space, the predicted variances vary significantly. For instance, in the BNN without replay, variance values range from under 100 to over 700. Given that variance plays a critical role in computing the acquisition function, this instability ultimately hinders the models' ability

to adapt effectively. In contrast, ConBO employs a grid-based generative approach with a variance filter, ensuring that the training data for the population model remains within a reasonable range and spans across the parameter space evenly. This regulation leads to more stable variance predictions and, consequently, better online adaptation performance. Finally, we found that ConBO-wo-GP performed better than ConBO-wo-Filter in early iterations, owing to the more regulated variance range.

Comparing *ConBO without GP* against GP-based approaches (Figure 4b), *ConBO without GP* outperforms standard BO, which typically starts optimization from scratch, leading to slower adaptation and worse performance in early iterations. This is reflected in its higher regret values, particularly in the initial stages of optimization. In addition, both *ConBO without GP* and TAF outperform the Single GP approach. This is also expected, as the single GP model lacks variance regulation, potentially leading to unstable model fitting when encountering a large number of observations and ineffective user-specific adaptation.

Goal 2: ConBO with and without the user-specific model. To address our second goal, we compare the performance of ConBO with and without the current user's GP model in Figure 4c. It demonstrates that incorporating the user-specific model to generate acquisition values allows ConBO to further enhance its performance, particularly in later iterations. This highlights the importance of leveraging a user-specific model to support the adaptation process, leading to more personalized results.

Goal 3: Computational efficiency. To answer our third goal, we show the average computation duration at each iteration across all users in Figure 4d. This highlights a significant drawback of GP-based approaches. The computation time of a single GP increases cubically with the number of observations. Specifically, the computation time of a single GP surpasses that of BNN-based approaches around the 11th or 12th synthetic user, corresponding to approximately 350 observations. This number is easily reached in real-world deployments involving a sequence of actual users. On the other hand, TAF's computation time grows linearly with the number of prior models (i.e., previously encountered users). Notably, the computation time of TAF surpasses the BNN-based methods from the 4th synthetic user onward. This increase in computation cost makes the GP-based solutions unsuitable for scenarios with a potentially unlimited number of users in a real-world continual learning setting. In contrast, the BNN-based approaches, including ConBO, do not suffer from the increasing computation duration.

A.6 McCormick (Test 2): Task and Settings

As with the Branin simulation, we generate 15 user functions for the McCormick function. As McCormick is a simpler function, each user function is optimized over 7 iterations. For each user function, we perform a grid search to determine its optimal performance. Based on this, we compute the regret at each iteration. Additionally, we record the computation time at each iteration across all user functions and methods. In this simulation, we set the shift range to $shift\_range = 0.5$ and the scale range to $scale\_range = 0.2$.

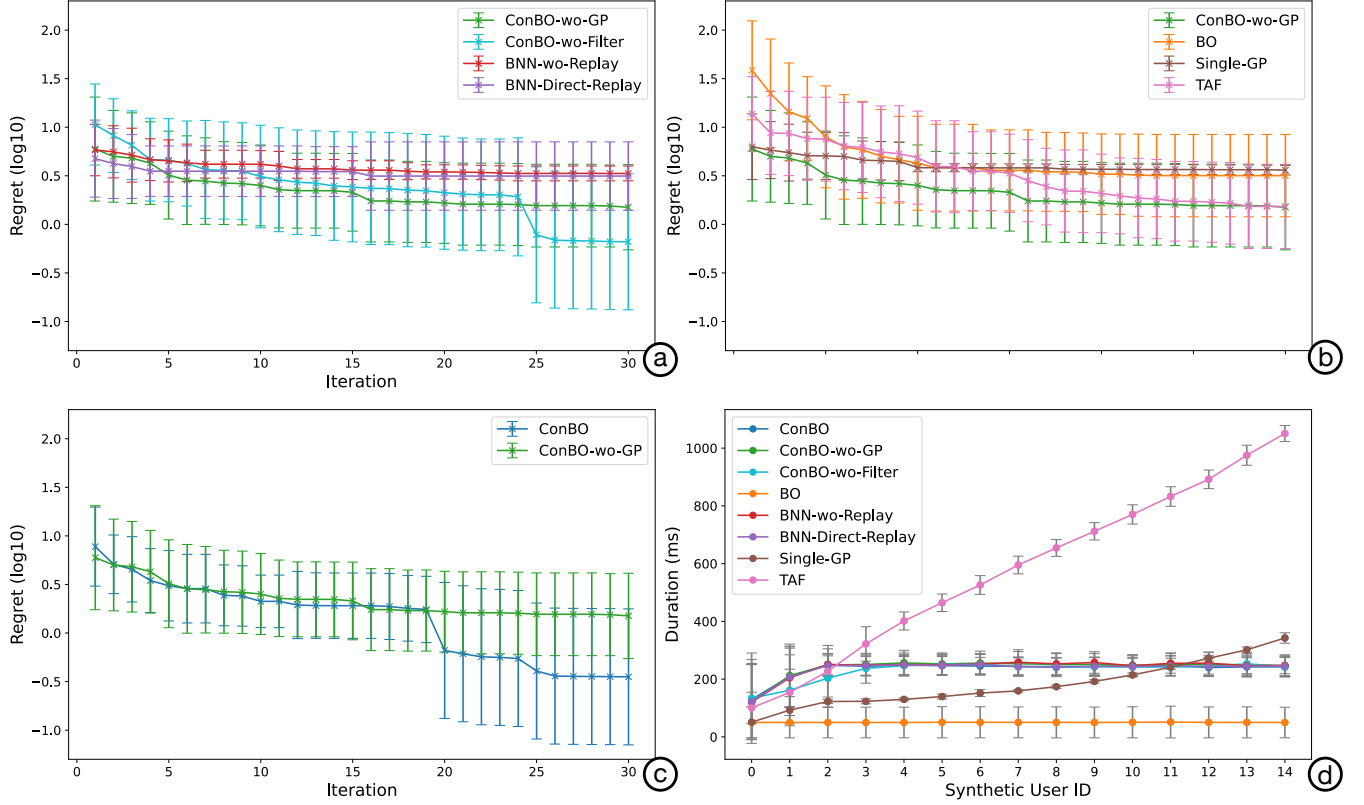Below are the detailed settings for each approach:

Figure 4: Results of our simulation study using the Branin function (Test 1). All error bars represent one standard deviation. (a) The regret values over the iterations of different BNN-based methods. We highlight that our *ConBO without GP* has better performance over other BNN-based population models. (b) The regret values over the iterations of GP-based methods, highlighting *ConBO without GP* outperforms Single GP and delivers comparable performance to TAF. (c) The regret values over the iterations to compare ConBO with and without integrating the user-specific model. The result shows that, with the support of the user-specific model, ConBO can improve its performance further. (d) The mean computation time spent in one iteration for each user. The result highlights the computation costs of GP-based population models (Single GP and TAF) increase quickly when accumulating more data.

1. ConBO. The configuration of ConBO is similar to the one used in the Branin function (Test 1). However, due to the limited optimization budget of 7 iterations, the initial number of random exploration points, r_0, is set to 3, with a decay rate d_r of 3. We combine the EI values from both the population model and the user-specific model, with weighting factors $\alpha_1 = 3$ and $\alpha_2 = 0.15$. Additionally, a variance threshold of $\lambda = 10$ is applied.

2. ConBO without GP. This configuration mirrors ConBO, with the exception that α_1 is set to 7, meaning all EI values are derived exclusively from the population model, without incorporating the user-specific model.

3. ConBO without Variance Filter. This variant's setting is similar to ConBO, except for disabling the variance filter, meaning that all the estimates across the design space from every previous user model are used for population model training.

4. BNN without Replay. This approach follows a setup similar to the Branin simulation, with the only difference being the number

of random exploration points. Here, the initial r_0 is set to 3, with a decay rate d_r of 3.

5. BNN with Direct Replay. This also mirrors the its setting in the Branin simulation but differs in the number of random exploration points. Here, the initial r_0 is set to 3, with a decay rate d_r of 3.

6. Single GP. This approach's setting is similar to one used in the Branin simulation, with the only difference being the number of random exploration points. Here, the initial r_0 is set to 3, with a decay rate d_r of 3.

7. Transfer Acquisition Function (TAF). This approach shares similar settings as in the Branin simulation, where $r_0 = 3$ and the decay rate $d_r = 3$. Additionally, the weight decay parameters for the prior models are consistent with the Branin setup: $d_1 = 0$ and $d_2 = 0.15$.

8. Standard BO. For the 7 total iterations, the first 3 iterations are random searches, followed by 4 optimization iterations guided by the Expected Improvement acquisition function.
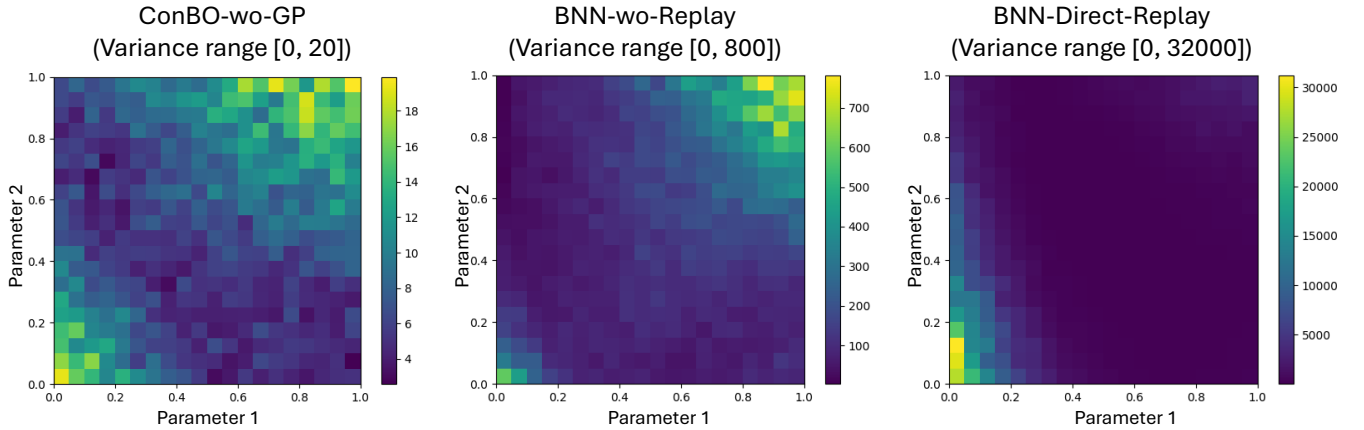
Figure 5: The predicted variance values derived from different BNN-based population model implementations after all 15 user functions. Our ConBO utilizes previous GPs to generate predicted means and variances and further utilizes a variance filter to remove unreliable predictions. Therefore, ConBO allows for more regulated variance predictions. On the other hand, other BNN-based approaches do not have an explicit mechanism to regulate the variance of the population model, making it unstable and changing drastically over the parameter space when trained with a large number of observations. This highlights the stability offered by our approach.

A.7 McCormick (Test 2): Results

Figure 6 presents the results of the McCormick simulation, which align with our findings from the Branin function.

Goal 1: population model performance comparison. To determine whether ConBo meets our first goal in this setting, we compare *ConBO without GP* to other approaches (Figure 6, a and b). Again, we separate approaches into BNN-based (shown in Figure 6, a) and GP-based (shown in Figure 4, b) groups. In Figure 6 (a), while all BNN-based methods converge by the final iteration, we observe that *ConBO without GP* and ConBO without Variance Filter outperform other BNN-based variations in the earlier iterations. This is consistent with our findings from the Branin function. In Figure 6 (b), we see that *ConBO without GP* outperforms standard BO overall and Single GP in early iterations. Both *ConBO without GP* and TAF provide similar performance throughout the simulation.

Goal 2: ConBO with and without the user-specific model. To answer our second goal, Figure 6 (c) compares the two variations of ConBo. In this case, incorporating the user-specific model does not lead to a significant difference in performance. This can potentially be attributed to the lower complexity of the McCormick function compared to the Branin function. As a result, variations in the user functions do not impact performance to the same extent.

Goal 3: Computational efficiency. To address our third goal, Figure 6 (d) shows the computation time per iteration across users. As expected, TAF shows a linearly increasing computation time. As observed in the previous simulated test, its computation time surpasses that of the BNN-based methods starting from the 4th synthetic user. Notably, since each user in the McCormick simulation has significantly fewer observations than in the Branin simulation (7 vs. 30 iterations for each user), the Single GP approach has not yet reached the point of rapidly escalating computational cost, as seen in Figure 4d. However, if the synthetic users continued, the

computation time for a single GP would surpass that of BNN-based methods at approximately 350 observations (around 50 synthetic users) as we learned from the previous simulated test.

A.8 Revisiting Previously Seen Synthetic Users (Test 3)

A.8.1 Task and Settings. In Test 3, we aim to demonstrate how ConBO's memory-replay mechanism addresses the potential forgetting issue in CHiLO. To focus specifically on the impact of the memory-replay mechanism, we compare two approaches: *1. ConBO (with Replay)* and *2. BNN without Replay.* The experimental settings for these two approaches are identical to those described in Test 1 (Appendix subsection A.4).

We use the Branin function as the base to generate 10 synthetic users, with a shift range of 0.4 and a scale range of 0.4. These ranges are slightly broader than those in Test 1, resulting in a more diverse set of synthetic users.

This test is divided into two phases: *initial optimization* and *re-optimization*. In initial optimization, both approaches continually optimize for the 10 synthetic users, and the resulting population models are stored as the **trained models**. In the subsequent re-optimization phase, these two **trained models** are used to re-optimize the same previously encountered synthetic users. This evaluates whether the population models effectively retain the memory of previously seen tasks and whether the optimizer improves its efficiency for users with characteristics similar to those it has seen before. Note that during re-optimization, we employ the same trained population models for each synthetic user (i.e., the re-optimization of individual users does not affect the model for other users). These trained population models capture the state after *initial optimization*. This analysis evaluates the ability of the population models to retain knowledge about prior users after all synthetic users in the initial optimization phase.
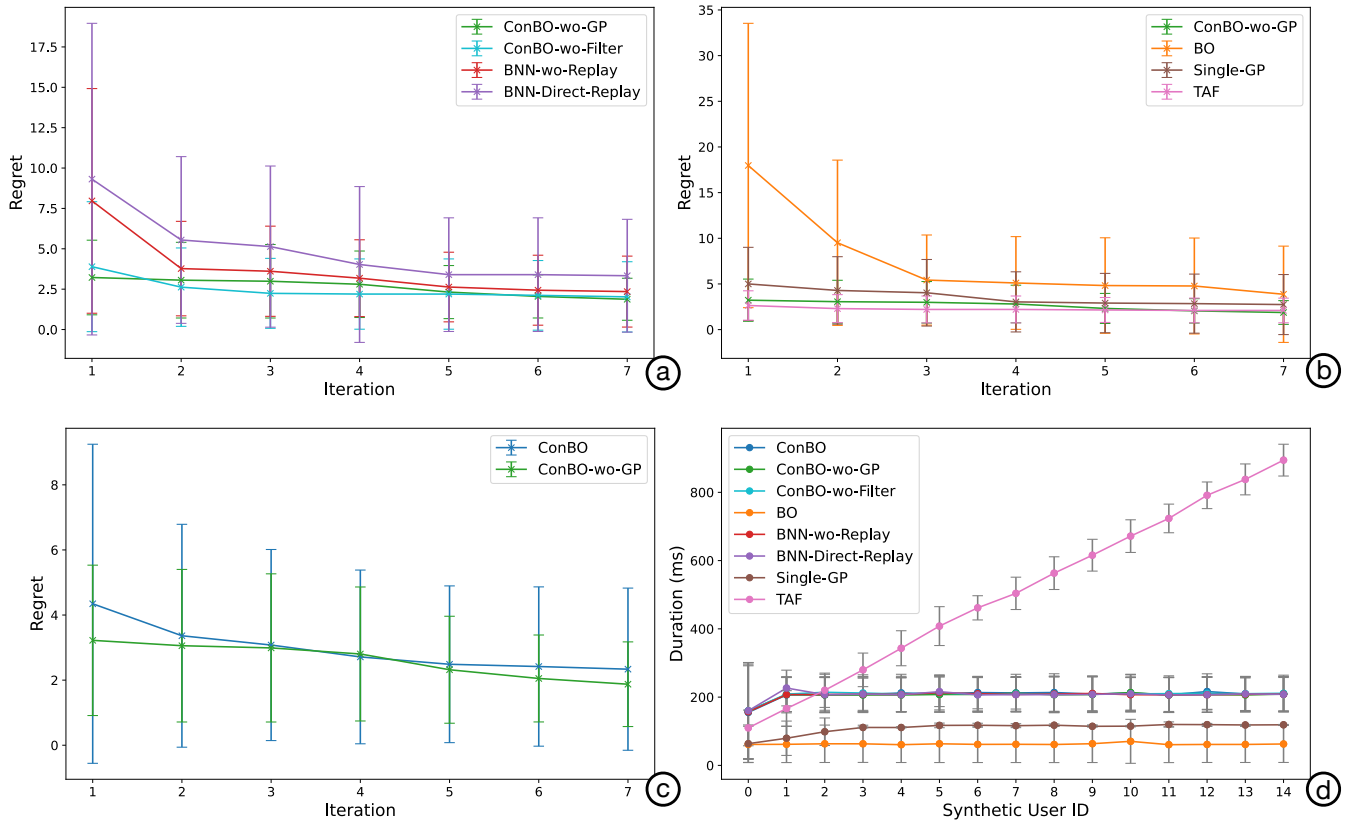
Figure 6: Results of our simulated study using the McCormick function (Test 2). All error bars represent one standard deviation. (a) The regret values over the iterations of different BNN-based methods. We highlight that our *ConBO without GP* has better performance over other BNN-based population models. (b) The regret values over the iterations of different GP-based methods. *ConBO without GP* outperforms standard BO overall and Single GP in early iterations. *ConBO without GP* and TAF provide similar performance. (c) The regret values over the iterations to compare ConBO with and without integrating the user-specific model. Incorporating the user-specific model does not lead to a significant difference in performance. (d) The mean computation time spent in one iteration for each user. The result highlights that the computation time of TAF increases linearly with the number of users.

*A.8.2   **Goal 4: The effect of memory replay on forgetting**.*
The results of the re-optimization are shown in Figure 7. For the re-optimization of users 1 to 5, the population model trained with *BNN without Replay* performs significantly worse compared to the model trained with *ConBO*. This suggests that, without a memory-replay mechanism, the population model struggles to retain the knowledge of earlier users, leading to reduced optimization efficiency when revisiting them. In comparison, for the more recent 6th to 10th synthetic users, the performance of *BNN without Replay* is comparable to that of *ConBO*.

A.9   Findings and Conclusion

Through the simulations, we explored the effectiveness of our proposed ConBO method compared to alternative approaches for constructing a population model for CHiLO. Particularly, our experiments on the Branin and McCormick functions aimed to address

three key goals: comparing population models' optimization performance, understanding the benefits of incorporating a user-specific model, and evaluating computational efficiency.

Our results show that ConBO outperforms other BNN-based approaches even when the user-specific model is not included, thanks to its regulated variance predictions and well-distributed training data across the parameter space. This ensures that ConBO avoids the pitfalls of overfitting or underfitting, providing consistent performance across different users. When the user-specific model is incorporated, ConBO demonstrates further improvements in performance, particularly in later iterations, allowing for more personalized and adaptive optimization.

In terms of computational efficiency, ConBO maintains a light-weight computational profile, even as more data is accumulated. This stands in contrast to other GP-based approaches, such as Single GP and TAF, which experience significant computational growth as the dataset scales, making them less suitable for continual learning applications. Furthermore, ConBO (with memory replay) showcases a faster convergence when re-optimizing the previously seen
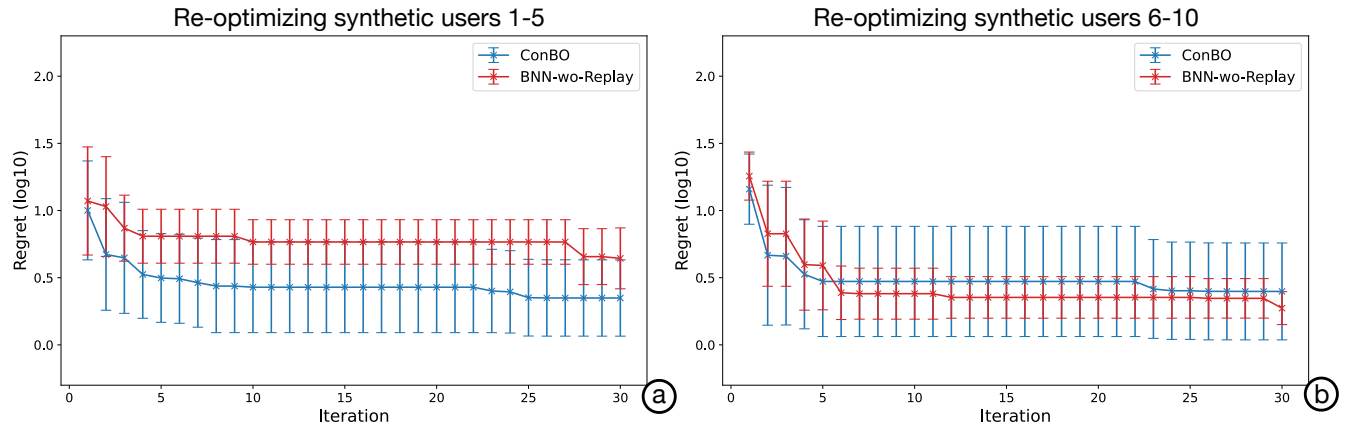
Figure 7: Results of our simulated test when *re-optimizing for the previously seen users* (Test 3). All error bars represent one standard deviation. (a) The performance for re-optimizing the 1st–5th synthetic users. (b) The performance for re-optimizing the 6th–10th synthetic users.

synthetic users. This highlights its ability to address the potential forgetting issue in CHiLO.

Overall, ConBO offers an effective balance between optimization performance and computational efficiency, making it a strong candidate for population model construction in CHiLO applications.

B ANALYSIS OF USER DIVERSITY

This section provides a detailed analysis of the diverse user characteristics to illustrate how ConBO effectively adapts to varying user performances. Specifically, we examine the optimal keyboard design for each individual user and their corresponding typing performance. Additionally, we show how the population model evolves over the sequence of users.

B.1 Participants Characteristics

The details of all participants are provided in Table 2, showcasing the diverse range of user characteristics in our user study. Note that all the participants had normal eyesight or corrected eyesight (glasses or contact lenses) during the study.

B.2 Optimal Keyboard Design for each User

Table 3 presents the optimal keyboard configuration and corresponding typing performance for each user under the three adaptation conditions Manual, Standard BO, and ConBO, according to our text entry study. Particularly, both ConBO and Standard BO identified a higher variety of optimal keyboard dimensions (from 32.5×11.37 to 88.12×30.84). However, Manual leads to less diverse keyboard dimension setting.

Our analysis shows that Manual exhibits less variation in keyboard configuration (std. width: 6.69 cm, distance: 10.84 cm) compared to Standard BO (std. width: 19.88 cm, distance: 12.87 cm) and ConBO (std. width: 16.25 cm, distance: 14.85 cm), suggesting

that users tend to be more conservative with their manual keyboard setup. Moreover, ConBO demonstrates the ability to remain adaptive to individual users with a growing user base.

B.3 Evolvement of the population model over Participants

ConBO's population model captures prior beliefs about expected performance based on the observed performance of previous participants. Table 4 shows the mean, minimum and maximum parameter values for the optimal keyboard configurations over the first 3, 6, 9, and 12 participants.

In Figure 8, we observe that the population model's predicted performance evolves in alignment with the observed user performance. Initially, after collecting data from the first three participants, the population model lacks sufficient information, resulting in a relatively uniform prior over the possible keyboard configurations. An early pattern emerges, suggesting that larger keyboard sizes tend to improve performance. After incorporating data from six participants, this pattern becomes more pronounced, with larger keyboard sizes showing improved performance. At the same time, the population model suggests that shorter distances between the user and the keyboard can enhance typing performance. By the time data from 12 participants is added, the population model adjusts, recognizing that some users perform better with the keyboard placed farther away and that the ideal distance may vary from person to person, while still generally favoring larger keyboard sizes.

This gradual learning process demonstrates the population model's ability to refine its estimates as it continuously adapts to new participant data, identifying performance patterns across previous users. The population model's improving prior belief about expected user performance leads to increasingly smaller regret with growing numbers of participants (see Figure 3).

Participant ID	Age	Gender	Height (cm)	Hand Length (cm)	Hand Breadth (cm)	Arm Length (cm)	Eyesight
1	26	Male	178	17.9	7.0	60	Corrected
2	38	Male	190	20.5	8	66	Normal
3	26	Male	186	19	7.5	60	Normal
4	30	Male	180	18.5	7.1	59	Corrected
5	27	Male	183	18.7	7.8	61	Normal
6	25	Female	160	16.5	6	47	Corrected
7	26	Male	192	20	8	70	Normal
8	28	Male	195	20.5	8.1	69	Normal
9	30	Male	169	17.2	6.3	54	Corrected
10	24	Male	185	17.9	6.9	65	Normal
11	25	Male	175	17	6.4	57	Corrected
12	28	Female	169	16.3	6.3	53	Corrected

Table 2: Participant characteristics.

Pariticipant ID	Width, Height [cm]			Distance [cm]			Typing Speed [NetWPM]		
	MANUAL	STANBO	CONBO	MANUAL	STANBO	CONBO	MANUAL	STANBO	CONBO
1	40.00, 14.00	37.5, 13.12	32.50, 11.37	49.21	29.00	43.67	25.00	16.65	24.76
2	50.00, 17.50	88.12, 30.84	88.12, 30.84	28.16	64.00	36.00	19.75	15.87	16.42
3	40.00, 14.00	58.13, 20.34	86.25, 30.19	54.47	53.00	31.00	10.24	13.03	12.47
4	50.00, 17.50	78.75, 27.56	76.87, 26.91	49.21	50.00	31.00	18.99	19.96	19.18
5	50.00, 17.50	20.62, 7.22	88.12, 30.84	28.16	41.00	38.00	17.45	9.63	14.02
6	60.00, 21.00	71.25, 24.94	82.50, 28.88	61.84	64.00	25.00	16.51	16.77	22.94
7	50.00, 17.50	61.88, 21.66	88.12, 30.84	43.95	47.00	31.00	15.07	14.27	13.12
8	60.00, 21.00	54.38, 19.03	76.87, 26.91	38.68	44.00	64.00	13.42	15.13	14.94
9	50.00, 17.50	31.87, 11.16	88.12, 30.84	38.68	37.00	63.00	12.76	14.25	14.28
10	50.00, 17.50	75.00, 26.25	75.00, 26.25	59.74	26.00	64.00	20.88	20.38	22.72
11	60.00, 21.00	58.13, 20.34	86.25, 30.19	43.95	57.00	59.00	13.56	12.12	11.31
12	50.00, 17.50	61.88, 21.66	61.88, 21.66	49.21	61.00	53.00	9.56	17.47	16.24

Table 3: Summary of keyboard dimensions, distance, and typing speed for each user across MANUAL, STANBO, and CONBO.

Group	Width [cm] (normalized)			Height [cm] (normalized)			Distance [cm] (normalized)		
	Mean	Min	Max	Mean	Min	Max	Mean	Min	Max
P 1-3	68.96 (0.71)	32.5 (0.23)	88.12 (0.97)	24.13 (0.71)	11.37 (0.23)	30.84 (0.97)	36.89 (0.29)	31.0 (0.15)	43.67 (0.46)
P 1-6	75.73 (0.80)	32.5 (0.23)	88.12 (0.97)	26.51 (0.80)	11.37 (0.23)	30.84 (0.97)	34.11 (0.22)	25.0 (0.00)	43.67 (0.46)
P 1-9	78.61 (0.84)	32.5 (0.23)	88.12 (0.97)	27.51 (0.84)	11.37 (0.23)	30.84 (0.97)	40.3 (0.38)	25.0 (0.00)	64.0 (0.97)
P 1-12	77.55 (0.83)	32.5 (0.23)	88.12 (0.97)	27.14 (0.83)	11.37 (0.23)	30.84 (0.97)	44.89 (0.50)	25.0 (0.00)	64.0 (0.98)

Table 4: Mean, minimum, and maximum of the optimal keyboard configurations for all participants and subgroups (normalized values according to Figure 8).
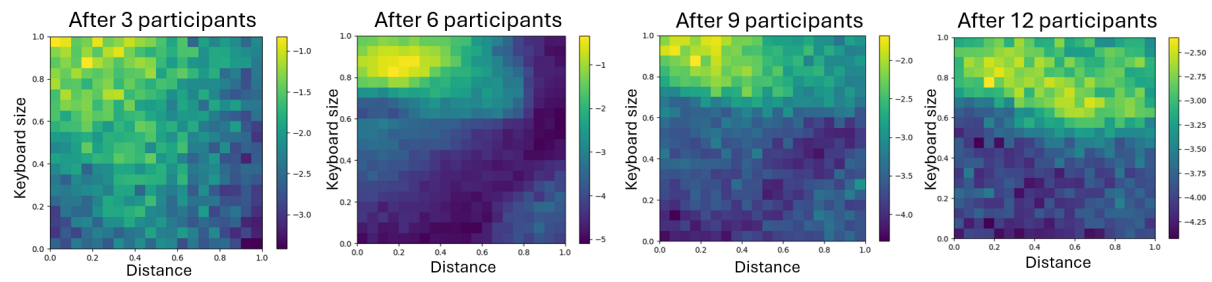
Figure 8: Estimated typing performance for different keyboard configurations according to the BNN population model with a growing user base. Note that the design parameters are normalized to the range $[0, 1]$, while predicted performance is normalized to $[-5, 5]$.