Select or Suggest? Reinforcement Learning-based Method for High-Accuracy Target Selection on Touchscreens

Zhi Li
Stony Brook University
New York, USA
zhili3@cs.stonybrook.edu

Maozheng Zhao
Stony Brook University
New York, USA
mazhao@cs.stonybrook.edu

Dibyendu Das
Stony Brook University
New York, USA
didas@cs.stonybrook.edu

Hang Zhao
Stony Brook University
New York, USA
zhao8@cs.stonybrook.edu

Yan Ma
Stony Brook University
New York, USA
yanma1@cs.stonybrook.edu

Wanyu Liu
STMS Lab, IRCAM, CNRS,
Sorbonne Université
Paris, France
abbywanyu.liu@ircam.fr

Michel Beaudouin-Lafon
Université Paris-Saclay, CNRS, Inria
Orsay, France
mbl@lri.fr

Fusheng Wang
Stony Brook University
New York, USA
fushwang@cs.stonybrook.edu

IV Ramakrishnan
Stony Brook University
New York, USA
ram@cs.stonybrook.edu

Xiaojun Bi
Stony Brook University
New York, USA
xiaojun@cs.stonybrook.edu

ABSTRACT

Suggesting multiple target candidates based on touch input is a possible option for high-accuracy target selection on small touchscreen devices. But it can become overwhelming if suggestions are triggered too often. To address this, we propose SATS, a Suggestion-based Accurate Target Selection method, where target selection is formulated as a sequential decision problem. The objective is to maximize the utility: the negative time cost for the entire target selection procedure. The SATS decision process is dictated by a policy generated using reinforcement learning. It automatically decides when to provide suggestions and when to directly select the target. Our user studies show that SATS reduced error rate and selection time over Shift [51], a magnification-based method, and MUCS, a suggestion-based alternative that optimizes the utility for the current selection. SATS also significantly reduced error rate over BayesianCommand [58], which directly selects targets based on posteriors, with only a minor increase in selection time.

CCS CONCEPTS

• **Human-centered computing** → **Human computer interaction (HCI)**; **Pointing**; **Touch screens**.

CHI '22, April 29-May 5, 2022, New Orleans, LA, USA

ACM ISBN 978-1-4503-9157-3/22/04…$15.00
https://doi.org/10.1145/3491102.3517472

KEYWORDS

Touch interaction, target selection, decision theory, reinforcement learning

ACM Reference Format:
Zhi Li, Maozheng Zhao, Dibyendu Das, Hang Zhao, Yan Ma, Wanyu Liu, Michel Beaudouin-Lafon, Fusheng Wang, IV Ramakrishnan, and Xiaojun Bi. 2022. Select or Suggest? Reinforcement Learning-based Method for High-Accuracy Target Selection on Touchscreens. In *CHI Conference on Human Factors in Computing Systems (CHI '22), April 29-May 5, 2022, New Orleans, LA, USA.* ACM, New York, NY, USA, 15 pages. https://doi.org/10.1145/3491102.3517472

1 INTRODUCTION

Accurately selecting targets with finger touch is an essential aspect of the user experience with touchscreen interactions. However, selecting a target with finger touch on a small touchscreen is error-prone, due to the small target sizes and imprecise touch input [10, 20, 21, 25, 26, 41, 51, 52]. Erroneous selections can cause tedious undo and redo actions. Besides, some selection errors are non-reversible and can lead to undesirable consequences. For example, if the "SEND" button on a keyboard is mis-selected while using an instant messaging application, the incomplete message is sent right away. One approach to mitigate this problem is *suggestion-based* input: target candidates near the touchpoint are presented as suggestions and the user confirms the selection with a second touch action.

Although a suggestion-based method could improve the accuracy of target selection, it comes with its own challenges for implementation. First, when should the suggestions be triggered? Ideally, a

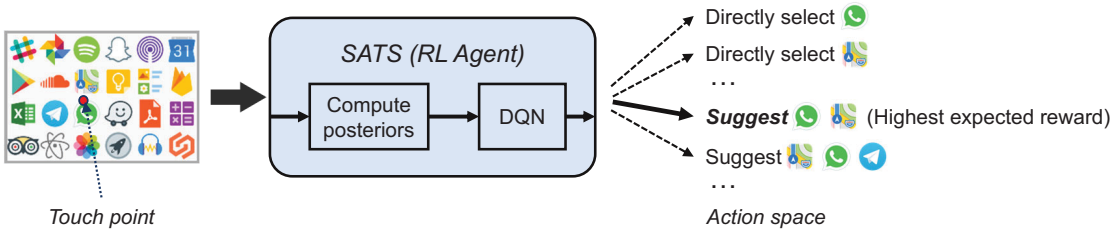


Figure 1: An overview of SATS. Given a touchpoint and menu layout, SATS, an agent trained via reinforcement learning, has a decision to make: directly selecting a target or showing suggestions. In this example, it shows two suggestions, because this action has the highest expected reward (utility).

system should only trigger suggestions when the input is ambiguous. Otherwise, it may overwhelm users and become distracting. Second, which candidates should be suggested? The number of suggestions should be large enough to include the intended target. On the other hand, if the number of suggestions is too large and unnecessary candidates are included, they become a distraction and will slow the user down.

In this paper, we introduce SATS, a Suggestion-based Accurate Target Selection method. We first view whether and which targets should be suggested as a decision problem: upon the observation of a touchpoint, the computer as an agent decides whether it should directly select a target candidate, or provide suggestions and ask for confirmation. In the latter case, it needs to further decide which candidates to suggest. The objective of the agent is to maximize the utility, which is defined as the negative time cost of the entire selection procedure. As we assume that the user will repeat the selection until the intended target is successfully selected if the current selection fails, using the negative time cost as the utility also captures the cost of erroneous selections. Under this setup, whether and which targets should be suggested becomes a *stochastic sequential decision problem* [2], because the agent may make sequential decisions if the user performs multiple selections to select the intended target.

We adopt a reinforcement learning-based approach, as shown in Fig. 1, to solve this sequential decision problem: the SATS agent interacts with the environment and learns an optimal policy for making the decisions using a Deep Q-Network (DQN) [38]. We simulate the interaction between the agent and environment using existing interaction models (e.g., using the Dual Gaussian model [6] to simulate the touchpoints). Each time the agent makes a decision, it receives a reward (utility) from the environment immediately, which is the negative of the time cost of the action. The agent learns the policy via the simulated interaction experience.

Our investigation shows that the SATS agent is able to learn an optimal policy that will trigger suggestions if the input is ambiguous, and will directly select the target if the input is deemed certain. Empirical evaluation shows that SATS significantly improves the accuracy of target selection over existing methods including Shift [51], BayesianCommand [58], and Visual Boundary, which selects a target using its boundaries. SATS also outperforms two other suggestion-based methods: Maximizing Utility for Current Selection (MUCS), which maximizes the utility of the current selection only, and Heuristic, which uses a simple heuristic for triggering suggestions.

2 RELATED WORK

This work is related to improving target selection on touchscreen devices, using decision theoretic approach and reinforcement learning to improve interaction experience with computers.

2.1 Improving Accuracy of Touchscreen Target Selection

Previous research shows that the performance of touchscreen target selection is affected by many factors, including hand postures [10, 21], finger angles [25, 26], body movement [20, 41]. These factors may affect the size and shape of the contact region. Also, the "fat finger" problem prevents the user from having direct feedback on where the touch point is [51, 52]. It further exacerbates the inaccuracy of target selection.

To improve the accuracy of touchscreen target selection, some works modeled the user's target selection behavior using probability and machine learning techniques [6, 9, 55] and carry the uncertainty during the input process [44, 45, 53]. Other works have approached target selection in a probabilistic way, i.e. using techniques such as Bayes Theorem. For example, Goodman et al. proposed a decoding algorithm for soft keyboard based on Bayes Theorem [22]. Bi et al. proposed to use a Gaussian distribution to model the user input distribution, and used it to compute the likelihood for a target to be selected [5]. Zhu et al. [58] further extended the method by incorporating a prior distribution for the targets based on input history, and proposed BayesianCommand, which selects the target with the highest posterior.

To provide more visual feedback, some works compensated the touchpoint offset based on the finger input angle [25, 26], or location on the screen [24]. And some utilized the back of the device as the input interface [16, 56]. Other approaches utilized different input methods such as gestures [36], sliding [10, 40, 57], rubbing [43], and multi-touch [3] to improve the target selection accuracy. Magnification-based methods are also widely used for high-accuracy target selection. For example, Vogel et al. proposed Shift [51], where a call-out window showing the magnified view of the area underneath the finger can be triggered by the user by long pressing. Before lifting the finger, the user can adjust the finger position to select the intended target.

Showing suggestions could be another approach to resolve the ambiguity in interaction [37]. One challenge of suggestion-based methods is to decide when suggestions should be provided and

what candidates should be included. Our SATS method uses a reinforcement learning-based approach to automatically make these decisions.

2.2 Decision-Theoretic Research in HCI

Decision theory studies how an agent makes decisions [7, 46]. It has been adopted to (1) model user's interaction with computers, or (2) guide the design of interfaces or intelligent interaction techniques.

The first line of research assumes that users' interaction with computers will converge to optimal actions (determined by the utility) which maximize the utility under constraints [27]. Indeed, Howes et al. [29] found that the individual's choices are boundedly optimal via two experiments on remembering internal and external resources. Under this assumption, research has been carried out to model users' interaction with computers. For example, Tseng et al. [50] proposed a computational model of how users perform visual search, where users try to maximize a utility related to the trade-off between speed and accuracy. Chen et al. [13] modeled the rational menu search by setting the goal to maximizing the trade-off between speed and accuracy during using the menu. Jokinen et al. [30] modeled how user learns to locate a key on different keyboards: a keyboard with familiar layouts or new keyboard layouts, based on utility learning. Toomim etc. [49] proposed a crowdsource-based approach to measure how the user makes choices, and quantifies it as a measure of utility.

The second line of research adopts the decision theory to improve the interaction experience with computers. For example, Horvitz [28] proposed a decision-theoretic approach to decide when the agent should suggest actions or intervene the interaction between the user and computers. Todi et al. [48] proposed adapting the menu layout by maximizing the utility, which was defined as the usefulness of a menu adaptation to the user. Lomas et al. [35] modeled the optimized interface design as a Multi-Armed Bandit problem, which optimizes the user engagement. The present work is also an example of research along this line. It proposes a decision-theoretic method to improve the accuracy of touch selection.

2.3 Reinforcement Learning in HCI

Reinforcement learning is a machine learning paradigm where an agent learns to take actions through interacting with its environment so that the expected cumulative reward can be maximized [47]. Typically, the environment is a form of Markov Decision Process (MDP). An MDP consists of the following 5 components $< S, A, T, R, \gamma >$: (1) A set of states S, which describes the state of the environment; (2) A set of actions A, which defines what action the agent can take; (3) A transition function T which defines the transition probability between two states given an action; (4) A reward function R, which defines the goal of the agent in the environment; and (5) A discount factor γ, which defines the agent's preference between seeking immediate or more distant rewards. Theoretically, by solving the *Bellman equation*, we can obtain the best actions to take, which is formally called *policy*, at different state of the environment. Sometimes, if the transition probability (and the reward function) is not known, we can use a model-free algorithm to solve the reinforcement learning problem, e.g. the Q-learning algorithm [54]. One of the recent advances for Q-learning is to use a deep learning technique, Deep Q-learning, which uses a neural network called Deep Q-Network (DQN) [38] to estimate the state-action value function (Q-value) in Q-learning.

In the HCI field, reinforcement learning has been adopted to (1) model user behavior and (2) create intelligent interfaces or interaction techniques. For example, Chen et al. [12] proposed to use a Partially Observable Markov Decision Process to model the eye gaze target selection process. Leino et al. [31] proposed the RL-KLM, a reinforcement learning-based keystroke-level model, which is able to generate user-like operations in simple realistic interaction tasks. Cheema et al. [11] used reinforcement learning to predict the interaction movements and arm fatigue in mid-air interaction. Do et al. [17] proposed a model to simulate point-and-click behavior for both stationary and moving targets. The point-and-click behavior was formulated as an MDP problem, and solved via deep reinforcement learning. Regarding creating intelligent interfaces or interaction techniques, Todi et al. [48] proposed a model-based reinforcement learning approach to design better menu systems. Gebhardt et al. [19] proposed to use reinforcement learning to learn when to show or hide the label of an object to better support visual search task in a mixed reality system. This work is another example of using reinforcement learning to create intelligent interaction techniques. It addresses a common problem in HCI: accurate target selection on touchscreen devices.

3 SATS: A SUGGESTION-BASED ACCURATE TARGET SELECTION METHOD

We propose SATS, a Suggestion-based Accurate Target Selection method. We first formulate whether and which target(s) should be suggested as a sequential decision problem [2]: upon the observation of a touchpoint, the computer as an agent decides whether it should directly select a target, or offer suggestions and ask for confirmation. The decision is sequential because a user may re-select should the current selection fails. The goal of the decision is to maximize the utility, which is the negative time cost of the selection procedure.

Based on this formation, we adopt reinforcement learning to solve this sequential decision problem. The computer as an agent learns an optimal policy using the Deep Q-Network (DQN) [38] based on simulations, where the interaction between the agent and the environment is simulated based on existing interaction models. The learned policy is used to automatically decide what action to take at each step. Fig. 2 shows an overview of how the agent interacts with the environment.

In the remainder of the section, we describe the assumptions for the reinforcement learning framework, the formulation of the problem, the implementation of the reinforcement learning agent SATS, and an alternative solver.

3.1 Theoretical Assumption

Before we dive into the detailed formulation, we introduce some theoretical assumptions as follows:

Touchpoint distribution model. Since we use a simulation-based method to train the agent, it is important to simulate how a user would land the touchpoints in target selection tasks. We adopted the Dual Gaussian model [5, 6] to simulate touch interaction.

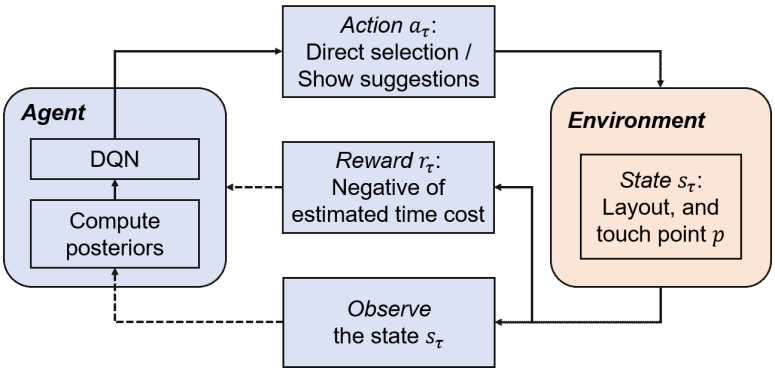


Figure 2: An overview of the reinforcement learning framework. The computer acts as the agent to interact with the environment. After observing a touchpoint made by the user, it decides whether to directly select a target or to show suggestions. If the decision is the latter, it also decides what suggestions should be shown.

The Dual Gaussian distribution model [5, 6] assumes that the touchpoint distribution for selecting a target t follows a Gaussian distribution. The probability of observing a touchpoint p given t as the intended target can be represented as:

$$P(p|t) = \frac{1}{2\pi\sigma_x\sigma_y\sqrt{1-\rho^2}} \exp\left[-\frac{z}{2(1-\rho^2)}\right], \tag{1}$$

where

$$z \equiv \frac{(p_x - t_x)^2}{\sigma_x^2} + \frac{2\rho(p_x - t_x)(p_y - t_y)}{\sigma_x\sigma_y} + \frac{(p_y - t_y)^2}{\sigma_y^2}. \tag{2}$$

(t_x, t_y) is the target center, σ_x and σ_y are the standard deviations of the users' touchpoints, and ρ is the correlation coefficient between x and y. We followed [6] to set $\rho = 0$ and used the following empirical model for σ_x^2 and σ_y^2:

$$\begin{aligned} \sigma_x^2 &= 0.0075 \times d_W^2 + 1.68, \\ \sigma_y^2 &= 0.0108 \times d_H^2 + 1.33, \end{aligned} \tag{3}$$

where d_W and d_H are the width and height (in mm) of the target.

Prior distribution of targets. We assumed that the distribution of the user's intended targets follows a Zipf distribution [42]. Based on the distribution, we can generate target selection frequencies in the simulation and user study. This assumption was made based on previous research in menu selection [14, 34], smartphone APP launching [39], and target triggering [1, 18, 58]. Specifically, the frequency of each target candidate can be modeled by the following equation:

$$f(l; s, N) = \frac{1/l^s}{\sum_{n=1}^{N}(1/n^s)}, \tag{4}$$

where N is the number of target candidates, $l \in \{1, 2, \ldots, N\}$ is the rank of each target candidate, and s is the value of the exponent characterizing the distribution.

While the distribution of the target candidates is not known to the agent, we adopted the frequency model introduced in [58] to estimate the prior probability for each candidate being the intended target prior to observing the touchpoint based on the selection history:

$$P(t_i) = \frac{k + c_i}{k \cdot N + \sum_{j=1}^{N} c_j}, \tag{5}$$

where N is the number of target candidates (e.g., the number of menu items), c_i is the number of times we have observed target t_i being selected. k is the pseudocount, a hyper-parameter of the distribution, and we used $k = 1$ as suggested by [58].

Empirical time cost model. We adopted the empirical models in [4, 14] to estimate the time cost for selecting a target from N target candidates. According to [14], we separated the selection time into a decision time and a touch action time:

$$R(N) = R_d(N) + R_{touch}, \tag{6}$$

where $R(N)$ is the time cost for selecting a target from N candidates, $R_d(N)$ is the time cost of deciding which object to select among the N candidates, and R_{touch} is the time cost of the motor action of selecting the target.

We used the empirical model for decision-making time (in seconds) in [14]: $R_d(N) = 0.08\log_2(N) + 0.24$. We adopted the FFitts Law [4] to estimate touch pointing time. We first used Equation 2 and the empirical parameter values reported by Bi et al.'s [4] to estimate the touch selection times when $ID = 0$ for 1D and 2D targets, and obtained the average value for these two types of targets as the final estimate: $R_{touch} = 0.129$ seconds.

3.2 Problem Formulation

We viewed the target selection problem as a *stochastic sequential decision problem* [2] and used a reinforcement learning approach to solve it. Fig. 3 shows how the problem is described in reinforcement learning framework. For a state s_0 of the environment, the agent can take one action from 4 types of action, including directly selecting a target candidate and showing 2, 3, or 4 suggestions. For each action the agent takes, the agent receives a reward immediately. If it successfully selects the correct target, the environment will enter the terminal state, noted T, otherwise it enters a new state, then the agent will repeat the selection procedure until a successful selection. For the action where the suggestions included the intended target, we assumed that a user successfully selected it because suggestions

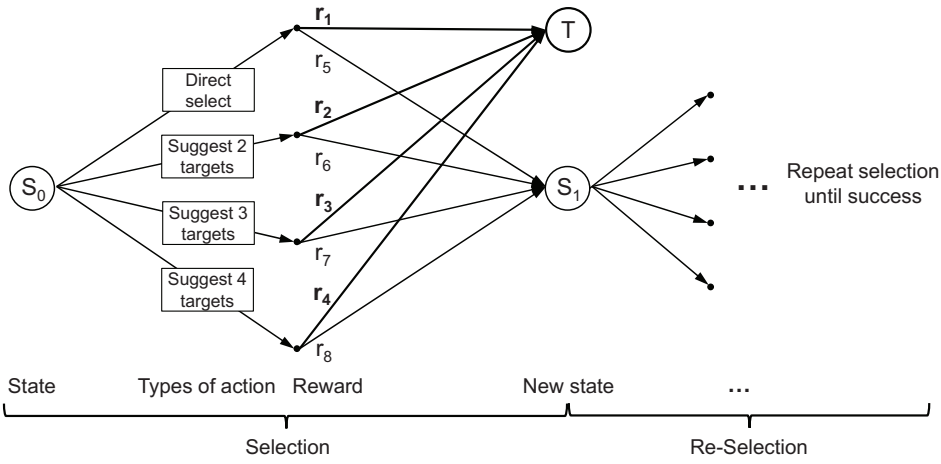


Figure 3: The target selection as a sequential decision problem, where s_i is a state, T is the terminal state, and r_i is a reward. The agent performs selections until successfully select the target.

are often displayed in large size, and the chance of making an error is very slim.

The elements in the reinforcement learning framework are as follows:

- **State**: At each time step τ, the environment has a state $s_\tau \in S$, and $s_\tau = \{T, p\}$, where T is the set of target candidates $T = \{t_1, t_2, \cdots, t_N\}$, including their location information. And $p = (p_x, p_y)$ is a touchpoint made by the user to select the intended target.
- **Observation**: The agent can fully observe the state T and p from the environment at each time step, denoted by o_τ. Based on the observation, the agent first computes the posterior for each candidate being the intended target following the Bayesian method introduced in BayesianCommand [58]. Specifically, we used Eq. 1 as the likelihood, and Eq. 5 to update the prior. Then the agent takes the top K posteriors among all target candidates as the input for the DQN. We chose $K = 4$ in the current implementation as in a common grid menu a touchpoint mostly has 4 closest target candidates which may cause ambiguity in selection.
- **Action**: The agent can take an action a_τ at each time step. The action space is constructed based on the four target candidates with the highest posteriors. The agent can select one target candidate directly, or show suggestions with 2, 3, or 4 target candidates. To include all possible actions, we consider all non-empty subsets of the four target candidates. Therefore, there are 15 actions, which can be categorized into four types: (1) Directly select a target candidate, (2) Suggest 2 target candidates, (3) Suggest 3 target candidates, (4) Suggest 4 target candidates. For example, a possible action can be: Suggest the target candidates with the 1st, 2nd, and 3rd highest posteriors.
- **Reward**: After taking an action, the agent receives a reward immediately from the environment. We set the reward as the negative value of the estimated time cost of each action based on Eq. 6. The time cost consists of three parts: (1) Select the intended target from N target candidates: $R(N)$. (2) Select the intended target from the suggestions if n suggestions are shown: $R(n)$. (3) Cancel the incorrect selection if it happens. We assumed the user needs to select a predefined cancel button, and the time cost is a constant $R(1)$ as the cancel button is at a fixed position on a particular UI. Following these assumptions, Table 1 summarizes the reward of the different actions in Fig. 3.
- **Discount rate** $0 \leq \lambda \leq 1$: The model receives a scalar reward after taking each action. By maximizing the value $E[\sum_{\tau=0}^{\infty} \lambda^\tau r(s_\tau, a_\tau)]$, the algorithm can derive the optimal policy.

We used a model-free algorithm, DQN [38], to learn the optimal policy. We chose DQN because it can handle the policy learning problem in which the state space is continuous and the action space is discrete [33]. The target selection problem is a good fit for DQN as the touchpoint p could be anywhere on the screen (i.e., continuous state space) and there are only a limited number of actions available (i.e., discrete action space). Instead of passing the raw touchpoint and layout information to the DQN, which is computationally expensive, the DQN takes the top four posteriors among the target candidates as input and outputs an action for the agent to take. In a grid layout, one touchpoint is adjacent to at most 4 candidates. Therefore, the candidate set with the four highest posteriors will likely include the intended target.

3.3 Implementation

We implemented the target selection environment within OpenAI Gym [8]. We were able to freely configure the layout and the target size for the environment, e.g. 4×6 grid layout and 3.5 mm target size (we assumed that the targets have the same width and height). The frequency of each target candidate to be selected was modeled by Zipf's law, as shown in Eq. 4. Given an intended target, the environment can generate a touchpoint by using the Dual Gaussian distribution in Eq. 1 to simulate user input. For the actions, if the agent chooses to show suggestions and the suggestions contain the

Selection result	Direct selection	Suggest 2 targets	Suggest 3 targets	Suggest 4 targets
Correct	$r_1 = -R(N)$	$r_2 = -R(N) - R(2)$	$r_3 = -R(N) - R(3)$	$r_4 = -R(N) - R(4)$
Incorrect	$r_5 = r_1 - R(1)$	$r_6 = r_2 - R(1)$	$r_7 = r_3 - R(1)$	$r_8 = r_4 - R(1)$

Table 1: The reward settings for the four kinds of actions when they select the correct and incorrect target. An incorrect selection has $R(1)$ as extra cost because users need to cancel the current selection.

correct target, we assumed it is a successful selection. After each trial, the agent estimates the prior probability by Eq. 5 based on the selection frequency in the history.

We used the model-free reinforcement learning algorithm DQN, and its implementation in the Stable-Baselines3 library[1] to solve the problem. The library is a set of implementations of reinforcement learning algorithms using Pytorch. The DQN was trained based on a replay buffer. The hyperparameters to train the DQN were as follows: MlpPolicy (a 3-layer neural network, hidden size: 64, activation function: ReLU) as the policy network, batch_size=128, total_timesteps=200,000. Other parameters were set to the default as in the Stable-Baselines3 library: learning_rate=0.0001, exploration_fraction=0.1, optimizer: Adam, gamma=0.99. The policy network usually converged after fewer than 600 updates. After the DQN is trained, we can load the pretrained policy network on Android/iOS devices using the Pytorch Mobile runtime[2].

3.4 Maximizing Utility for Current Selection (MUCS) as An Alternative Solver

Besides reinforcement learning, we may solve the decision problem with a Maximizing Utility for Current Selection (MUCS) based method. This method calculates the expected utility of each action, and executes the one with the highest utility. The expected utility of an action is calculated based on the posterior of the action being correct and its reward. More specifically, the expected utility of an action a (denoted by $EU(a)$) is calculated as:

$$EU(a) = P(a) \times \text{Reward of correct action} + (1 - P(a)) \times \text{Reward of incorrect action},$$

where $P(a)$ is the posterior probability of a being a correct action. If the action a is the direct selection, $P(a)$ is the posterior probability of the selected target being the intended target, calculated based on the method proposed in the BayesianCommand work [58]. If the action a is showing 2/3/4 suggestions, $P(a)$ is the sum of the posterior probabilities of the suggested targets. The utility of correct actions is the same as defined in Table 1. The utility for incorrect actions is the time cost defined in Table 1 minus $R(N)$, as we assume that the user will re-select if the current selection fails and he/she will succeed in re-selection. Taking the action of "showing two suggestions" as an example, its utility $EU(a)$ under these assumptions is defined as:

$$EU(a) = P(a) \times r_2 + (1 - P(a)) \times (r_6 - R(N)).$$

The main difference between this MUCS method and SATS is that MUCS only maximizes the reward for the current selection while SATS maximizes the accumulated reward of the whole selection procedure until a successful selection. We adopted reinforcement learning to solve the sequential decision problem as it can take future actions into consideration and maximize the reward beyond the current selection action. Nevertheless, we empirically compared these two methods and found that SATS outperformed MUCS (see details in Section 5). Next, we investigate how training factors would affect the performance of SATS via simulation.

4 SIMULATION BASED EVALUATION OF SATS

We first adopted a simulation-based approach to evaluate the performance of SATS, using the aforementioned Dual Gaussian model [5, 6] to simulate touch interaction. We evaluated how different training factors, including the s parameter in Zipf's law, the menu layout and the target size, affect the performance of SATS.

The evaluation metrics are: (1) Error rate: it is the number of trials where SATS makes at least one selection error divided by the total number of trials. (2) Selection time: it is the duration starting from the beginning of a trial to the moment that the intended target is finally selected. This time cost was estimated by the empirical model (Eq. 6).

4.1 Effects of Distribution of Target

As our model assumes that the distribution of intended targets in a menu layout follows Zipf's law, we firstly evaluated how varying the s value of Zipf's law in the training data affects the performance of SATS.

We trained a policy network under a specific s value of Zipf's law (with a layout and a target size), and tested it on environments with identical and different s values. More specifically, we trained three types of policy networks with $s = 1$, $s = 2$, and $s = 3$, respectively, and evaluated each of them on three types of testing environments with $s = 1$, $s = 2$, and $s = 3$, respectively. In both training and testing, the menu was a 4×6 grid layout, and the target sizes were 2.5 mm and 3.5 mm. The average error rates of policy networks trained on different s ($s = 1, s = 2, s = 3$) of Zipf's law across the testing environments were 6.48%, 7.35%, and 7.88% respectively, and the average selection times were 0.932, 0.932, and 0.921 seconds respectively. The difference between policies trained under different s value was small: there was only around 1% difference in error rate and 0.01 s difference in selection time, indicating that the s value in training has a minor influence on SATS's performance.

4.2 Effects of Layout

We also evaluated the performance of SATS trained on different target layouts using a similar approach. Specifically, we trained three types of policy networks with 4×6, 6×8, 10×10 layout respectively. For each type of policy network, we evaluated them on three types of testing environments with 4×6, 6×8, 10×10

[1] https://github.com/DLR-RM/stable-baselines3
[2] https://pytorch.org/mobile/home/

layout respectively. In both training and testing, we used $s = 1$ and target sizes 2.5 mm and 3.5 mm.

The average error rates of policy networks trained on different layouts (4×6, 6×8, 10×10) across the testing environments were 10.37%, 10.45%, and 9.17% respectively, and the average selection times were 1.185, 1.187, and 1.179 seconds respectively. The difference between policies trained under different layouts was small, which indicates that the layout in training has a minor influence on SATS's performance.

4.3 Effects of Target Size

We then evaluated the performance of SATS trained on datasets generated by a single target size and a combination of target sizes. Specifically, we trained four policy networks with target size 2.5 mm, 3.5 mm, (2.5 and 3.5 mm) together, (2.5, 3, 3.5, and 4 mm) together, where $s = 1$ and layout is 4×6. For the last two policy networks, we generated the training dataset using two/four environments with the corresponding target size. We evaluated the four policy networks on two testing environments with setting $s = 1$, 4×6 layout and two target sizes, 2.5 mm and 3.5 mm respectively. Note that the policy networks trained on 2.5 mm and 3.5 mm were only evaluated on the testing environment with the same target size, and we averaged the results for the two models.

The average error rates of policy networks trained on different target size (same as the testing environment, [2.5, 3.5 mm] together, [2.5, 3, 3.5, 4 mm] together) across the testing environments were 11.3%, 12.4%, and 12.45% respectively, and the average selection times were 1.102, 1.107, and 1.106 seconds respectively. The policy network trained for specific target size and the policy network trained on a combination of target sizes has similar performance, indicating that it is not necessary to train a policy network for different target sizes.

Overall, our simulation-based evaluation shows that the distribution of target, layout, and target sizes in the training settings have only minor effects on the performance of trained policy networks.

4.4 A General Policy Network for SATS

Based on the above result, we trained a general policy network based on the setting $s = 1$, 10×10 layout, 4 target sizes (2.5, 3, 3.5, 4 mm together), and compared it with policy networks trained with specific settings as in the testing environment. We evaluated the policy networks in the testing environments in Section 4.1 and 4.2 (10 testing environments in total). We found that the average performance of the general policy network is similar to the policy network trained on specific settings. Specifically, the testing error rates were 8.03% and 7.65% respectively, and the testing selection times were 1.052 and 1.050 seconds respectively. Therefore, we used this general policy network for SATS.

In the next sections, we report three controlled experiments that systematically evaluated SATS. In User Study I, we compare SATS with two suggestion-based methods: (1) MUCS, which maximizes the utility of the current selection, and (2) Heuristic, which uses a simple heuristic for triggering suggestions based on the highest posterior. Such a comparison will tell us whether using reinforcement learning can improve suggestion accuracy. In User Study II, we compare SATS with methods that directly selected targets without showing any suggestions. In User Study III, we compare SATS with Shift [51], in which a user manually triggers a call-out window and adjusts the input finger to confirm the target selection. The comparison with Shift will tell us whether automatically providing suggestions can improve target selection performance over manually triggering a magnification window.

5 USER STUDY I: COMPARE SATS WITH SUGGESTION BASED TARGET SELECTION METHODS

To understand whether SATS improves performance over other suggestion-based methods, we conducted a user study to compare SATS with (1) MUCS, as described in Section 3.4, and (2) a simple heuristic-based method which we refer to as Heuristic. Heuristic directly selects the top candidate if its posterior exceeds a threshold δ; otherwise Heuristic suggests the top four candidates according to their posteriors. We set the threshold δ to 0.7 via a simulation-based study. In the study, we performed a grid search in the range of [0.5, 0.95] with a step size of 0.05 to search for the optimal δ, which minimized the estimated selection time for 1000 simulated trials. We assumed that the user would repeat the trial until the selection succeeded, and the time cost of selection actions was calculated according to Table 1. The simulation study suggested that $\delta = 0.7$ had the lowest selection time within the range [0.5, 0.95].

5.1 Participants and Apparatus

Eighteen adults (4 female, 14 male) between 21 and 36 years old (average 27.7±3.6) participated in the study. Seventeen of them used the right index finger, and the other one used the right middle finger for target selection. Ten participants had the experience of using Android/Apple watches or other small-screen devices. We used a Ticwatch S smartwatch with a 45mm diameter screen. The watch runs Google's Wear OS, which is a version of the Android operating system designed for smartwatches and other wearables.

5.2 Design

We adopted a [3×2] within-subjects design. There were two independent variables: (1) target selection method with 3 levels (SATS, MUCS, Heuristic), (2) target size with 2 levels (2.5 mm and 3.5 mm). The target sizes were similar to those used in the previous work BayesianCommand [58] and Shift [51], which represented the small-sized targets a user might encounter on a small touchscreen device (e.g., a hyperlink on a webpage is approximately 2.5 mm wide on a smartphone [4]). Similar to BayesianCommand, we used a 4×6 grid layout to display 24 items, as shown in Fig. 4a. A grid layout reflects the commonly used layout for presenting menu items or icons on touchscreen devices. We randomly selected 12 items as target candidates and used the same set of target candidates across participants and conditions. For each method × target size condition, there were 2 blocks, each with 50 trials. Before the 100 formal trials, the user practiced 5 trials for each method. The frequency of each target to be selected was generated based on Zipf's law, where the parameter s was 1, and the total trials was 50. The generated frequencies were 16, 8, 5, 4, 3, 3, 2, 2, 2, 2, 2, 1. The frequencies were assigned randomly to the 12 target candidates, and the order of target candidates was randomized. Similar to [1, 23, 58], we balanced

the frequency assignments on the 12 target candidates across all participants and conditions, so that each target item was selected an equal number of times. We also counterbalanced the orders of the 6 (3 selection methods × 2 target sizes) experimental conditions across 18 participants according to a Latin Square.

5.3 Procedure

The participant was first introduced to the target selection task where they selected targets with finger touch on a smartwatch. They were allowed to use whatever fingers they preferred. There were 24 items shown in a 4 × 6 grid layout, and the target to be selected was shown above the grid layout, as shown in Fig. 4a.

(a) (b) (c)

Figure 4: Implementation for user study I. (a) Target selection page: the participants will select a target in the menu layout bounded by the grey rectangle, (b) Suggestion in SATS: SATS shows suggestions with three target candidates (in the red rectangle). (c) Selection result page: once a target candidate is selected, the participant needs to confirm the selection result by clicking 'OK' if the selection is correct, 'RESELECT' otherwise.

For the three methods, we told the participant that a suggestion window may show up after a touch, e.g. Fig. 4b shows a suggestion window with 3 target candidates, and the participant can choose from the suggested target candidates. The suggested target candidates were ordered by their posteriors and were shown in one row, or in a 2×2 grid if there were four target candidates. The target size in the suggestion window was set to 2.5 times the target size in the grid menu to ensure that the participant can easily select them.

For all methods, the participant needed to decide whether to cancel the selection after a target candidate was selected. We used a selection result page as shown in Fig. 4c. If an incorrect target was selected, the "OK" button was disabled and the participant selected the "RESELECT" button to cancel the selection. If the correct target was selected, the "RESELECT" button was disabled and the participant selected the "OK" button to proceed to the next trial. We included the disabled button to make sure that the participant would redo the trial if the target selection failed, and would advance to the next trial if the target selection succeeded.

The study started after we introduced the task. Fig. 5 shows a participant doing the experiment. The participant was asked to finish the target selection as accurately and quickly as possible. At the end of the study, participants were asked to rate their preference over the three methods on a scale of 1 to 5 (1: dislike, 5: like very much). They also answered a subset of NASA-TLX questions to measure the workload of the target selection task, including mental and physical demand. The rating was on a scale from 1 to 10, from least demanding to most demanding. The study for each participant lasted about 30 minutes. The study in total included 18 participants × 3 methods × 2 target sizes × 100 trials = 10,800 trials.

Figure 5: A participant is selecting a 3.5 mm target

5.4 Results

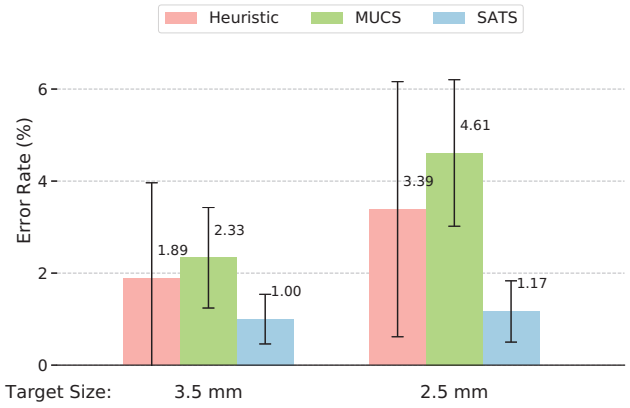
5.4.1 Error Rate. Error rate is the number of trials in which the participant makes at least one selection error divided by the total number of trials. The results for error rate (Fig. 6a) showed that for both target sizes, SATS had the lowest and MUCS had the highest error rate.

A repeated measures ANOVA showed significant main effects for target selection method ($F_{2,34} = 4.86$, $p < 0.05$) and target size ($F_{1,17} = 4.78$, $p < 0.05$) on error rate. No significant interaction effect was observed between these two independent variables ($F_{2,34} = 1.53$, $p = 0.23$). Pairwise mean comparisons with Bonferroni adjustments showed that the difference was significant for SATS vs. MUCS ($p < 0.001$), but not for SATS vs. Heuristic ($p = 0.22$) or MUCS vs. Heuristic ($p = 0.81$). Further, Cohen's effect size values suggested a large difference for SATS vs. MUCS ($d = -0.85$), and small differences for SATS vs. Heuristic ($d = -0.31$) and MUCS vs. Heuristic ($d = 0.19$).

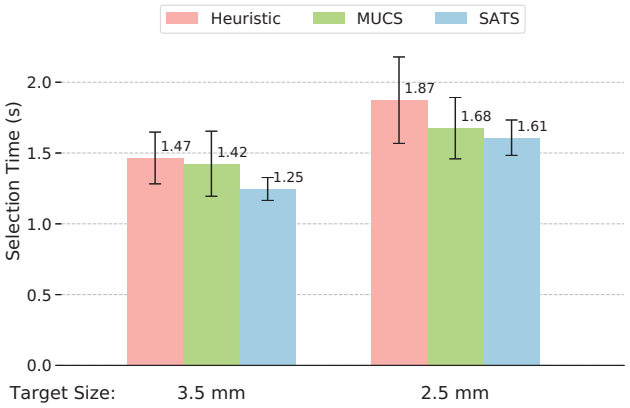
5.4.2 Selection time. Selection time is the duration starting from the beginning of a trial to the moment that the intended target is finally selected. The results for selection time (Fig. 6b) showed that SATS achieved the fastest input speed, while Heuristic had the longest selection time.

A repeated measures ANOVA showed significant main effects for target selection method ($F_{2,34} = 3.44$, $p < 0.05$) and target size ($F_{1,17} = 29.57$, $p < 0.001$) on selection time. No significant interaction effect was observed between the two independent variables ($F_{2,34} = 0.839$, $p = 0.44$). Pairwise mean comparisons with Bonferroni adjustment showed that the difference was significant for SATS vs. Heuristic ($p < 0.05$), but there was no significant difference between SATS vs. MUCS ($p = 0.28$) and MUCS vs. Heuristic ($p = 0.46$). Furthermore, Cohen's effect size values suggested small differences for SATS vs. MUCS ($d = -0.29$) and MUCS vs. Heuristic ($d = -0.24$), and a medium difference for SATS vs. Heuristic ($d = -0.50$).

We also analyzed the total numbers of suggestions that were triggered by SATS, MUCS, and Heuristic in Fig. 7. MUCS showed the



(a) Mean error rate with 95% CI



(b) Mean selection time with 95% CI

Figure 6: Results for user study I by target size × method. SATS outperformed MUCS and the Heuristic in both error rate and selection time.

least suggestions among the three methods, which explains why it had the highest error rate. On the other hand, Heuristic showed the most suggestions, which explains why Heuristic had the longest selection time. By using reinforcement learning to dynamically show suggestions, SATS outperformed MUCS and Heuristic in both error rate and selection time.

As SATS takes future actions into consideration when learning the policy, its reward estimation is closer to the actual cost of selection than that of MUCS as participants repeated selection until successfully selecting the intended target. SATS sometimes recommended different actions than MUCS, resulting in the difference in performance. Fig. 8 is an example showing MUCS and SATS recommending different actions.

5.4.3 Subjective Feedback. The results for subjective feedback are shown in Fig. 9. For overall preference, the median ratings of SATS, MUCS, and Heuristic were 4.0, 4.0, and 3.0 respectively. The median ratings for mental demand were 3.0, 4.5, and 5.5 for the three methods respectively, and the median ratings for physical demand were 3.0, 5.0, and 6.0 respectively. Non-parametric Friedman tests

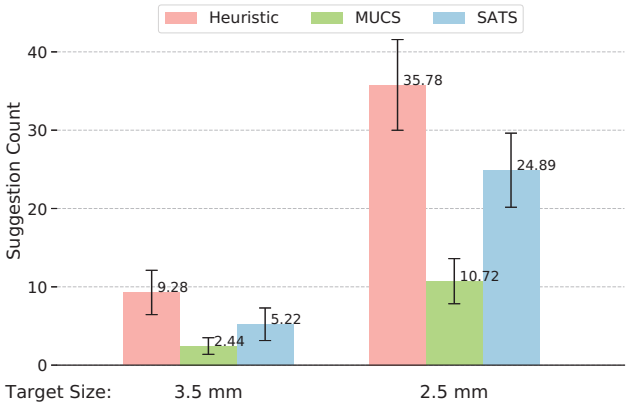


Figure 7: Suggestion count (95% CI) for SATS, MUCS and Heuristic per 100 trials

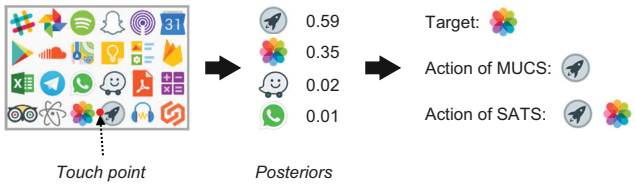


Figure 8: An example where SATS suggested the correct target while MUCS selected an incorrect one

showed significant main effects of selection methods on two metrics: overall preference ($X_r^2(2) = 9.58, p < 0.01$) and physical demand ($X_r^2(2) = 10.98, p < 0.01$). There was no significant main effect of mental demand ($X_r^2(2) = 5.88, p = 0.053$).

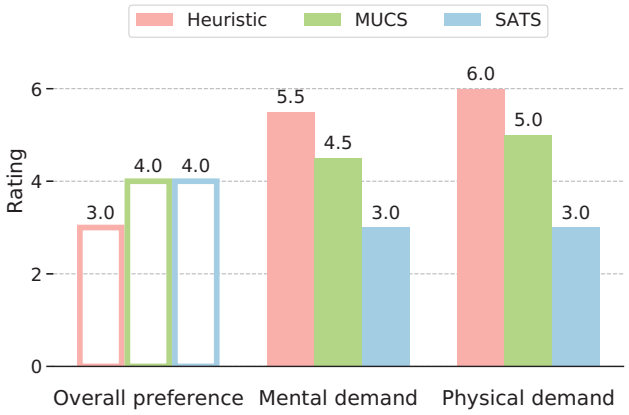


Figure 9: Median subjective ratings of overall preference, mental demand, and physical demand in study I. For overall preference, higher ratings are better. For mental and physical demand, lower ratings are better.

6 USER STUDY II: COMPARE SATS WITH DIRECT SELECTION BASED METHODS

In this study, we compared SATS with two direct selection-based methods, BayesianCommand and Visual Boundary (direct touch). We included BayesianCommand and Visual Boundary as baselines because: 1) BayesianCommand is a high-accuracy, direct selection-based method, and 2) Visual Boundary is the most commonly used method.

6.1 Participants and Apparatus

Eighteen adults (7 female, 11 male) between 24 and 36 years old (average 28.7±2.9) participated in the study. 17 of them used the right index finger to select targets, the other one used the left index finger. 11 participants had the experience of using Android/Apple watch or other small-screen devices. We also used a Ticwatch S smartwatch with a 45mm diameter screen as in study I.

6.2 Design and Procedure

We adopted a [3 × 2] within-participant design. There were two independent variables: (1) target selection method with 3 levels (SATS, BayesianCommand, and Visual Boundary), (2) target size with 2 levels (2.5 mm and 3.5 mm).

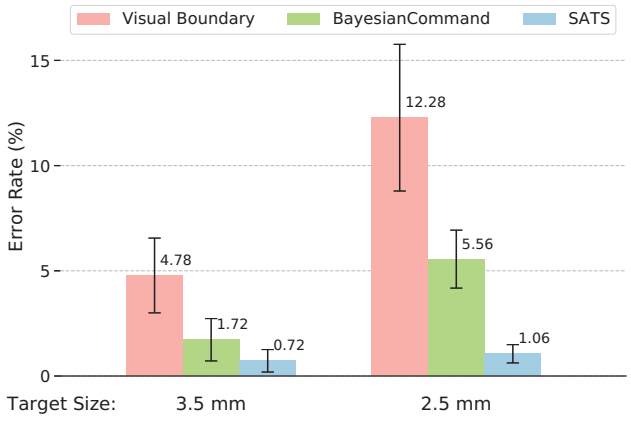
The BayesianCommand and Visual Boundary method did not show any suggestions, and a target candidate was selected once the user lifted the finger off the screen. BayesianCommand selected the target candidate with the highest posterior. The Visual Boundary selected the target candidate whose boundaries contained the touchpoint. Other settings for the study, i.e. the targets arrangement and procedure, were the same as in study I. The study for each participant lasted about 30 minutes. The orders of 6 experimental conditions (3 selection methods × 2 target sizes) were counterbalanced across 18 participants using a Latin Square. In total, we collected 18 users × 3 methods × 2 target sizes × 100 trials = 10,800 trials.

6.3 Results

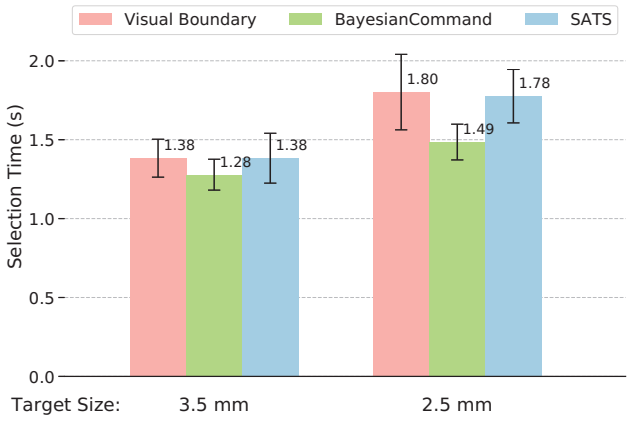
6.3.1 Error Rate. The results for error rate (Fig. 10a) showed that for both target sizes, SATS had the lowest error rate and Visual Boundary had the highest error rate.

A repeated measures ANOVA showed significant main effects for target selection method ($F_{2,34} = 40.13$, $p < 0.001$) and target size ($F_{1,17} = 52.46$, $p < 0.001$), and a significant interaction effect method × target size ($F_{2,34} = 15.28$, $p < 0.001$) on error rate. Pairwise mean comparisons with Bonferroni adjustment showed the differences were significant for SATS vs. BayesianCommand ($p < 0.001$), SATS vs. Visual Boundary ($p < 0.001$), and BayesianCommand vs. Visual Boundary ($p < 0.001$). Further, Cohen's effect size values suggested large differences for SATS vs. BayesianCommand ($d = -0.94$), SATS vs. Visual Boundary ($d = -1.19$), and BayesianCommand vs. Visual Boundary ($d = -0.87$).

We also presented the statistical information of how many times the suggestions were triggered for SATS and the attempt count for the three methods to successfully select the targets in 100 trials, as shown in Table 2. SATS had the smallest attempt count, while it had a relatively high suggestion count, especially when the target size is 2.5 mm. This is because when the target is too small, input is more likely to be ambiguous.



(a) Mean error rate with 95% CI



(b) Mean selection time with 95% CI

Figure 10: Results for user study II by target size × method. Both SATS and BayesianCommand outperformed the baseline Visual Boundary. SATS outperformed BayesianCommand in error rate, but slightly increased selection time (around 200 ms).

6.3.2 Selection Time. The results for selection time (Fig. 10b) showed that SATS has almost the same selection time as Visual Boundary, while the average selection time for BayesianCommand was about 200 ms lower than the other two methods.

A repeated measures ANOVA showed significant main effects for target selection method ($F_{2,34} = 5.97$, $p < 0.01$) and target size ($F_{1,17} = 42.86$, $p < 0.001$) on selection time. No significant interaction effect was observed. Pairwise mean comparisons with Bonferroni adjustment showed the differences were significant for SATS vs. BayesianCommand ($p < 0.01$) and BayesianCommand vs. Visual Boundary ($p < 0.01$), but not for SATS vs. Visual Boundary ($p = 1$). Further, Cohen's effect size values suggested medium differences for SATS vs. BayesianCommand ($d = 0.58$) and BayesianCommand

Target size	Suggestion count	Attempt count		
	SATS	SATS	BayesianCommand	Visual Boundary
3.5 mm	4.33 (1.46)	100.78 (0.63)	101.83 (1.06)	105.40 (2.08)
2.5 mm	23.61 (3.98)	101.06 (0.43)	105.78 (1.42)	115.11 (5.08)

Table 2: Suggestion count and attempt count (95% CI) for three methods in 100 trials

vs. Visual Boundary ($d = -0.55$), and a small difference for SATS vs. Visual Boundary ($d = -0.03$).

6.3.3 Subjective Feedback. The results for subjective feedback are shown in Fig. 11. For overall preference, the median ratings of SATS, BayesianCommand, Visual Boundary were 5, 4, and 3 respectively. SATS had the highest median preference. The median ratings for mental demand were 3.5, 5.0, and 6.5 for the three methods respectively, and the median ratings for physical demand were 3.0, 4.0, and 5.5 respectively. Non-parametric Friedman tests showed significant main effects for selection method on two metrics: overall preference ($X_r^2(2) = 15.10, p < 0.001$) and mental demand ($X_r^2(2) = 12.04, p < 0.01$). There was no significant main effect of physical demand ($X_r^2(2) = 5.72, p = 0.057$).

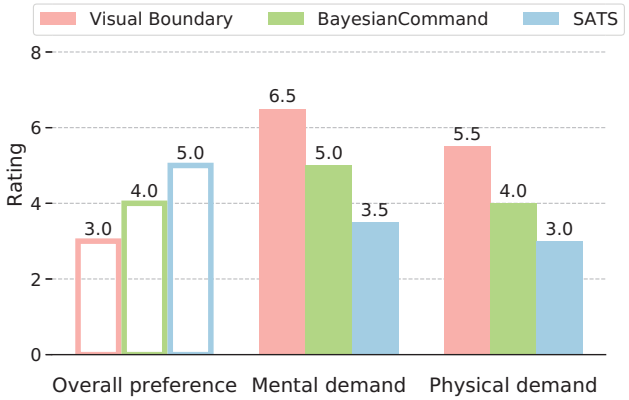


Figure 11: Median subjective ratings of overall preference, mental demand and physical demand in study II. For overall preference, higher ratings are better. For mental and physical demand, lower ratings are better.

7 USER STUDY III: COMPARE SATS WITH SHIFT

In this experiment, we compared SATS with Shift [51], a magnification-based method in which a user can manually trigger a call-out window and adjust the finger to confirm the target selection. As magnification-based methods have been widely adopted on touchscreen devices (e.g., iPhone) to assist target selection, we wanted to assess whether automatically providing suggestions could outperform manually triggering a magnification window.

7.1 Participants and Apparatus

Twelve adults (4 female, 8 male) between 21 and 36 years old (average 28±4.41) participated in the study. They selected targets using the preferred input fingers. Nine of them used the right index finger for selection, two used the left index finger, and another one used the right thumb. 8 participants had the experience of using Android/Apple watch or other small-screen devices. We also used a Ticwatch S smartwatch with a 45mm diameter screen as in study I and II.

7.2 Design and Procedure

We adopted a [2 × 2] within-participant design, where the two independent variables were: (1) target selection method with 2 levels (SATS and Shift), (2) target size with 2 levels (2.5 mm and 3.5 mm). Similar to the previous two user studies, for each method × target size condition, there were 2 blocks, each with 50 trials. Before the 100 formal trials, the user practiced 5 trials for each method.

For Shift, a call-out window showing the view underneath the user's finger magnified 1.5 times appeared right after the user touched the screen, as shown in Fig. 12. To maintain high efficiency, the call-out window appeared as soon as the finger touched the screen. Such a design required no dwelling time for triggering the call-out window. This feature was close to the original work [51] which set an extremely short dwelling time (<5 ms) for targets smaller than 3.5 mm. The call-out window remained static as the finger was on the screen, and it disappeared as soon as the input finger was lifted. Before lifting the finger, the user can move the finger to adjust the touch position, and as feedback, a red cross showed the real-time touch point in the call-out window. The user may also ignore the call-out window and directly select the target with finger touch. To accommodate the limited screen size of the watch, we set the diameter of the call-out window to 8.75 mm (100 pixels on the watch), and placed it 8.75 mm above the initial contact point of the finger.

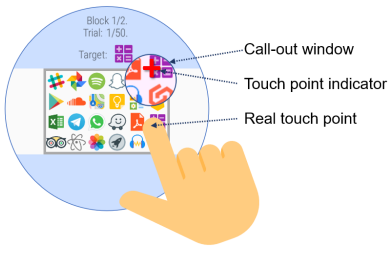
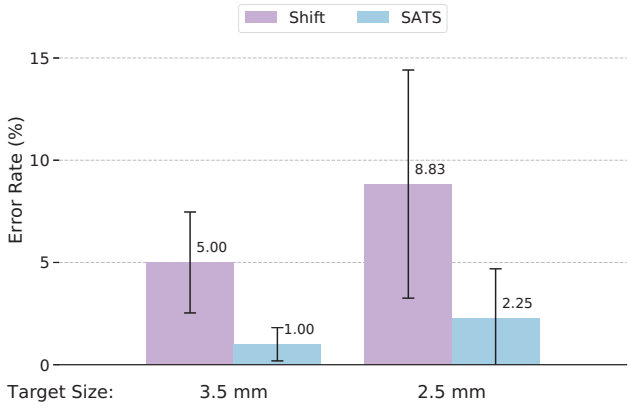


Figure 12: Implementation of Shift

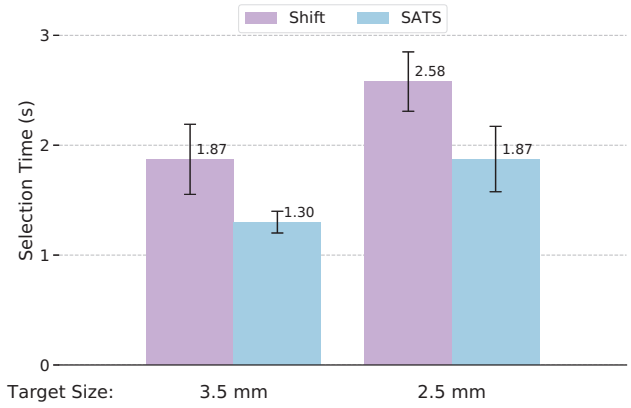
Other settings for the study, i.e. targets arrangement and procedure, were the same as in the study I and II. The user study for each user lasted about 20 minutes. The orders of the four experimental conditions (2 selection methods × 2 target sizes) were counterbalanced across participants using a Latin Square. In total, we collected 12 users × 2 methods × 2 target sizes × 100 trials = 4,800 trials.

7.3 Results

7.3.1 Error Rate. We plotted the mean error rate with 95% confidence interval (CI) for SATS and Shift in Fig. 13a. SATS outperformed Shift in error rate in both target sizes.



(a) Mean error rate with 95% CI



(b) Mean selection time with 95% CI

Figure 13: Results for user study III by target size × method. SATS outperformed Shift in both error rate and selection time.

A repeated measures ANOVA showed significant main effects for target selection method ($F_{1,11} = 9.92$, $p < 0.01$), and target size ($F_{1,11} = 5.10$, $p < 0.05$) on error rate. No significant interaction effect was observed between the two independent variables ($F_{1,11} = 5.01, p = 0.38$). Pairwise mean comparisons with Bonferroni adjustment indicated that the difference was significant for SATS and Shift ($p < 0.01$). Cohen's effect size value suggested a medium difference for SATS vs. Shift ($d = -0.70$) [15]. As reported in Section 7.2, the 100 trials of each user for each method × target size condition was divided into 2 blocks. The mean error rates of SATS for 3.5 mm targets were 1.17% for block 1, and 0.83% for block 2, and for 2.5 mm targets were 2.5% for block 1, and 2.0% for block 2. The mean error rates of Shift for 3.5 mm targets were 3.67% for block 1, and 6.33% for block 2, and for 2.5 mm targets were 9.5% for block 1, and 8.17% for block 2. As shown, the error rates were similar across blocks for each method × target size condition, except for Shift × 3.5 mm target where block 2 had higher error rate.

7.3.2 Selection Time. The results for selection time (Fig. 13b) showed that SATS also outperforms Shift in selection speed. A repeated measures ANOVA showed significant main effects for target selection method ($F_{1,11} = 41.94, p < 0.001$) and target size ($F_{1,11} = 41.26, p < 0.001$) on selection time. No significant interaction effect between selection method and target size was observed ($F_{1,11} = 0.30, p = 0.59$). Pairwise mean comparisons with Bonferroni adjustment showed that the difference was statistically significant between the two selection methods ($p < 0.001$). Cohen's effect size value suggested a large difference for SATS vs. Shift ($d = -1.20$). The mean selection times of SATS for 3.5 mm targets were 1.38 s for block 1, and 1.22 s for block 2, and for 2.5 mm targets were 2.08 s for block 1, and 1.67 s for block 2. The mean selection times of Shift for 3.5 mm targets were 1.88 s for block 1, and 1.86 s for block 2, and for 2.5 mm targets were 2.62 s for block 1, and 2.54 for block 2. As shown, for each method × target size condition, the selection times were similar across blocks for Shift, while the select time for SATS in the block 2 was shorter than that in the block 1.

7.3.3 Suggestion Usage. SATS on average showed suggestions in 6.5 (95% CI 2.62) trials per 100 trials for 3.5 mm target: 3.17 trials for 2 suggestions, 1.5 trials for 3 suggestions, and 1.83 trials for 4 suggestions. For 2.5 mm targets, it showed suggestions for 26.5 (95% CI: 8.739) per 100 trials: 2.3 trials for 2 suggestions, 5.3 trials for 3 suggestions, and 19 trials for 4 suggestions. It matched our expectation that suggestions would be more frequently displayed for trials with smaller targets. In Shift, the call-out window appeared as soon as the finger landed on the screen. Therefore it was turned on for all trials. Such a design followed the previous research [51] where a negligible dwelling time (< 5ms) was adopted for small target (< 5.2 mm) selection.

7.3.4 Subjective Feedback. The results for subjective feedback are shown in Fig. 14. They indicate that the participants had a higher preference for SATS than Shift, with median ratings 5 and 2.5 respectively. The median ratings for mental demand were 3.0 and 6.5 for the two methods respectively, and the median ratings of the physical demand were 3.5 and 8.0 respectively. The Wilcoxon signed-rank tests showed that the differences were statistically significant for user preference ($p < 0.001$), mental demand ($p < 0.001$), and physical demand ($p < 0.001$).

8 DISCUSSION

8.1 Performance of SATS

The user study results show that SATS achieved higher target selection accuracy over other suggestion-based approaches, direct selection-based approaches, as well as a magnification-based

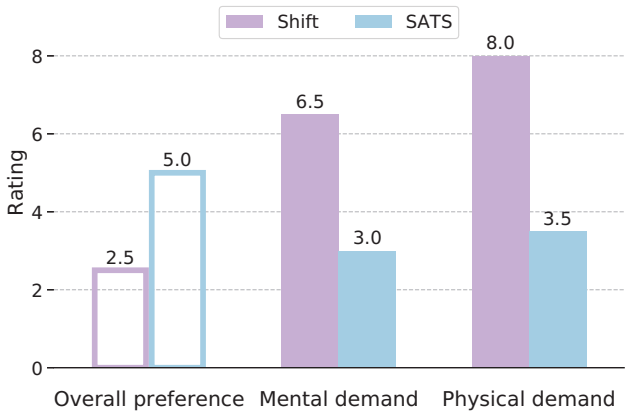


Figure 14: Median subjective ratings of overall preference, mental demand and physical demand in study III. For overall preference, higher ratings are better. For mental and physical demand, lower ratings are better.

method. Compared with the suggestion-based method MUCS, SATS reduced error rate from 3.47% to 1.09%, and reduced selection time from 1.55 seconds to 1.43 seconds, a 7.74% reduction. Meanwhile, compared to Heuristic, SATS reduced error rate from 2.64% to 1.09%, and reduced selection time from 1.67 seconds to 1.43 seconds, a 14.37% reduction. Compared with the direct selection methods, SATS also reduced the error rate of BayesianCommand from 3.64% to 0.89%, while the selection time of SATS was only around 200 ms slower than BayesianCommand, i.e. 1.58 seconds vs 1.39 seconds. At the same time, SATS had almost the same selection time as the most commonly used Visual Boundary method, which was 1.59 seconds, but reduced error rate from 8.53% down to 0.89%. Compared with the manually triggering magnification window-based approach Shift, SATS reduced error rate from 6.92% to 1.63%, and reduced selection time from 2.23 seconds to 1.59 seconds, a 28.70% reduction.

The extremely low error rate of SATS in our user studies suggests that SATS is a high-accuracy selection method. It will especially be beneficial when touch selection errors are costly, such as in conditions where touch selection errors are non-reversible, e.g. clicking a "pay" button in online shopping or clicking the "send" button in live messaging, or in conditions where correcting errors is tedious, e.g. when the "undo" button is hard to access. That said, the selection time of SATS is around 200 ms longer than BayesianCommand, indicating that in conditions where touch selection errors are not costly and can be easily reversed, BayesianCommand is also an appropriate method.

8.2 Select or Suggest in SATS

To understand when SATS directly selected a target, and when it showed 2, 3, 4 suggestions which could be a combination of any target candidates from the top 4 candidates (see details in Section 3.2), we summarize the statistical information for the top 4 posteriors for the four types of action across the three user studies in Table 3. As shown, SATS directly selected the target if the top posterior was extremely high (i.e., 90%). It included 2, 3, or 4 suggestions if the top posterior dropped to around 59%, 56%, and 48%. Such a policy is well-matched with our expectation that SATS directly selected the target if there was high confidence in determining the target, and it provided suggestions if there was high uncertainty. For actions showing 2, 3, or 4 suggestions, the sum of posteriors of all suggestions was more than 90%, indicating that SATS had a broad coverage of possible target candidates, and was able to adjust the number of suggested candidates to accommodate the drop of the top posterior.

8.3 Execution Time Cost

Although SATS uses a neural network to predict action, it is an efficient method that can be used on devices with limited computational abilities. We recorded the loading time and prediction time of the policy network of SATS for 30 trials on the Android watch. It turned out that the average loading time of the policy network is 1244.17 (95% CI: 33.32) ms and the average prediction time is 17.37 (95% CI: 0.86) ms. Therefore, prediction is very fast, and we can load the policy network once when the system or an application starts, which does not affect the target selection performance.

8.4 Future Work

The policy network of SATS is pre-trained based on simulated interaction experiences. The interaction between the agent and the environment is simulated based on existing interaction models, i.e. the Dual Gaussian model for touch point distribution and the empirical time cost model. This means that SATS uses the same policy network for all users. A possible future work is to fine-tune the pre-trained policy network based on the user's real interaction data to create a personalized target selection method.

The core idea behind SATS, using reinforcement learning to train an optimal action policy, could be generalized beyond touch based target selection. The input to the policy network is the posteriors of target candidates. In theory, the method can work with other target selection methods or modalities as long as the posteriors of selecting targets can be obtained. For example, previous work showed that the posteriors of selecting targets can be calculated for gestural input [58], and for gaze-based input [32]. We could then follow a similar procedure of training the policy network for SATS to obtain an optimized action policy for these two different target (command) selection methods.

9 CONCLUSION

In this paper, we frame the question of whether to suggest candidates and which candidates to suggest in a suggestion-based target selection task as a sequential decision problem, and solve it with reinforcement learning. Our investigation led to SATS, a Suggestion-based Accurate Target Selection method. In SATS, the computer acts as an agent that assists the user in target selection. It learns an optimal policy via a Deep Q-Network trained on simulated interactions between the agent and the environment. Following the optimal policy, the agent shows suggestions if the input is ambiguous, and directly selects the target if the input is certain. Empirical evaluations showed that SATS significantly improved the accuracy

Type of action	# of occurrences	Mean±standard deviation of top 4 posteriors			
	out of 10,226 attempts	Highest	2nd highest	3rd highest	4th highest
Direct selection	8,598 (123)	0.90±0.10	0.06±0.06	0.02±0.03	0.01±0.01
Suggest 2 targets	262 (0)	0.59±0.06	0.38±0.06	0.02±0.01	0.01±0.01
Suggest 3 targets	382 (0)	0.56±0.06	0.31±0.07	0.07±0.03	0.03±0.01
Suggest 4 targets	984 (7)	0.48±0.09	0.26±0.07	0.14±0.05	0.08±0.03

Table 3: Statistics of four types of action introduced in Section 3.2 taken by SATS in 10,226 attempts in the three studies, including the occurrence of actions: total occurrence (occurrence where the direct selection was wrong, or the intended target was not included in the suggestions), and mean±standard deviation of the top-four posteriors.

of target selection over two suggestion-based methods: Maximizing Utility for Current Selection (MUCS), which maximizes the utility of the current selection only, and Heuristic, which uses a simple heuristic for triggering suggestions. Compared to existing approaches, SATS also achieved higher accuracy than Shift [51], BayesianCommand [58] and Visual Boundary, which selects a target using its boundaries.

ACKNOWLEDGMENTS

We thank the anonymous reviewers for their insightful comments, and our user study participants. This work was supported by NIH awards R01EY030085, R01HD097188, NSF awards 1815514, 2125147, 2113485, 1805076, 1936027 and ALS Association grant ALS 20-MALS-538.

REFERENCES

[1] Caroline Appert and Shumin Zhai. 2009. Using strokes as command shortcuts: cognitive benefits and toolkit support. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. ACM, Boston, MA, USA, 2289–2298.

[2] Richard E Bellman and Lotfi Asker Zadeh. 1970. Decision-making in a fuzzy environment. *Management science* 17, 4 (1970), B–141.

[3] Hrvoje Benko, Andrew D Wilson, and Patrick Baudisch. 2006. Precise selection techniques for multi-touch screens. In *Proceedings of the SIGCHI conference on Human Factors in computing systems*. ACM, Montréal, Québec, Canada, 1263–1272.

[4] Xiaojun Bi, Yang Li, and Shumin Zhai. 2013. FFitts law: modeling finger touch with fitts' law. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. ACM, Paris, France, 1363–1372.

[5] Xiaojun Bi and Shumin Zhai. 2013. Bayesian touch: a statistical criterion of target selection with finger touch. In *Proceedings of the 26th annual ACM symposium on User interface software and technology*. ACM, St. Andrews Scotland, United Kingdom, 51–60.

[6] Xiaojun Bi and Shumin Zhai. 2016. Predicting finger-touch accuracy based on the dual Gaussian distribution model. In *Proceedings of the 29th Annual Symposium on User Interface Software and Technology*. ACM, Tokyo, Japan, 313–319.

[7] Rafal Bogacz. 2007. Optimal decision-making theories: linking neurobiology with behaviour. *Trends in cognitive sciences* 11, 3 (2007), 118–125.

[8] Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. 2016. Openai gym. *arXiv preprint arXiv:1606.01540* (2016), 1–4.

[9] Daniel Buschek and Florian Alt. 2015. TouchML: A machine learning toolkit for modelling spatial touch targeting behaviour. In *Proceedings of the 20th International Conference on Intelligent User Interfaces*. ACM, Atlanta, Georgia, USA, 110–114.

[10] Daniel Buschek and Florian Alt. 2017. ProbUI: Generalising touch target representations to enable declarative gesture definition for probabilistic GUIs. In *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems*. ACM, Denver, Colorado, USA, 4640–4653.

[11] Noshaba Cheema, Laura A Frey-Law, Kourosh Naderi, Jaakko Lehtinen, Philipp Slusallek, and Perttu Hämäläinen. 2020. Predicting mid-air interaction movements and fatigue using deep reinforcement learning. In *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems*. ACM, Honolulu, HI, USA, 1–13.

[12] Xiuli Chen, Aditya Acharya, Antti Oulasvirta, and Andrew Howes. 2021. An Adaptive Model of Gaze-based Selection. In *CHI 2021: ACM Conference on Human Factors in Computing Systems*. ACM, Online virtual, 1–11.

[13] Xiuli Chen, Gilles Bailly, Duncan P Brumby, Antti Oulasvirta, and Andrew Howes. 2015. The emergence of interactive behavior: A model of rational menu search. In *Proceedings of the 33rd annual ACM conference on human factors in computing systems*. ACM, Seoul, Republic of Korea, 4217–4226.

[14] Andy Cockburn, Carl Gutwin, and Saul Greenberg. 2007. A predictive model of menu performance. In *Proceedings of the SIGCHI conference on Human factors in computing systems*. ACM, San Jose, California, USA, 627–636.

[15] Jacob Cohen. 2013. *Statistical power analysis for the behavioral sciences*. Academic press.

[16] Wenzhe Cui, Suwen Zhu, Zhi Li, Zheer Xu, Xing-Dong Yang, IV Ramakrishnan, and Xiaojun Bi. 2021. BackSwipe: Back-of-device Word-Gesture Interaction on Smartphones. In *Proceedings of CHI 2021 - the SIGCHI Conference on Human Factors in Computing Systems*. ACM, Online virtual, 1–19.

[17] Seungwon Do, Minsuk Chang, and Byungjoo Lee. 2021. A Simulation Model of Intermittently Controlled Point-and-Click Behaviour. In *Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems*. ACM, Online virtual, 1–17.

[18] Stephen R Ellis and Robert J Hitchcock. 1986. The emergence of Zipf's law: Spontaneous encoding optimization by users of a command language. *IEEE transactions on systems, man, and cybernetics* 16, 3 (1986), 423–427.

[19] Christoph Gebhardt, Brian Hecox, Bas van Opheusden, Daniel Wigdor, James Hillis, Otmar Hilliges, and Hrvoje Benko. 2019. Learning cooperative personalized policies from gaze data. In *Proceedings of the 32nd Annual ACM Symposium on User Interface Software and Technology*. ACM, New Orleans, LA, USA, 197–208.

[20] Mayank Goel, Leah Findlater, and Jacob Wobbrock. 2012. WalkType: using accelerometer data to accomodate situational impairments in mobile touch screen text entry. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. ACM, Austin, Texas, USA, 2687–2696.

[21] Mayank Goel, Jacob Wobbrock, and Shwetak Patel. 2012. Gripsense: using built-in sensors to detect hand posture and pressure on commodity mobile phones. In *Proceedings of the 25th annual ACM symposium on User interface software and technology*. ACM, Cambridge, Massachusetts, USA, 545–554.

[22] Joshua Goodman, Gina Venolia, Keith Steury, and Chauncey Parker. 2002. Language modeling for soft keyboards. In *Proceedings of the 7th international conference on Intelligent user interfaces*. ACM, San Francisco, California, USA, 194–195.

[23] Tovi Grossman, Pierre Dragicevic, and Ravin Balakrishnan. 2007. Strategies for accelerating on-line learning of hotkeys. In *Proceedings of the SIGCHI conference on Human factors in computing systems*. ACM, San Jose, California, USA, 1591–1600.

[24] Niels Henze, Enrico Rukzio, and Susanne Boll. 2011. 100,000,000 taps: analysis and improvement of touch performance in the large. In *Proceedings of the 13th International Conference on Human Computer Interaction with Mobile Devices and Services*. ACM, Stockholm, Sweden, 133–142.

[25] Christian Holz and Patrick Baudisch. 2010. The generalized perceived input point model and how to double touch accuracy by extracting fingerprints. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. ACM, Atlanta, Georgia, USA, 581–590.

[26] Christian Holz and Patrick Baudisch. 2011. Understanding touch. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. ACM, Vancouver, BC, Canada, 2501–2510.

[27] Kasper Hornbæk and Antti Oulasvirta. 2017. What is interaction?. In *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems*. ACM, Denver, Colorado, USA, 5040–5052.

[28] Eric Horvitz. 1999. Principles of mixed-initiative user interfaces. In *Proceedings of the SIGCHI conference on Human Factors in Computing Systems*. ACM, Pittsburgh, Pennsylvania, USA, 159–166.

[29] Andrew Howes, Geoffrey B Duggan, Kiran Kalidindi, Yuan-Chi Tseng, and Richard L Lewis. 2016. Predicting short-term remembering as boundedly optimal

strategy choice. *Cognitive Science* 40, 5 (2016), 1192–1223.

[30] Jussi PP Jokinen, Sayan Sarcar, Antti Oulasvirta, Chaklam Silpasuwanchai, Zhenxin Wang, and Xiangshi Ren. 2017. Modelling learning of new keyboard layouts. In *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems*. ACM, Denver, Colorado, USA, 4203–4215.

[31] Katri Leino, Kashyap Todi, Antti Oulasvirta, and Mikko Kurimo. 2019. Computer-supported form design using keystroke-level modeling with reinforcement learning. In *Proceedings of the 24th International Conference on Intelligent User Interfaces: Companion*. ACM, Marina del Ray, California, USA, 85–86.

[32] Zhi Li, Maozheng Zhao, Yifan Wang, Sina Rashidian, Furqan Baig, Rui Liu, Wanyu Liu, Michel Beaudouin-Lafon, Brooke Ellison, Fusheng Wang, et al. 2021. BayesGaze: A Bayesian Approach to Eye-Gaze Based Target Selection. In *Graphics Interface 2021*. Online virtual, 231–240.

[33] Timothy P Lillicrap, Jonathan J Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. 2015. Continuous control with deep reinforcement learning. *arXiv preprint arXiv:1509.02971* (2015), 1–14.

[34] Wanyu Liu, Gilles Bailly, and Andrew Howes. 2017. Effects of frequency distribution on linear menu performance. In *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems*. ACM, Denver, Colorado, USA, 1307–1312.

[35] J Derek Lomas, Jodi Forlizzi, Nikhil Poonwala, Nirmal Patel, Sharan Shodhan, Kishan Patel, Ken Koedinger, and Emma Brunskill. 2016. Interface design optimization as a multi-armed bandit problem. In *Proceedings of the 2016 CHI conference on human factors in computing systems*. ACM, San Jose, California, USA, 4142–4153.

[36] Yuexing Luo and Daniel Vogel. 2015. Pin-and-cross: A unimanual multitouch technique combining static touches with crossing selection. In *Proceedings of the 28th Annual ACM Symposium on User Interface Software & Technology*. ACM, Charlotte, NC, USA, 323–332.

[37] Jennifer Mankoff, Scott E Hudson, and Gregory D Abowd. 2006. Interaction techniques for ambiguity resolution in recognition-based interfaces. In *ACM SIGGRAPH 2006 Courses*. ACM, Boston, Massachusetts, USA, 6–es.

[38] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. 2013. Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602* (2013), 1–9.

[39] Alistair Morrison, Xiaoyu Xiong, Matthew Higgs, Marek Bell, and Matthew Chalmers. 2018. A Large-Scale Study of iPhone App Launch Behaviour. In *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems*. ACM, Montreal, QC, Canada, 1–13.

[40] Tomer Moscovich. 2009. Contact area interaction with sliding widgets. In *Proceedings of the 22nd annual ACM symposium on User interface software and technology*. ACM, Victoria, BC, Canada, 13–22.

[41] Josip Musić and Roderick Murray-Smith. 2016. Nomadic input on mobile devices: the influence of touch input technique and walking speed on performance and offset modeling. *Human–Computer Interaction* 31, 5 (2016), 420–471.

[42] Mark EJ Newman. 2005. Power laws, Pareto distributions and Zipf's law. *Contemporary physics* 46, 5 (2005), 323–351.

[43] Alex Olwal, Steven Feiner, and Susanna Heyman. 2008. Rubbing and tapping for precise and rapid selection on touch-screen displays. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. ACM, Florence, Italy, 295–304.

[44] Julia Schwarz, Scott Hudson, Jennifer Mankoff, and Andrew D Wilson. 2010. A framework for robust and flexible handling of inputs with uncertainty. In *Proceedings of the 23nd annual ACM symposium on User interface software and technology*. ACM, New York, New York, USA, 47–56.

[45] Julia Schwarz, Jennifer Mankoff, and Scott Hudson. 2011. Monte carlo methods for managing interactive state, action and feedback under uncertainty. In *Proceedings of the 24th annual ACM symposium on User interface software and technology*. ACM, Santa Barbara, California, USA, 235–244.

[46] Katie Steele and H. Orri Stefánsson. 2020. Decision Theory. In *The Stanford Encyclopedia of Philosophy* (Winter 2020 ed.), Edward N. Zalta (Ed.). Metaphysics Research Lab, Stanford University.

[47] Richard S Sutton and Andrew G Barto. 2018. *Reinforcement learning: An introduction*. MIT press.

[48] Kashyap Todi, Gilles Bailly, Luis A Leiva, and Antti Oulasvirta. 2021. Adapting user interfaces with model-based reinforcement learning. In *CHI 2021: ACM Conference on Human Factors in Computing Systems*. ACM, Online virtual, 1–13.

[49] Michael Toomim, Travis Kriplean, Claus Pörtner, and James Landay. 2011. Utility of human-computer interactions: Toward a science of preference measurement. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. ACM, Vancouver, BC, Canada, 2275–2284.

[50] Yuan-Chi Tseng and Andrew Howes. 2015. The adaptation of visual search to utility, ecology and design. *International Journal of Human-Computer Studies* 80 (2015), 45–55.

[51] Daniel Vogel and Patrick Baudisch. 2007. Shift: a technique for operating pen-based interfaces using touch. In *Proceedings of the SIGCHI conference on Human factors in computing systems*. ACM, San Jose, California, USA, 657–666.

[52] Daniel Vogel and Géry Casiez. 2012. Hand occlusion on a multi-touch tabletop. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. ACM, Austin, Texas, USA, 2307–2316.

[53] David J Ward, Alan F Blackwell, and David JC MacKay. 2000. Dasher—a data entry interface using continuous gestures and language models. In *Proceedings of the 13th annual ACM symposium on User interface software and technology*. ACM, San Diego, California, USA, 129–137.

[54] Christopher JCH Watkins and Peter Dayan. 1992. Q-learning. *Machine learning* 8, 3-4 (1992), 279–292.

[55] Daryl Weir, Simon Rogers, Roderick Murray-Smith, and Markus Löchtefeld. 2012. A user-specific machine learning approach for improving touch accuracy on mobile devices. In *Proceedings of the 25th annual ACM symposium on User interface software and technology*. ACM, Cambridge, Massachusetts, USA, 465–476.

[56] Daniel Wigdor, Clifton Forlines, Patrick Baudisch, John Barnwell, and Chia Shen. 2007. Lucid touch: a see-through mobile device. In *Proceedings of the 20th annual ACM symposium on User interface software and technology*. ACM, Newport, Rhode Island, USA, 269–278.

[57] Koji Yatani, Kurt Partridge, Marshall Bern, and Mark W Newman. 2008. Escape: a target selection technique using visually-cued gestures. In *Proceedings of the SIGCHI conference on Human Factors in Computing Systems*. ACM, Florence, Italy, 285–294.

[58] Suwen Zhu, Yoonsang Kim, Jingjie Zheng, Jennifer Yi Luo, Ryan Qin, Liuping Wang, Xiangmin Fan, Feng Tian, and Xiaojun Bi. 2020. Using Bayes' Theorem for Command Input: Principle, Models, and Applications. In *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems*. ACM, Honolulu, HI, USA, 1–15.